42

# CARGA, TRANSFORMACIÓN Y PRESENTACIÓN DE INFORMACIÓN PROVENIENTE DE ARCHIVOS PLANOS UTILIZANDO EL LENGUAJE BASE SAS


**UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso**

## AGRADECIMIENTOS Y DEDICATORIA

A mi madre . . .

# INDICE

# INTRODUCCIÓN

Cuando terminamos la carrera y comenzamos a trabajar aprendemos que existen situaciones que nunca nadie nos menciona y muchas otras que escuchamos miles de veces (y que en muchos casos no son ciertas), la verdad es que en el ambiente de sistemas no todo es como lo imaginamos. Un caso muy particular es la forma en la que algunas empresas "grandes" manipulan la información, por ejemplo, ¿Cómo cree que están desarrollados los sistemas de información que registran las transacciones de los tarjeta habientes de un banco?, talvez pensemos en algún Sistema desarrollado con una herramienta que utiliza un lenguaje 4GL y un manejador de Base de Datos muy potente como Oracle, Sybase o Informix; sin embargo en algunos casos esto no es así, creo que un factor importante que no tomamos en cuenta es que estas empresas estaban funcionando mucho antes de que los lenguajes 4GL existieran lo mismo que los más modernos y eficientes manejadores de Bases de Datos. Las necesidades tecnológicas han orillado a estas empresas a alinearse dentro de los estándares del mercado y los de la misma empresa ya que las aplicaciones que surgen día a día dentro de esta "Sistemas que buscan automatizar y eficientar los procesos internos", toman para su desarrollo las más modernas herramientas. A pesar de ello, las antiguas aplicaciones no siempre son transportadas a la nueva tecnología y requieren de mantenimiento, ya que sería mucho más costoso y en algunas ocasiones incluso peligroso tratar de transportarlas. Muchas de las empresas que aun cuentan con este tipo de tecnología han evaluado la posibilidad de modificar estas aplicaciones, pero no siempre lo han llevado a la practica.

De acuerdo a lo planteado anteriormente resulta que ahora dentro de una misma empresa nos encontramos la mezcla de "antiguas y nuevas" tecnologías las cuales requieren una de la otra. por lo cual debe existir interacción entre los sistemas de información antiguos y las nuevas aplicaciones (las cuales muchas de las veces reciben información de las antiguas) Esta situación ha llevado a muchos negocios a enfrentarse a un nuevo problema, **LA TRANSFERENCIA DE INFORMACIÓN ENTRE APLICACIONES QUE NO SON COMPATIBLES**, ya sea por que no funcionan sobre la misma plataforma, no están desarrolladas bajo el mismo lenguaje o simplemente por que la Base de Datos donde reside la información es distinta y no compatible ¿Cuál es el problema?, algunas aplicaciones existentes fueron desarrolladas con la tecnología de hace 30 años y contienen información vital para el adecuado funcionamiento de las aplicaciones que surgen como respuesta a las necesidades de los negocios, es aquí donde encontramos la importancia de saber la forma adecuada de manipular la información, ya que muchas veces esta transferencia de información se realiza utilizando archivos planos mejor conocidos como archivos de texto, cuyo contenido no tiene ningún tipo de estructura de columnas y renglones como los que se manejan con las tablas dentro de un manejador de base de datos, de hecho en algunas ocasiones encontraremos archivos de texto con estructuras un tanto complejas que pueden ser difíciles de leer Hasta ahora esta forma de transferencia de información entre sistemas NO compatibles es una de las más utilizadas.

Además de lo anterior, actualmente miles de empresas en nuestro país y el resto del mundo, se están enfrentando al reto de convertir sus datos en información que sea útil para soportar la toma de decisiones. Dicho proceso, implica la extracción de información de los sistemas operacionales, la cual, además de provenir de bases de datos formales (Oracle,Sybase,Informix, etc), en la mayoría de los casos también incluyen los ya mencionados archivos planos.

Es por esto la importancia de saber manipular este tipo de archivos ya que tarde o temprano nos encontramos con este problema.

El presente trabajo pretende dar a conocer a los alumnos de la carrera de Matemáticas Aplicadas y Computación una herramienta que les permita manipular archivos planos, de una forma sencilla, presentando rápidos y eficientes resultados, utilizando siempre las mejores prácticas de programación adquiridas durante la carrera. las cuales permitirán optimizar código, tiempos de desarrollo y procesamiento además de esfuerzos y costos.

En el capitulo 1 se explican detalles acerca del Sistema SAS y la empresa que lo comercializa, además muestra una visión global de las soluciones que SAS Institute Inc. ofrece. Durante el capitulo 2 se explica la forma en que funciona SAS con el objetivo de familiarizar al lector con el ambiente de trabajo de la herramienta. El capitulo 3 muestra como realizar la carga de archivos planos de estructuras simples, jerárquicas, múltiples registros por observación y múltiples observaciones por registro Con los datos cargados en el Sistema SAS se procede a transformar la información esto es cubierto durante los capítulos 4 y 5. Finalmente en el capitulo 6 se muestra como presentar la información *utilizando procedimientos sencillos para la generación de reportes y gráficas.*

# CAPITULO 1

# SAS

Conocer el Sistema SAS para que se utiliza y la empresa que lo desarrolla y comercializa.

En 1993 la estrategia MVA permitió a SAS Institute liberar en forma casi simultánea nuevas versiones del Sistema SAS para los siguientes sistemas operativos: MVS, CMS, VSE, OpenVMS para VAX y AXP, OS/2 2.0, Windows, Windows NT, AIX, HP-UX, RISC/ULTRIX, Solaris y ConvexOS.

A través de años de desarrollo, el Sistema SAS se ha transformado en un completo Sistema de Entrega de Información. Ha ido creciendo hasta integrar más de 50 aplicaciones modulares; que permiten a las organizaciones un completo control sobre sus datos, desde el acceso, manejo y análisis, hasta su presentación. El Sistema SAS incluye herramientas para desarrollo de aplicaciones orientado a objetos, capacidades avanzadas de procesamiento cliente/servidor, nueva tecnología de visualización de datos, y acceso ilimitado a los datos.

En marzo de 1995 SAS Institute lanzó su primera versión del Sistema SAS en la Apple Macintosh y actualmente se está trabajando en el desarrollo de la versión 9 del Sistema SAS.

**¿DÓNDE ESTA?**

La oficina matriz de SAS Institute se encuentra localizados en Cary, Carolina del Norte, E.U. Con oficinas regionales y centros de capacitación a lo largo de todo el mundo, esto es presencia global que garantiza un excelente servicio a sus clientes, ya que durante 25 años SAS ha establecido oficinas, subsidiarias y distribuidores en más de 55 países y distritos al rededor del mundo. Esta estructura garantiza a los usuarios un rápido y fácil acceso al Soporte Técnico, capacitación y servicios de consultaría en cualquier localidad y en el lenguaje que se requiera. Cada año más oficinas son abiertas con el objetivo de mejorar la atención y servicios locales.

Oficinas SAS Inc.:

| REGION | LOCATION |
|---|---|
| Asia Pacifico y Latino América | Buenos Aires, Argentina / Sydney, Australia / Canberra, Australia / Melbourne, Australia / Perth, Australia / Brisbane, Australia / Adelaide, Australia / Sao Paulo, Brasil / Rio de Janeiro, Brasil / Brasilia, Brasil / Beijing, China / Shanghai, China / Guangzhou, China / Santiago, Chile / Bogota, Colombia / Hong Kong / Montevideo, Uruguay / Mumbai, India / New Delhi, India / Pune, India / Jakarta, Indonesia / Tokyo, Japan / Osaka, Japan / Kita Kyushu, Japan / Seoul, Korea / Kuala Lumpur, Malaysia / Mexico City, México / Wellington, New Zealand / Auckland, New Zealand / Lahore Cantt, Pakistan / Manila, Philippines / San Juan, Puerto Rico – Singapore / Bangkok, Thailand / Taipei, Taiwan / Kaohsiung, Taiwan / Hsinchu, Taiwan / Caracas, Venezuela |
| Oficinas en Canada | Toronto, Notaric / Montreal, Canada / Ottawa, Canada / Vancouver, Canada / Calgary, Canada / Quebec City, Canada (Sainte/Foy) |
| Oficinas en Europa | Bandariya, Arabia Saudita/ Wien, Austria / Leuven, Belgica/ Praha, Checoslovakia / Kobenhavn, / Dinamarca / Skanderborg, Dinamarca / Heidelberg, Alemania/ Hqtr Europa/ Espoo, Finlandia / Gregy-Sur-Yerres, Francia / Lyon Cedex, Francia / Aix En Provence, Francia / Nantes Cedex, Francia / Toulouse Cedex, Francia / Heidelberg, Alemania / Koln, Alemania / Berlin, Alemania / Muenchen, Alemania / Hamburg, Alemania / Frankfurt, Alemania / Athenas, Greecia / Budapest, Hungria / Dublin, Iranda / Milan, Italia / Roma, Italia / Venecia, Italia / Internationale, Luxenburgo / Herzilya, Israel / Huizen, Holanda/ Trollasen, Noruega/ Stavanger, Noruega / Vilnius, Lithuania / Warszawa, Polonia / Lisboa, Portugal / Bucharest, Rumania / Johannesburg Soud-Africa (Houghton / Capetown, Soud-Africa (Howardplace / Moscow, Russia / Ljubljana, Slovenia / Madrid, España/ Barcelona, España / Bratislava, Slovak Republic / Sollentuna, Suiza / Bruettisellen, Suecia / Istanbul, Turkia / Dubai, Emiratos Arabes unidos/ Marlow, Reino Unido/ Glasgow, Reino Unido/ Manchester, Reino Unido |

| | |
|---|---|
| Oficinas en Estados Unidos | Fayetteville, Arkansas / Bentonville, Arkansas (Lowell) / Little Rock, Arkansas (Russellville) / Bella Vista, Arkansas / Phoenix, Arizona / Irvine, California / San Diego, California / San Diego, California / San Francisco, California / Silicon Valley, California(San Jose / Los Angeles California / Palo Alto, California / Morgan Hill California / Denver, Colorado (Greenwood Village / Woodland Park, Colorado / Hartford, Connecticut(Glastonbury) / Southbury Connecticut / Norwalk, Connecticut / New Haven, Connecticut / New Milford, Connecticut / Orlando, Florida / Ft. Lauderdale, Florida / Miami, Florida @'g' / Tampa, Florida / Jacksonville, Florida / New Smyrna Beach, Florida / Atlanta, Georgia / Chicago, Illinois / Carbondale, Illinois / Itasca, Illinois / Indianapolis, Indiana / Carmel, Indiana / Kansas City, Kansas (Overland Park) / Louisville, Kentucky / New Orleans, Louisiana / Harvey, Louisiana / Boston, Massachusetts / Rockville, Maryland / Abingdon, Maryland / Detroit, Michigan (Southfield) / Grand Rapids, Michigan / Fremont, Michigan / Minneapolis, Minnesota / St. Louis, Missouri / Cary, North Carolina - Institute HQ / Charlotte, North Carolina / Wilmington, North Carolina / Greensboro,North Carolina(Oak Ridge) / Charlotte, North Carolina / Raleigh, North Carolina / Cary, North Carolina (Ibiomatics) / Omaha, Nebraska / Bedminster, New Jersey / Mendham, New Jersey / Williamston, New Jersey / Mahwah, New Jersey / Cedar Crest, New Mexico / New York, New York / Albany, New York / Whitesboro, New York / Cazenovia New York / Amherst, New York / Cincinnati, Ohio / Cleveland, Ohio (Independence) / Westerville, Ohio / Dayton, Ohio / Tulsa, Oklahoma / Portland, Oregon / Eugene, Oregon / Pittsburgh, Pennsylvania / Philadelphia, Pennsylvania (Wayne) / San Juan, Puerto Rico / Greenville, South Carolina (Roebuck) / Charleston, South Carolina(Mt Pleas / Spartanburg, South Carolina / Nashville, Tennessee / Memphis, Tennessee / Austin, Texas / Dallas, Texas / Houston, Texas / Ft. Worth, Texas / Salt Lake City, Utah / American Fort, Utah / Richmond, Virginia (Glen Allen) / Seattle, Washington / Milwaukee, Wisconsin (Brookfield) / Weirton, West Virginia / Seneca, New York |

**¿QUÉ HACE?**

SAS Institute desarrolla, vende y soporta el proceso de Data Warehouse y Soporte a Decisiones.

Este software ofrece un conjunto de herramientas para la entrega de información que da el soporte a la toma de decisiones, esto permite a las compañías transformar sus datos de una forma muy variada lo cual les permite tomar decisiones exitosas. El software SAS accede datos desde casi cualquier plataforma y formato. Permite limpiar y transformar la información de forma que sea entendible y útil para los tomadores de decisiones, además puede almacenar esta información de forma abierta y eficiente en una estructura de Data Warehouse dentro del propio SAS o en cualquier otro tipo de repositorio (Oracle, Sybase, Informix, otra base de datos o inclusive archivos planos).

Para explorar datos SAS provee de herramientas para análisis multidimensional (OLAP) consultas y reporteo. Sistemas de Información Ejecutiva (EIS), Minería de Datos, visualización de datos, además de la capacidad de desarrollo de aplicaciones, soluciones cliente/servidor y herramientas WEB.

Además SAS entrega soluciones de negocio empacadas que sirven para consolidación financiera, reporteo y análisis clínico.

**CLIENTES**

Los clientes de SAS Institute vienen de las grandes industrias como Bancos, farmacéuticas, manufactureras, telecomunicaciones y gobierno. Todos con las mismas necesidades básicas, hacer mejor, decisiones más estratégicas para ganar ventaja competitiva.

SAS Institute soporta y organiza anualmente conferencias de grupos de usuarios de SAS de distintas industrias y partes del mundo se conocen e intercambian ideas con líderes programadores de SAS, esto les da una oportunidad de influenciar las prioridades de desarrollo y la dirección de las tecnologías.

La oportunidad de conocer las necesidades de los usuarios ha aumentado la lealtad de clientes en la industria del software. Más del 98% de los clientes eligen renovar sus licencias de este software año tras año.

La mayoría de los clientes de SAS México tienen el software SAS en más de una plataforma. Ellos abarcan desde los sectores de: Industria y Comercio, Educación, Banca, Servicios Financieros y Seguros, entre otros.

Algunos de los clientes de SAS México

| | |
|---|---|
| ACCIVAL | ITESM CD. MEXICO |
| A.C. Nielsen | ITESM EDO.MEXICO |
| Aeromexico | ITESM MONTERREY |
| American Express | ITESM MORELOS |
| Banamex | KEMET MATAMOROS |
| Bancomer | KEMET NVO. LEON |
| Bancomer/Promex | MATISS |
| Bayer de México | MEXICANA DE AVIACION |
| C.A.F.E.T. | NEWELL ARAÑO Y AS |
| C.I.B. Noroeste | PEMEX |
| C.I.MAT/Univ. Gto | PHARMACIA & UPJOHN |
| C.I.Quim. Aplicada | Philip Morris |
| Casa de Bolsa INVERLAT | PMI |
| CIMMYT | Procter & Gamble |
| CIMVESTAV / MERIDA | Promeco |
| CIMVESTAV / QRO | Sai Consultores |
| CINSA | Secretaria de Comercio y Fomento |
| CNBV | Secretaria de Educación P. |
| Colegio de la Frontera Sur | Sedesol Progresa |
| Colegio de Posgraduados | Smartketing |
| Comisión Federal de Electricidad | Técnica Omya |
| EDS | Tubería Nacional (Adición) |
| ENEP Acatlan | Tubos de Acero (Mexico) |
| Estadísticos y Clínicos Asociados | UAM Xochimilco |
| Euroamerican Capital Corporation Limited | Univ. Autónoma de cd.Juarez |
| FUMIAF | Univ. Autónoma de Coahuila |
| G. de Villa y Asociados | Univ. Autónoma de Nayarit |
| GNP | Univ. Autónoma de Zacatecas |
| HEWITT ASOCIATES LLC | Univ. Michoacana de Hidalgo |
| Hospital General | Univ.de las Americas Puebla |
| Hylsa | Univ.Metropolitana Azcapot. |
| IBM | Universidad Anahuac |
| IIMAS | Universidad Antonio Narro |
| INCOM | Universidad Autónoma de Hidalgo |
| Ind. Vinícolas Pedro Domec | Universidad de Cuautitlan |
| INEGI | Universidad de Chapingo |
| ININ | Universidad de Monterrey |
| Inst. Tecnológico de Sonora | Universidad de Sonora |
| Instituto Manantlan | Video Centro |
| Inv. Integral de Mercados | Zurich Compañía de Seguros |
| ISOSA | |

**VENTAJAS FRENTE A LA COMPETENCIA**

Entre las ventajas que el Sistema SAS ofrece en comparación con otros sistemas en el mercado, podemos mencionar:

- Estabilidad de la Compañía - Mientras que otras empresas de software han desaparecido del mercado, o han tenido que fusionarse con otras, o simplemente han despedido personal para poder sobrevivir; los ingresos de SAS Institute han crecido a razón de un 22% anual desde hace 24 años, y el número de su personal ha crecido en un porcentaje mayor durante los dos últimos años.

- Presencia en México de SAS Institute - Desde 1992 SAS Institute está presente en México en forma directa y no a través de un distribuidor. Esto le asegura a los clientes, que tendrán el respaldo de una empresa en México que provee soporte técnico y consultoría y que detrás del personal de SAS Institute en México, se encuentran los 3,000 empleados de la matriz en Cary, Carolina del Norte.

- Confianza en el software - Mientras que el Sistema SAS pasa por las pruebas más rigurosas de control de calidad antes de liberar una nueva Versión, para que así, el usuario pueda confiar plenamente en que sus aplicaciones críticas de negocios no se verán afectadas por las nuevas versiones. Los productos de software de otras compañías reciben reportes negativos de la prensa especializada, por las numerosas fallas que contienen.

- Potencial de herramientas de análisis y manejo de datos - Desde sus comienzos a principio de la década de los 70's SAS Institute ha sido el líder de la industria en lo que se refiere a la potencia de sus herramientas de análisis.

- Reconocimientos de la Industria - SAS ha ganado el premio de Producto del año en la categoría de Data Warehouse por dos años Consecutivos según la revista DATAMATION en febrero del 96 y 97. DM Review World Class Solutions Award 96 por Meta Group y DCI ganador en 2 categorías Decision Support y Service and Support SAS fue el único nominado en las 4 categorías. Meta Data Mining Gold Rush Agosto del 96 otorgada por Meta Group y DCI. Ganadores LTV Steel/SAS Institute Data Warehousing Institute's Data Mining Competition Julio del 96. Y actualmente ha ganado premios como mejor herramienta de Data Warehouse según DM en 1999 y el 2000.

- Potencial de herramientas de desarrollo de aplicaciones - Los clientes en el área de Sistemas de Información Ejecutiva, escogen el Sistema SAS como herramienta de desarrollo, no por lo atractivo de las presentaciones que son posibles producir, sino por la extrema flexibilidad que proporciona, y la facilidad para integrar todas las herramientas de análisis disponibles en el Sistema SAS.

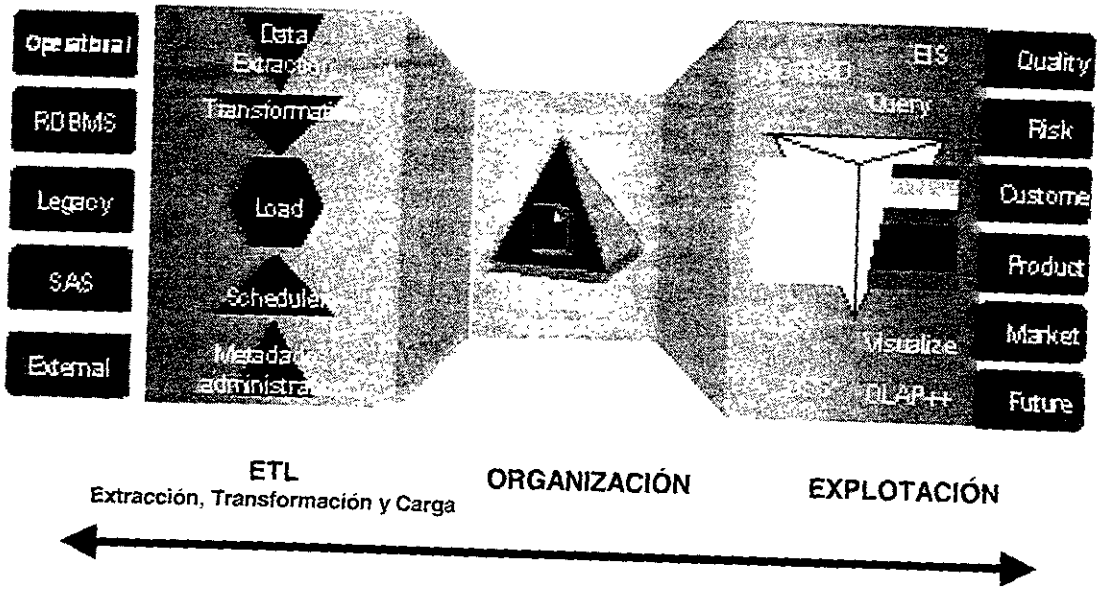
Finalmente, el Sistema SAS provee una solución integral, eficiente, flexible y fácil de usar para todas las áreas operativas de una empresa.

## 1.3 USOS DEL SISTEMA SAS

El Sistema SAS provee la forma más eficiente de transformar los datos en información. Los datos y las aplicaciones que corren en diferentes plataformas de hardware pueden ser integrados en un único ambiente de software y de esta manera entregar a los usuarios finales información relevante y actualizada.

El software SAS es un sistema de entrega de información que provee acceso transparente a cualquier fuente de datos, incluyendo archivos planos, archivos jerárquicos, y los más importantes proveedores de bases de datos relacionales. También incluye su propia base de datos de información para almacenamiento y manejo.

Expresado de la manera más simple, SAS tiene métodos de reunir los datos provenientes de diferentes fuentes en un modelo único, para separar físicamente los sistemas de datos operacionales de una empresa de sus sistemas de soporte a decisiones. Consiste en un repositorio de información construido utilizando datos de los sistemas de diferentes departamentos de la empresa, de manera tal que los datos puedan ser fácilmente modelados y analizados por los gerentes de las áreas de negocios.




Siendo SAS Institute el único proveedor de una solución completa para construir una este tipo de repositorios, él mismo dispone de todos los elementos críticos necesarios para optimizar los datos en que se basa la toma de decisiones estratégica. Estos elementos incluyen:

- Herramientas de acceso a los datos en cualquier formato y plataforma.
- Un motor robusto e integrado de transformación de datos, que funciona en múltiples plataformas.
- La ubicación y arquitectura ideales del repositorio de soporte a la toma de decisiones.
- Herramientas para análisis multi-dimesional, OLAP, consultas, reporteo, Minería de Datos, análisis estadístico y otros.
- El Sistema SAS también soporta los principales protocolos de comunicación, cubre los cinco modelos de procesamiento cliente/servidor de acuerdo a Gartner Group.
- El Sistema SAS soporta un amplio rango de aplicaciones, destacándose: análisis estadístico, análisis gráfico de datos, análisis de datos guiado, mejoramiento de la calidad, diseño experimental,

dministración de proyectos, programación lineal y no-lineal, generación de reportes y gráficas. ıanipulación y despliegue de imágenes, sistemas de información geográfica, visualización nultidimensional de datos, aplicaciones de multimedia, sistemas de información ejecutiva, etc.

esumen SAS le proporciona a las compañías la experiencia y las herramientas para la implementación ,oluciones como:

- Balanced Scorecard
- Claims Reserving
- Cost Management
- Credit Analysis
- Customer Relationship Management
- Database Marketing
- e-Discovery
- Enterprise Performance Management
- Executive Information Systems
- Experimental Design
- Exploratory Data Analysis
- Financial Consolidation, Reporting & Analysis
- Financial Management
- Forecasting
- Geographic Reporting
- Guided Data Analysis
- Human Resources Management
- IT Systems Management
- Leveraging ERP Systems
- Management Science
- Market Research
- Operations Research
- Performance Management
- PharmaHealth Technologies
- Portfolio Analysis
- Process Management
- Project Management
- Public Sector
- Quality Improvement
- Risk Management
- SAS Business Templates
- Statistical Analysis
- Supplier Relationship Management
- Statistical Process Control

La solución a cualquier problema comienza con una fuerte base. No importa que problema de negocio se tenga que resolver, todas las soluciones SAS están construidas con una arquitectura para la entrega de información (Information Delivery Architecture por sus siglas en ingles IDA). Entre las tecnologías que se ofrecen se encuentran:

- Data Warehousing
- Data Mining
- Web Enablement
- Wireless Technologies
- OLAP
- Collaborative Technologies
- Integration Technologies

Para mayor información referente a cualquiera de las soluciones mencionadas, consultar la página: www.sas.com/soluciones .

# CAPITULO 2

# COMENZANDO CON SAS

Utilizar el ambiente de trabajo del sistema SAS y comprender la lógica del lenguaje *BASE SAS*.

## 2.1 CONCEPTOS FUNDAMENTALES

### EL AMBIENTE SAS

El Software SAS es manejado utilizando, dentro del ambiente de trabajo de SAS se pueden abrir y cerrar ventanas. Una ventana

- Es abierta con:
  - ✓ Un comando que llame a la ventana
  - ✓ Una selección del menú
- Es cerrada con:
  - ✓ El comando END ( **FILE** → **END** del menú principal)
  - ✓ Con el comando *nombre de la ventana* OFF ( **FILE** → **CLOSE** del menú principal)
- Puede ser abierta pero no visible.
- Permanece abierta hasta especificar que sea cerrada.

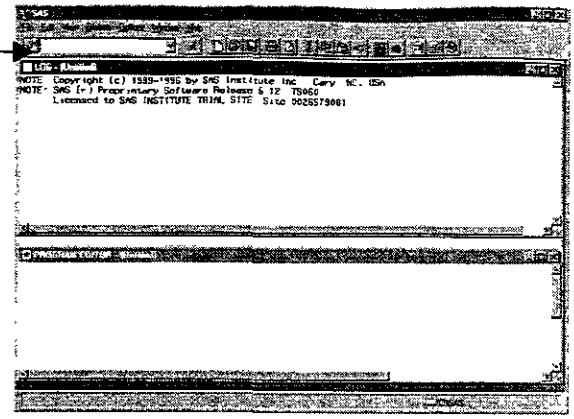
Tres ventanas le permiten al usuario conectarse directamente al sistema SAS y administrar los aspectos básicos de cualquier sesión SAS

| VENTANA | FUNCION |
|---|---|
| **PROGRAM EDITOR** | Proporciona un editor de texto para el desarrollo de Programas SAS |
| **LOG** | Muestra información sobre la ejecución de programas SAS y proporciona al usuario mensajes de error, avisos y notas. |
| **OUTPUT** | Muestra automáticamente la salida generada. |

Una ventana puede estar activa a la vez. Para activar una ventana se debe

- Usar el nombre de la ventana en la línea de comando
- Posicionar el cursor en cualquier lugar dentro de la ventana y presionar ENTER
- Posicionar el Mouse dentro del a ventana usando un click
- Seleccionar en el menú principal WINDOW y después seleccionar el nombre de la ventana
- Seleccionar en el menú principal GLOBALS y después seleccionar el nombre de la ventana
- Si la ventana ha sido minimizada, doble click para regresarla a su tamaño normal

**Línea de Comandos**



Algunos de los comandos más utilizados son:

| COMANDO | ACCIÓN |
|---|---|
| **PGM** | Activa la ventana del Program Editor |
| **OUTPUT** | Activa la ventana del OUTPUT |
| **LOG** | Activa la ventana del LOG |
| **RECALL** *n* | Llama al Program Editor el *n-ésimo* programa ejecutado |
| **FILE** *'nombre del archivo'* | Almacena el contenido de la ventana activa en el archivo especificado. |
| **INCLUDE** *'nombre del archivo'* | Carga en la ventana activa el contenido del archivo especificado |
| **CLEAR** *'nombre de la ventana'* | Limpia el contenido de la ventana activa o de la ventana especificada |
| **ZOOM** | Minimiza o maximiza la ventana activa |
| **PMENU** | Activa y desactiva una línea de comando en las ventanas abiertas. |
| **HELP** | Hace un llamado a la ventana de ayuda. Es también posible utilizar este comando, seguido de los datos que se desea buscar en la ayuda, esto permitirá tener un acceso mucho más rápido a la información que se busca. |
| **SUBMIT** | Ejecuta el programa que se encuentra contenido en la ventana del PROGRAM EDITOR. |
| **NUMBERS** | Activa y desactiva la numeración de líneas en el PROGRAM EDITOR |
| **BYE** | Salir del Sistema |

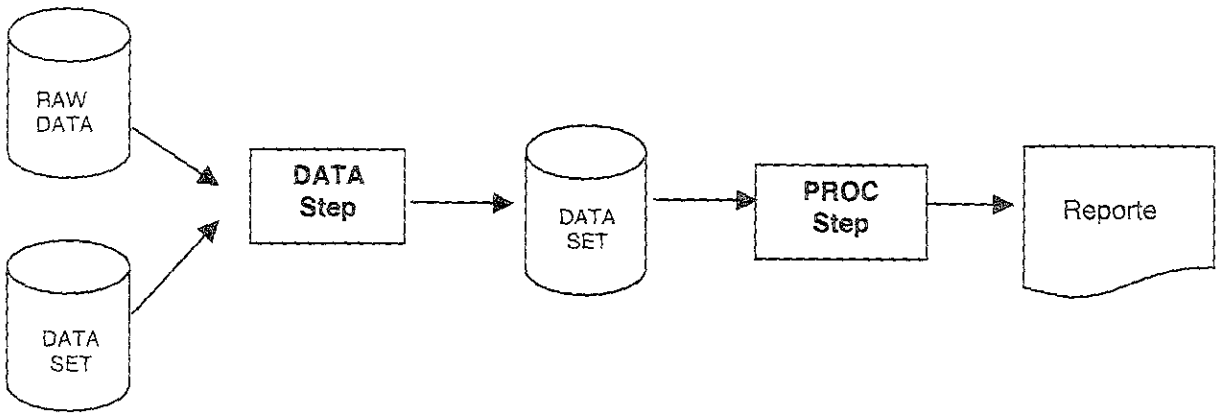
Estos comandos están disponibles dentro de las opciones del menú principal, cualquiera de las ventanas abiertas a consecuencia de utilizar alguno de los comandos anteriormente descritos, podrá ser cerrada de la siguiente manera.

| COMANDO | ACCIÓN |
|---|---|
| **CLOSE** | Cierra la ventana activa y salva los cambios efectuados en esta (si es que los hubo). |
| **CANCEL** | Cierra la ventana activa sin salvar los cambios efectuados en esta (si es que los hubo). |
|  | Utilizando el botón estándar para cerrar ventanas. Cierra la ventana activa y salva los cambios efectuados en esta (si es que los hubo). |

**PROCESANDO SAS**

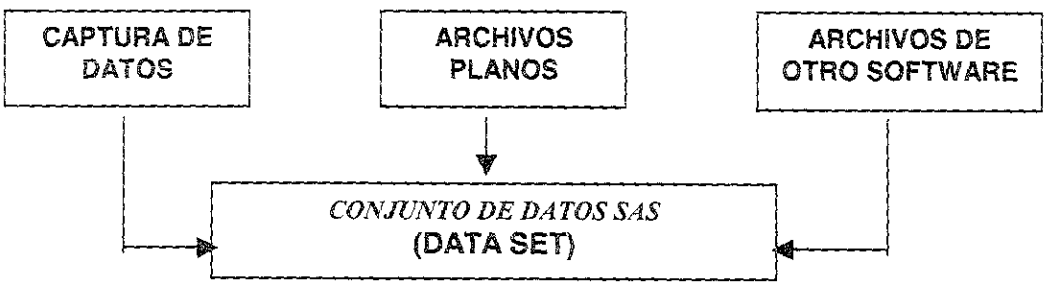
Un programa SAS es una secuencia de pasos. Dentro del lenguaje BASE SAS, hay solo dos tipos de pasos:

- Pasos DATA que son típicamente utilizados para crear archivos tipo SAS (que llamaremos DATA SETS SAS), pero pueden también ser utilizados para producir reportes.
- Pasos PROC que son utilizados para procesar DATA SETS SAS (generar reportes y gráficas, editar datos. ordenar datos, etc.) y en algunos casos para crear data sets.




**DATA SETS SAS**

Con el Sistema SAS se tiene acceso a los datos almacenados en cualquier formato.




Al leer los datos, se puede utilizar el sistema SAS para modificarlos o transformarlos, de distintas formas, antes de hacer análisis, reportes, generar gráficas etc.

El sistema SAS trabaja con información que deberá encontrarse en un conjunto de datos SAS para poder utilizar procedimientos SAS y analizarlos. Antes de iniciar una sesión de SAS para realizar alguna tarea se requiere saber como se relacionan los datos que vamos a utilizar con un conjunto de datos SAS.

Para la generación de informes o análisis de datos con el software SAS, estos deben encontrarse en una forma en que el sistema SAS pueda reconocer fácilmente. Esa forma se llama DATA SET y esto es un conjunto de datos SAS almacenados en archivos de datos tipo SAS y consta de:

- Una parte descriptora: Que contiene atributos e información acerca de los datos.
- Una parte de datos: Que contiene los valores de los datos.

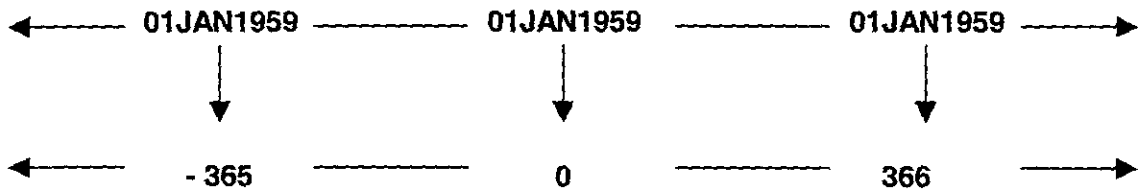
Los valores de los datos, en la parte de datos de un conjunto de datos SAS, están organizados en una tabla rectangular.

Data Set MARCH: Parte de datos

| FLIGHT | DEST | DATE | FREIGHT |
|---|---|---|---|
| 387 | CPH | 08MAR1995 | 367 |
| 439 | | 08MAR1995 | 187 |
| 622 | LON | 08MAR1995 | . |
| 872 | LAX | 08MAR1995 | 293 |
| 271 | PAR | 08MAR1995 | 279 |

Las columnas de la tabla se llaman VARIABLES

- Las variables corresponden a los campos y cada variable tiene un nombre.
- Hay dos tipos de variables: CHARACTER y NUMERIC.
- Las variables de tipo CHARACTER pueden tener de 1 a 200 caracteres de longitud.
- Las variables de tipo NUMERIC son almacenadas como números de punto flotante en 8 bytes por default este espacio permite almacenar hasta 16 o 17 dígitos significantes.
- No existe limite en el número de variables que pueden ser almacenados en un DATA SET.
- Las variables de tipo fecha son almacenadas como un tipo especial de variable numérica, esta fecha SAS es interpretada como el número de días transcurridos entre el 01 de ENERO de 1960 hasta la fecha que sé esta almacenando, ejemplo:




Las filas son llamadas OBSERVACIONES.

- Las observaciones corresponden a los registros o líneas de datos.
- No existe limite en el número de observaciones que pueden ser almacenadas en un DATA SET.

La estructura rectangular de un data set SAS, implica que existe un valor por cada variable por cada observación.

- Un valor missing de una variable de tipo NUMERIC, es desplegado con un punto.
- Un valor missing de una variable de tipo CHARACTER, es desplegado como un blanco.

SAS Data Set MARCH: Parte de datos

| FLIGHT | DEST | DATE | FREIGHT |
|---|---|---|---|
| 387 | CPH | 08MAR1995 | 367 |
| 439 | | 08MAR1995 | 187 |
| 622 | LON | 08MAR1995 | . |
| 872 | LAX | 08MAR1995 | 293 |
| 271 | PAR | 08MAR1995 | 279 |

La parte descriptora de un DATA SET contiene:

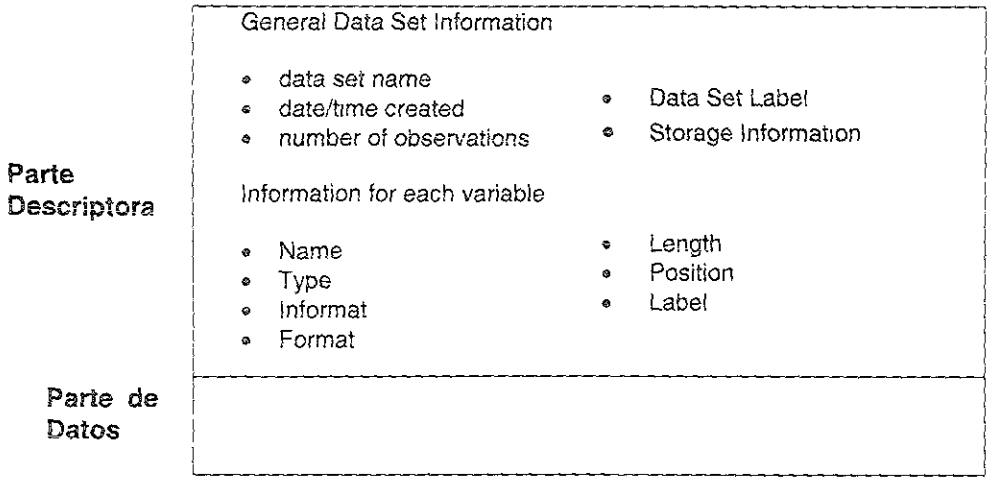
Información general acerca del DATA SET como:

- Nombre del DATA SET
- Fecha y hora en la que fue creado
- Número de observaciones

Información de los atributos de cada una de las variables en el DATA SET como:

- Nombre de la variable
- Tipo (NUMERIC o CHARACTER)
- Longitud (en Bytes)
- Formato
- Etiqueta

**SAS DATA SET**




El nombre de un DATA SET tiene dos niveles, el primero que determina si el DATA SET es Temporal o Permanente hace referencia a la localidad donde se encuentra almacenado. La forma general de nombrar DATA SET es la siguiente:

***LIBRERÍA.DATA SET***

*LIBRERÍA* Es una asignación lógica a una locación física en el disco.
*DATA SET* Especifica un archivo SAS dentro de una LIBRERÍA.

Cada vez que se abre una sesión de SAS esta crea automáticamente un directorio de trabajo, al cual asigna una librería llamada WORK, la cual es eliminada junto con su contenido, al cerrar la sesión.
Es por esta razón que al leer, guardar o crear un DATA SET SAS se hace de la siguiente manera:

- Temporal (referenciando la librería WORK u omitiendo el nombre de la librería).
- Permanente (referenciando otra librería que se halla declarado previamente).

☞ **CUANDO SE LEEN VOLUMENES DE DATOS MUY GRANDES, ES RECOMENDABLE QUE LAS LIBRERÍAS DE LECTURA Y ESCRITURA, SE LOCALICEN EN DISTINTAS UNIDADES DE DISCO PARA EVITAR LEER Y ESCRIBIR DESDE EL MISMO LUGAR, ESTO OPTIMIZARA LOS TIEMPOS DE RESPUESTA EN LOS PROCESOS. PARA MAYOR INFORMACIÓN REFERENTE A LAS LIBRERÍAS SE PUEDE CONSULTAR ALGUNO DE LOS MATERIALES INCLUIDOS EL LA BIBLIOGRÁFIA.**

*Estas librerías deben ser declaradas dentro del programa o al iniciar la sesión de SAS.*

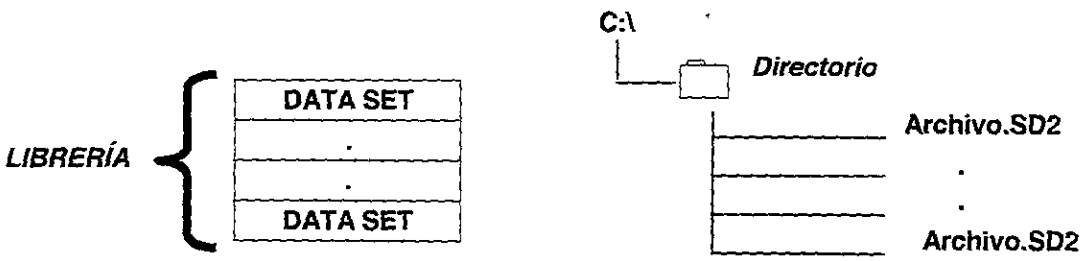
Las reglas en SAS para nombrar Data Set y variables son las mismas:

- Deben tener de uno a ocho caracteres de longitud.
- Comenzar con una letra (entre A y Z) o un guión bajo ( _ ).
- Continuar con alguna combinación de números, letras y/o guión bajo.

**LIBRERIAS SAS**

Una librería SAS es una colección de archivos SAS. Una librería SAS tiene diferentes representaciones dependiendo del Sistema Operativo en el cual se esta trabajando. (Durante este trabajo se hará referencia a una plataforma que utiliza Windows).

Una librería es una referencia temporal que utiliza SAS para almacenar información. La asignación de las librerías que se utilizarán durante una sesión de trabajo, deberán ser realizadas al inicio del mismo. La librería WORK es asignada automáticamente al iniciar una sesión.



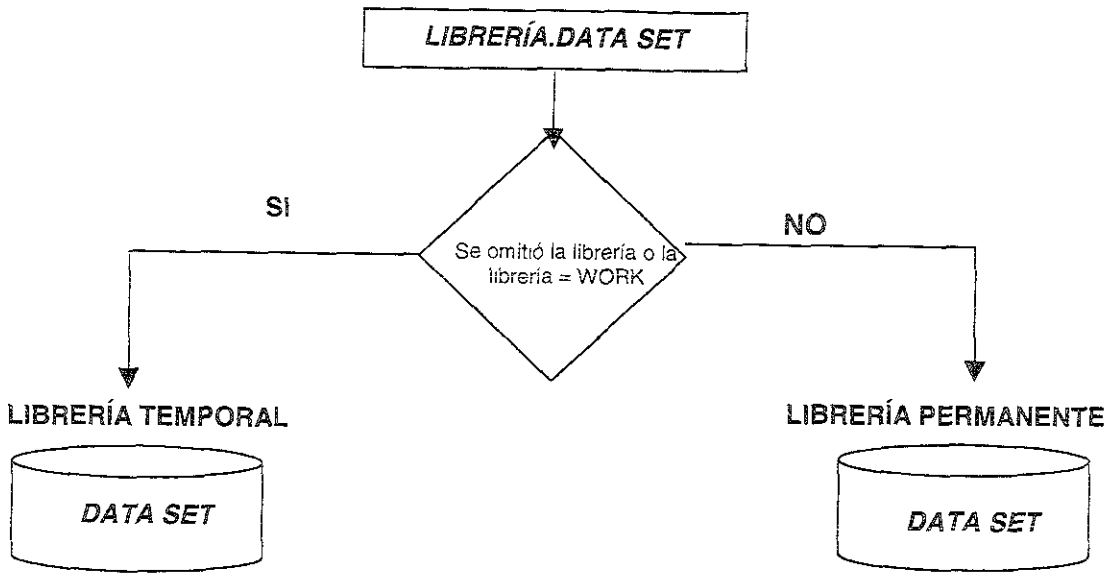

Por lo tanto y en este contexto, entendemos que el nombre de un archivo SAS esta formado por dos niveles, donde, el primer nivel determina si el DATA SET es temporal o permanente y también indica donde esta almacenado el DATA SET.

***LIBRERÍA.DATA SET***

***LIBRERÍA*** Es un nombre que esta asociado con una locación física. (En el caso de cualquier Sistema Operativo, esto sería el equivalente a un directorio).

***DATA SET*** Refiriere o especifica el nombre del archivo SAS dentro de la librería.

Si el primer nivel no es especificado (***LIBRERÍA***), entonces la librería utilizada por default es la librería WORK (librería Temporal que se asigna automáticamente al iniciar una sesión de SAS), por lo tanto el DATA SET queda almacenado de forma Temporal, ya que el contenido dentro de esta librería es borrado al final de cada sesión de SAS.

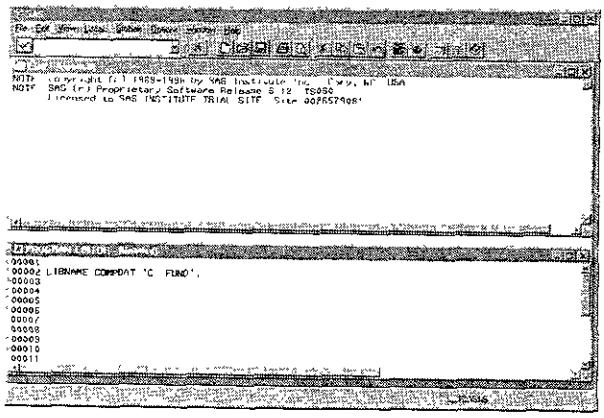


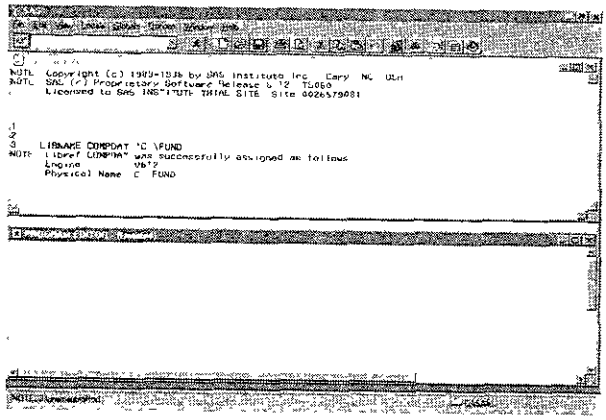

Se puede utilizar la sentencia LIBNAME para declarar una librería, la sintaxis es:

**LIBNAME** *LIBRERÍA* '*Directorio a donde apunta la librería*';

*LIBRERÍA* Este nombre debe ser asignado respetando las reglas para asignación de nombres dentro de SAS

Ejemplo: Se quiere asignar una librería llamada COMPDAT al directorio C:\FUND.




Una vez que esta sentencia es ejecutada, aparece un mensaje en la ventana del LOG que indica que la librería ha sido asignada.

**NOTA**: En el ejemplo puede observarse como al ejecutar el programa, este desaparece de la ventana del PROGRAM EDITOR, para traerlo nuevamente se requiere ejecutar el comando RECALL como se menciono anteriormente.

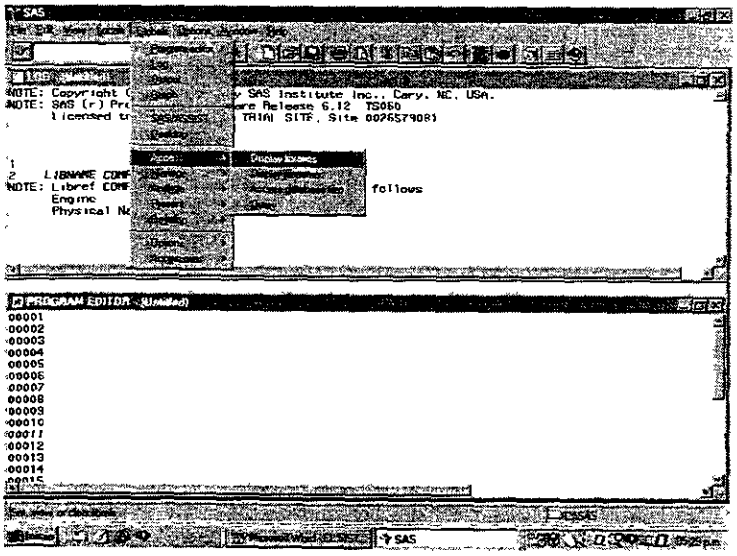
Es importante mencionar que una librería puede hacer referencia a solo un directorio del Sistema Operativo, si se desea ver más directorios utilizando SAS, se debe asignar una librería distinta para cada uno de estos. Un directorio puede ser accedido a través de una o más librerías.

☞ **LAS NUEVAS VERSIONES DEL SISTEMA SAS TIENEN LA HABILIDAD DE PODER DECLARAR LIBRERÍAS DE LIBRERÍAS, ES DECIR LIBRERÍAS QUE APUNTEN HACIA OTRAS LIBRERÍAS, LAS VENTAJAS DE ESTO SON VARIAS: NO SE REQUIERE DECLARAR MÚLTIPLES LIBRERÍAS EN UN PROGRAMA ADEMÁS DE QUE PERMITE INTEGRAR DISTINTAS PLATAFORMAS Y FUENTES DE DATOS.**

### INVESTIGANDO LA ESTRUCTURA DE UN DATA SET

Como se menciono anteriormente, un DATA SET esta formado por una parte de datos y por una parte descriptora, en la segunda, se encuentra toda la información referente al DATA SET y a los datos que se almacenan en este, esta información puede ser accedida utilizando el comando VAR o como se muestra a continuación:

La opción del menú que se muestra desplegará los nombres de todas las librerías asignadas durante la sesión de SAS en la cual se esta trabajando. Este mismo resultado es producido al seleccionar el botón  de la barra de herramientas.

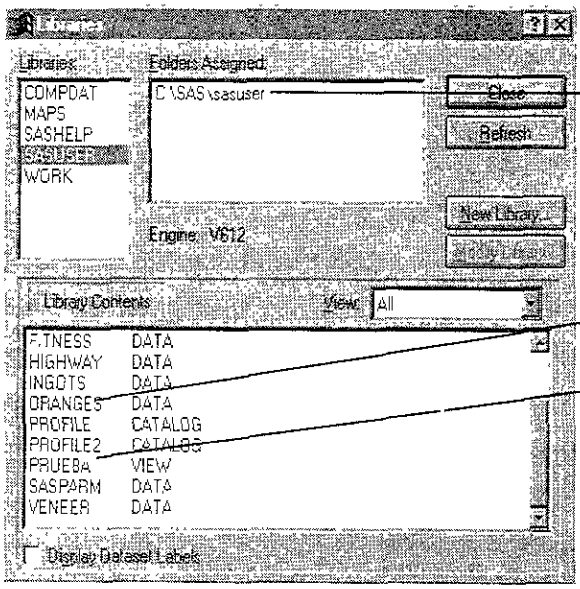


A través de esta ventana, se puede conocer el contenido de una librería, el tipo de archivo almacenados en esta y también la dirección a la cual esta apuntando la librería. El tipo de archivos que se desplegaran son archivos SAS, pueden existir otro tipo de archivos almacenados en ese directorio, pero a través de esta ventana solo se podrá acceder archivos SAS, entre los cuales se distinguen los siguientes tipos:

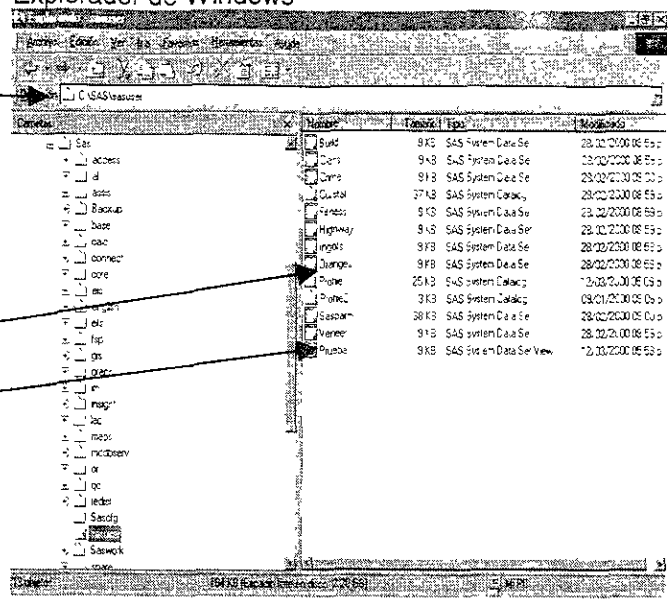
- Archivos tipo DATA los cuales si observáramos desde el explorador de Windows, serán aquellos con una extensión .SD2 (SAS Ver.6.12) o .SAS7BDAT (versiones posteriores de SAS), estos archivos son los llamados DATA SETS contiene información en formato SAS y están formados por una parte descriptora y una parte de datos.
- Archivos tipo VIEW, llamados vistas de datos, dentro de SAS son tratados y referenciados igual que un DATA SET sin embargo estos solamente contienen la parte descriptora del archivo, lo cual provoca que mantengan una relación con los archivos a partir de los cuales fueron generados. Desde el explorador de Windows los distinguimos por su extensión .SV2. (SAS Ver. 6.12) o .SAS7BVAT (versiones posteriores de SAS).
- Archivos tipo CATALOG los cuales son utilizados para almacenar distintos tipos de objetos SAS. Desde el explorador de Windows pueden ser ubicados como aquellos archivos con una extensión .SC2 (SAS Ver. 6.12) o .SAS7BCAT (versiones posteriores de SAS).

Los archivos de tipo CATALOG y VIEW, no son motivo de este trabajo, por lo cual no se mencionan más detalles al respecto.

Librería SASUSER vista desde la ventana LIBNAME de SAS

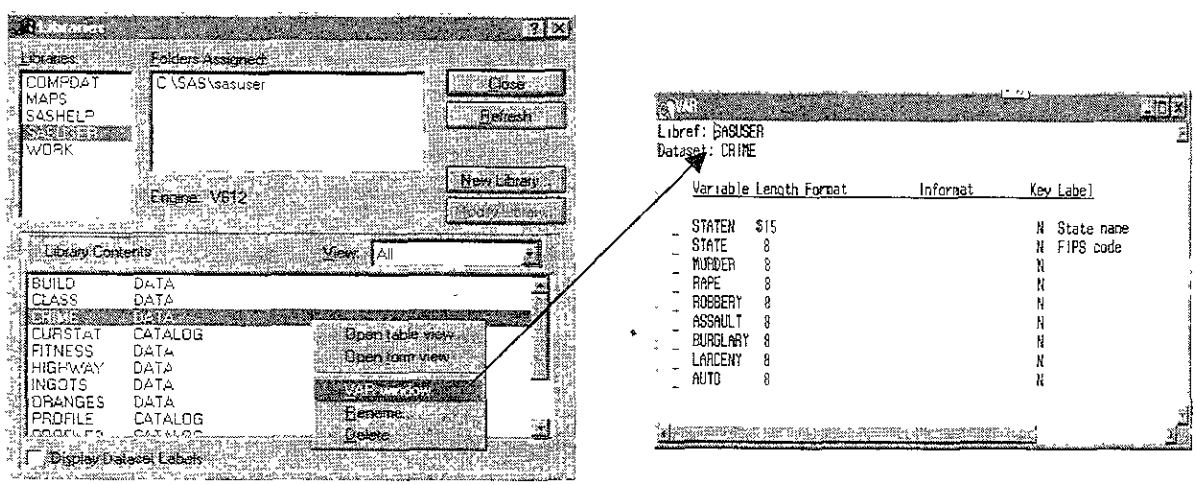



Contenido del directorio C:\SAS\SASUSER al que apunta la librería SASUSER, visto desde el Explorador de Windows



Como se puede observar los archivos SAS son transportables entre equipos y plataformas de distintos tipos. esta facilidad de transportar información de un lugar a otro y reutilizarla le da a SAS una ventaja ante otras herramientas.

Una vez dentro de la ventana de las librerías y conociendo el contenido de estas, a continuación se muestra como conocer el contenido de los archivos tipo DATA (DATA SET).
Para conocer la parte descriptora de un DATA SET solo tendremos que seleccionar la librería donde esta ubicado, seleccionar el DATA SET y con el botón derecho del mouse seleccionar la opción VAR WINDOW del POPMENU, como se muestra a continuación



esta opción permite conocer la parte descriptora de un DATA SET donde se puede observar:

- La librería donde esta almacenado el DATA SET (Encabezado de la Ventana)
- El nombre del DATA SET (Encabezado de la ventana)
- Los Nombres de las variables dentro del DATA SET (Columna **VARIABLE**)
- El tipo de la variable (**$** o espacio en blanco) $ indica que la variable es de tipo carácter espacio en blanco indica que la variable es de tipo numérico.
- La longitud de la variable (Columna **LENGTH**)
- El formato de cada variable (Columna **FORMAT**)
- El informat de cada variable (Columna **INFORMAT**)
- Un identificador para saber si es o no un campo llave dentro de la tabla (Columna **KEY**)
- Una etiqueta (Columna **LABEL**)

La misma información podrá ser desplegada utilizando el comando VAR (explicado anteriormente Pág. 13).

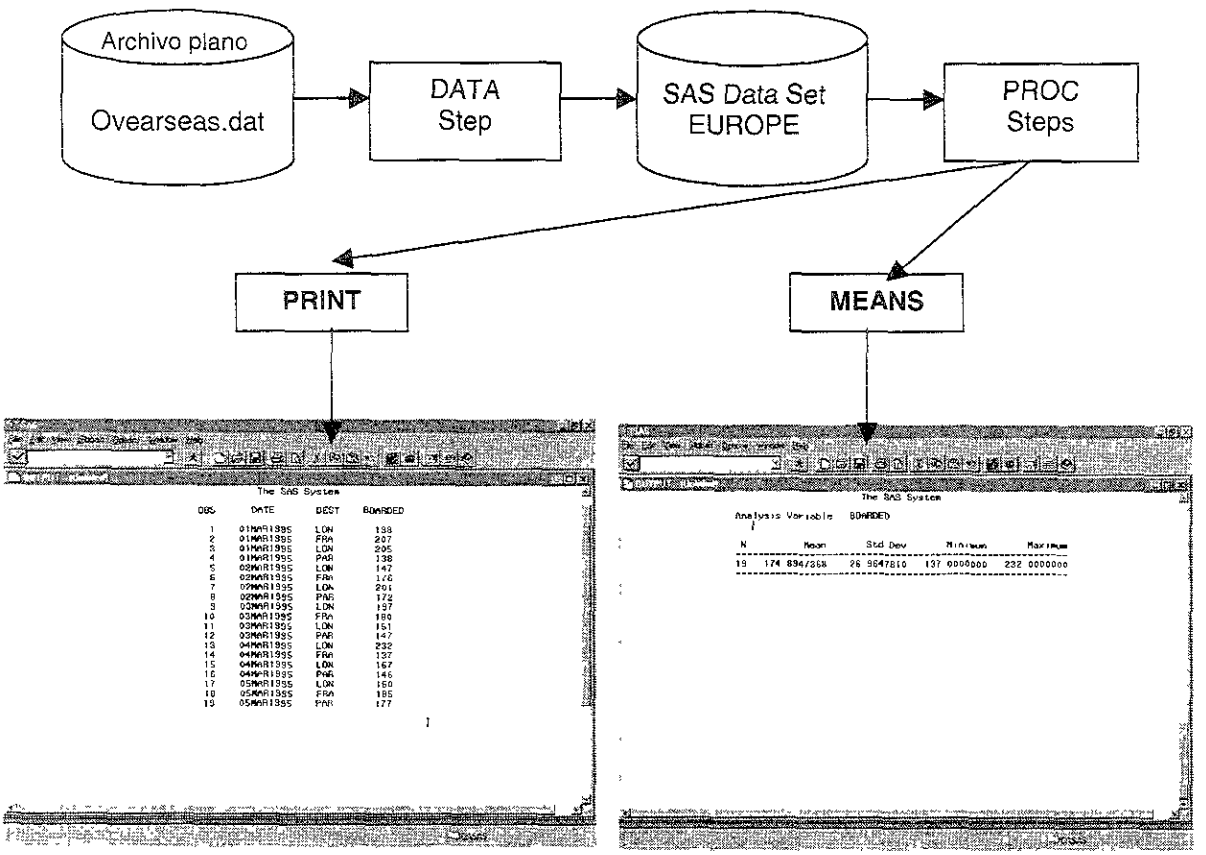
Para conocer la parte de datos del DATA SET, bastara con llamar nuevamente la ventana de las librerías. Seleccionar la librería donde se encuentra ubicado el DATA SET, seleccionar el DATA SET, con el botón derecho del mouse seleccionar la opción OPEN TABLE VIEW del POPMENU o dar un doble click.

## 2.2 UN PROGRAMA SAS

Ejemplo: Los datos de pasajeros que volaran a Europa durante la primera semana de Marzo, son almacenados en un archivo externo. Escribir un programa que genere un DATA SET SAS tomando los datos desde el archivo externo, imprimir el DATA SET y producir un reporte.

Programa:

```
data europe;
  infile 'c:\fund\overseas.dat';
  input @1 date $9. @10 dest $3. @13 boarded 3.;
run;
proc print data=europe;
proc means data=europe;
run;
```




La gráfica muestra la secuencia lógica de un programa donde, el primer paso es la carga de datos y creación de un data set (todo esto a través del paso DATA), y posteriormente la creación de los reportes utilizando los procedimientos adecuados (en el ejemplo PROC PRINT para el reporte tipo lista y PROC MEANS para el reporte estadístico).

Cuando un programa SAS es ejecutado, la salida generada por el Sistema SAS esta dividida en dos partes:

**LOG** Contiene Información acerca de proceso del Programa SAS, incluyendo algún mensaje de WARNING o de ERROR.

**OUTPUT** Contiene reportes generados por los Procedimientos SAS (PROC) y los pasos DATA.

SAS LOG

```
53
54   data europe;
55     infile 'c:\fund\overseas.dat';
56     input @1 date $9. @10 dest $3. @13 boarded 3.;
57   run;

NOTE: The infile 'c:\fund\overseas.dat' is:
      FILENAME=c:\fund\overseas.dat,
      RECFM=V,LRECL=256

NOTE: 19 records were read from the infile 'c:\fund\overseas.dat'.
      The minimum record length was 15.
      The maximum record length was 15.
NOTE: The data set WORK.EUROPE has 19 observations and 3 variables.
NOTE: The DATA statement used 11.97 seconds.


58   proc print data=europe;

NOTE: The PROCEDURE PRINT used 1.37 seconds.


59   proc means data=europe;
60   run;

NOTE: The PROCEDURE MEANS used 0.22 seconds.
```

Cada paso en un programa SAS esta formado de una o más sentencias SAS.

Algunas sentencias SAS pueden ser utilizadas:

- Solo en pasos DATA
- Solo en pasos PROC
- En ambos

Las sentencias SAS:

- Generalmente comienzan con una palabra llave que sirve como identificador.
- Siempre terminan con punto y coma.
- Puedan estar en mayúsculas y/o minúsculas.

```
data europe;
  infile 'c:\fund\overseas.dat';
  input date $ 1-9 dest $ 11-13 boarded 13-15;
run;
proc print data=europe;
proc tabulate data=europe;
    class date dest;
    var boarded;
    table date,dest*boarded*sum;
run;
```

Las sentencias SAS son de formato libre:

- Pueden comenzar y finalizar en cualquier columna.
- Una sentencia puede continuar en varias líneas.
- Varias sentencias pueden ser escritas en la misma línea.
- Los espacios en blanco pueden ser utilizados para separar las palabras.

```
data europe;
  infile 'c:\fund\overseas.dat';
  input date $ 1-9 dest $ 11-13 boarded 13-15;
run;
proc print data=europe;
proc tabulate data=europe;
    class date dest;
    var boarded;
    table date,dest*boarded*sum;
run;
```

☞ UN BUEN ESTILO DE PROGRAMACIÓN ES AQUEL QUE COMIENZA A ESCRIBIR LOS PASOS DATA, PROC Y RUN EN LA PRIMERA COLUMNA E IDENTA LAS DEMÁS SENTENCIAS.

SAS *detecta el comienzo de un paso cuando encuentra:*

- Una sentencia DATA
- Una sentencia PROC
- Una sentencia RUN (para pasos DATA y algunos procedimientos)
- Una sentencia QUIT (para algunos procedimientos)

El comienzo de un nuevo paso, implica el final del paso anterior.

Los comentarios dentro de un programa SAS comienzan con */** y finalizan con ***/** para comentar solo una línea se utiliza * .

☞ DENTRO DE SAS COMO EN CUALQUIER OTRO LENGUAJE DE PROGRAMACION QUE SE UTILICE, ES IMPORTANTE TOMAR EN CUENTA LAS BUENAS PRACTICAS DE PROGRAMACIÓN, ESTO SIGNIFICA ENTENDER Y ANALIZAR COMO FUNCIONA INTERNAMENTE, BUSCANDO QUE LOS ALGORITMOS IMPLEMENTADOS, SEAN LO MÁS EFICIENTES POSIBLE, ADEMÁS NO OLVIDAR LA DOCUMENTACIÓN DE ESTOS.

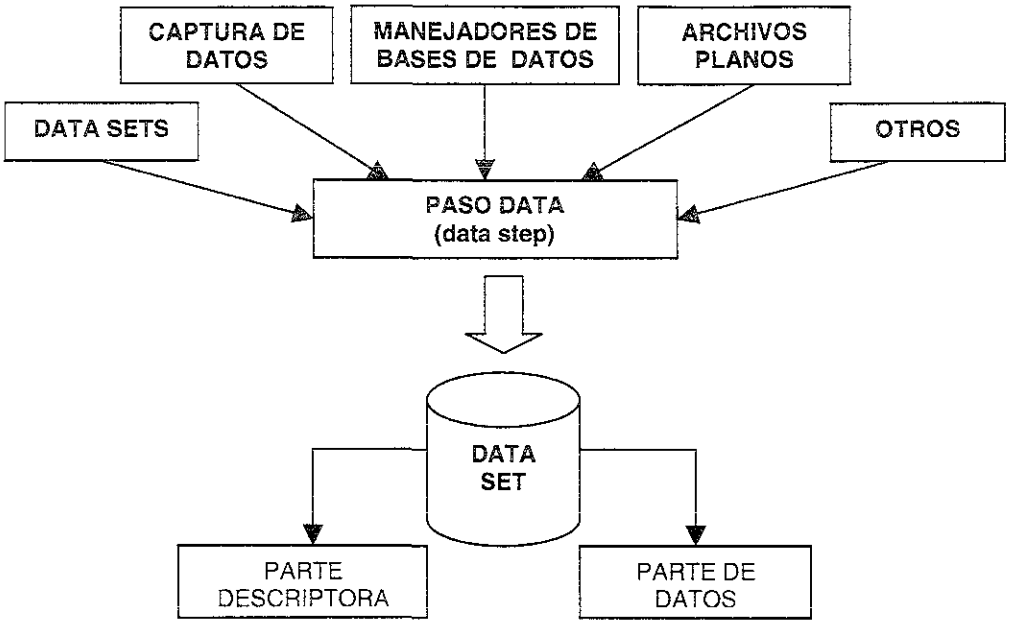
# CAPITULO 3

# MANEJO DE ARCHIVOS PLANOS

Aprender a utilizar sentencias del lenguaje BASE SAS para la lectura de archivos planos y la generación de data sets.

## 3.1 SENTENCIAS PARA LA LECTURA DE ARCHIVOS PLANOS

SAS es una herramienta que permite el acceso a distintas fuentes de datos. algunas de estas se muestran en el siguiente diagrama:




Como se observa, un DATA SET puede ser creado accediendo datos de fuentes tales como:

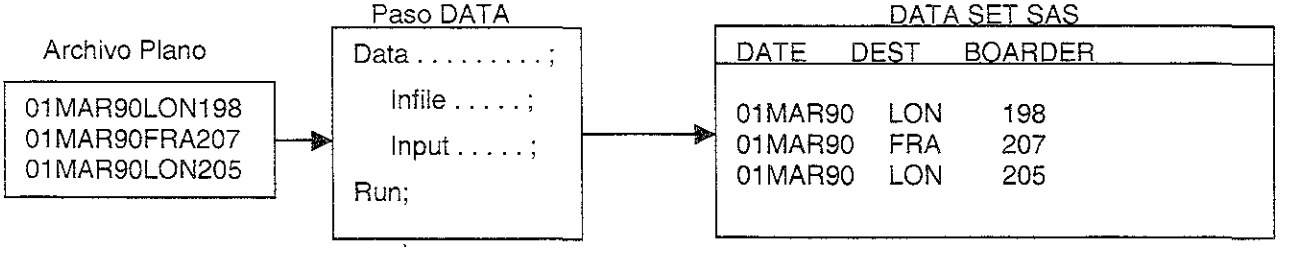
- Captura de datos
- Sistemas Manejadores de Bases de Datos (ACCESS, ORACLE, DBASE, SYBASE, INFORMIX)
- Archivos planos (TXT, DAT, ASCII, etc.)
- DATA SETS SAS
- Otros (EXCEL, ERP's)

Este trabajo se enfoca a los casos de acceso a archivos planos y DATA SETS SAS.

☞ ES IMPORTANTE SABER QUE NO TODAS LAS LECTURAS SE REALIZAN DE LA MISMA MANERA Y TAMBIÉN QUE UN PASO DATA, RECIBE UN ARCHIVO DE ENTRADA Y PUEDE GENERAR NINGÚN ARCHIVO O HASTA 255 ARCHIVOS DE SALIDA, EXCEPTO EN EL CASO DE LA LECTURA DE DATA SETS, DONDE SE PUEDEN ACCEDER HASTA 255 DATA SETS DE ENTRADA GENERANDO 0 Y HASTA 255 ARCHIVOS DE SALIDA

Un archivo plano no cuenta con una estructura de tabla definida, por lo cual al momento de realizar la lectura de un archivo de este tipo, es preciso definir la estructura que tendrá el DATA SET de salida (esto es definir la parte descriptora del DATA SET que se esta creando).

☞ SIEMPRE QUE EN SAS SE REQUIERA GENERAR MAS DE UN DATA SET DE SALIDA, TOMANDO COMO ENTRADA EL MISMO DATA SET EN AMBOS CASOS, SE RECOMIENDA REALIZAR LA ESCRITURA DE LOS DOS O MÁS DATA SETS DE SALIDA, EN UN MISMO PASO DATA CON ESTO SE EVITARA LA LECTURA REPETITIVA DE TABLAS, EL BENEFICIO SERÁ EL RESULTADO EN EL TIEMPO DE PROCESO



La lectura del archivo plano se realiza utilizando un paso data como se muestra en la gráfica, el resultado es un data set SAS.

Un paso DATA comienza con la sentencia DATA, de esta sentencia las dos mayores funciones son:

- Señalar el comienzo de un paso DATA.
- Nombrar el DATA SET que será creado.

Un paso DATA comienza con la palabra reservada DATA, seguida del nombre o nombres de el o los DATA SETS que serán creados en ese paso, posteriormente las siguientes sentencias de lectura

**INFILE** Identifica el nombre del archivo externo que será cargado, debe ser declarado antes de ejecutar la sentencia INPUT.

Sintaxis:

**INFILE** ' *Nombre del archivo* ';

**INPUT** Define las variables que tendrá el DATA SET de salida y el orden en el cual estas serán leídas desde el archivo de entrada especificado en la sentencia INFILE.

Sintaxis:

**INPUT** *control del cursor nombre de la variable $nombre del informatw.d ;*

*Control del Cursor:*

*@n* Ir a la columna *n*
*+n* Mover el cursor *n* posiciones

*$nombre del informatw.d*

| | |
|---|---|
| *$* | Solo para informat de tipo carácter |
| *nombre del informat* | Indica el nombre del informat utilizado |
| *w* | Es una opción para indicar el ancho del campo |
| . | Es un delimitador requerido |
| *d* | Es una opción que especifica el # de decimales en el informat. |

☞ UN INFORMAT ESPECIFICA EL ANCHO DEL CAMPO DE ENTRADA Y COMO LEER LOS DATOS QUE SE ALMACENARÁN EN EL CAMPO.

Ejemplo: Escribir una sentencia INPUT que lea los siguientes campos:

| CAMPO | COLUMNAS | COMENTARIOS |
|---|---|---|
| Last Name | 1-15 | Carácter |
| First Name | 21-35 | Carácter |
| Gender | 40 | M o F |
| Salary | 46-51 | Numérico |

**INPUT @1 lname $15. @21 fname $15. @40 gender $1. @46 salary 6.;**

La lectura con informats es apropiada para leer:

- Datos numéricos y Carácter estándar y no-estándar.
- Valores de fechas para convertirlas a fechas SAS.

Ejemplo de algunos *informats*

7. o 7.0 Lee siete columnas de datos numéricos
7.2 Lee siete columnas de datos numéricos e inserta un punto decimal tomando dos decimales.
$6. Lee 6 columnas con datos de tipo carácter, removiendo espacios en blanco.
SCHAR6. Lee 6 columnas con datos de tipo carácter, respetando espacios en blanco.
COMMA6. Lee 6 columnas de datos de tipo numérico, omitiendo los caracteres como signos de pesos, comas, puntos.

**NOTA**: Todos los informats deben llevar un punto al final como parte del nombre, este punto es utilizado como el delimitador del informat.

☞ DENTRO DE SAS ES RECOMENDABLE LEER Y ALMACENAR LOS VALORES DE FECHAS COMO VALORES NUMÉRICOS

Finalmente el paso DATA es terminado utilizando la sentencia RUN.

Todas y cada una de las sentencias dentro de un paso DATA deben finalizar con ; como se muestra a continuación:

Ejemplo: Escriba un paso DATA que lea un archivo con la siguiente estructura:

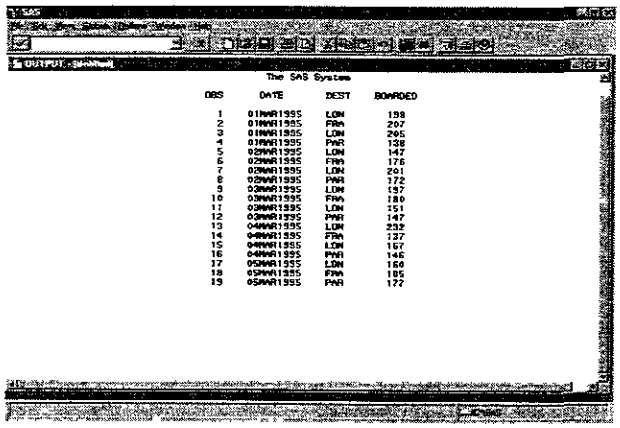
```
                1       1       2       2
1---5----0----5----0----5----0----5
01MAR1995LON198
01MAR1995FRA207
01MAR1995LON205
```

Programa:

```
data europe;
  infile 'c:\fund\overseas.dat';
  input date $ 1-9 dest $ 10-12 boarded 13-15;
proc print data=europe;
run;
```

Resultado :

Al ejecutar el programa utilizando  o el comando SUBMIT la salida en la ventana del OUTPUT es la siguiente:

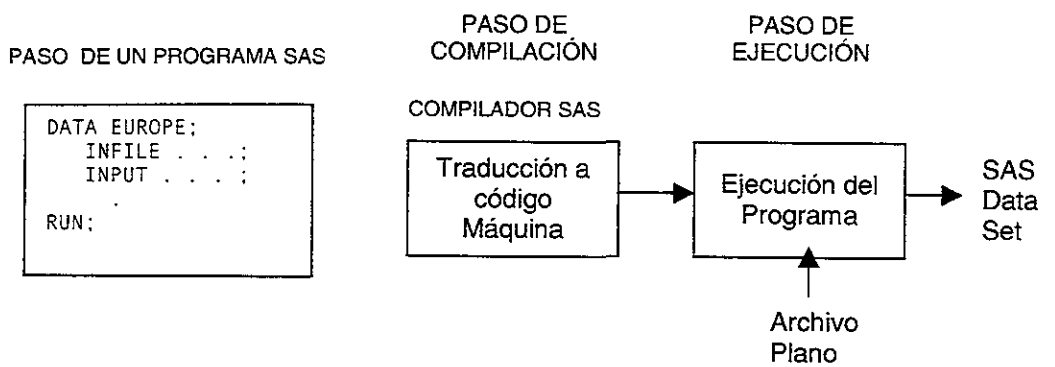


The SAS System

| OBS | DATE | DEST | BOARDED |
|---|---|---|---|
| 1 | 01MAR1995 | LON | 198 |
| 2 | 01MAR1995 | PAR | 207 |
| 3 | 01MAR1995 | LON | 205 |
| 4 | 01MAR1995 | PAR | 138 |
| 5 | 02MAR1995 | LON | 147 |
| 6 | 02MAR1995 | PAR | 176 |
| 7 | 02MAR1995 | LON | 201 |
| 8 | 02MAR1995 | PAR | 172 |
| 9 | 03MAR1995 | LON | 197 |
| 10 | 03MAR1995 | PAR | 180 |
| 11 | 03MAR1995 | LON | 151 |
| 12 | 03MAR1995 | PAR | 147 |
| 13 | 04MAR1995 | LON | 232 |
| 14 | 04MAR1995 | PAR | 137 |
| 15 | 04MAR1995 | LON | 167 |
| 16 | 04MAR1995 | PAR | 146 |
| 17 | 05MAR1995 | LON | 160 |
| 18 | 05MAR1995 | PAR | 185 |
| 19 | 05MAR1995 | PAR | 177 |

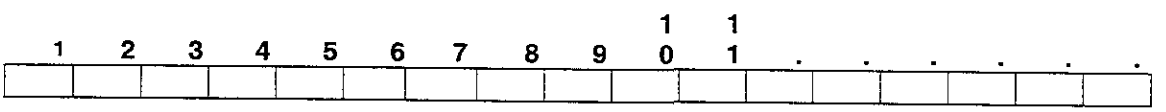
☞ SI AL INICIAR LA LECTURA, NO SE ESPECIFICA LA POSICIÓN DONDE SE DESEA COMENZAR LA LECTURA COMENZARÁ SIEMPRE EN LA COLUMNA 1. CUANDO SE OMITE UNA ESPECIFICACIÓN DEL CURSOR (*@N O + N* ) LA LECTURA DE LA SIGUIENTE VARIABLE, COMENZARA A PARTIR DE LA SIGUIENTE COLUMNA DE DONDE SE REALIZO LA ULTIMA LECTURA.

Cuando un programa SAS es ejecutado, internamente el Sistema SAS, compila el programa y después lo ejecuta.




Al momento de compilar el Sistema SAS,

- Crea un buffer para almacenar el registro que esta leyendo desde el archivo externo.




- El *PROGRAM DATA VECTOR* es creado para almacenar la observación actual.

| **NAME:** | **DATE** | **DEST** | **BOARDED** |
|---|---|---|---|
| **TYPE:** | **$** | **$** | **N** |
| **LENGTH:** | **9** | **3** | **8** |
| **VALUE:** | | | |

- La parte descriptora del DATA SET es creada.

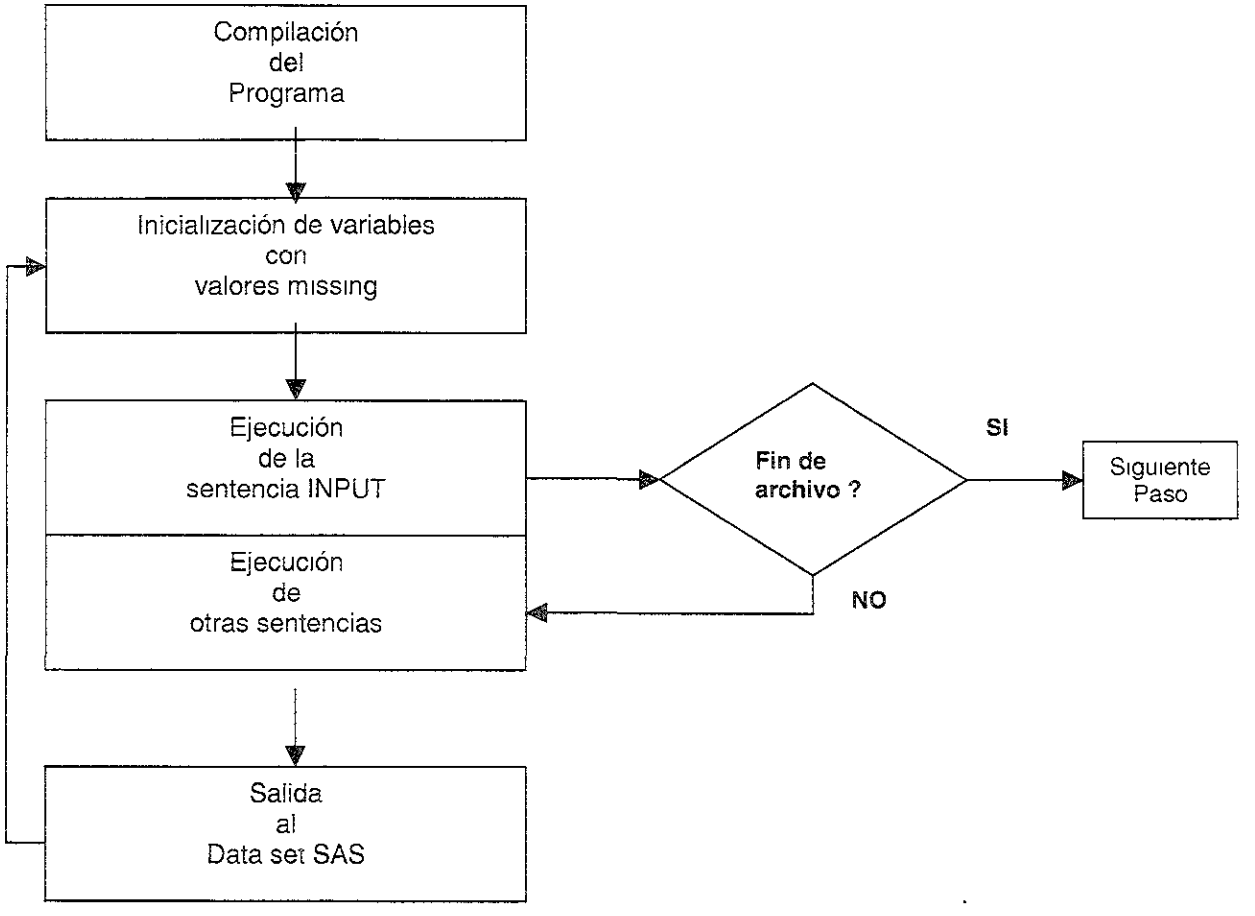
EUROPE

| | DATE<br>$<br>9 | DEST<br>$<br>3 | BOARDED<br>N<br>8 |
|---|---|---|---|
| **WORK DISK SPACE** | | | |

Al momento de ejecutar el programa:

- Se inicializan las variables del PDV con valores missing antes de cada ejecución del paso DATA.
- Cada sentencia es ejecutada secuencialmente.
- La sentencia INPUT lee el siguiente registro desde el archivo externo identificado por la sentencia INFILE y lo vacía dentro del PDV.
- Otra sentencia puede modificar los valores de la observación actual.
- Los valores que se encuentran dentro del PDV son escritos al DATA SET al final del paso DATA.
- El flujo del programa es regresado al comienzo del paso.
- El paso es ejecutado hasta que es encontrado el final del archivo que esta leyendo.

Ciclo de ejecución de un programa SAS



**LECTURA DE DATOS CON INFORMATS**

Como se puede observar en la salida del programa anterior, la fecha se almacena como un dato de tipo carácter. Para verificarlo se puede consultar la parte descriptora del DATA SET creado en el programa, ésto se puede realizar utilizando la sentencia VAR.

En este caso, la variable DATE fue almacenada como una variable de tipo carácter ya que en la sentencia INPUT utilizada para leerla se especifico que la lectura se hiciera para una variable de tipo carácter. Sin embargo como ya se menciono, las fechas dentro del Sistema SAS deben ser almacenadas como datos de tipo numérico, con la finalidad de realizar posteriormente algún tipo de operación.

Para leer este tipo de valores, será necesario utilizar *informats* como los que a continuación se describen.

*MMDDYYw.* Lee fechas de la forma 101695 (MMDDYY6.) ó 10/16/95 (MMDDYY8.) ó 10-16-1995 (MMDDYY10.)

*DDMMYYw.* Lee fechas con la forma 161095 (DDMMYY6.) ó 16/10/95 (DDMMYY8.) ó 16-10-1995 (DDMMYY10.)

*YYMMDDw.* Lee fechas con la forma 951016 (YYMMDD6.) ó 95/10/16 (YYMMDD8.) ó 1995-10-16 (YYMMDD10.)

*DATEw.* Lee fechas con la forma 16OCT95 (DATE7.) ó 16OCT1995 (DATE9.).

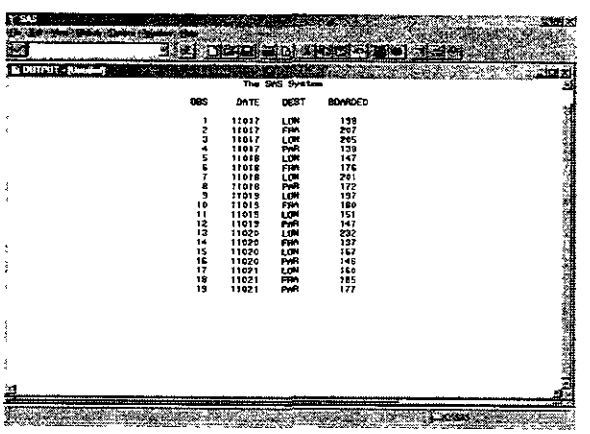
El objetivo de utilizar los informats es interpretar los valores que se están leyendo. En un dato de tipo fecha el informat que se utilice depende del formato en el que este almacenado el valor de la fecha. Internamente SAS almacenara este valor como el número de días transcurridos a partir del 01JAN60 hasta la fecha que se esta almacenando.

Dentro de un paso DATA se utiliza de la siguiente manera

Programa:

```
data europe;
  infile 'c:\fund\overseas.dat';
  input date date9. dest $ 10-12 boarded 13-15;
proc print data=europe;
run;
```

Resultado :



Como se observa en la siguiente salida el valor de la variable DATE representa una fecha en número de días sin formato. En el siguiente capitulo se muestra como dar formato a los datos en las salidas.

### ERRORES EN LOS DATOS

En algunas ocasiones cuando se trata de validar información, sucede que algunos de los datos contenidos en los archivos que se leen no son datos validos, los errores más comunes son encontrar algún tipo de carácter cuando se lee una variable de tipo numérico ó encontrar que la fecha que se esta leyendo no tiene el formato indicado. Cuando este tipo de errores suceden mientras se esta cargando información el proceso que se esta ejecutando no termina, ni se queda a la mitad, incluso realiza las lecturas de todos y cada uno de los registros incluidos en el archivo que se esta cargando, lo que sucede como se verá a continuación, es que aquellos datos de los registros donde se detectaron errores, son almacenados con valores nulos. en el caso de datos numéricos este valor es representado por un ( . ) punto y en el caso de los datos tipo carácter será representado por un espacio en blanco.

Ejemplo: Escriba un paso DATA que lea los datos almacenados en un archivo llamado MARFLT.DAT que contiene los siguientes datos:

| DESCRIPCIÓN DEL CAMPO | COLUMNAS | TIPO DE DATO |
|---|---|---|
| Número de Vuelo | 1 – 3 | Carácter |
| Fecha | 4 – 9 | Fecha con formato (mmddyy) |
| Destino | 18 – 20 | Carácter |
| Pasajeros que abordaron | 34 – 36 | Numérico |
| Pasajeros Transferidos | 37 – 39 | Numérico |
| Pasajeros que NO pagaron | 40 – 42 | Numérico |

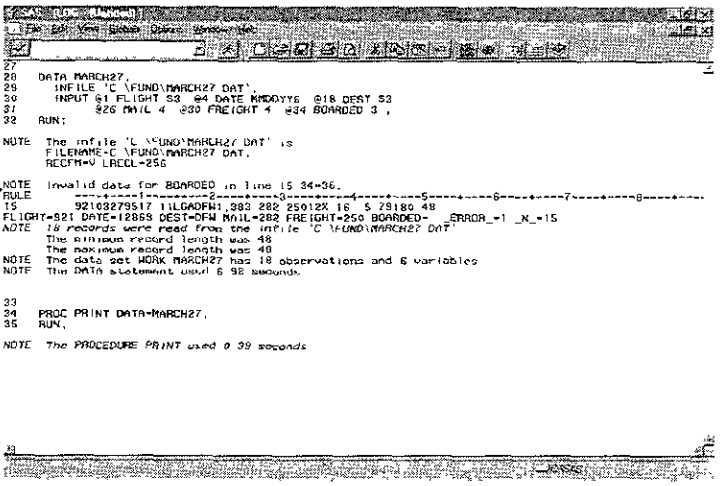
Ejecutar el programa y examinar los mensajes en la ventana del LOG, posteriormente utilizar un PROC PRINT para imprimir los datos que se almacenaron en el DATA SET.

Programa:

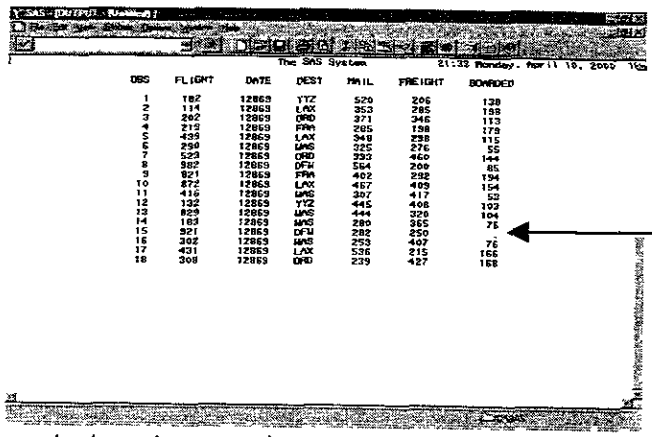
```
data march;
  infile 'c:\fund\marflt.dat';
  input  flight $1-3 date mmddyy6. dest $18-20
         mail 26-29 freight 30-33 boarded 34-36;
run;
```

Resultado: Al ejecutar el programa observamos la ventana del LOG

Como se puede ver el la ventana del LOG, se encontró un dato invalido en la línea 16 columnas 34-36 y este dato corresponde a la variable *BOARDED* de tipo numérico.




Finalmente al ver la salida en la ventana del OUTPUT



| OBS | FLIGHT | DATE | DEST | MAIL | FREIGHT | BOARDED |
|---|---|---|---|---|---|---|
| 1 | 182 | 12869 | YYZ | 520 | 206 | 138 |
| 2 | 114 | 12869 | LAX | 353 | 285 | 198 |
| 3 | 262 | 12869 | ORD | 371 | 346 | 113 |
| 4 | 219 | 12869 | PHX | 285 | 198 | 178 |
| 5 | 439 | 12869 | LAX | 348 | 298 | 115 |
| 6 | 298 | 12869 | MKE | 335 | 276 | 55 |
| 7 | 523 | 12869 | ORD | 292 | 460 | 144 |
| 8 | 382 | 12869 | DFW | 564 | 209 | 85 |
| 9 | 921 | 12869 | PHX | 402 | 292 | 194 |
| 10 | 872 | 12869 | LAX | 467 | 403 | 154 |
| 11 | 416 | 12869 | MKE | 307 | 417 | 52 |
| 12 | 132 | 12869 | YYZ | 445 | 406 | 102 |
| 13 | 829 | 12869 | MKE | 444 | 320 | 104 |
| 14 | 183 | 12869 | MKE | 280 | 365 | 75 |
| 15 | 921 | 12869 | DFW | 282 | 250 | . |
| 16 | 302 | 12869 | MKE | 253 | 407 | 76 |
| 17 | 431 | 12869 | LAX | 536 | 215 | 166 |
| 18 | 308 | 12869 | ORD | 229 | 427 | 168 |

Encontramos el dato faltante representado por un . debido a que la variable es de tipo numérico

Es importante notar que el proceso concluyo exitosamente y todas las variables contenidas en el registro dentro del cual se encontró el error fueron leídas de forma correcta.

En la mayoría de los casos la corrección de este tipo de errores en los datos no corresponde a la persona que esta realizando la extracción si no a quien genero el archivo plano. Sin embargo cualquiera que sea la situación se debe contar con las reglas que permitan saber que tipo de tratamiento se dará cuando se encuentren este tipo de errores. Finalmente esto es lo que se conoce como validación de la información.

☞ AL GENERAR VARIABLES DENTRO DE UN DATA SET, VALE LA PENA TOMAR UN POCO DE TIEMPO Y DECIDIR EL TIPO QUE ÉSTA TENDRÁ, DEPENDIENDO DE LOS VALORES QUE SERÁN ALMACENADOS.

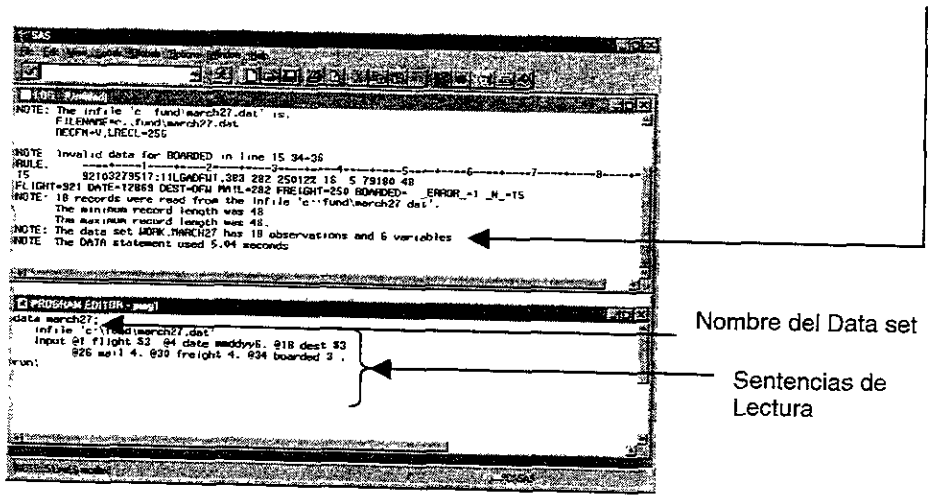
## 3.2 CREACIÓN DE DATA SETS PERMANENTES

Los data sets pueden ser almacenados de forma temporal y de forma permanente.

Los data sets se almacenan de forma permanente omitiendo el nombre de la librería al asignar el nombre del data set que se esta creando, o utilizando la librería WORK.

Ejemplo: Los datos de los vuelos de Marzo, son almacenados en un archivo plano. Crea un data set temporal llamado MARCH27 que tome los datos del archivo plano. Imprime el data set resultante.

Al ejecutar el programa el resultado en la ventana del LOG indica donde fue creado el data set.



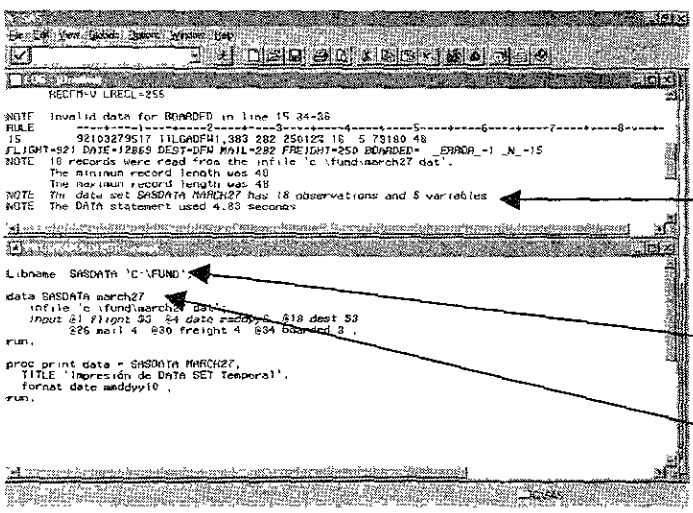

Al momento de imprimir este data set, se debe llamar desde la localidad donde se encuentra almacenado (en el ejemplo WORK.MARCH27) como se indica a continuación:

Programa:

```
Proc print data=WORK.MARCH27;
Run;
```

Un data set puede ser almacenado de forma permanente utilizando otra librería distinta de WORK. La librería que sea seleccionada para almacenar los datos, debe ser asignada apropiadamente a una localidad física antes de ser referenciada.

Ejemplo: Alterar el programa anterior, para almacenar los datos del DATA SET MARCH27 de forma permanente en una librería llamada SASDATA. Para tal efecto se incluirá en el programa la sentencia LIBNAME que asignara la librería a una localidad.




El data set es creado en la localidad indicada

Para acceder data set almacenados de forma permanente debe,

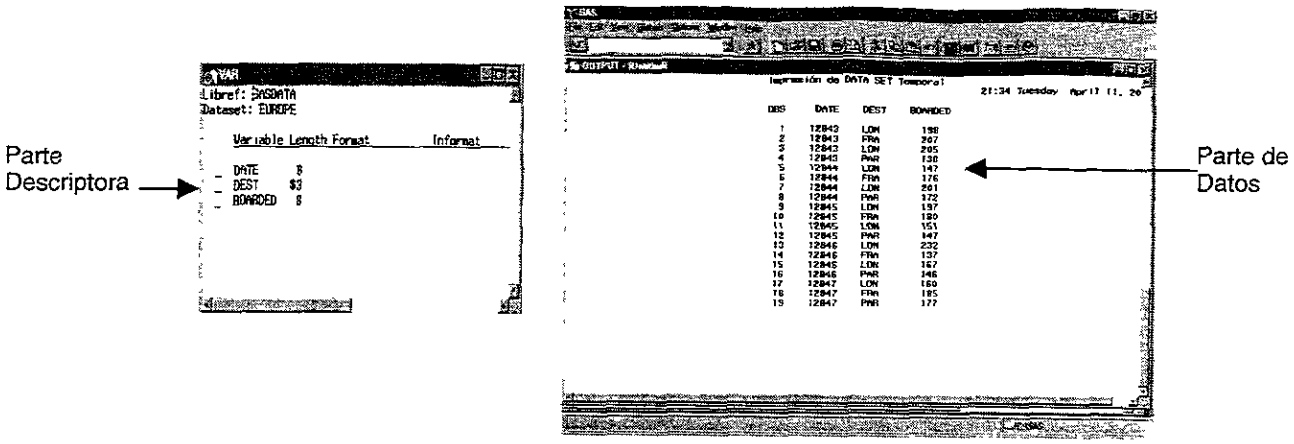
- Ejecutarse una sentencia LIBNAME para asignar la librería la localidad donde se desea almacenar los datos.
- Preceder el nombre del data set con el nombre apropiado de la librería.

☞ *ES RECOMENDABLE ALMACENAR DATA SETS DE FORMA PERMANENTE, SOLO EN AQUELLOS CASOS EN QUE SEA ESTRICTAMENTE NECESARIO, ESTO REDUCIRA TIEMPOS DE ESCRITURA Y AHORRO DE ESPACIO*

**ASIGNACIÓN DE ATRIBUTOS A LAS VARIABLES**

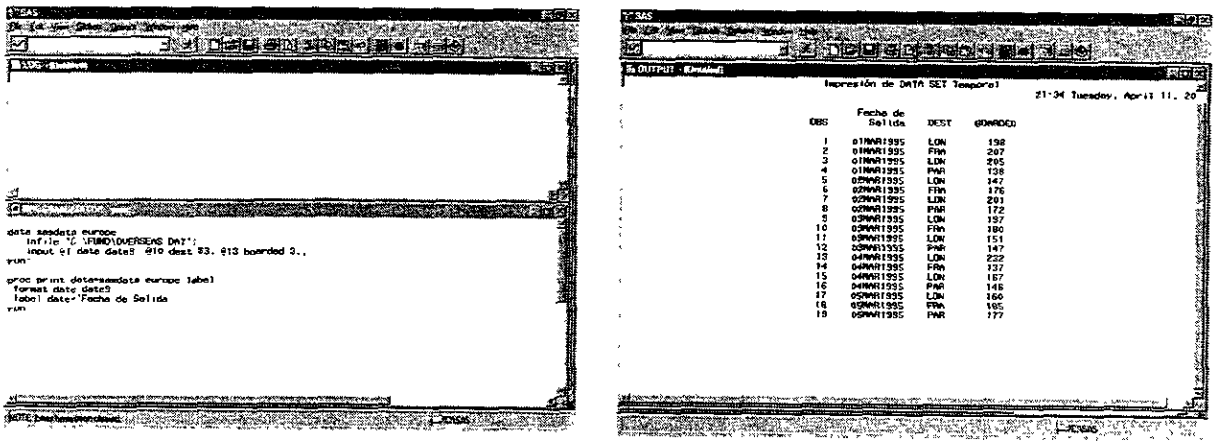
Cuando una variable es creada en un paso DATA el nombre, tipo y longitud de esta son automáticamente asignados. Sin embargo atributos como el formato y la etiqueta no son asignados.

Cuando la variable es utilizada en un paso posterior su nombre, es desplegado con el propósito de identificar esta y sus valores son mostrados utilizando el formato asignado por el sistema.




Las sentencias LABEL y FORMAT pueden ser utilizadas en un paso siguiente para desplegar una etiqueta descriptiva en lugar del nombre de la variable y para desplegar los valores en un formato más significativo.

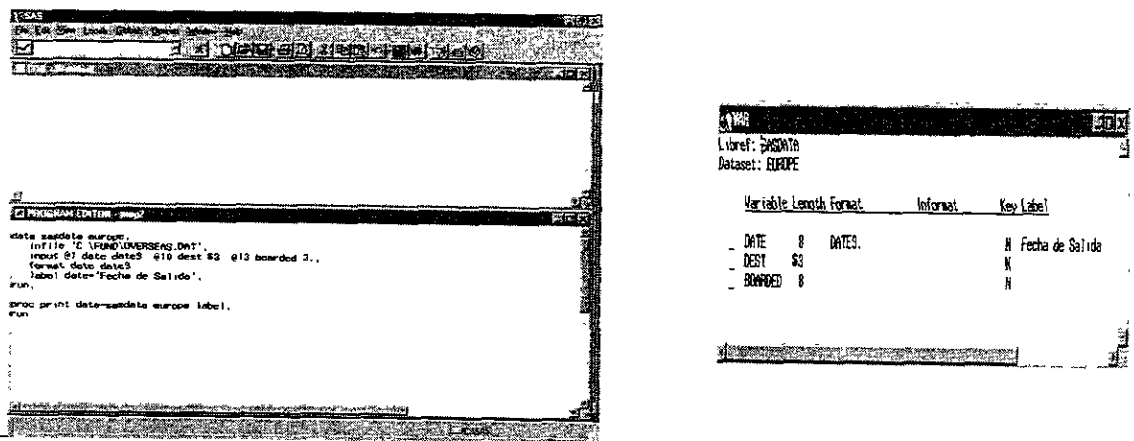
**ES RECOMENDABLE ESPECIFICAR LAS CARACTERISTICAS COMPLETAS DEL DATA SET CUANDO ESTE ES CREADO, DE ESTA FORMA, CADA VEZ QUE ÉSTE SEA UTILIZADO, LAS CARACTERSITICAS DE LAS VARIABLES QUE LO FORMAN ESTARAN DISPONIBLES.**



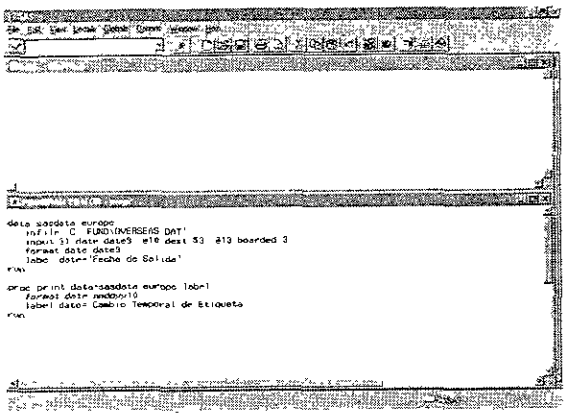
Estos atributos solo tienen efecto durante el paso PROC en el cual son utilizados, la parte descriptora del data set y los atributos por default de las variables no son alterados.

Sin embargo las sentencias LABEL y FORMAT pueden también ser incluidas en el paso DATA con el objetivo de almacenar esta información en la parte descriptora del DATA SET.

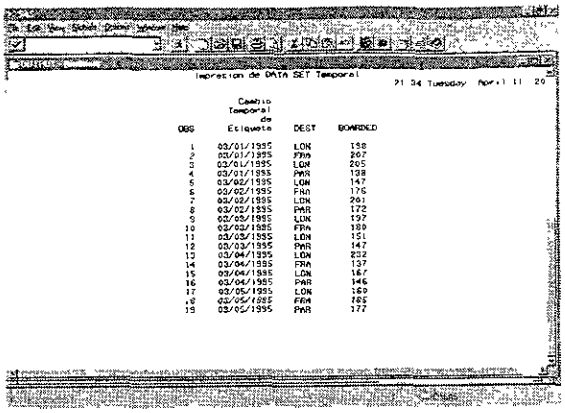
Estos atributos estarán automáticamente disponibles en pasos PROC subsecuentes.



Cuando las sentencias LABEL y FORMAT son utilizadas en pasos PROC estos sustituyen (solamente durante este paso) a los atributos definidos en la parte descriptora del DATA SET utilizado.



```
data salida europa;
   infile 'C:\FUNDAMENTOS\VUELOS.DAT'
   input @1 fecha date9. @10 dest $3. @13 boarded 3.
   format fecha date9.
   label fecha='Fecha de Salida';
run;

proc print data=salida europa label;
   format fecha ddmmyy10.
   label fecha='Cambio Temporal de Etiqueta';
run;
```



Impresion de DATA SET Temporal

21:34 Tuesday, April 11, 20

| OBS | Cambio Temporal de Etiqueta | DEST | BOARDED |
|---|---|---|---|
| 1 | 03/01/1995 | LON | 158 |
| 2 | 03/01/1995 | FRA | 207 |
| 3 | 03/01/1995 | LON | 205 |
| 4 | 03/01/1995 | PAR | 138 |
| 5 | 03/02/1995 | LON | 147 |
| 6 | 03/02/1995 | FRA | 176 |
| 7 | 03/02/1995 | LON | 261 |
| 8 | 03/02/1995 | PAR | 172 |
| 9 | 03/02/1995 | LON | 197 |
| 10 | 03/02/1995 | FRA | 180 |
| 11 | 03/03/1995 | LON | 151 |
| 12 | 03/03/1995 | PAR | 147 |
| 13 | 03/04/1995 | LON | 232 |
| 14 | 03/04/1995 | FRA | 137 |
| 15 | 03/04/1995 | LON | 167 |
| 16 | 03/04/1995 | PAR | 146 |
| 17 | 03/05/1995 | LON | 160 |
| 18 | 03/05/1995 | FRA | 205 |
| 19 | 03/05/1995 | PAR | 177 |

## 3.3 CREACIÓN DE VARIABLES Y SELECCIÓN DE OBSERVACIONES

Se pueden utilizar sentencias de asignación para:

- Crear nuevas variables
- Modificar valores de variables existentes

Forma general de una sentencia de asignación:

*Variable=expression;*

Cuando una sentencia de este tipo es encontrada, el Sistema SAS evalúa la expresión y asigna el valor resultante a la variable.

Algunos de los operadores que están disponibles para cálculos aritméticos son:

| OPERADOR | ACCION | EJEMPLO | PRIORIDAD |
|---|---|---|---|
| + | SUMA | SUM = X + Y; | III |
| - | RESTA | DIFF = X – Y; | III |
| * | MULTIPLICACIÓN | MULTI = X * Y; | II |
| / | DIVISIÓN | DIVIDE = X / Y; | II |
| ** | EXPONENCIAL | RAIZ = X ** Y; | I |
| - | NÚMEROS NEGATIVOS | NEGATIVO = - X; | I |

Reglas Adicionales:

- Operaciones de prioridad I son ejecutadas antes de las operaciones de prioridad II y estas antes de las de prioridad III.
- Operaciones consecutivas con la misma prioridad son ejecutadas:
  - De derecha a Izquierda las de prioridad I
  - De Izquierda a Derecha las de prioridad II y III
- Los paréntesis pueden ser utilizados para controlar el orden de las operaciones.

Ejemplo: Datos de vuelos de Dallas a Los Angeles son almacenados en un archivo plano con la estructura que se muestra en la siguiente tabla. Utilizar un paso DATA para crear un DATA SET llamado DFWLAX que tome los datos de el archivo plano y genere una nueva variable llamada TOTAL que muestre la suma del total de pasajeros en cada vuelo.

| DESCRIPCIÓN DEL CAMPO | COLUMNAS | TIPO DE DATO |
|---|---|---|
| Número de Vuelo | 1 – 3 | Carácter |
| Fecha | 4 – 9 | Fecha con formato (mmddyy) |
| Destino | 18 – 20 | Carácter |
| Pasajeros que abordaron | 34 – 36 | Numérico |
| Pasajeros Transferidos | 37 – 39 | Numérico |
| Pasajeros que NO pagaron | 40 – 42 | Numérico |

Programa:

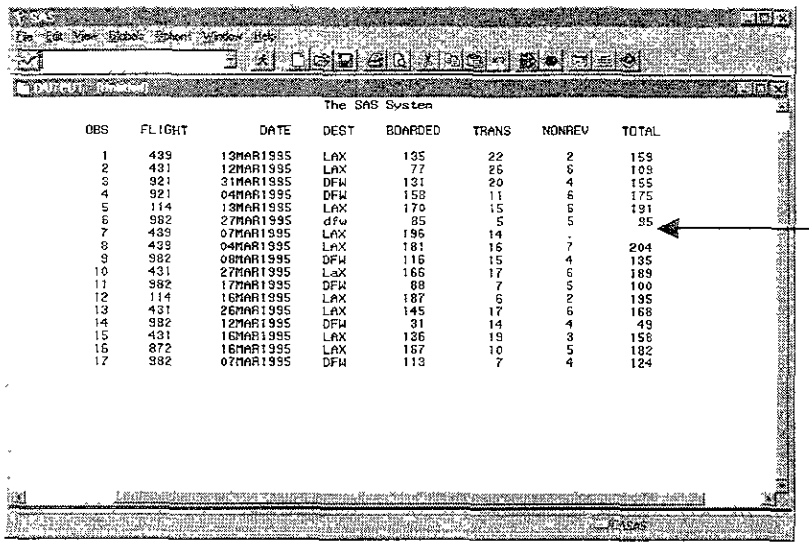
```
data dfwlax;
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=boarded+trans+nonrev;
run;

proc print data=dfwlax;
  format date date9.;
run;
```

Al momento de compilar el programa anterior, las siguientes variables son creadas dentro del PDV

**P D V**

| FLIGHT | DATE | DEST | BOARDED | TRANS | NONREV | TOTAL |
|---|---|---|---|---|---|---|
| | | | | | | |

Resultado



The SAS System

| OBS | FLIGHT | DATE | DEST | BOARDED | TRANS | NONREV | TOTAL |
|---|---|---|---|---|---|---|---|
| 1 | 439 | 13MAR1995 | LAX | 135 | 22 | 2 | 159 |
| 2 | 431 | 12MAR1995 | LAX | 77 | 26 | 6 | 109 |
| 3 | 921 | 31MAR1995 | DFW | 131 | 20 | 4 | 155 |
| 4 | 921 | 04MAR1995 | DFW | 158 | 11 | 6 | 175 |
| 5 | 114 | 13MAR1995 | LAX | 170 | 15 | 6 | 191 |
| 6 | 982 | 27MAR1995 | dfw | 85 | 5 | 5 | 95 |
| 7 | 439 | 07MAR1995 | LAX | 195 | 14 | . | |
| 8 | 439 | 04MAR1995 | LAX | 181 | 16 | 7 | 204 |
| 9 | 982 | 08MAR1995 | DFW | 116 | 15 | 4 | 135 |
| 10 | 431 | 27MAR1995 | LAX | 166 | 17 | 6 | 189 |
| 11 | 982 | 17MAR1995 | DFW | 88 | 7 | 5 | 100 |
| 12 | 114 | 16MAR1995 | LAX | 187 | 6 | 2 | 195 |
| 13 | 431 | 26MAR1995 | LAX | 145 | 17 | 6 | 168 |
| 14 | 982 | 12MAR1995 | DFW | 31 | 14 | 4 | 49 |
| 15 | 431 | 16MAR1995 | LAX | 136 | 19 | 3 | 158 |
| 16 | 872 | 16MAR1995 | LAX | 167 | 10 | 5 | 182 |
| 17 | 982 | 07MAR1995 | DFW | 113 | 7 | 4 | 124 |

Como se puede observar en esta impresión, existen algunos valores nulos de la variable TOTAL, esto es debido a que alguno de los valores utilizados con operadores aritméticos es un valor nulo. Este problema será resuelto con el uso de funciones como se mostrará en el capitulo 5.

- CUANDO SE REQUIERE AGREGAR ALGÚN CALCULO ARITMETICO DENTRO DE UN PROGRAMA, ES RECOMENDABLE UTILIZAR FUNCIONES, LAS CUALES PERMITEN EVITAR QUE LOS VALORES MISSING AFECTEN EL RESULTADO.

### ASIGNACIÓN CONDICIONAL DE VALORES

Dentro de SAS es posible utilizar las sentencias IF-THEN y ELSE para asignar valores a las variables que dependan de las condiciones especificadas.

La forma general de utilizar esta sentencia es:

**IF** *expression* **THEN** *statement*;

**ELSE** *statement*;

*Expression* Es cualquier expresión SAS valida.
*Statement* Es cualquier sentencia SAS ejecutable.

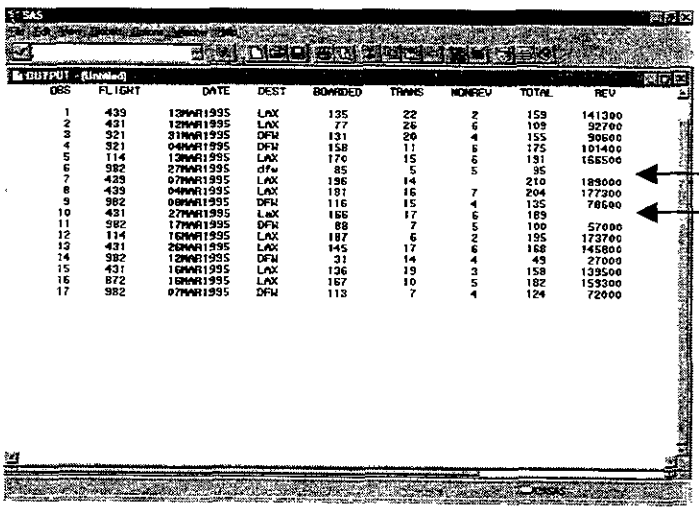
**NOTA**:La sentencia **ELSE** es opcional.

Ejemplo: Modificar el paso DATA anterior para crear otra variable llamada REV que sea el monto obtenido del total de pasajeros BOARDED y TRANS. Asumir que el costo del vuelo a Los Angeles es de $900 y el del vuelo a Dallas es de $600.

Programa:

```
data dfwlax;
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev);
   if dest='DFW' then rev=SUM(boarded,trans)*600;
   else if dest='LAX' then rev=SUM(boarded,trans)*900;
run;

proc print data=dfwlax;
  format date date9.;
run;
```

Resultado:

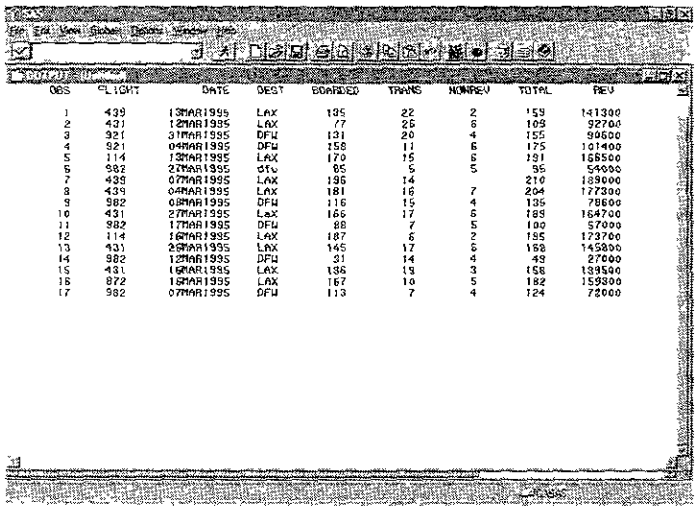


| OBS | FLIGHT | DATE | DEST | BOARDED | TRANS | NONREV | TOTAL | REV |
|---|---|---|---|---|---|---|---|---|
| 1 | 439 | 13MAR1995 | LAX | 135 | 22 | 2 | 159 | 141300 |
| 2 | 431 | 12MAR1995 | LAX | 77 | 26 | 6 | 109 | 92700 |
| 3 | 321 | 31MAR1995 | DFW | 131 | 20 | 4 | 155 | 90600 |
| 4 | 321 | 04MAR1995 | DFW | 158 | 11 | 6 | 175 | 101400 |
| 5 | 114 | 13MAR1995 | LAX | 170 | 15 | 6 | 191 | 166500 |
| 6 | 382 | 27MAR1995 | dfw | 85 | 5 | 5 | 95 | . |
| 7 | 439 | 07MAR1995 | LAX | 196 | 14 | . | 210 | 189000 |
| 8 | 439 | 04MAR1995 | LAX | 181 | 16 | 7 | 204 | 177300 |
| 9 | 982 | 08MAR1995 | DFW | 116 | 15 | 4 | 135 | 78600 |
| 10 | 431 | 27MAR1995 | LaX | 166 | 17 | 6 | 189 | . |
| 11 | 382 | 17MAR1995 | DFW | 88 | 7 | 5 | 100 | 57000 |
| 12 | 114 | 16MAR1995 | LAX | 187 | 6 | 2 | 195 | 173700 |
| 13 | 431 | 26MAR1995 | LAX | 145 | 17 | 6 | 168 | 145800 |
| 14 | 382 | 12MAR1995 | DFW | 31 | 14 | 4 | 49 | 27000 |
| 15 | 431 | 16MAR1995 | LAX | 136 | 19 | 3 | 158 | 139500 |
| 16 | 872 | 18MAR1995 | LAX | 167 | 10 | 5 | 182 | 153300 |
| 17 | 382 | 07MAR1995 | DFW | 113 | 7 | 4 | 124 | 72000 |

Los valores nulos observados en el resultado, son consecuencia de la comparación utilizada en la sentencia IF - THEN - ELSE, donde se esta comparando contra los valores de la variable DEST en mayúsculas y como se puede observar en la impresión no todos los valores de esta variable cumplen con ese requisito. Para evitar este error lógico en el programa se debe utilizar la función UPCASE () la cual nos dará como resultado todos los valores de la variable DEST en mayúsculas para poder compararlos sín este tipo de error.

Solución:

```
data dfwlax;
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev);
   if UPCASE(dest)='DFW' then rev=SUM(boarded,trans)*600;
   else if UPCASE(dest)='LAX' then rev=SUM(boarded,trans)*900;
run;
```



| OBS | FLIGHT | DATE | DEST | BOARDED | TRANS | NONREV | TOTAL | REV |
|---|---|---|---|---|---|---|---|---|
| 1 | 439 | 13MAR1995 | LAX | 195 | 22 | 2 | 159 | 141300 |
| 2 | 431 | 12MAR1995 | LAX | 77 | 26 | 6 | 109 | 92700 |
| 3 | 921 | 31MAR1995 | DFW | 131 | 20 | 4 | 155 | 99600 |
| 4 | 921 | 04MAR1995 | DFW | 158 | 11 | 6 | 175 | 101400 |
| 5 | 114 | 13MAR1995 | LAX | 170 | 15 | 6 | 191 | 166500 |
| 6 | 982 | 27MAR1995 | dfw | 85 | 5 | 5 | 95 | 54000 |
| 7 | 439 | 07MAR1995 | LAX | 196 | 14 | | 210 | 189000 |
| 8 | 439 | 04MAR1995 | LAX | 181 | 16 | 7 | 204 | 177300 |
| 9 | 982 | 08MAR1995 | DFW | 116 | 15 | 4 | 135 | 78600 |
| 10 | 431 | 27MAR1995 | LAX | 166 | 17 | 6 | 189 | 164700 |
| 11 | 982 | 17MAR1995 | DFW | 88 | 7 | 5 | 100 | 57000 |
| 12 | 114 | 16MAR1995 | LAX | 187 | 6 | 2 | 195 | 173700 |
| 13 | 431 | 29MAR1995 | LAX | 145 | 17 | 6 | 168 | 145800 |
| 14 | 982 | 12MAR1995 | DFW | 31 | 14 | 4 | 48 | 27000 |
| 15 | 431 | 16MAR1995 | LAX | 136 | 15 | 3 | 154 | 138600 |
| 16 | 872 | 18MAR1995 | LAX | 167 | 10 | 5 | 182 | 159300 |
| 17 | 982 | 07MAR1995 | DFW | 113 | 7 | 4 | 124 | 72000 |

**EJECUCIÓN DE VARIAS SENTENCIAS DE FORMA CONDICIONAL**

Las sentencias DO y END pueden ser utilizadas para ejecutar un grupo de sentencias basadas en una condición.

Forma general de la sentencia DO y END

**IF** *expression* **THEN**

**DO;**
*Statement1*;
*Statement2*;
.
.

**END;**

**ELSE**

**DO;**

*Statement1*;
*Statement2*;
.
.

**END;**

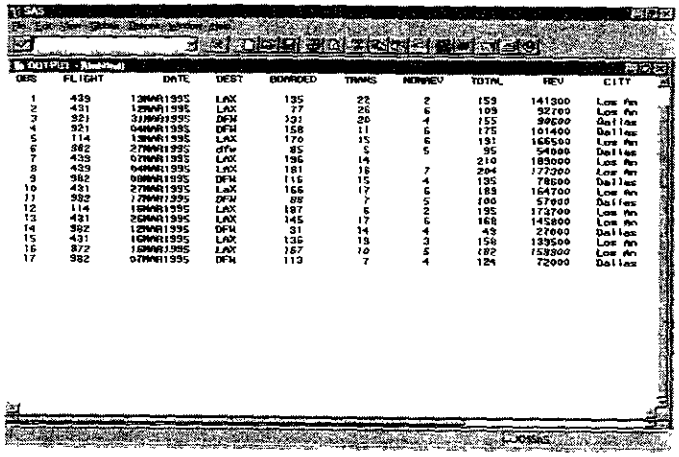
☞ **EN ESTE CASO LAS SENTENCIAS DO Y END NO SON UTILIZADAS COMO PARTE DE UN CICLO, SU FUNCIÓN AQUÍ ES LA DE AGRUPAR LAS SENTENCIAS QUE DEBERÁN EJECUTARSE SI LA CONDICIÓN DE IF/THEN/ELSE ES VERDADERA O FALSA. ES REQUISITO UTILIZARLAS CUANDO SE NECESITA EJECUTAR MÁS DE UNA SENTENCIADESPUES DE IF/THEN/ELSE.**

Ejemplo: Alterar el paso data anterior para generar una nueva variable llamada CITY y asignar un valor apropiado basado en el valor de la variable DEST.

Programa:

```
data dfwlax;
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev);
   if UPCASE(dest)='DFW' then
   do;
        rev=SUM(boarded,trans)*600;
        city='Dallas';
   end;
   else if UPCASE(dest)='LAX' then
   do;
        rev=SUM(boarded,trans)*900;
        city='Los Angeles';
   end;
run;

proc print data=dfwlax;
  format date date9.;
run;
```

Resultado



| OBS | FLIGHT | DATE | DEST | BOARDED | TRANS | NONREV | TOTAL | REV | CITY |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 439 | 13MAR1995 | LAX | 135 | 22 | 2 | 159 | 141300 | Los An |
| 2 | 431 | 12MAR1995 | LAX | 77 | 26 | 6 | 109 | 92700 | Los An |
| 3 | 921 | 31MAR1995 | DFW | 191 | 20 | 4 | 215 | 99600 | Dallas |
| 4 | 921 | 04MAR1995 | DFW | 158 | 11 | 6 | 175 | 101400 | Dallas |
| 5 | 114 | 13MAR1995 | LAX | 170 | 15 | 6 | 191 | 165600 | Los An |
| 6 | 882 | 27MAR1995 | DFW | 85 | 5 | 5 | 95 | 54000 | Dallas |
| 7 | 439 | 07MAR1995 | LAX | 196 | 14 | 3 | 210 | 185000 | Los An |
| 8 | 439 | 04MAR1995 | LAX | 181 | 18 | 7 | 206 | 177200 | Los An |
| 9 | 982 | 08MAR1995 | DFW | 116 | 15 | 4 | 135 | 78600 | Dallas |
| 10 | 431 | 27MAR1995 | LAX | 165 | 17 | 6 | 188 | 164700 | Los An |
| 11 | 982 | 17MAR1995 | DFW | 88 | 7 | 5 | 100 | 57600 | Dallas |
| 12 | 114 | 15MAR1995 | LAX | 187 | 6 | 2 | 195 | 173700 | Los An |
| 13 | 431 | 26MAR1995 | LAX | 145 | 17 | 6 | 168 | 145800 | Los An |
| 14 | 982 | 12MAR1995 | DFW | 31 | 14 | 4 | 49 | 27600 | Dallas |
| 15 | 431 | 16MAR1995 | LAX | 136 | 19 | 3 | 158 | 139500 | Los An |
| 16 | 872 | 18MAR1995 | LAX | 167 | 10 | 5 | 182 | 153900 | Los An |
| 17 | 982 | 07MAR1995 | DFW | 113 | 7 | 4 | 124 | 72000 | Dallas |

¿Por qué el valor de la variable CITY es truncado cuando su valor es igual a "Los Angeles" ?

Cuando una variable es generada asignando directamente un valor a esta sin que se haya declarado con anterioridad, la nueva variable toma las características del primer valor que se le asigna. En el ejemplo se

puede observar que dentro del programa, el primer valor que se le asigna a la variable CITY es Dallas cuyo valor es de solo 5 caracteres, por esto cuando el valor " Los Angeles " es asignado, la variable solamente almacena los 5 primeros caracteres de dicho valor. Para resolver este problema se requiere del uso de la sentencia LENGTH la cual se describe a continuación.

La sentencia LENGTH es utilizada para definir la longitud de una variable.

La forma general de la sentencia LENGTH

**LENGTH** *variable(s) $ longitud*

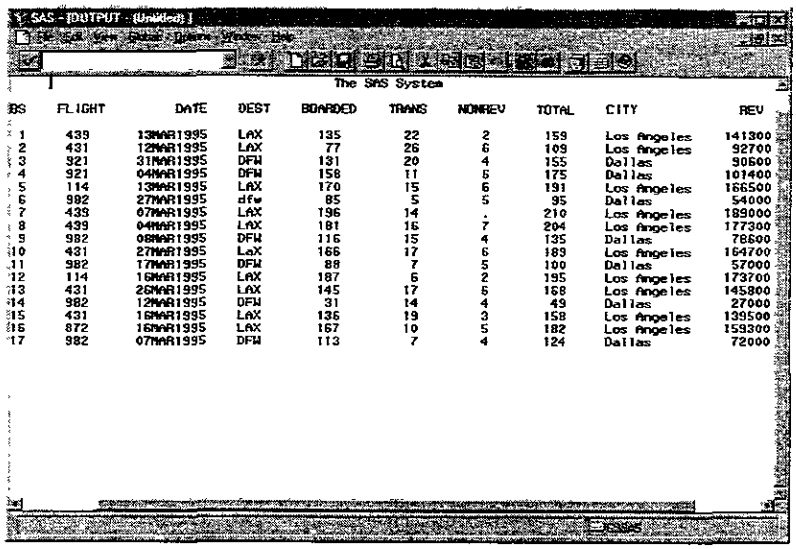
**$** Solo para variables de tipo carácter.

Ejemplo: Alterar el programa anterior de manera que los valores de la variable CITY no sean truncados.

Programa :

```
data dfwlax;
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev);
   length city $11;
   if UPCASE(dest)='DFW' then
   do;
        rev=SUM(boarded,trans)*600;
        city='Dallas';
   end;
   else if UPCASE(dest)='LAX' then
   do;
        rev=SUM(boarded,trans)*900;
        city='Los Angeles';
   end;
run;

proc print data=dfwlax;
  format date date9.;
run;
```

Resultado:



The SAS System

| OBS | FLIGHT | DATE | DEST | BOARDED | TRANS | NONREV | TOTAL | CITY | REV |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 439 | 13MAR1995 | LAX | 135 | 22 | 2 | 159 | Los Angeles | 141300 |
| 2 | 431 | 12MAR1995 | LAX | 77 | 26 | 6 | 109 | Los Angeles | 92700 |
| 3 | 921 | 31MAR1995 | DFW | 131 | 20 | 4 | 155 | Dallas | 80600 |
| 4 | 921 | 04MAR1995 | DFW | 158 | 11 | 6 | 175 | Dallas | 101400 |
| 5 | 114 | 13MAR1995 | LAX | 170 | 15 | 6 | 191 | Los Angeles | 166500 |
| 6 | 982 | 27MAR1995 | dfw | 85 | 5 | 5 | 95 | Dallas | 54000 |
| 7 | 439 | 07MAR1995 | LAX | 196 | 14 | . | 210 | Los Angeles | 189000 |
| 8 | 439 | 04MAR1995 | LAX | 181 | 16 | 7 | 204 | Los Angeles | 177300 |
| 9 | 982 | 08MAR1995 | DFW | 116 | 15 | 4 | 135 | Dallas | 78600 |
| 10 | 431 | 27MAR1995 | LAX | 166 | 17 | 6 | 189 | Los Angeles | 164700 |
| 11 | 982 | 17MAR1995 | DFW | 88 | 7 | 5 | 100 | Dallas | 57000 |
| 12 | 114 | 16MAR1995 | LAX | 187 | 6 | 2 | 195 | Los Angeles | 173700 |
| 13 | 431 | 26MAR1995 | LAX | 145 | 17 | 6 | 168 | Los Angeles | 145800 |
| 14 | 982 | 12MAR1995 | DFW | 31 | 14 | 4 | 49 | Dallas | 27000 |
| 15 | 431 | 16MAR1995 | LAX | 136 | 19 | 3 | 158 | Los Angeles | 139500 |
| 16 | 872 | 16MAR1995 | LAX | 167 | 10 | 5 | 182 | Los Angeles | 159300 |
| 17 | 982 | 07MAR1995 | DFW | 113 | 7 | 4 | 124 | Dallas | 72000 |

**SELECCIONANDO VARIABLES**

Cuando se leen archivos y se transforman los valores de las variables utilizando funciones, puede suceder que algunas de las variables que se leen no se necesiten posteriormente. También sucede que al trabajar con un archivo con muchas variables, solo se necesiten leer algunas de estas y no todas.

En cualquiera de estos casos se pueden utilizar las opciones del paso DATA. DROP= o KEEP= , estas opciones controlan la información que es leída desde el DATA SET y también la que es escrita al mismo.

☞ **LA IDEA DE LEER O ESCRIBIR SOLO CIERTAS VARIABLES ES PODER OPTIMIZAR LOS TIEMPOS DE PROCESO, TRABAJANDO SOLO CON LAS VARIABLES QUE REALMENTE SE REQUIEREN.**

La forma general de utilizar estas opciones es la siguiente:

*SAS-Data-set(DROP= variables)*

*SAS-Data-set(KEEP= variables)*

Volviendo al ejemplo, es común que algunos de los datos que se utilizan durante el programa para crear nuevas variables no requieran ser almacenados después de ser utilizados, en tal caso, será necesario al momento de almacenar el data set que se esta creando desechar las variables que no son necesarias, o lo que es lo mismo, almacenar solo aquellas que serán de utilidad en el futuro (entre otras muchas cosas por cuestión de espacio). Para tal efecto se utilizaran las opciones antes descritas.

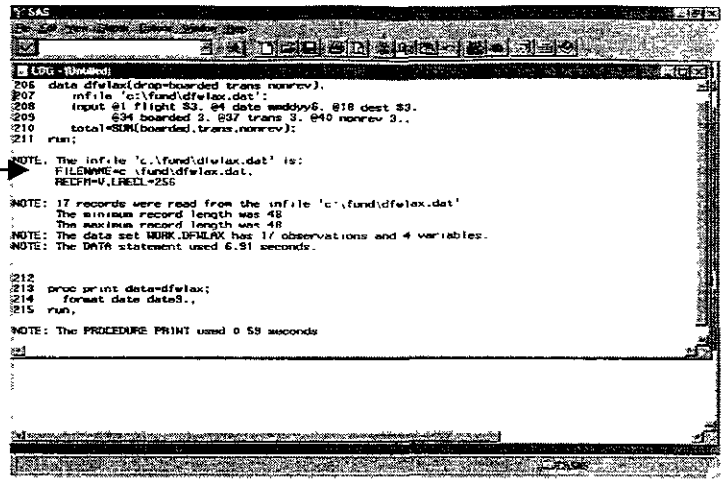
Ejemplo: Tomando como base el programa anterior, donde se genero la variable TOTAL a partir de los valores de las variables BOARDED, TRANS y NONREV, se desea almacenar el valor de la variable TOTAL, sin almacenar las variables a partir de las cuales fue generada.

Programa:

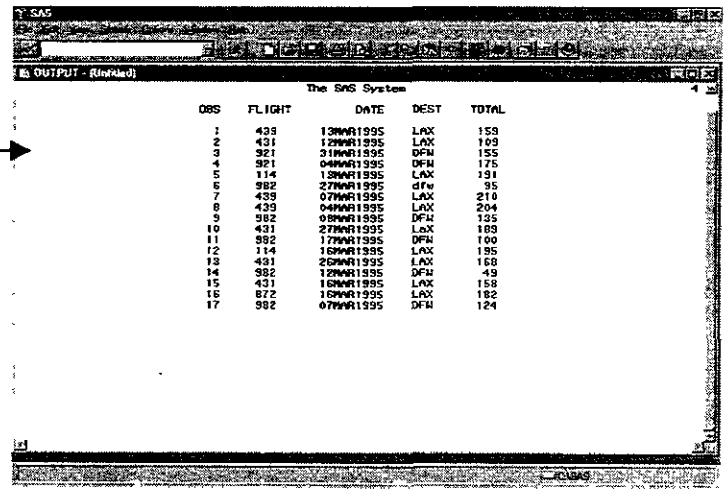
```
data dfwlax(drop=dest boarded trans nonrev);
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev);
   if UPCASE(dest)='DFW' then
   do;
        rev=SUM(boarded,trans)*600;
        city='Dallas';
   end;
   else if UPCASE(dest)='LAX' then
   do;
        rev=SUM(boarded,trans)*900;
        city='Los Angeles';
   end;
run;

proc print data=dfwlax;
  format date date9.;
run;
```

Resultado

Al ejecutar el programa y revisar la ventana del LOG, se observa que solo 4 variables fueron almacenadas en el data set y estas son aquellas no incluidas en la opción DROP




Finalmente esta información podrá ser verificada al momento de la impresión en la ventana del OUTPUT donde solo se observan las variables incluidas en el data set, ademas se puede ver que las variables BOARDED, TRANS y NONREV no aparecen.




Dentro del PDV se genera una casilla para todas y cada una de las variables, solo que esta vez al momento de escribir al data set se excluirán aquellas incluidas en la opción DROP.

☞ ¿CUÁNDO UTILIZAR DROP Y CUÁNDO UTILIZAR KEEP?

SI TIENE 100 VARIABLES DE LAS CUALES SOLO DESEA RETENER 10. ¿QUÉ ES MÁS FÁCIL? , ESCRIBIR LA OPCIÓN DROP CON UNA LISTA DE 90 VARIABLES QUE SE DESECHARAN Ó ESCRIBIR LA OPCIÓN KEEP CON LA LISTA DE LAS 10 VARIABLES QUE DESEA RETENER.

VALE LA PENA TOMAR UNOS MINUTOS ANTES DE COMENZAR UN PROGRAMA, PARA ANALIZAR CUALES SON LAS VARIABLES QUE REALMENTE SE UTILIZARAN, DE ESTA MANERA SE LOGRARA OPTIMIZAR LOS TIEMPOS DE LECTURA Y ESCRITURA

**SELECCIONANDO OBSERVACIONES**

Se puede utilizar la sentencia IF para controlar que observaciones serán escritas al data set.

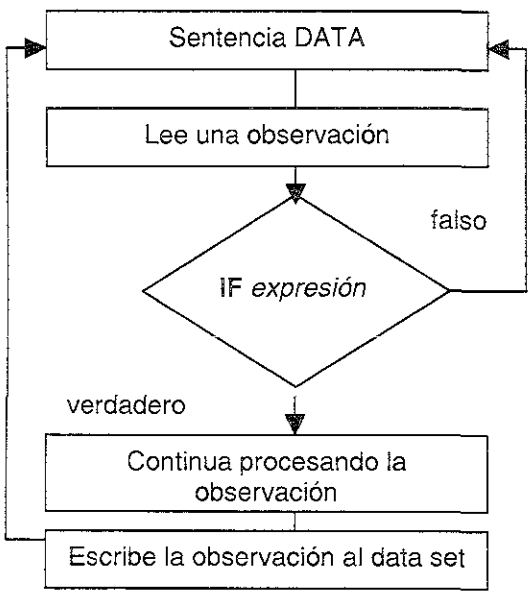
Forma general de la sentencia IF

IF *expresión;*

*Expresión* Puede ser cualquier expresión SAS válida.

La sentencia IF

- Regresa el control al inicio del paso data si la expresión es falsa (la observación no es escrita).
- Permite que la observación pase y sea procesada solo cuando la expresión es verdadera.

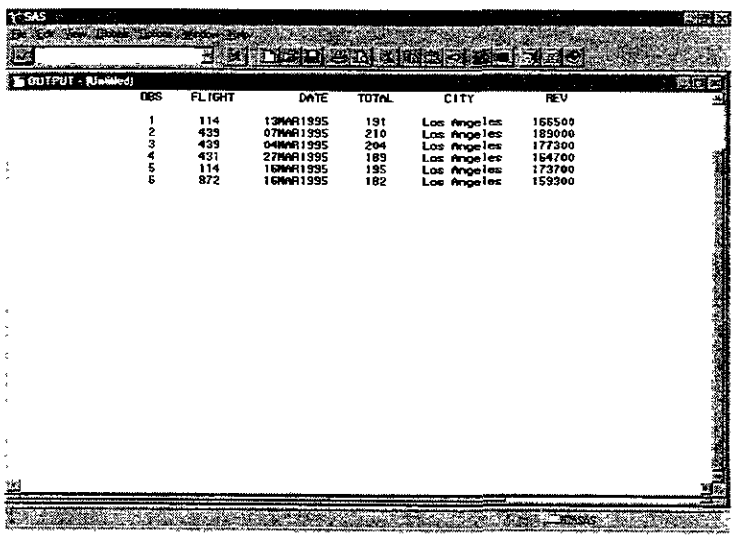


Ejemplo: Alterar el programa anterior para seleccionar solo aquellos registros donde el total de pasajeros sea mayor a 175

Programa :

```
data dfwlax(drop= dest boarded trans nonrev);
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev);
   if total gt 175;
   length city $11;
   if UPCASE(dest)='DFW' then
   do;
        rev=SUM(boarded,trans)*600;
        city='Dallas';
   end;
   else if UPCASE(dest)='LAX' then
   do;
        rev=SUM(boarded,trans)*900;
        city='Los Angeles';
   end;
run;

proc print data=dfwlax;
  format date date9.;
run;
```

Resultado:



| OBS | FLIGHT | DATE | TOTAL | CITY | REV |
|---|---|---|---|---|---|
| 1 | 114 | 13MAR1995 | 191 | Los Angeles | 166500 |
| 2 | 439 | 07MAR1995 | 210 | Los Angeles | 189000 |
| 3 | 439 | 04MAR1995 | 204 | Los Angeles | 177300 |
| 4 | 431 | 27MAR1995 | 189 | Los Angeles | 164700 |
| 5 | 114 | 16MAR1995 | 195 | Los Angeles | 173700 |
| 6 | 872 | 16MAR1995 | 182 | Los Angeles | 159300 |

Otra forma de seleccionar las observaciones que deseamos escribir al data set, es utilizando la siguiente sentencia.

Forma general de utilizar esta expresión

**IF** *expresión* **THEN DELETE**:

Si *expresión* es

- Verdadera, la sentencia DELETE es ejecutada y el control del proceso regresa al inicio del paso data (la observación es borrada).
- Falsa, la sentencia DELETE no es ejecutada y el proceso continua con la siguiente sentencia en el paso data.

No existe diferencia entre el uso de esta sentencia y la anterior.

En este contexto, surge una pregunta, ¿Cómo tendría que hacer, para seleccionar solo aquellas observaciones donde la variable **date** es menor al 7 de Marzo de 1995? . La dificultad que se enfrenta aquí es en referencia al manejo de variables de tipo fecha dentro de SAS. Como se menciono anteriormente, las fechas dentro de SAS son almacenadas como un número de días transcurridos a partir del 1 de Enero de 1960 y hasta la fecha que se esta almacenando, por lo cual si se requiere utilizar una fecha dentro de la *expresión* sería necesario conocer el número de días que esa fecha representa.

Se puede utilizar un valor de fecha constante para generar fechas que sean reconocidas como una fecha SAS.

La sentencia, '*ddMMMyy* ' **D**, crea un valor de fecha SAS a partir del valor que recibe con el formato de fecha especificado. Es importante utilizar las comillas y la D después de la definición de la fecha, ya que estos son los delimitadores que permiten a SAS interpretar estos valores.

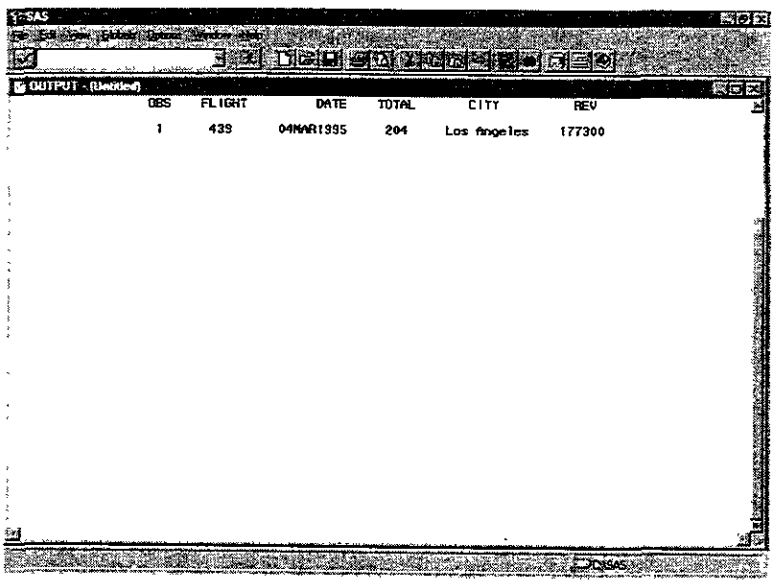
*dd* Es uno o dos dígitos para el valor del día
*MMM* Son tres letras de abreviación del mes en inglés (JAN, FEB, MAR etc.).
*yy* Es un valor de dos dígitos para el año.

Ejemplo: **if date lt '07MAR95'd;**
Alterar el paso data anterior utilizando un valor constante de fecha para seleccionar solo aquellos vuelos que partieron antes del 3 de Marzo de 1990 con más de 175 pasajeros.

Programa:

```
data dfwlax(drop= dest boarded trans nonrev);
   infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev);
   if total gt 175 and date lt '07MAR95'd;
   length city $11;
   if UPCASE(dest)='DFW' then
   do;
        rev=SUM(boarded,trans)*600;
        city='Dallas';
   end;
   else if UPCASE(dest)='LAX' then
   do;
        rev=SUM(boarded,trans)*900;
        city='Los Angeles';
   end;
run;

proc print data=dfwlax;
  format date date9.;
run;
```

Resultado:



**ACUMULANDO TOTALES**

Dentro de un paso DATA se pueden utilizar sentencias para decidir que variables dentro del PDV serán inicializadas con valores missing y cuales deben conservar su valor.

Ejemplo: Un empleado de la compañía manufacturera almacena en un renglón de un archivo plano el monto de la aportación que la compañía hace en su fondo de retiro. Cada observación representa la contribución por un año comenzando en 1997.

```
1
1---5----0

 500
1000
1500
2200
2700
```

Crear un data set llamado RETIRO tomando los datos del archivo plano, con las variables MONTO ANU y TOTAL. Esta última variable debe contener el valor acumulado del monto de retiro año tras año.

Utilizar la sentencia RETAIN para crear las variables YEAR y TOTAL, inicializar estas variables con valores apropiados y retener sus valores a través de la ejecución del paso data.

Como se menciono anteriormente la forma en la cual trabaja SAS es leyendo los archivos de forma secuencial, esto debido a que SAS es capaz de detectar automáticamente el fin de un registro y por supuesto el fin de un archivo.

Cuando un fin de registro es encontrado todas las variables en el PDV son inicializadas con valores nulos, por lo cual cualquier valor que haya tomado cualquiera de las variables existentes, desaparece, por este motivo es imposible realizar operaciones con los valores anteriores de una variable a menos que se utilice la sentencia RETAIN.

La sentencia RETAIN es utilizada regularmente para:

- Asignar un valor inicial a una variable retenida.
- Prevenir la inicialización de variables cada vez que el paso data es ejecutado.

Forma general de la sentencia RETAIN

**RETAIN** *variables valor inicial* . . . . ;

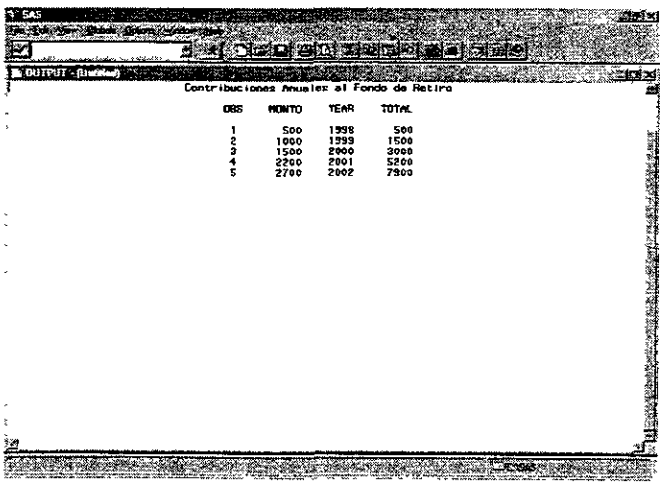
La sentencia RETAIN

- Se ejecuta al momento de compilación y crea la variable si es que esta no existe.
- Inicializa la variable retenida con un valor nulo antes de la primer ejecución del paso data si no es asignado un valor inicial.

Programa:

```
data retiro;
   infile 'c:\fund\retiro.txt';
   input monto 1-4;
   retain year 1997 total 0;
   year=year+1;
   total=total+monto;
run;

proc print data=retiro;
  title 'Contribuciones Anuales al Fondo de Retiro';
run;
```

Resultado:



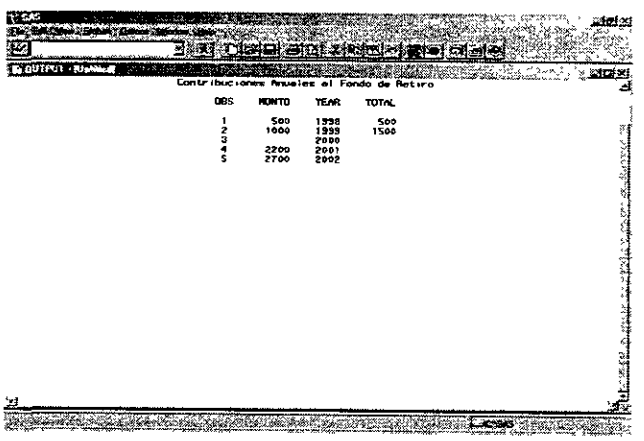
Contribuciones Anuales al Fondo de Retiro

| OBS | MONTO | YEAR | TOTAL |
|---|---|---|---|
| 1 | 500 | 1998 | 500 |
| 2 | 1000 | 1999 | 1500 |
| 3 | 1500 | 2000 | 3000 |
| 4 | 2200 | 2001 | 5200 |
| 5 | 2700 | 2002 | 7900 |

¿ Qué pasaría con los valores de TOTAL y MONTO si existiera algún valor missing en alguna de las observaciones?

```
1
1---5----0

 500
1000
  .
2200
2700
```

Resultado obtenido al ejecutar el programa anterior, sobre el conjunto de datos con un valor missing



Contribuciones Anuales al Fondo de Retiro

| OBS | MONTO | YEAR | TOTAL |
|---|---|---|---|
| 1 | 500 | 1998 | 500 |
| 2 | 1000 | 1999 | 1500 |
| 3 | | 2000 | |
| 4 | 2200 | 2001 | |
| 5 | 2700 | 2002 | |

Se puede utilizar la sentencia suma para acumular valores de variables.

La forma general de utilizar la sentencia suma:

*Variable + expresión*

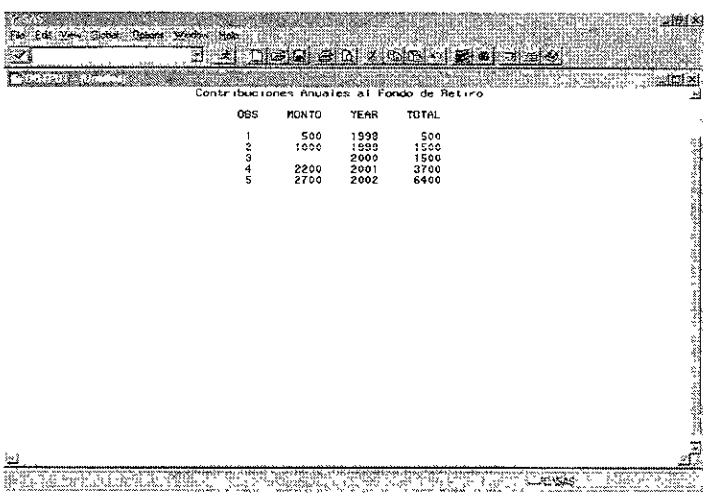
- Variable es retenida e inicializada con un valor de cero.
- El valor actual de la variable es sumada con el resultado de la expresión.
- La expresión es ignorada si el resultado es nulo.

Ejemplo: Repite el ejemplo anterior utilizando la sentencia suma.

Programa:

```
data retiro;
   infile 'c:\fund\retiro.txt';
   input monto 1-4;
   retain year 1997 total 0;
   year=year+1;
   total+monto;
run;

proc print data=retiro;
  title 'Contribuciones Anuales al Fondo de Retiro';
run;
```

Resultado:



Contribuciones Anuales al Fondo de Retiro

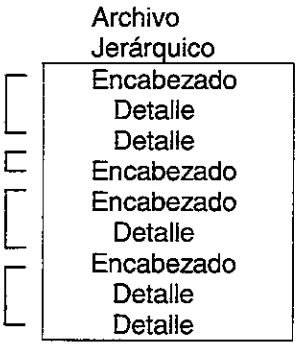
| OBS | MONTO | YEAR | TOTAL |
|---|---|---|---|
| 1 | 500 | 1998 | 500 |
| 2 | 1000 | 1999 | 1500 |
| 3 | | 2000 | 1500 |
| 4 | 2200 | 2001 | 3700 |
| 5 | 2700 | 2002 | 6400 |

☞ CUANDO UNA SENTENCIA DE SUMA A+B ES UTILIZADA DENTRO DE UN PROGRAMA, UN RETAIN IMPLICITO ES UTILIZADO PARA RETENER EL VALOS DE LA VARIABLE EN LA CUAL SE ESTAN ACUMULANDO LOS VALORES, DE ESTA FORMA CADA VEZ QUE EL PROGRAM DATA VECTOR SE INICIALIZA LOS VALORES DE DICHA VARIABLE SE MANTIENEN PARA SEGUIR REALIZANDO LAS ACUMULACIONES

## 3.4 MANEJO DE ARCHIVOS JERÁRQUICOS

Muchos archivos que son jerárquicos en su estructura constan de:

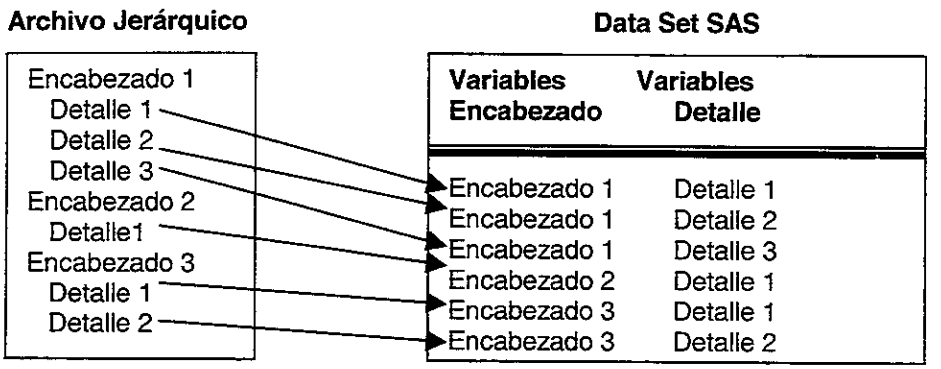
- Un registro de encabezado
- Uno ó más registros de detalle relacionados al encabezado



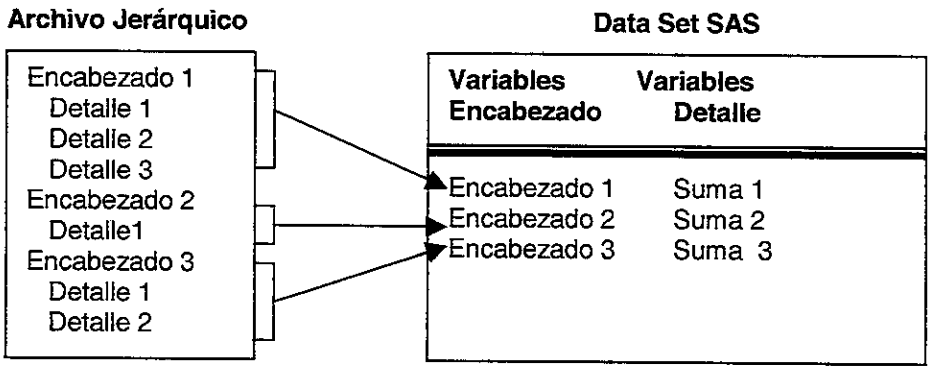

Típicamente cada registro contiene un campo que identifica si es que ese es un registro detalle o encabezado.

Un archivo jerárquico puede ser leído utilizando SAS para crear un data set que contenga una observación:

- Por cada registro detalle, almacenando el encabezado del registro como parte de cada observación.




- Por cada encabezado, almacenando el detalle de información sumarizado como parte del registro.



Ejemplo: Un archivo contiene la variable ID que identifica a los empleados y estos registros están marcados como el encabezado de la observación (E), seguidos de estos, el archivo contiene registros de detalle donde se almacena una fecha y una medida de colesterol (C).

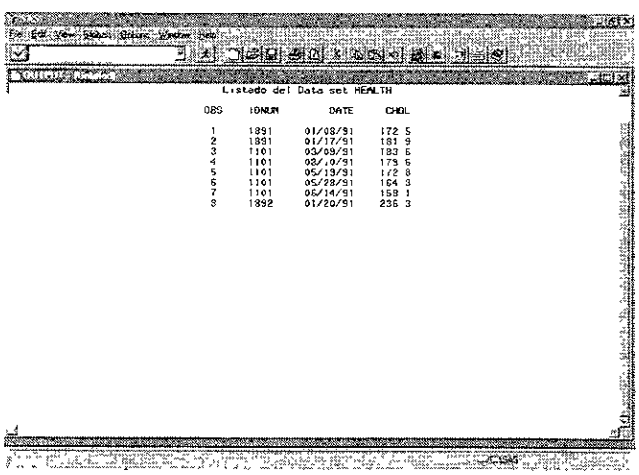
```
    1   1   2   2   3   3
1---5----0----5----0----5----0----5

E 1891
C 010891 172.5
C 011791 181.9
E 1101
C 030991 183.6
C 031091 179.6
C 051991 172.8
C 052891 164.3
C 061491 159.1
E 1892
C 012091 236.3
```

Crear un data set con una observación por cada registro de detalle leído para cada empleado. Imprimir el resultado.

Programa:

```
data health(drop=type);
  retain idnum;
  infile 'c:\fund\chol.txt';
  input type $1. @;
  if type='E' then input @3 idnum $4.;
  if type='C';
  input @3 date mmddyy6. +1 chol 5.;
run;

proc print data=health;
   format date mmddyy8.;
   title 'Listado del Data set HEALTH';
run;
```

Resultado



Listado del Data set HEALTH

| OBS | IDNUM | DATE | CHOL |
|---|---|---|---|
| 1 | 1891 | 01/08/91 | 172.5 |
| 2 | 1891 | 01/17/91 | 181.9 |
| 3 | 1101 | 03/09/91 | 183.6 |
| 4 | 1101 | 03/10/91 | 179.6 |
| 5 | 1101 | 05/13/91 | 172.8 |
| 6 | 1101 | 05/28/91 | 164.3 |
| 7 | 1101 | 06/14/91 | 159.1 |
| 8 | 1892 | 01/20/91 | 236.3 |

Ejemplo: Crear un data set, a partir de la lectura del archivo plano que se muestra a continuación, en el cual las tres primeras líneas son el encabezado del registro cuyo principio esta identificado (A) conteniendo, nombre y dirección del empleado.

Los registros del encabezado son seguidos por un número variado de registros de detalle. Los registros de detalle contiene el nombre, la relación (S = cónyuge C= hijo), y el genero (F = mujer , M = hombre), para cada dependiente del empleado.

Escribir un paso data que calcule el total de la deducción de hospitalización para elaborar el cheque correspondiente a cada empleado. El plan indica una deducción de $50 por cada empleado, $25 por cónyuge y $10 por cada hijo. Crear un data set con una observación por empleado con el total del cheque. Imprimir el data set resultante.

```
1   1   2   2   3   3
1---5----0----5----0----5----0----5

A ADAMS, SUSAN R
  441 GLENWOOD AVE
  RALEIGH, NC 27606
  MICHAEL C M
  LINDSEY C F
A ALEXANDER, JULIAN T
  2316 BEDFORD AVE
  RALEIGH, NC 27607
  SHERRY S F
  EMILY C F
  BENJAMIN C M
  THOMAS C M
  JOHN C M
A APPLE, TROY L
  2480 BRADY STREET
  CARY, NC 27511
A ARTHUR, BARBARA
  222 FAIR LANE
  NEW YORK, NEW YORK 10044
  RICHARD S M
  STEPHANIE C F
  DANIELLE C F
```

Este tipo de archivo es difícil de leer por que se debe leer el siguiente registro para determinar si es que el registro actual es el último de un grupo de registros relacionados.

Cuando se escriben programas que lean encabezados, se debe prever la marca de fin de archivo que detenga el paso data. Para esto se puede utilizar la opción END = en la sentencia INFILE para crear una variable que permita detectar cuando el último registro del archivo esta siendo leído.

Forma general de la opción END = ,

**INFILE** *nombre del archivo* **END** = *variable*

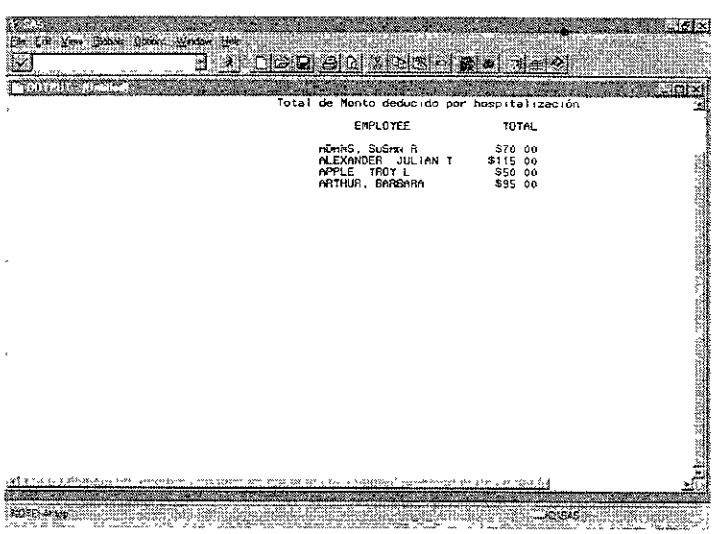
*Variable* Es una variable numérica temporal cuyos valores pueden ser

**0** Antes de que el último registro sea leído
1 Cuando el último registro es leído.

Programa:

```
data count(drop= type name relation);
  infile 'c:\fund\hospital.txt' end=last;
  input @1 type $ 1 @;
  if type='A' then
     do;
      if _n_ gt 1 then output;
      input @3 employee $20. / /;
      total=50;
     end;
  else
    do;
     input name $ relation $;
     if relation='S' then total+25;
     else total+10;
    end;
  if last then output;
  retain employee;
run;

proc print data= count noobs;
  title 'Total de Monto deducido por hospitalización';
  format total dollar8.2;
run;
```

Resultado

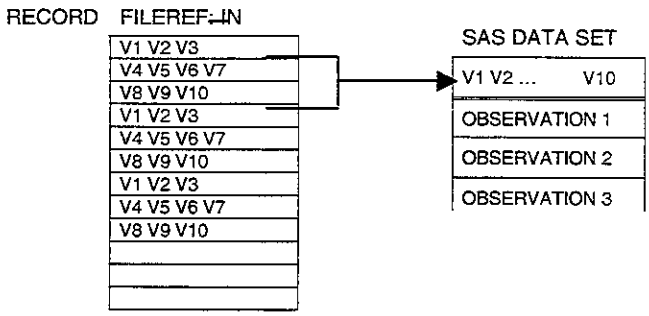



☞ LA VARIABLE "LAST" QUE CONTIENE EL VALOR DE LA OPCIÓN END EN LA SENTENCIA INFILE, PUEDE TENER CUALQUIER OTRO NOMBRE DE VARIABLE SAS VALIDO ÉSTA ES GENERADA COMO VARIABLE DE CONTROL EN EL PDV PERO NO ES ESCRITA AL DATA SET DE SALIDA

## 3.5 MÚLTIPLES REGISTROS POR OBSERVACIÓN

Los archivo planos pueden contener múltiples registros que deberán ser combinados en una sola observación, este tipo de archivos pueden ser leídos utilizando múltiples sentencias INPUT como se muestra:

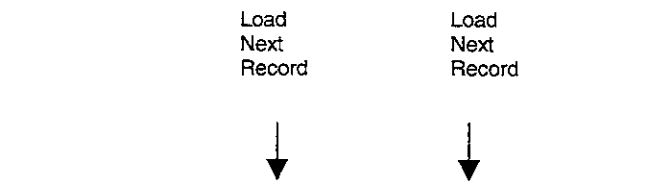
```
data test;
   infile in;
   input v1 ... v3;
   input v4 ... v7;
   input v8 ... v10;
run;
```

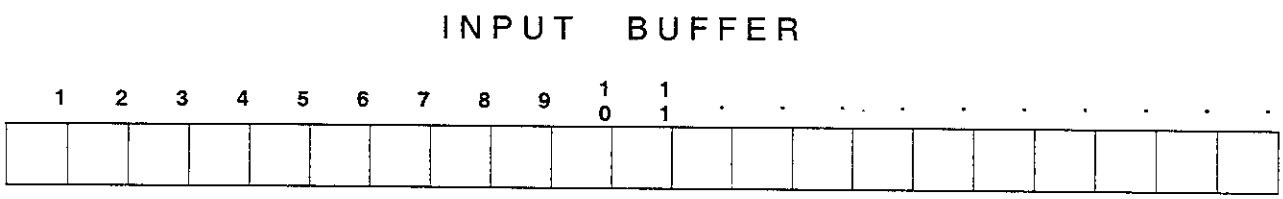



Cada sentencia INPUT llena el input buffer con el valor del siguiente registro. Esto quiere decir que cada sentencia INPUT que es encontrada dentro del programa realiza un salto de línea dentro del archivo que se esta leyendo, de tal forma que la lectura se realiza de forma secuencial.

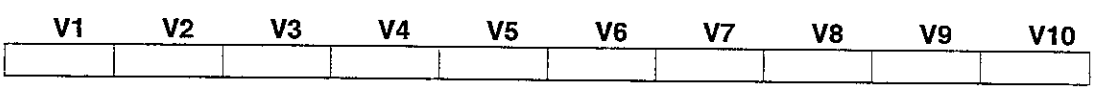
En el ejemplo cada ejecución de este paso DATA lee tres registros distintos.

De la misma forma que se utiliza la sentencia INPUT, también puede ser utilizado el llamado control apuntador de línea que es representado por una diagonal invertida / esta diagonal también representa un salto de línea al momento de llevar a cabo la lectura de un archivo plano, usualmente es utilizada para saltar registros que no se desean leer y al igual que la sentencia INPUT llena el buffer solo con un registro a la vez.




**Program Data Vector**



EN ESTE TIPO DE LECTURAS, EL PROGRAM DATA VECTOR ES LLENADO POR COMPLETO AL TERMINAR DE LEER EL NÚMERO DE REGISTROS INDICADOS EN LA SENTENCIA INPUT UNA VEZ LLENO, ES VACIADO AL DATA SET DE SALIDA ESTE TIPO DE LECTURA ES ÚTIL PARA OPTIMIZAR LOS TIEMPOS DE PROCESAMIENTO DE LOS DATOS

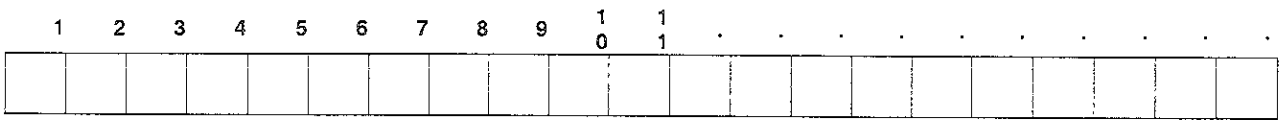
Ejemplo: Un archivo contiene información acerca de la dirección de los empleados en tres registros distintos. Crear un DATASET que contenga toda la información en un solo registro.

```
         1    1    2    2
1---5----0----5----0----5----

Farr, Sue
15 Greenwood St.
Anaheim  CA 90066
Cox, Kay B
1623 N. Avon Pl.
Anaheim  CA 90062
```

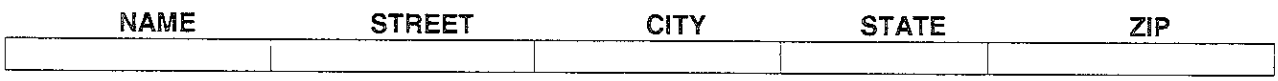
Programa:

```
data address;
  infile rawdata;
  input name $ 1-20 /
        street $ 1-20/
        city $ 1-10 state $ 11-12 zip $14-18;
run;
```

INPUT BUFFER




Program Data Vector

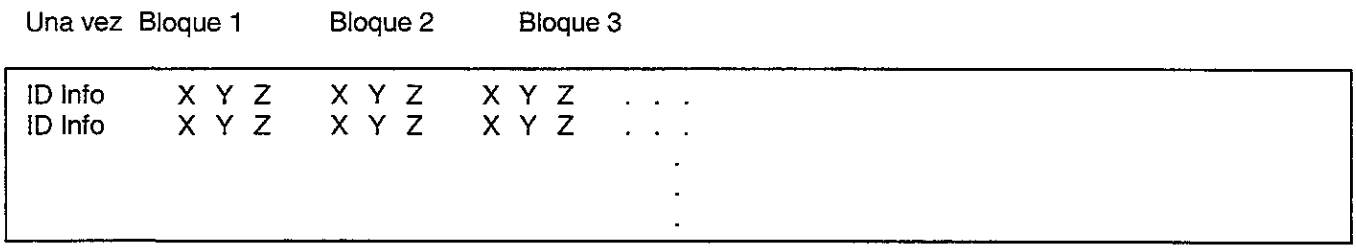


## 3.6 MÚLTIPLES OBSERVACIONES POR REGISTRO

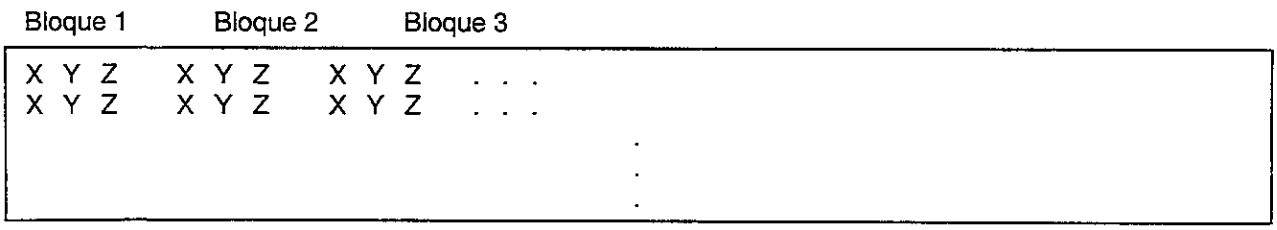
Algunos archivos contienen datos para varias observaciones en un solo registro.

Los archivos de este tipo, son clasificados dentro de dos categorías:

- Aquellos donde cada registro contiene un campo identificador (ID) que aparece una vez por línea seguido por bloques de datos que representan observaciones separadas.




- Otros donde cada registro contiene bloques separados de datos y cada bloque representa una observación independiente.




**NOTA:** Los bloques de datos y los campos dentro de los bloques pueden o no estar separados por espacios en blanco y el número de bloques por registro puede ser el mismo o variable.

Ejemplo: Se les pregunto a los empleados acerca de el tipo de actividades que les gustaría tener disponibles dentro de la compañía. Cada registro del siguiente archivo, contiene el número identificador del empleado (ID) seguido por las tres selecciones de actividades.

```
1  1  2  2  3  3
1---5----0----5----0----5----0----5
```

```
1011 SKIING MOVIES FOOTBALL
1031 FOOTBALL AEROBICS MOVIES
1032 MOVIES FOOTBALL AEROBICS
```

Crear un data set que tenga una observación por cada selección del empleado. Imprimir el data set.

Programa:

```
data survey;
   infile 'c:\fund\actividades.txt';
   input idnum $4. @;
   input activity $ @;
   rank=1;
   output;
   input activity $ @;
   rank=2;
   output;
   input activity $ @;
   rank=3;
   output;
run;

proc print data=survey noobs;
   title ' Listado del data set SURVEY';
run;
```

☞ LA FUNCIÓN DE @ AL FINAL DE LA LÍNEA ES PARA DETENER LA LECTURA Y EVALUAR ALGUNO DE LOS VALORES CUANDO UNA @ ES UTILIZADA EL CURSOR SE DETIENE EN LA POSICIÓN ACTUAL, A PARTIR DE AQUI COMENZARA LA LECTURA INDICADA POR LA SIGUIENTE INSTRUCCION INPUT, ES DECIR, EL SIGUIENTE INPUT NO CAMBIA LA LÍNEA DE LECTURA

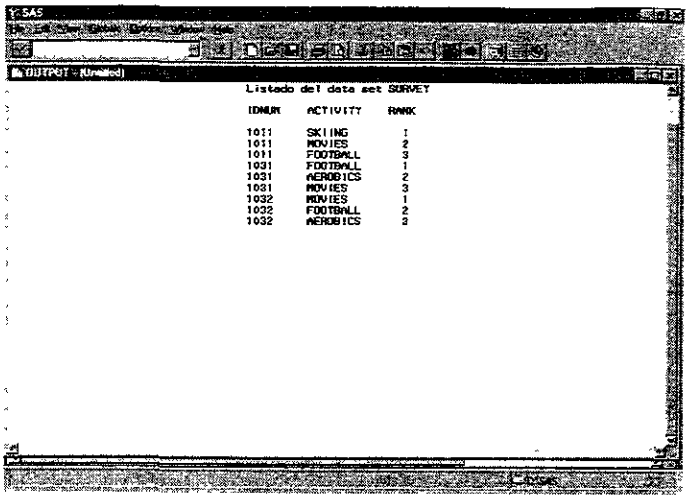
EN ESTE EJERCICIO, SE MUESTRA LA FORMA MÁS LARGA DE HACER EL PROGRAMA, SIN EMBARGO SIGUIENDO CON LAS BUENAS PRÁCTICAS DE PROGRAMACIÓN, LO IDEAL ES SIMPLIFICAR EL CÓDIGO, EL CUAL EN ESTE CASO ES MUY REPETITIVO

En el programa, la línea `input idnum $4. @;` realiza la lectura de la variable IDNUM.

En las sentencias posteriores:

```
input activity $ @;      (lee la actividad correspondiente)
rank=1;          (asigna valor de la variable RANK)
output;          (escribe al data set)
```

Resultado:



Listado del data set SURVEY

| IDNUM | ACTIVITY | RANK |
|---|---|---|
| 1011 | SKIING | 1 |
| 1011 | MOVIES | 2 |
| 1011 | FOOTBALL | 3 |
| 1031 | FOOTBALL | 1 |
| 1031 | AEROBICS | 2 |
| 1031 | MOVIES | 3 |
| 1032 | MOVIES | 1 |
| 1032 | FOOTBALL | 2 |
| 1032 | AEROBICS | 3 |

Ejemplo: Los registros diarios de la venta de café son almacenados de forma secuencial generando varios registros. Cada registro tiene almacenados los datos de 4 días.

```
1  1  2  2  3  3  4  4  5  5  6  6
1---5----0----5----0----5----0----5----0----5----0----5----0----5----0----5

02APR91 2750.31 03APR91 4820.75 04APR91 3952.81 05APR91 4005.90
06APR91 3715.14 09APR91 3295.41 10APR91 3817.50 11APR91 4242.30
12APR91 3945.60 13APR91 2583.66 16APR91 3108.16 17APR91 3795.41
18APR91 4150.89 19APR91 4959.00 20APR91 3870.41 23APR91 2992.35
24APR91 3377.45 25APR91 3802.76 26APR91 3672.00 27APR91 4195.00
30APR91 4210.81
```

Crear un data set con una observación por día, almacenar las fechas como un valor de tipo fecha SAS. Imprimir el data set.

**NOTA**:Se asume que la longitud del registro es de 80 caracteres.

Para resolver este problema se puede utilizar @@ que indica una lectura secuencial con el patrón definido en la sentencia INPUT.

La sentencia @@

- Es utilizada típicamente para leer múltiples observaciones desde un mismo registro.
- No debe ser utilizado con el puntero @ en la sentencia INPUT, ni con la opción MISSOVER.

Cuando se utiliza @@

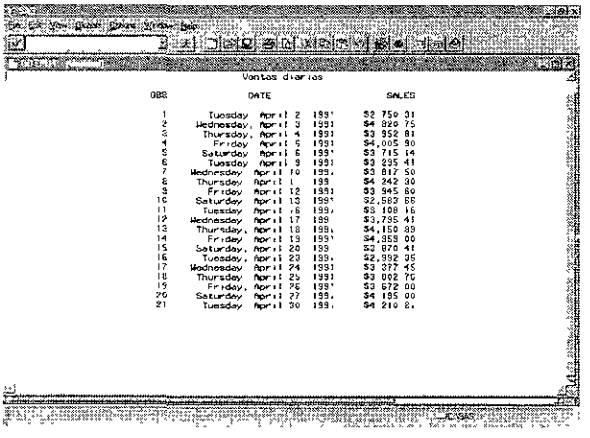
- La ejecución de la sentencia INPUT es secuencial, esto es que un nuevo registro es cargado al buffer si el fin de registro es encontrado en el registro actual.

Programa:

```
data sales;
   infile 'c:\fund\ventas.txt';
   input date : date7. sales @@;
run;

proc print data=sales;
   format date weekdate. sales dollar10.2;
   title 'Ventas diarias';
run;
```

☞ ES MUY IMPORTAMTE RESALTAR EL HECHO DE POR QUE NO SE DEBEN UTILIZAR LAS SENTENCIAS @@ CON LA OPCION MISSOVER, ESTA OPCIÓN TIENE COMO OBJETIVO INDICAR AL PUNTERO QUE DENTRO DE UN REGISTRO PUEDEN EXISTIR VALORES MISSING, LA OPCIÓN EVITA QUE EL PUNTERO SE MUEVA DE REGISTRO BUSCANDO LA INFORMACIÓN QUE LE HACE FALTA PARA LLENAR EL PDV, SI ESTA OPCIÓN ES UTILIZADA CON @@, EL PROGRAMA ENTRARIA EN UN CICLO, YA QUE AL ENCONTRAR UN VALOR MISSING PENSARIA QUE ES PARTE DEL PATRÓN

Resultado:



Ventas diarias

| OBS | DATE | SALES |
|---|---|---|
| 1 | Tuesday, April 2, 199 | $2,750.31 |
| 2 | Wednesday, April 3, 199 | $4,820.75 |
| 3 | Thursday, April 4, 199 | $3,952.81 |
| 4 | Friday, April 5, 199 | $4,005.90 |
| 5 | Saturday, April 6, 199 | $3,715.14 |
| 6 | Tuesday, April 9, 199 | $3,235.41 |
| 7 | Wednesday, April 10, 199 | $3,817.50 |
| 8 | Thursday, April 1, 199 | $4,242.30 |
| 9 | Friday, April 12, 199 | $3,945.60 |
| 10 | Saturday, April 13, 199 | $2,582.86 |
| 11 | Tuesday, April 16, 199 | $3,108.16 |
| 12 | Wednesday, April 17, 199 | $3,795.41 |
| 13 | Thursday, April 18, 199 | $4,150.89 |
| 14 | Friday, April 19, 199 | $4,358.00 |
| 15 | Saturday, April 20, 199 | $2,870.41 |
| 16 | Tuesday, April 23, 199 | $2,992.35 |
| 17 | Wednesday, April 24, 199 | $3,377.45 |
| 18 | Thursday, April 25, 199 | $3,002.70 |
| 19 | Friday, April 26, 199 | $3,672.00 |
| 20 | Saturday, April 27, 199 | $4,185.00 |
| 21 | Tuesday, April 30, 199 | $4,210.8 |

# CAPITULO 4

# LECTURA Y COMBINACIÓN DE ARCHIVOS SAS (DATA SETS)

Utilizar sentencias del lenguaje BASE SAS para la lectura, transformación y combinación de archivos SAS (data sets).

## 4.1 LECTURA DE UN DATA SET

Como ya se menciono, un data set puede ser creado utilizando como fuente de información archivos planos, tablas de bases de datos, archivos EXCEL y otros entre los cuales se encuentran data sets SAS. Durante este capitulo se mostrara la forma de generar nuevos data sets a partir de data sets existentes.

La sentencia SET es utilizada en un paso data para leer un data set existente.

La forma general de utilizar esta sentencia es:

**SET** *nombre del data set*;

Todas las variables y observaciones son:

- Automáticamente leídas directamente del data set especificado en la sentencia SET (a menos de que se especifique otra cosa) y escritas a el PDV.
- Automáticamente copiadas desde el PDV a el nuevo data set especificado en la sentencia DATA (a menos de que se especifique otra cosa).

Para crear un data set partiendo de uno existente.

```
data new;
  set old;

    Sentencias adicionales para hacer modificaciones

run;
```

También se puede remplazar un data set existente

```
data old;
  set old;

    Sentencias adicionales para hacer modificaciones

run;
```

Al momento de compilación

- La sentencia SET lee la parte descriptora del data set de entrada.
- PDV es creado conteniendo todas las variables leídas desde el data set de entrada además de cualquier nueva variable creada en el paso data.
- La parte descriptora del nuevo data set es creada.

☞ LA SENTENCIA SET SOLO PUEDE SER UTILIZADA DENTRO DE UN PASO DATA

Ejemplo:

DATA SET PASAJERO

| NAME: | DEST | BOARDED | TRANS | NONREV |
|---|---|---|---|---|
| TYPE: | $ | N | N | N |
| LENGTH: | 3 | 8 | 8 | 8 |
| | LAX | 172 | 18 | 6 |
| | LAX | 167 | 13 | 5 |
| | DFW | 49 | 19 | . |

```
data TOTAL;
  set PASAJERO;
  total=sum(boarded,trans,nonrev);
run;
```

P D V

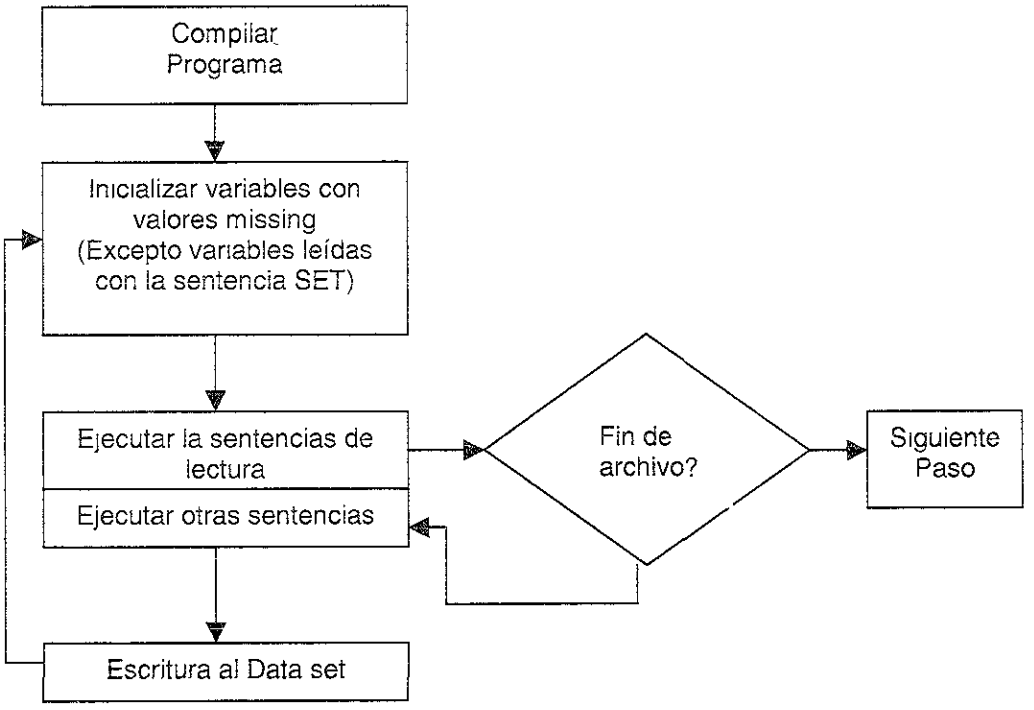
| NAME: | DEST | BOARDED | TRANS | NONREV | TOTAL |
|---|---|---|---|---|---|
| TYPE: | $ | N | N | N | N |
| LENGTH: | 3 | 8 | 8 | 8 | 8 |
| VALOR | | | | | |

DATA SET TOTAL

| NAME: | DEST | BOARDED | TRANS | NONREV | TOTAL |
|---|---|---|---|---|---|
| TYPE: | $ | N | N | N | N |
| LENGTH: | 3 | 8 | 8 | 8 | 8 |

Al momento de ejecución

- Cada sentencia es ejecutada secuencialmente.
- La sentencia SET lee la siguiente observación desde el data set entrante y la escribe dentro del PDV.
- Otra sentencia podría entonces modificar la observación actual.
- Los valores del PDV son escritos al nuevo data set al final del paso data.
- El flujo del programa es regresado al principio del paso data.
- Los siguientes pasos son ejecutados hasta que se encuentra el fin de archivo.

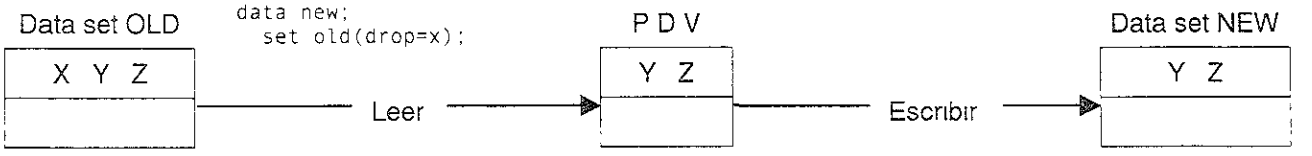



Cuando se escribe un paso DATA que lee un data set existente pueden utilizar las sentencias DROP = o KEEP =.

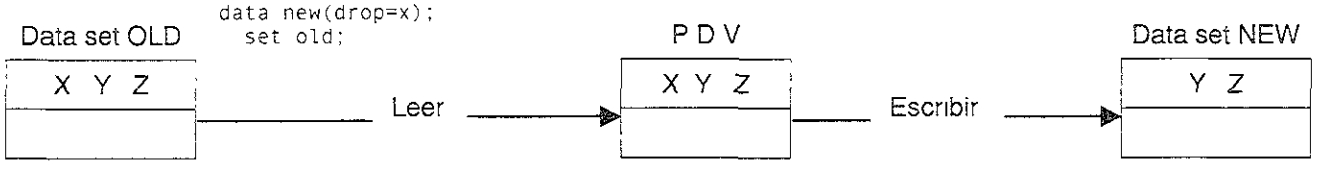
- En el data set entrante (el que se esta leyendo con la sentencia SET).
- En el data set saliente (el que se esta creando con la sentencia DATA).

Si las opciones DROP = o KEEP = son utilizadas en una

- Sentencia SET, estas opciones afectan los datos que se están leyendo además de definir que datos se escribirán en el PDV y cuales deben ser escritos al data set saliente.




- Sentencia DATA esta afecta los datos que se están escribiendo al nuevo data set desde el PDV.



Ejemplo: Escribir un paso data para generar un data set llamado WEIGHT que lea solo las variables FLIGHT, DATE, MAIL Y FREIGHT desde el data set WASHDC. Crear una variable llamada POUNDS que contenga la suma de los valores de las variables MAIL y FREIGHT.

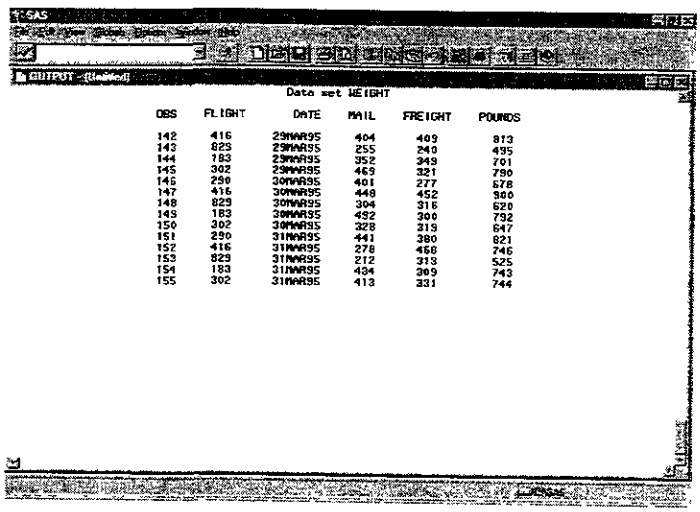
Programa:

```
data weight;
  set sasdata.washdc(keep=flight date mail freight);
  pounds=sum(freight,mail);
run;

proc print data=weight;
  title 'Data set WEIGHT';
run;
```

**P D V**

| FLIGHT | DATE | MAIL | FREIGHT | POUNDS |
|---|---|---|---|---|
| | | | | |

Resultado:



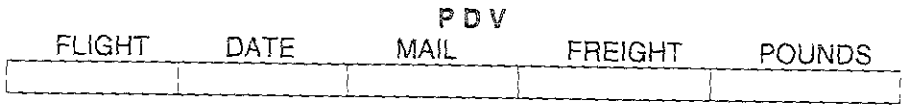
Data set WEIGHT

| OBS | FLIGHT | DATE | MAIL | FREIGHT | POUNDS |
|---|---|---|---|---|---|
| 142 | 416 | 29MAR95 | 404 | 409 | 813 |
| 143 | 829 | 29MAR95 | 255 | 240 | 495 |
| 144 | 183 | 29MAR95 | 352 | 349 | 701 |
| 145 | 302 | 29MAR95 | 469 | 321 | 790 |
| 146 | 290 | 30MAR95 | 401 | 277 | 678 |
| 147 | 416 | 30MAR95 | 448 | 452 | 900 |
| 148 | 829 | 30MAR95 | 304 | 316 | 620 |
| 149 | 183 | 30MAR95 | 492 | 300 | 792 |
| 150 | 302 | 30MAR95 | 328 | 319 | 647 |
| 151 | 290 | 31MAR95 | 441 | 380 | 821 |
| 152 | 416 | 31MAR95 | 278 | 468 | 746 |
| 153 | 829 | 31MAR95 | 212 | 313 | 525 |
| 154 | 183 | 31MAR95 | 434 | 309 | 743 |
| 155 | 302 | 31MAR95 | 413 | 331 | 744 |

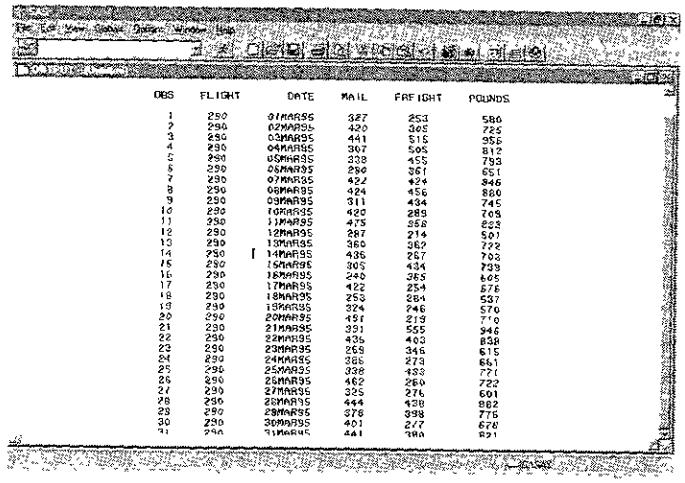
Alterar el paso DATA anterior para seleccionar solo las observaciones donde el número de vuelo sea igual a 290.

Programa:

```
data weight;
  set sasdata.washdc(keep=flight date mail freight);
  if flight='290';
  pounds=sum(freight,mail);
run;

proc print data=weight;
  title 'Data set WEIGHT';
run;
```




Resultado:

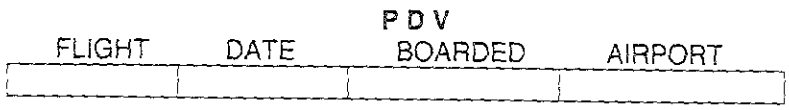


| OBS | FLIGHT | DATE | MAIL | FREIGHT | POUNDS |
|---|---|---|---|---|---|
| 1 | 290 | 01MAR95 | 327 | 253 | 580 |
| 2 | 290 | 02MAR95 | 420 | 305 | 725 |
| 3 | 290 | 03MAR95 | 441 | 515 | 956 |
| 4 | 290 | 04MAR95 | 307 | 505 | 812 |
| 5 | 290 | 05MAR95 | 338 | 455 | 793 |
| 6 | 290 | 06MAR95 | 290 | 361 | 651 |
| 7 | 290 | 07MAR95 | 422 | 424 | 846 |
| 8 | 290 | 08MAR95 | 424 | 456 | 880 |
| 9 | 290 | 09MAR95 | 311 | 434 | 745 |
| 10 | 290 | 10MAR95 | 420 | 289 | 709 |
| 11 | 290 | 11MAR95 | 475 | 358 | 833 |
| 12 | 290 | 12MAR95 | 287 | 214 | 501 |
| 13 | 290 | 13MAR95 | 360 | 362 | 722 |
| 14 | 290 | 14MAR95 | 436 | 267 | 703 |
| 15 | 290 | 15MAR95 | 305 | 434 | 739 |
| 16 | 290 | 16MAR95 | 240 | 365 | 605 |
| 17 | 290 | 17MAR95 | 422 | 254 | 676 |
| 18 | 290 | 18MAR95 | 253 | 284 | 537 |
| 19 | 290 | 19MAR95 | 324 | 246 | 570 |
| 20 | 290 | 20MAR95 | 491 | 219 | 710 |
| 21 | 290 | 21MAR95 | 291 | 565 | 946 |
| 22 | 290 | 22MAR95 | 436 | 403 | 839 |
| 23 | 290 | 23MAR95 | 269 | 346 | 615 |
| 24 | 290 | 24MAR95 | 386 | 273 | 661 |
| 25 | 290 | 25MAR95 | 338 | 433 | 771 |
| 26 | 290 | 26MAR95 | 462 | 260 | 722 |
| 27 | 290 | 27MAR95 | 325 | 276 | 601 |
| 28 | 290 | 28MAR95 | 444 | 438 | 882 |
| 29 | 290 | 29MAR95 | 378 | 398 | 776 |
| 30 | 290 | 30MAR95 | 401 | 277 | 678 |
| 31 | 290 | 31MAR95 | 441 | 380 | 821 |

Crear un data set temporal llamado AIRPORT leyendo solo las variables FLIGHT, DATE y BOARDED desde el data set WASHDC. Crear una variable llamada AIRPORT asignando el nombre del aeropuerto (Dulles o National) basado el número de vuelo.

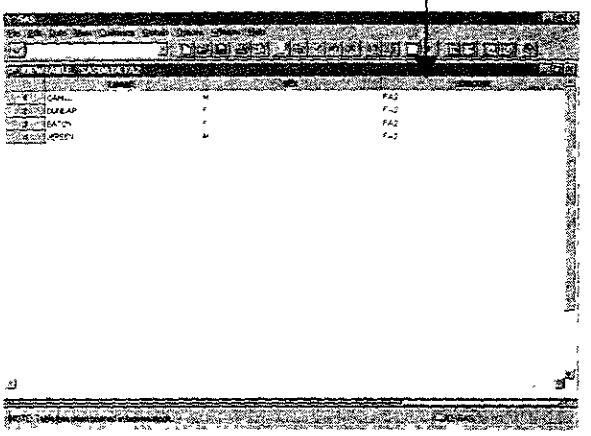
Programa:

```
data airport;
  set sasdata.washdc(keep=flight date mail freight);
  length airport $ 8;
  if flight in ('183','290','416') then airport='Dulles';
  else if flight in ('302','829') then airport='National';
run;

proc print data=airport;
  title 'Data set AIRPORT';
run;
```

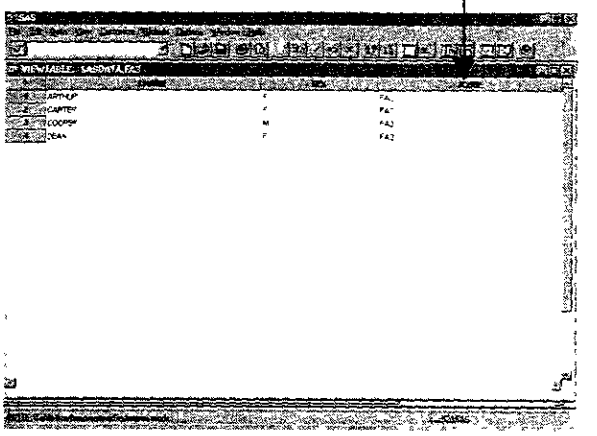


Notar las diferencias en los nombres de las variables.

**FA2**



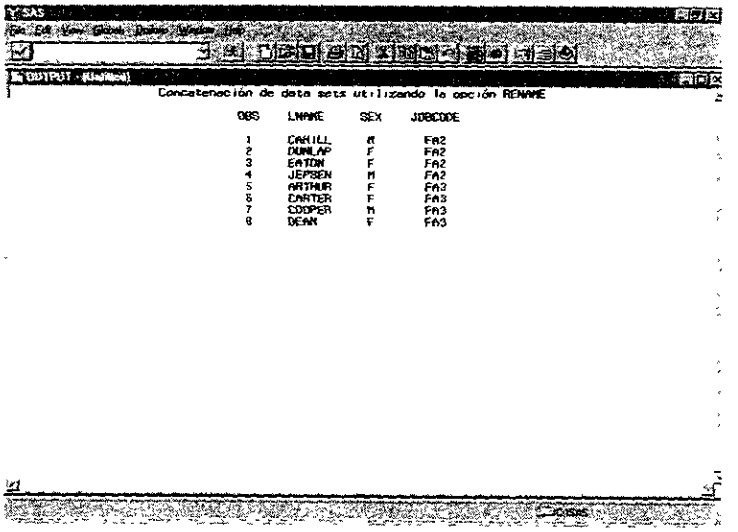
**FA3**



Programa:

```
data new;
  set sasdata.fa2 sasdata.fa3(rename=(jcode=jobcode));
run;

proc print data=new;
 title 'Concatenación de data sets utilizando la opción RENAME';
run;
```

Resultado:



Concatenación de data sets utilizando la opción RENAME

| OBS | LNAME | SEX | JOBCODE |
|---|---|---|---|
| 1 | CARILL | M | FA2 |
| 2 | DUNLAP | F | FA2 |
| 3 | EATON | F | FA2 |
| 4 | JEPSEN | M | FA2 |
| 5 | ARTHUR | F | FA3 |
| 6 | CARTER | F | FA3 |
| 7 | COOPER | M | FA3 |
| 8 | DEAN | F | FA3 |

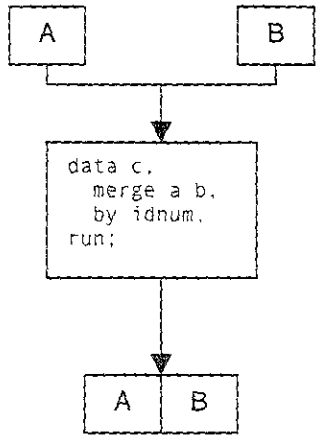
### A SENTENCIA MERGE

a sentencia MERGE es utilizada para unir observaciones correspondientes de 2 ó más data sets.

.a forma general de la sentencia MERGE es

```
DATA data set;
  MERGE data set1  data set2 . . . ;
  BY  variables;
RUN;
```

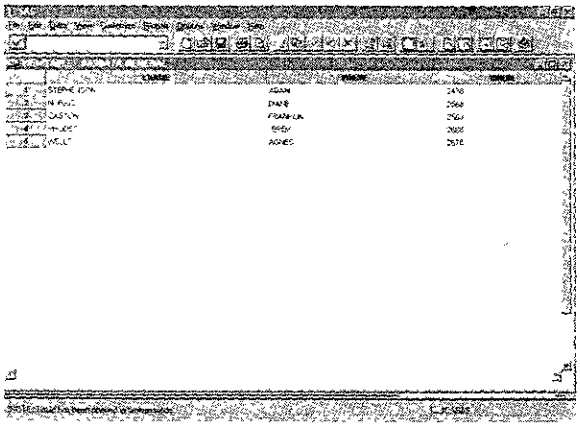
- Cualquier número de data sets pueden ser combinados con la sentencia MERGE.
- Los data sets entrantes deben estar ordenados por la (s) variable (es) que tienen en común y que será utilizada en la sentencia BY.
- El PDV y el data set resultante, contienen todas las variables presentes en los data sets entrantes a menos que se utilicen las opciones DROP = o KEEP =.
- Los data set deben estar ordenados por la (s) misma variable (s) que se utilizan en la sentencia BY.



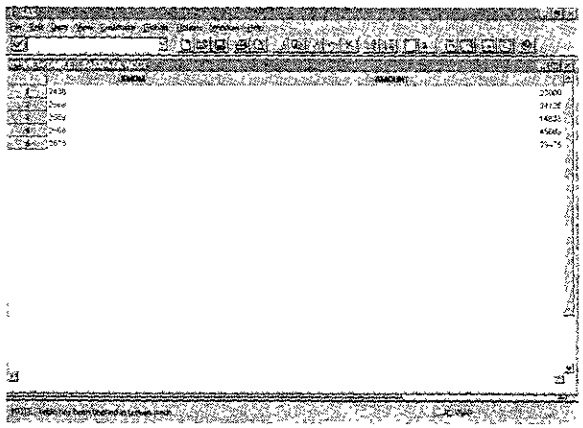

Las observaciones son leídas de acuerdo al orden de los valores de la variable utilizada en la sentencia BY y cuyos valores sean los mismos para ambas tablas.

Ejemplo: Utilizar un paso DATA para unir el data set GENERAL y el data set ACCT utilizando la variable IDNUM en la sentencia BY. Ambos data sets están ordenados por la variable IDNUM.

GENERAL



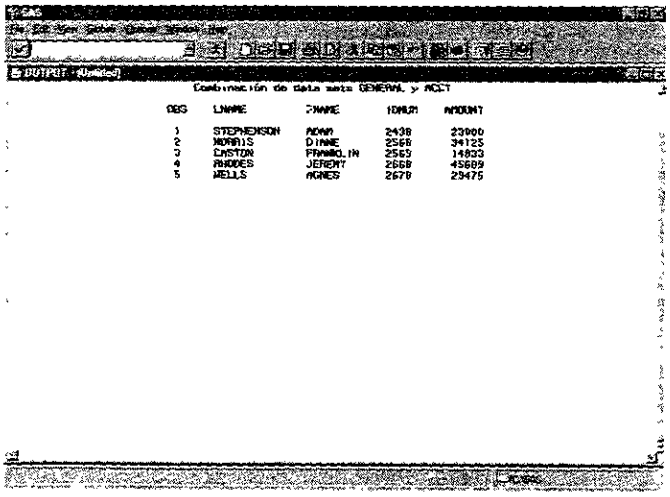
ACCT



Programa:

```
data combine;
   merge sasdata.general sasdata.acct;
   by idnum;
run;

proc print data=combine;
   title 'Combinación de data sets GENERAL y ACCT';
run;
```

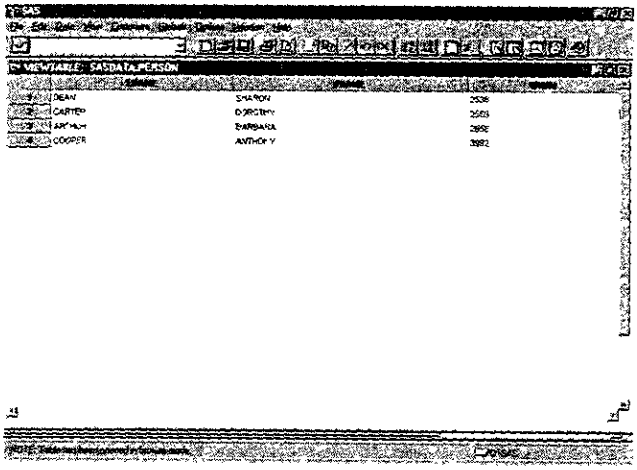
Resultado:



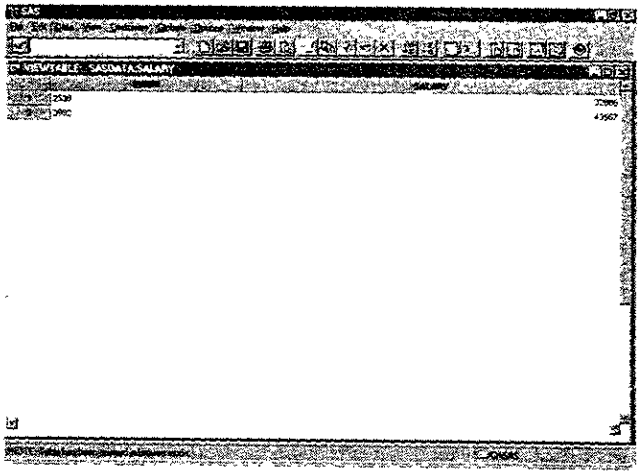
Los valores leídos con la sentencia MERGE son almacenados con valores nulos dentro del PDV cuando el valor de la variable utilizada en la sentencia BY es distinta en ambos data sets.

Ejemplo: Utilizar un paso DATA para unir los data sets PERSON y SALARY por IDNUM. Ambos data sets esta ordenados por la variable IDNUM.

PERSON



SALARY



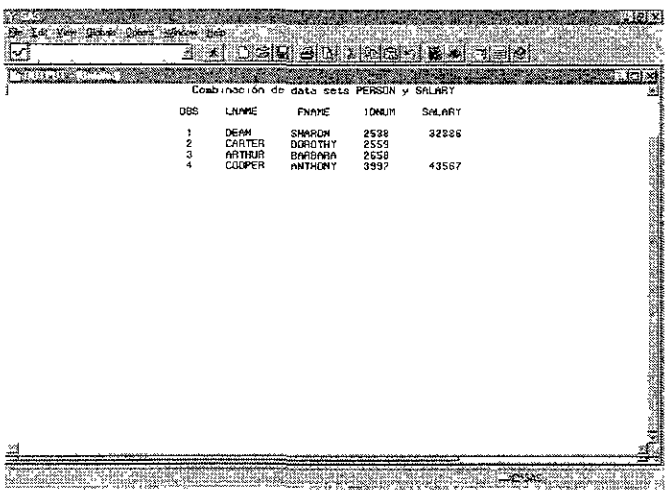
Programa:

```
data persal;
  merge sasdata.person sasdata.salary;
  by idnum;
run;

proc print data=persal;
   title 'Combinación de data sets PERSON y SALARY';
run;
```

**P D V**

| LNAME | FNAME | IDNUM | SALARY |
|---|---|---|---|
| | | | |

Resultado:



Combinación de data sets PERSON y SALARY

| OBS | LNAME | FNAME | IDNUM | SALARY |
|---|---|---|---|---|
| 1 | DEAN | SHARON | 2538 | 32386 |
| 2 | CARTER | DOROTHY | 2559 | |
| 3 | ARTHUR | BARBARA | 2658 | |
| 4 | COOPER | ANTHONY | 3997 | 43567 |

☞ *LA SENTENCIA MERGE PUEDE SER UTILIZADA CON MÁS DE UNA TABLA , SIN AMBARGO LA RECOMENDACION ES USARLO SOLO CON DOS TABLAS A LA VEZ (EN UN MISMO PASO DATA, LA COMBINACION DE MÁS TABLAS CON ESTA SENTENCIA PUEDE NO SER TAN EFICIENTE, EN CASO DE QUERER CONBINAR MÁS DE DOS TABLAS DE ESTA FORMA, LO RECOMENABLE ES UTLIZAR TANTOS PASOS DATA COMO SEA NECESARIO O RECURRIR AL PROC SQL (PARA MAYOR DETALLE, CONSULTAR MANUALES EN LA BIBLIOGRÁFIA)*

Cuando se leen múltiples data sets en un paso DATA. Se puede utilizar la opción IN= para detectar cual data set esta contribuyendo a la observación que se esta generando.

Forma general de la opción IN

*Data set* (**IN**=*variable*);

*Variable* es una variable numérica temporal con valores

0 Indica que el data set no contribuyó con la observación generada en el data set.

1 Indica que el data set contribuyó con la observación generada en el data set.

Ejemplo: Desplegar los valores de las variables IN= cuando se combinan data sets.

**A**

| ID |
|---|
| 1 |
| 2 |

**B**

| ID |
|---|
| 1 |
| 3 |

Programa:

```
data combine;
  merge sasdata.a(in=ina) sasdata.b(in=inb);
  by id;
run;

proc print data=combine noobs;
run;
```

**P D V**

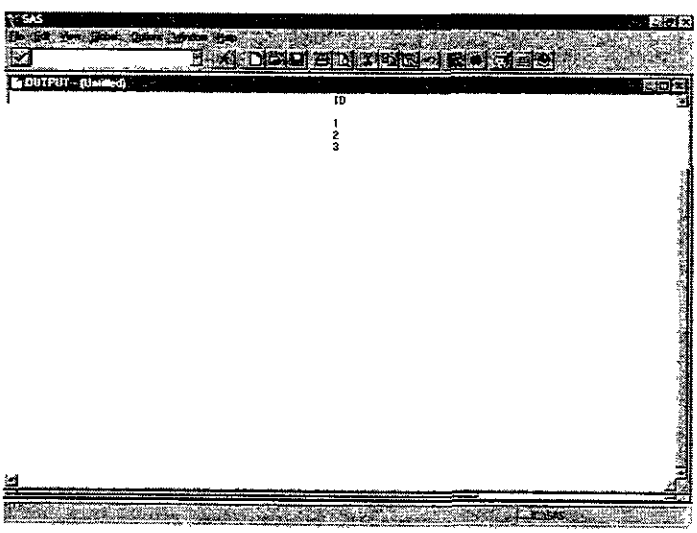
| ID | INA | INB | _N_ |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 0 | 2 | 2 |
| 0 | 1 | 3 | 3 |

☞ SE PUEDE NOTAR QUE LOS VALORES DE 1 EN AMBAS VARIABLES (INA E INB) SE DA SOLO EN EL CASO EN QUE EL VALOR DE LA VARIABLE UTILIZADA EN LA SENTENCIA BY COINCIDE EN AMBOS DATA SET. EN CUALQUIER OTRO CASO AMBAS VARIABLES TIENE VALORES DISTINTOS DEPENDIENDO DE DONDE VENGA EL VALOR QUE SE ESTA ALMACENANDO EN LA OBSERVACIÓN ACTUAL.

La variable _N_ indica el número de observación que se esta leyendo y es una variable de control interno que se genera siempre dentro del PDV y cuyo valor puede ser utilizado dentro de un programa.

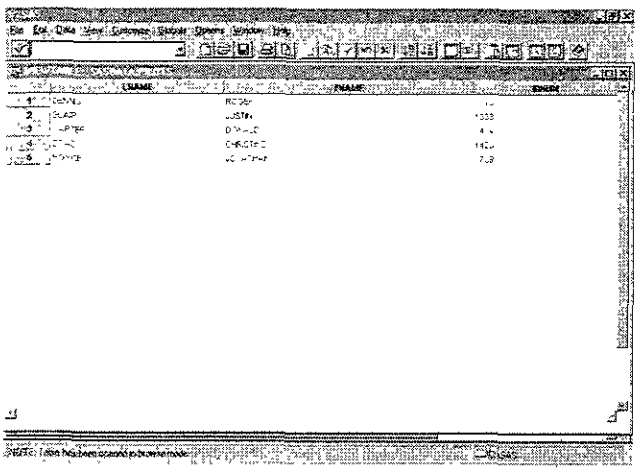
**NOTA:** Las variables INA e INB son creadas en el PDV pero no se escriben al data set de salida.

Resultado:

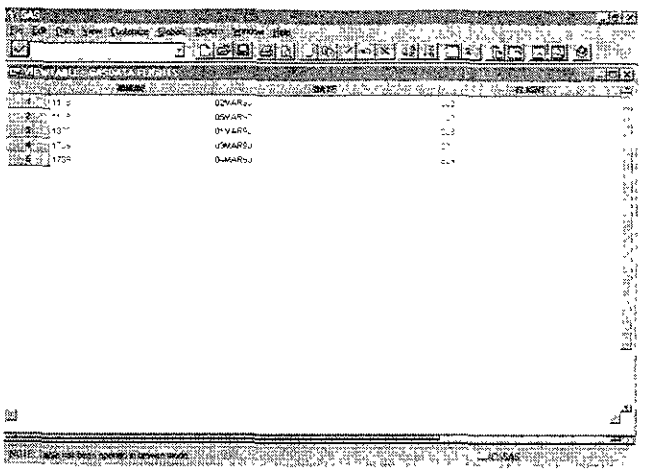


Ejemplo: Utilizar un paso DATA para combinar los data sets PILOTS y FLIGHTS por la variable IDNUM ambos data sets se encuentran ordenados por esta variable. El data set resultante debe contener solo las observaciones correspondientes a aquellos pilotos con vuelos asignados.

PILOTS



FLIGHTS

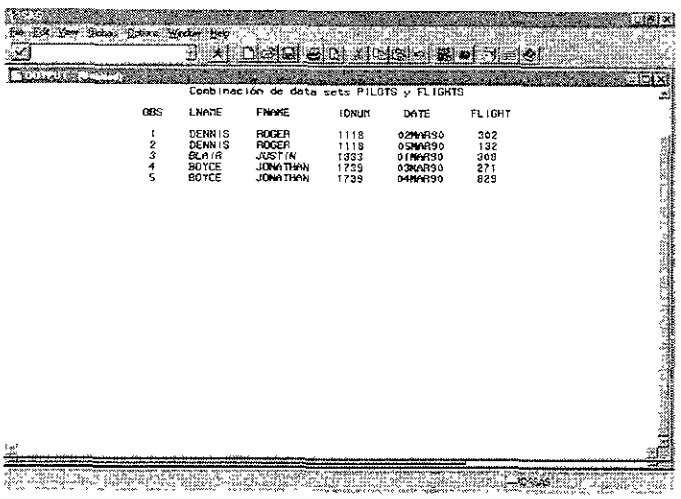


Programa:

```
data schedule;
   merge sasdata.pilots sasdata.flights(in=flying);
   by idnum;
   if flying=1;
run;

proc print data=schedule;
   title 'Combinación de data sets PILOTS y FLIGHTS';
run;
```

ES RECOMENDABLE UTILIZAR ESTE TIPO DE SENTENCIA **IF** Y LAS OPCIONES **IN** CON LA FINALIDAD DE ESCRIBIR Y PROCESAR HACIA EL DATA SET DE SALIDA, SOLO A QUELLOS REGISTROS QUE CUMPLAN CIERTAS CONDICIONES .

Resultado:



Combinación de data sets PILOTS y FLIGHTS

| OBS | LNAME | FNAME | IDNUM | DATE | FLIGHT |
|---|---|---|---|---|---|
| 1 | DENNIS | ROGER | 1118 | 02MAR90 | 302 |
| 2 | DENNIS | ROGER | 1118 | 05MAR90 | 132 |
| 3 | BLAIR | JUSTIN | 1333 | 01MAR90 | 369 |
| 4 | BOYCE | JONATHAN | 1739 | 03MAR90 | 271 |
| 5 | BOYCE | JONATHAN | 1739 | 04MAR90 | 829 |

# CAPITULO 5

# TRANSFORMACIÓN BÁSICA DE DATOS UTILIZANDO FUNCIONES

Conocer y utilizar las funciones básicas del lenguaje BASE SAS para transformar datos.

## 5.1 FUNCIONES SAS

Una función SAS es una rutina que regresa un valor determinado por los argumentos recibidos.

Las funciones en SAS están disponibles para:

- Ejecutar operaciones aritméticas (como la raíz cuadrada).
- Calcular estadísticas (como sumas y promedios).
- Manipular fechas SAS (como conocer el día de la semana).
- Ejecutar otras tareas.

La forma general de utilizar las funciones SAS es:

*Nombre de la función (argumento1, argumento2, . . . );*

Cada argumento debe ser separado del otro por una coma, la mayoría de las funciones aceptan argumentos que son:

- Constantes
- Variables
- Expresiones
- Funciones

Todas las funciones aritméticas en SAS como la función SUM (que suma los valores de los argumentos) ignoran los valores nulos, si es que existe alguno dentro de los valores que recibe como argumentos.

Forma general de utilizar la función SUM

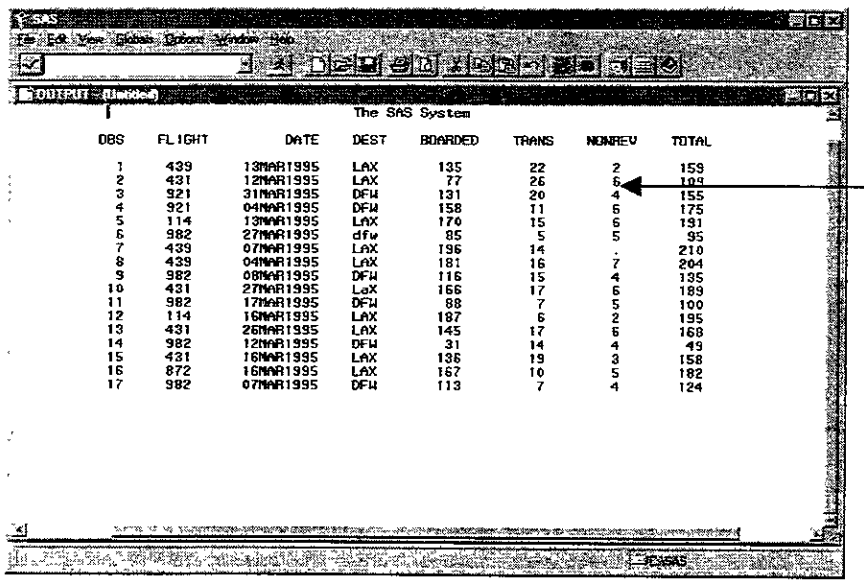
***SUM****(argumento1. argumento2, . . . );*

Si se aplica la función anterior dentro del programa, se puede observar que el resultado de la variable TOTAL no se ve afectado por los valores MISSING que recibe como argumentos.

Programa:

```
data dfwlax;
  infile 'c:\fund\dfwlax.dat';
   input @1 flight $3. @4 date mmddyy6. @18 dest $3.
         @34 boarded 3. @37 trans 3. @40 nonrev 3.;
   total=SUM(boarded,trans,nonrev),
run;

proc print data=dfwlax;
 format date date9.;
run;
```

☞ SIEMPRE QUE SEA POSIBLE, ES RECOMENDABLE UTILIZAR FUNCIONES DENTRO DE UN PROGRAMA, YA QUE CON ESTO SE EVITARÁ OBTENER ERRORES POR POSIBLES VALORES MISSING EN LOS CALCULOS.



The SAS System

| OBS | FLIGHT | DATE | DEST | BOARDED | TRANS | NONREV | TOTAL |
|---|---|---|---|---|---|---|---|
| 1 | 439 | 13MAR1995 | LAX | 135 | 22 | 2 | 159 |
| 2 | 431 | 12MAR1995 | LAX | 77 | 26 | 6 | 109 |
| 3 | 921 | 31MAR1995 | DFW | 131 | 20 | 4 | 155 |
| 4 | 921 | 04MAR1995 | DFW | 158 | 11 | 6 | 175 |
| 5 | 114 | 13MAR1995 | LAX | 176 | 15 | 6 | 191 |
| 6 | 982 | 27MAR1995 | dfw | 85 | 5 | 5 | 95 |
| 7 | 439 | 07MAR1995 | LAX | 196 | 14 | . | 210 |
| 8 | 439 | 04MAR1995 | LAX | 181 | 16 | 7 | 204 |
| 9 | 982 | 08MAR1995 | DFW | 116 | 15 | 4 | 135 |
| 10 | 431 | 27MAR1995 | LaX | 166 | 17 | 6 | 189 |
| 11 | 982 | 17MAR1995 | DFW | 88 | 7 | 5 | 100 |
| 12 | 114 | 16MAR1995 | LAX | 187 | 6 | 2 | 195 |
| 13 | 431 | 26MAR1995 | LAX | 145 | 17 | 6 | 168 |
| 14 | 982 | 12MAR1995 | DFW | 31 | 14 | 4 | 49 |
| 15 | 431 | 16MAR1995 | LAX | 136 | 19 | 3 | 158 |
| 16 | 872 | 16MAR1995 | LAX | 167 | 10 | 5 | 182 |
| 17 | 982 | 07MAR1995 | DFW | 113 | 7 | 4 | 124 |

Se observa que uno de los valores que recibe la función SUM es nulo. Sin embargo en el resultado, solo se consideran aquellas variables con valores distintos de nulos.

Algunas funciones toman:

- Varios valores en cualquier orden
- Un número especifico de argumentos en cierto orden

**NOTA:** Para funciones que toman varios valores en cualquier orden, se pueden ordenar los argumentos en una lista utilizando la palabra OF seguida de la lista de valores:

***SUM** (x1, **of** x10 - x20, x25) ;*

A continuación se describen algunas de las funciones más utilizadas para la manipulación de datos. Existen muchas otras funciones que no serán descritas dentro de este trabajo, para mayor información y detalle, puede consultar alguno de los libros incluidos en la bibliografía.

## 5.2 FUNCIONES PARA MANEJO DE DATOS NUMÉRICOS

La manipulación de valores numéricos se puede realizar utilizando algunas de las funciones que SAS tiene para el manejo de datos de este tipo, como :

**FUNCIONES PARA TRUNCAR**

| | |
|---|---|
| **ROUND** | Redondea los valores de acuerdo al número de decimales especificados como argumento. |
| **CEIL** | Regresa el entero más cercano, mayor ó igual al valor del argumento. |
| **INT** | Regresa la parte entera del valor que recibe como argumento. |

Sintaxis:

**ROUND** (*argumento1, argumento2*);

*argumento1* Valor que será redondeado.
*argumento2* Número de decimales en el resultado.

**CEIL** (*argumento1*);

*argumento1* Valor numérico base para obtener el entero más cercano.

**INT** (*argumento1*);

*argumento1* Valor numérico base para obtener la parte entera.

**NOTA**: Todos los que reciben estas funciones deben ser numéricos.

Ejemplo: Comparar los resultados de las funciones ROUND, CEIL e INT

Se tiene el siguiente DATA SET

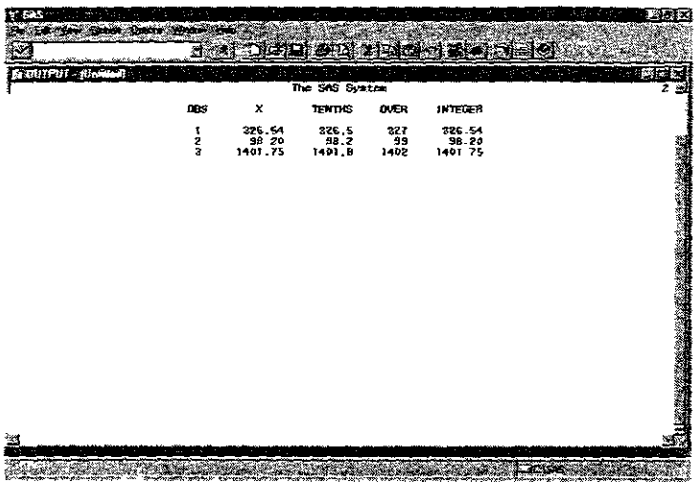
SASDATA.TRUNC

| X |
|---|
| 326.54 |
| 98.2 |
| 1401.75 |

Programa:

```
data chgnum;
   set sasdata.trunc;
   tenths=round(x,.1);
   over=ceil(x);
   integer=int(x);
proc print data=chgnum;
run;
```

Resultado:




**FUNCIONES QUE CALCULAN ESTADISTICAS**

Se pueden utilizar funciones para calcular muestras estadísticas basadas en un grupo de variables. Algunas de las funciones que se pueden utilizar para este objetivo son:

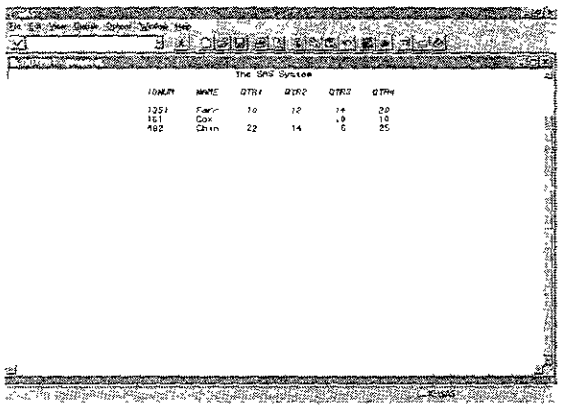
| | |
|---|---|
| **SUM** | Suma los valores de los argumentos. |
| **MEAN** | Media aritmética (promedio) de los valores. |
| **MIN** | Valor mínimo. |
| **MAX** | Valor máximo. |
| **VAR** | Varianza de los valores. |
| **STD** | Desviación estándar de los valores. |

Estas funciones

- Aceptan cualquier argumento en cualquier orden.
- Utilizan los mismos argumentos que los procedimientos estadísticos de SAS.
- Ignoran valores nulos.

Ejemplo: El Data Set SASDATA.DONATE contiene las contribuciones trimestrales realizadas por los empleados. Imprimir un reporte que despliegue el total y el promedio de las contribuciones realizadas para cada empleado.

Contenido del Data Set SASDATA.DONATE

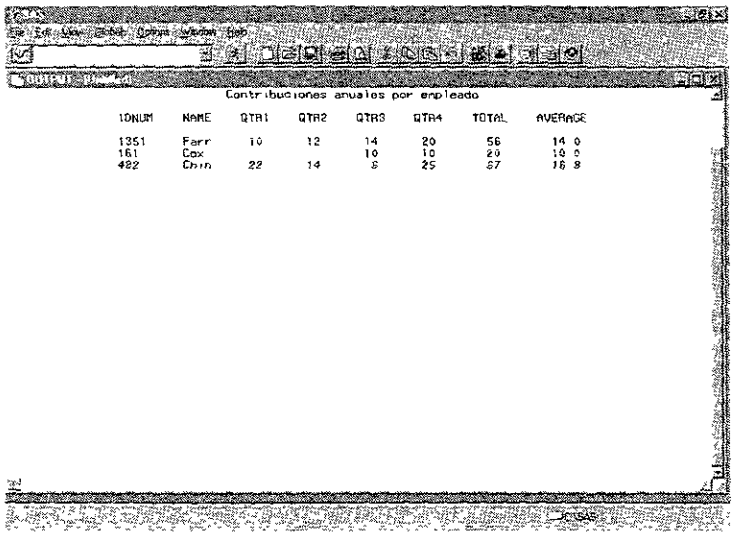


Programa:

```
data stats;
   set sasdata.donate(drop=name);
   total=sum(qtr1,qtr2,qtr3,qtr4);
   average=round(mean(of qtr1-qtr4),.1)
proc print data=stats;
   title 'Employee Contributions for the Year';
run;
```

Para este caso se utilizaron las funciones SUM, ROUND y MEAN. Vale la pena notar que en la segunda el argumento que se utiliza es otra función (MEAN).

Resultado:



Contribuciones anuales por empleado

| IDNUM | NAME | QTR1 | QTR2 | QTR3 | QTR4 | TOTAL | AVERAGE |
|---|---|---|---|---|---|---|---|
| 1351 | Farr | 10 | 12 | 14 | 20 | 56 | 14.0 |
| 161 | Cox | | | 10 | 10 | 29 | 10.0 |
| 482 | Chin | 22 | 14 | 8 | 25 | 87 | 16.8 |

**FUNCION PARA GENERAR NÚMEROS ALEATORIOS**

Se pueden generar números aleatorios utilizando una función numérica llamada RANUNI, esta regresa un número entre 0 y 1 desde una distribución uniforme.

Forma general de la función RANUNI:

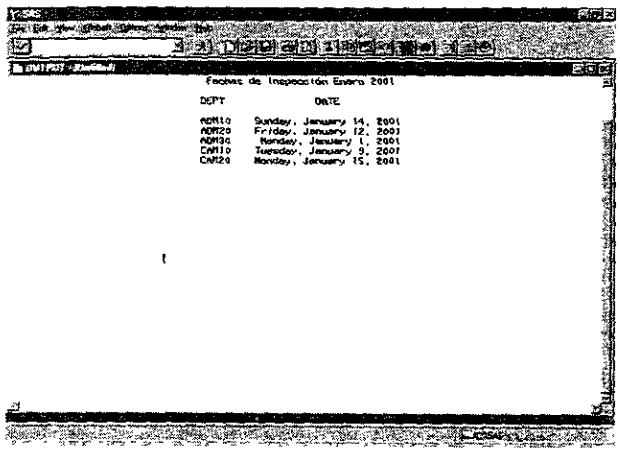
**RANUNI** (*semilla*);

Ejemplo: Cada departamento será inspeccionado por sorpresa durante los primeros quince días del mes. Crea un data set que contenga una observación por departamento con la fecha de inspección sorpresa durante Enero del 2001.

Programa:

```
data inspect(keep=dept date);
   set sasdata.dept;
   day =ceil(ranuni(0)*15);
  date=mdy(6,day,2001);

proc print data=inspect noobs;
   format date weekdate.;
   title 'Inspection Dates for June';
run;
```

Resultado:




SIEMPRE QUE SE DESEA REALIZAR ALGÚN CALCULO ESTADÍSTICO SE RECOMIENDA VERIFICAR DENTRO DE TODAS LAS FUNCIONES QUE SAS OFRECE PARA SABER SI EXISTE ALGUNA QU REALIZA EL CALCULO DE NUESTRO INTERÉS, ANTES DE DUPLICAR ESFUERZOS TRATANDO DE PROGRAMAR ESTAS FUNCIONES POR UNO MISMO. LA LISTA DE FUNCIONES DISPONIBLES, PUEDE CONSULTARSE TECLEANDO EL COMANDO *FUNCTIONS* DESDE LA LINEA DE COMANDOS.

## 5.3 FUNCIONES PARA MANEJO DE DATOS CARÁCTER

Para la descripción de las siguientes funciones, será de gran utilidad plantear un problema, que nos ayude a *comprender claramente los usos de estas.*

Algunas de las funciones que se discutirán en esta sección son:

| | |
|---|---|
| **INDEX** | Busca un carácter o cadena de caracteres dentro de un valor, regresa el # de posición donde encontró el valor buscado. |
| **LEFT** | Alinea los valores a la Izquierda |
| **RIGHT** | Alinea los valores a la Derecha. |
| **SCAN** | Busca por # de palabra utilizando separadores. |
| **SUBSTR** | Extrae una subcadena de una cadena. |
| **TRIM** | Elimina espacios en blanco a la derecha. |
| **UPCASE** | Convierte el valor a Mayúsculas. |
| **LOWCASE** | Convierte el valor a Minúsculas. |

Además de un operador que nos permitirá realizar la concatenación de valores de tipo carácter.

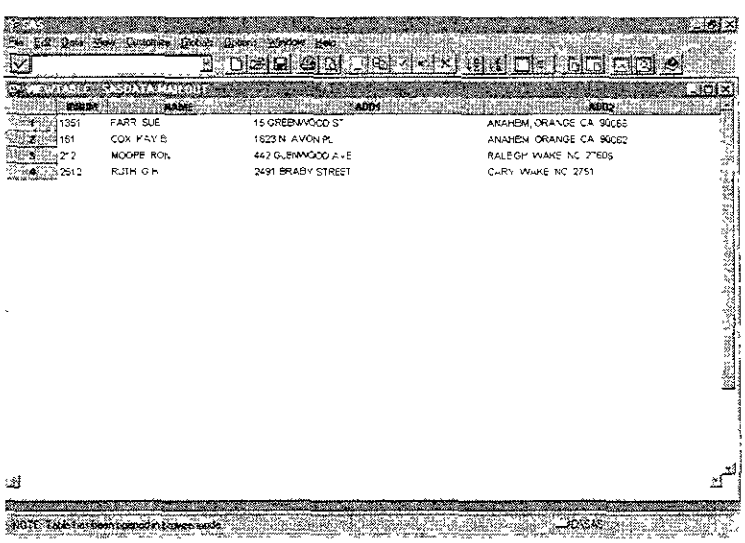
A continuación la descripción de nuestro ejemplo y la solución utilizando las funciones antes descritas.

*APLICACIÓN PARA ENVÍO DE CORREOS*

Una compañía Manufacturera cuenta con un data set SASDATA.MAILOUT con los datos actuales de los empleados. El departamento de Recursos Humanos, quiere utilizar este data set para crear otro adecuado para el envío de correos a los empleados.

> **NOTA**: *La variable IDNUM es una variable de tipo carácter cuyo ultimo valor representa el sexo* del empleado. **1** Mujer y **2** Hombre.

Contenido del data set SASDATA.MAILOUT



El resultado que se desea obtener es la siguiente tabla:

Data Set **LABELS**

| FULLNAME | ADD1 | ADD2 |
|---|---|---|
| Ms. Sue Farr | 15 Greenwood St. | Anaheim, CA 90066 |
| Ms. Kay B Cox | 1623 N. Avon Pl. | Anaheim, CA 90062 |
| Mr. Ron Moore | 442 Glenwood Ave. | Raleigh, NC 27606 |
| Mr. G H Ruth | 2491 Brady Street | Cary, NC 27511 |

Primero se generara el campo FULLNAME, el cual debe incluir titulo del empleado "Mr." o "Ms." Como se describió anteriormente, el sexo del empleado esta definido por el último número del campo IDNUM el cual puede estar formado por 4, 3 ó menos caracteres.

Como se puede observar en el data set SASDATA.MAILOUT el campo IDNUM se encuentra alineado a la izquierda, lo primero que haremos será alinear la información a la derecha para así poder tomar el último carácter de este campo y generar a partir de este el valor de una nueva variable llamada TITLE que indique el sexo del empleado. Para este proceso se utilizaran las funciones RIGHT (para alinear) y SUBSTR (para extraer el último dígito del campo).

Ejemplo:

**SUBSTR**(*string, start, length*)

*String* Es el valor carácter, constante o expresión, de donde se extraerá la subcadena.

*Start* Es la posición a partir de la cual se comenzara a extraer la subcadena.

*Length* Es la longitud (número de caracteres) que serán leídos desde la posición *start.*

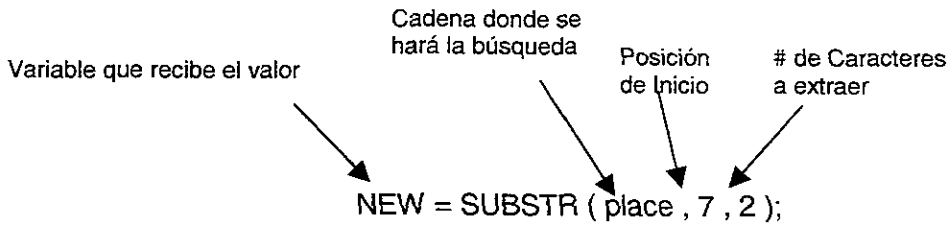
Se tiene la variable PLACE de tipo carácter de 14 posiciones.

**PLACE**
**$ 14**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **C** | **A** | **R** | **Y** | **,** | | **N** | **C** | | **2** | **7** | **5** | **1** | **1** |

Se desea extraer la información correspondiente a la posición 7 del campo (el estado NC).

La sentencia SUBSTR correcta sería:




El resultado NEW = ' NC '

**RIGHT** (*argumento*)

*Argumento* Cualquier valor de tipo carácter, constante o alguna expresión

| IDNUM |
|---|
| $ 4 |
| 1 6 1 |

IDNUM_R = RIGHT( IDNUM);

| iDNUM _R |
|---|
| $ 4 |
| 1 6 1 |

☞ SI LA LONGITUD DE LA VARIABLE QUE RECIBE EL VALOR NO ES PREDEFINIDA, ESTA TOMARÁ EL MISMO VALOR (LONGITUD) QUE EL ARGUMENTO DE LA FUNCIÓN

Utilizando esta información en el programa anterior este quedaría como sigue:

Programa :

```
data labels;
   set sasdata.mailout;
   length title $ 3;
   if substr(right(idnum),4)='1' then title='Ms.';
   else if substr(right(idnum),4)='2' then title='Mr.';
run;
```

El resultado de este programa es un data set llamado LABELS, que además de contener todas las variables que vienen de SASDATA.MAILOUT, ahora también contiene la nueva variable TITLE.

La siguiente tarea dentro de nuestro ejemplo es tomar el nombre del empleado y transformarlo.

| | **NAME** |
|---|---|
| Forma actual del campo NAME | Farr, Sue |
| Forma requerida para el campo FULLNAME | Sue Farr |

Para realizar este cambio se requiere de la función SCAN, la cual regresa la *n*-ésima palabra de una cadena de un valor de tipo carácter tomando como base un separador.

Sintaxis:

**SCAN**(*string, n, delimiters*)

*String* Puede ser un valor carácter constante, variable o expresión.
*N* Especifica la *n*-ésima palabra a extraer desde la cadena (*string*).
*Delimiters* Define el carácter que delimita (separa) las palabras.

| **TEXT**<br>**$ 40** |
|---|
| A TEXT STRING WITH BLANKS AS DELIMITERS |

1. ¿Cuántas palabras se pueden encontrar en la variable TEXT utilizando el blanco como delimitador? [1]

2. ¿Cuál sería el resultado de SCAN (TEXT, 4, ' ')?

| **TEXT**<br>**$ 40** |
|---|
| A :TEXT STRING :WITH BLANKS AS: DELIMITERS |

3. ¿Cuántas palabras se pueden encontrar en la variable TEXT utilizando los dos puntos como delimitador? [2]

4. ¿Cuál sería el resultado de SCAN (TEXT, 4, ' : ') ?

Si un delimitador no es especificado dentro de la función SCAN todos los siguientes caracteres servirán como separadores por default.

Blanco . < ( + & ! $ * ) ; - / , %

Algunas consideraciones que deben tomarse en cuenta al utilizar la función SCAN:

- Cualquier carácter o conjunto de caracteres pueden servir como delimitador
- La longitud de la variable que recibe el valor es de 200 caracteres a menos que se defina previamente con una longitud distinta(utilizando la sentencia LENGHT).
- Delimitadores antes de la primer palabra no tiene efecto.
- Si existen dos o más delimitadores juntos, serán tratados como un solo delimitador.
- Si el número de palabras en la cadena es menos a la n-ésima palabra, la función SCAN regresara un valor nulo.

☞ NO SE RECOMIENDA UTILIZAR LETRAS COMO DELIMITADORES.

Dentro del ejemplo la función SCAN servirá para generar un par de variables FIRSTN y LASTN las cuales contendrán el nombre y apellido del empleado sucesivamente. Estas nuevas variables serán unidas más adelante con la variable TITLE para formar la variable FULLNAME.

[1] Respuesta: 1 ( 7 )
2 ( WITH )
[2] Respuesta: 3 ( 4 )
4 ( DELIMITERS )

Programa :

```
data labels;
   set sasdata.mailout;

   /* Generación de la variable TITLE */
   length title $ 3;
   if substr(right(idnum),4)='1' then title='Ms.';
   else if substr(right(idnum),4)='2' then title='Mr.';

   /* Generación de las variables FIRSTN y LASTN */
   length firstn $ 7 lastn $ 7;
   firstn = scan(name,2,',');
   lastn = scan(name,1,',');

run;
```

Para la unión de las variables se utilizaran los operadores de concatenación.
La concatenación es una función que nos permita unir dos cadenas de caracteres.

Dependiendo de los caracteres disponibles en los teclados, los símbolos para concatenación pueden ser:

- Dos barras verticales ( || )
- Dos barras verticales partidas ( ¦¦ )
- Dos signos de exclamación ( !! )

Forma general de utilizar el operador de concatenación

***string1* !! *string2***

*strig1* y *strig2* Son caracteres constantes, variables o expresiones.

La sentencia dentro del programa sería: *FULLNAME = firstn !! lastn;*

| FIRSTN<br>$ 7 | LASTN<br>$ 7 | FULLNAME<br>$ 14 |
|---|---|---|
| Sue | Farr | Sue Farr |

¿Cuál es la longitud de FULLNAME?

**NOTA**: Si la variable que recibe el valor no tiene una longitud predefinida, su longitud es la suma de las longitudes de las cadenas que se concatenan.
La operación de concatenación no elimina espacios en blanco.

☞ ES RECOMENDABLE DEFINIR PREVIAMENTE Y DE LA MANERA MÁS EXACTA POSIBLE LA LONGITUD DE LA NUEVA VARIABLE UTILIZANDO LA SENTENCIA *LENGTH*.

La función TRIM remueve los espacios en blanco a la derecha del argumento que recibe.

**TRIM**(*argumento*)

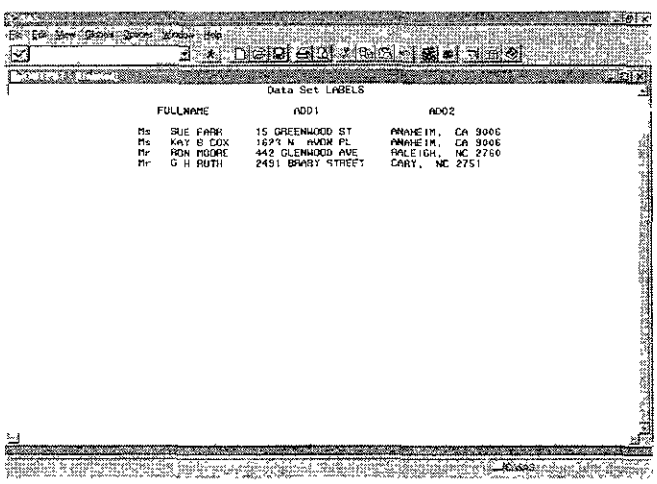
*Argumento* Es un valor carácter constante, variable o expresión.

| FIRSTN<br>$ 7 | LASTN<br>$ 7 | FULLNAME<br>$ ? |
|---|---|---|
| Sue | Farr | |

Programa:

```
data labels;
   set sasdata.mailout;

   /* Generación de la variable TITLE */
   length title $ 3;
   if substr(right(idnum),4)='1' then title='Ms.';
   else if substr(right(idnum),4)='2' then title='Mr.';

   /* Generación de las variables FIRSTN,LASTN y FULLNAME */
   length firstn $ 7 lastn $ 7;
   firstn = scan(name,2,',');
   lastn = scan(name,1,',');
   fullname=title!!' '!!trim(firstn)!!' '!!lastn;

   /* Otras modificaciones */
   length city $ 33 state $ 3 zip $ 5;
   city=scan(add2,1,',');
   state=scan(add2,3,',');
   zip=scan(add2,4,',');
   add2=trim(city)!!', '!!state!!zip;
run;

proc print data=labels noobs;
   var fullname add1 add2;
   title 'Data Set LABELS';
run;
```

Resultado



*APLICACIÓN DE BUSQUEDAS*

El data set SASDATA.HISTORY contiene información acerca de la historia laboral de cada empleado.

La información histórica almacenada en la variable JOBHIST consiste de:

- Cada puesto que el empleado a desempeñado.
- El año en el que el empleado comenzó en ese puesto.

Ejemplo: Crear un reporte que muestre a todos los empleados que han ocupado el puesto de Contador (Acct) y el año en que comenzaron a desempeñarlo.

Para determinar quien ha trabajado como contador, se requiere buscar dentro de la variable JOBHIST la cadena **Acct**.

La función INDEX busca dentro de una cadena de caracteres la locación del carácter especificado, el valor que regresa es la posición donde localizo el carácter buscado.

**INDEX**(*target, value*)

*Target* Es el valor carácter, constante o expresión donde buscar.

*Value* Es el valor carácter a ser buscado dentro del *target*.

La función INDEX regresa,

- La posición inicial del primer valor de *value* dentro de *target* si es que *value* es encontrado.
- Un 0 si el valor no es encontrado

Ejemplo: TEXT = " Este texto contiene una frase clave ";

N=INDEX(text,"frase");

| **TEXT**<br>**$ 40** | **N**<br>**Num 8** |
|---|---|
| Este texto contiene una frase clave | |

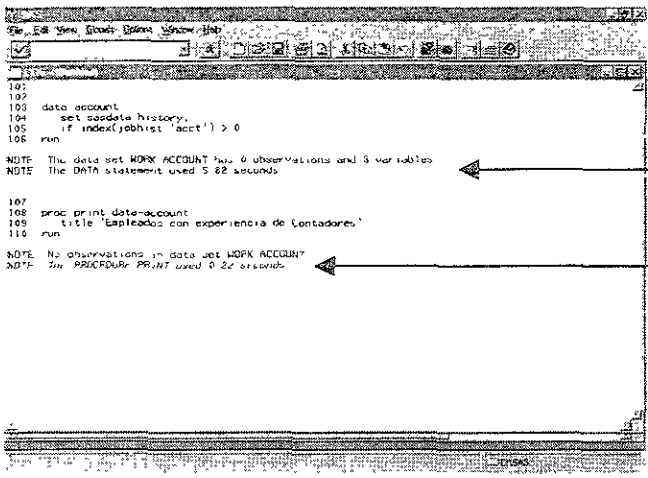
Posición 25

**NOTA:** La búsqueda de *value* dentro de *target* es tan literal que son consideradas letras mayúsculas, minúsculas, espacios en blanco (antes o después), etc.
El resultado arrojado por la función INDEX es un dato de tipo numérico.

Programa:

```
data account;
   set sasdata.history;
   if index(jobhist,'acct') > 0;
run;
proc print data=account;
   title 'Empleados con experiencia de Contadores';
run;
```

Resultado:




*El LOG muestra que el data set* resultante no contiene observaciones. esto es debido a que la comparación del valor ACCT se realizó en minúsculas y los datos dentro del data set están en mayúsculas.
Esto reitera la importancia de realizar las búsquedas de la forma más exacta posible.

Para resolver este problema existen dos soluciones,

1 Realizar la comparación con mayúsculas.
2. Utilizar las funciones UPCASE y LOWCASE.

Por supuesto que la solución más adecuada es la segunda, ya que de esta forma se asegura que el valor contra el cual se compara es el correcto no importando como se encuentre capturado dentro del data set.

Las funciones UPCASE y LOWCASE

- Convierten todas las letras del argumento en mayúsculas o minúsculas (según la función).
- No tienen efecto sobre números o caracteres especiales.

Ejemplo: Utilizar la función UPCASE para buscar todas las formas del valor **Acct.** Extraer además el año en que el empleado desempeño el puesto, incluirlos en un reporte.

Programa

```
data account(drop=location jobhist);
   set sasdata.history;
   location=index(upcase(jobhist),'ACCT');
   if location > 0;
   length year $ 4;
   year=substr(jobhist,location+5,4);
run;
proc print data=account;
   title 'Empleados con experiencia de Contadores';
run;
```

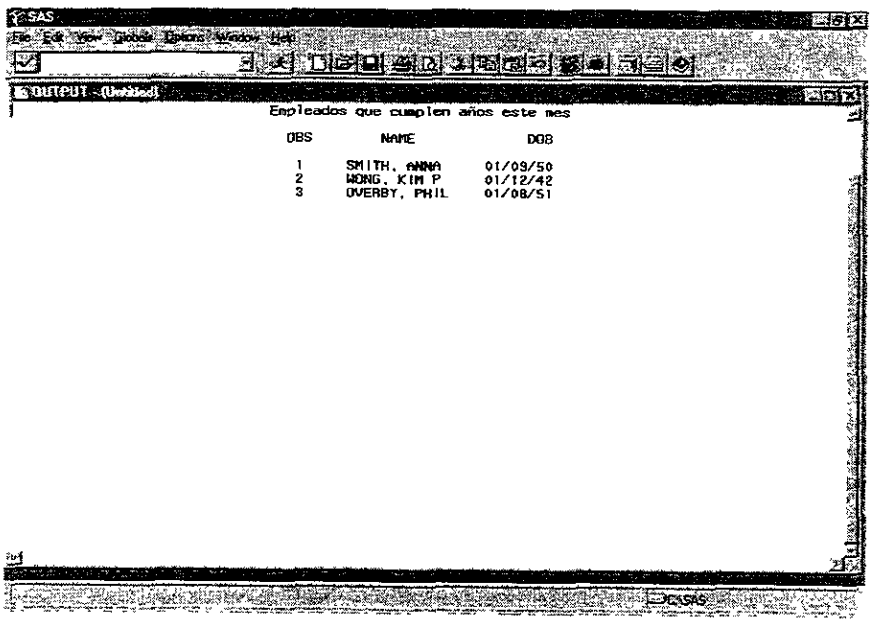
El data set SASDATA.STAFF contiene información acerca de las fechas de cumpleaños de los empleados de una Compañía Manufacturera.

Ejemplo: Generar un reporte que despliegue una lista con todos los empleados que cumplen años durante este mes. Este programa fue ejecutado el 1 de Enero del 2001.

Programa:

```
data birthday;
   set sasdata.staff(keep=name dob);
   dubmonth=month(dob);
   curmonth=month(today());
   if dobmonth=curmonth;
proc print data=birthday;
   title 'Employee who have birthadys this month';
   format dob mmddyy8.;
run;
```

Resultado:



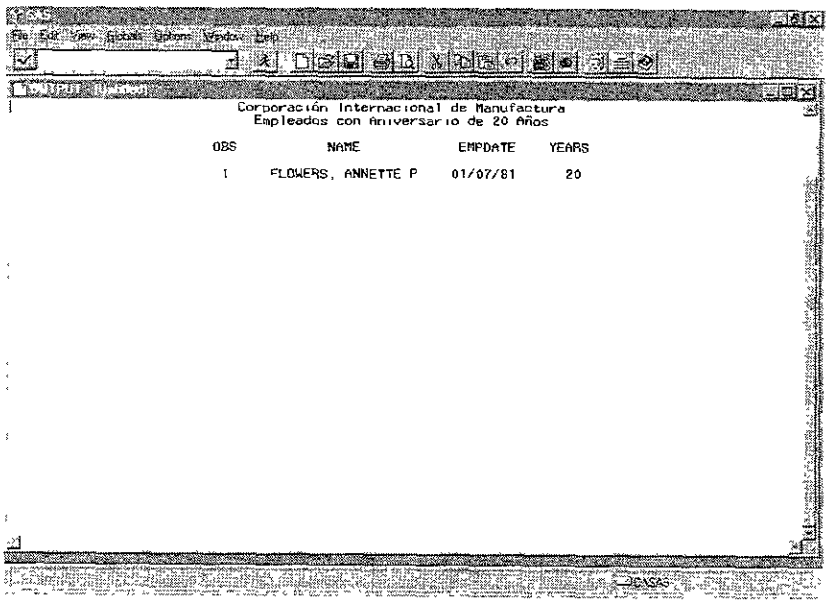
El data set SASDATA.MASTER contiene información acerca de las fechas en las cuales los empleados fueron contratados.

Ejemplo: Genera un reporte que despliegue una lista de todos los empleados que cumplen 20 años de servicio durante este mes. Este programa fue ejecutado el 1 de Enero del 2001.

Programa:

```
data anniver;
   set sasdata.master(keep=name empdate);
   years=intck('year',empdate,today());
   if years=20 and month(empdate)=month(today());
proc print data=birthday;
   title 'Corporación Internacional de Manufactura';
   title2 'Empleados con Aniversario de 20 Años';
run;
```

Resultado:



# CAPITULO 6

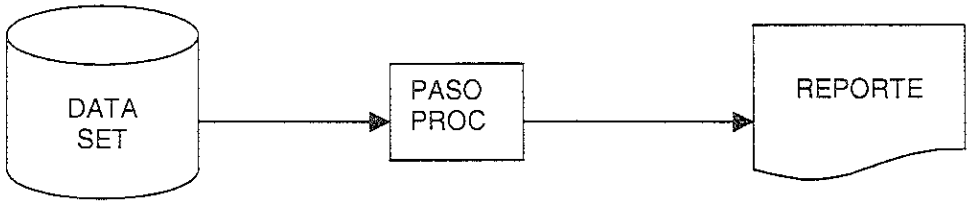
# VISUALIZACIÓN BÁSICA DE DATOS (UTILIZANDO PROCEDIMIENTOS SAS)

Aprender a generar reportes básicos para presentar información utilizando procedimientos del lenguaje BASE SAS.

## 6.1 REPORTES

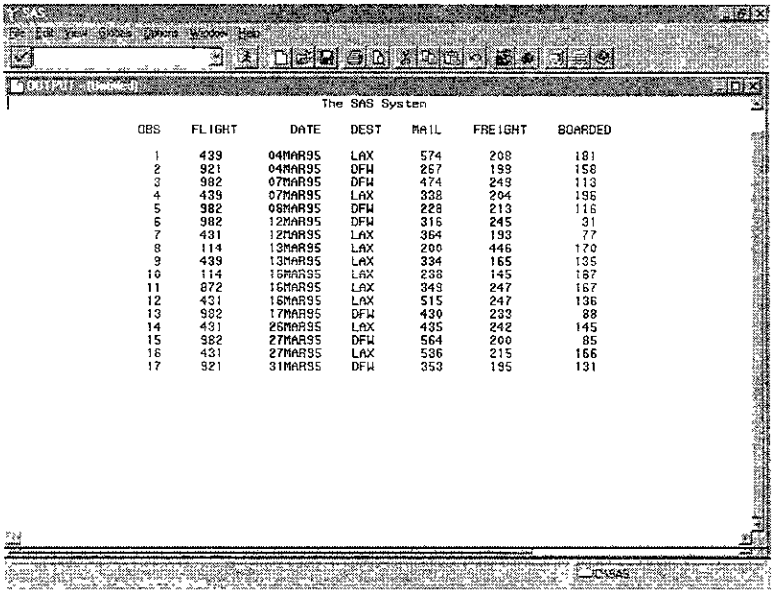
Una vez que los datos han sido accedidos, es conveniente realizar una revisión para saber si la carga se llevo a cabo correctamente. Con este objetivo, se pueden utilizar múltiples procedimientos disponibles dentro de SAS, como los que se describen a continuación.

Se pueden utilizar varios tipos de procedimientos para generar reportes desde un data set.




El tipo más común de reporte es un reporte de tipo lista que contiene una línea por cada observación en el data set. Este tipo de reportes son típicamente generados utilizando el procedimiento PRINT. Un reporte de este tipo puede ser generado utilizando muy pocas líneas de código.

Ejemplo de un reporte de tipo lista.



The SAS System

| OBS | FLIGHT | DATE | DEST | MAIL | FREIGHT | BOARDED |
|---|---|---|---|---|---|---|
| 1 | 439 | 04MAR95 | LAX | 574 | 208 | 181 |
| 2 | 921 | 04MAR95 | DFW | 267 | 199 | 158 |
| 3 | 982 | 07MAR95 | DFW | 474 | 249 | 113 |
| 4 | 439 | 07MAR95 | LAX | 338 | 204 | 196 |
| 5 | 982 | 08MAR95 | DFW | 228 | 213 | 116 |
| 6 | 982 | 12MAR95 | DFW | 316 | 245 | 31 |
| 7 | 431 | 12MAR95 | LAX | 364 | 193 | 77 |
| 8 | 114 | 13MAR95 | LAX | 200 | 446 | 170 |
| 9 | 439 | 13MAR95 | LAX | 334 | 165 | 135 |
| 10 | 114 | 15MAR95 | LAX | 238 | 145 | 187 |
| 11 | 872 | 16MAR95 | LAX | 348 | 247 | 167 |
| 12 | 431 | 16MAR95 | LAX | 515 | 247 | 136 |
| 13 | 982 | 17MAR95 | DFW | 430 | 233 | 88 |
| 14 | 431 | 25MAR95 | LAX | 435 | 242 | 145 |
| 15 | 982 | 27MAR95 | DFW | 564 | 200 | 85 |
| 16 | 431 | 27MAR95 | LAX | 536 | 215 | 166 |
| 17 | 921 | 31MAR95 | DFW | 353 | 195 | 131 |

La mayoria de los reportes pueden ser mejorados agregando sentencias y opciones para generar:

- Títulos y pies de página.
- Encabezados descriptivos para las columnas.
- Formato a los valores.
- Totales por columna.
- Subtotales por columna.
- Salto de página cuando los subgrupos cambian.

## 6.2 PROC PRINT

Para producir un reporte de tipo lista.

- Utilizar el procedimiento PRINT.
- Especificar el data set.

Forma General de utilizar el procedimiento PRINT.

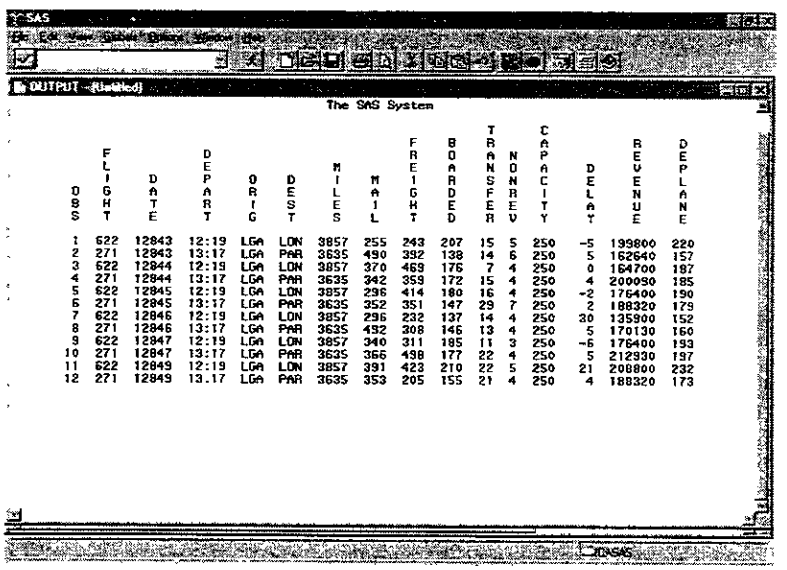
**PROC PRINT** DATA = *data set*;
**RUN**;

Ejemplo: Utiliza un procedimiento PRINT para generar un reporte de tipo lista del data set SASDATA.LONPAR.

Programa:

```
proc print data=SASDATA.LONPAR;
run;
```

Resultado:



The SAS System

| OBS | FLIGHT | DATE | DEPART | ORIG | DEST | MILES | MAIL | FREIGHT | BOARDED | TRANSFER | NONREV | CAPACITY | DELAY | REVENUE | DEPLANE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 622 | 12843 | 12:19 | LGA | LON | 3857 | 255 | 243 | 207 | 15 | 5 | 250 | -5 | 199800 | 220 |
| 2 | 271 | 12843 | 13:17 | LGA | PAR | 3635 | 490 | 392 | 138 | 14 | 6 | 250 | 5 | 162640 | 157 |
| 3 | 622 | 12844 | 12:19 | LGA | LON | 3857 | 370 | 469 | 176 | 7 | 4 | 250 | 0 | 164700 | 187 |
| 4 | 271 | 12844 | 13:17 | LGA | PAR | 3635 | 342 | 359 | 172 | 15 | 4 | 250 | 4 | 209080 | 185 |
| 5 | 622 | 12845 | 12:19 | LGA | LON | 3857 | 296 | 414 | 180 | 16 | 4 | 250 | -2 | 176400 | 190 |
| 6 | 271 | 12845 | 13:17 | LGA | PAR | 3635 | 352 | 351 | 147 | 29 | 7 | 250 | 2 | 188320 | 179 |
| 7 | 622 | 12846 | 12:19 | LGA | LON | 3857 | 296 | 232 | 137 | 14 | 4 | 250 | 30 | 135900 | 152 |
| 8 | 271 | 12846 | 13:17 | LGA | PAR | 3635 | 432 | 308 | 146 | 13 | 4 | 250 | 5 | 170130 | 160 |
| 9 | 622 | 12847 | 12:19 | LGA | LON | 3857 | 340 | 311 | 185 | 11 | 3 | 250 | -6 | 176400 | 193 |
| 10 | 271 | 12847 | 13:17 | LGA | PAR | 3635 | 366 | 438 | 177 | 22 | 4 | 250 | 5 | 212930 | 197 |
| 11 | 622 | 12849 | 12:19 | LGA | LON | 3857 | 391 | 423 | 210 | 22 | 5 | 250 | 21 | 200800 | 232 |
| 12 | 271 | 12849 | 13:17 | LGA | PAR | 3635 | 353 | 205 | 155 | 21 | 4 | 250 | 4 | 188320 | 173 |

- Por default una columna con el número de las observaciones es impreso en la columna de la izquierda.
- Todas las variables del DATASET son impresas en el orden en el cual fueron almacenadas en el DATASET.
- Sin la opción DATA =, el procedimiento PRINT imprime el DATASET creado más recientemente.

Ejemplo:

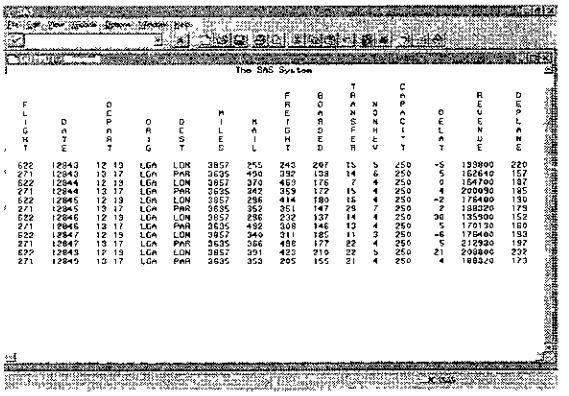
```
proc print data=SASDATA.LONPAR;
run;
```

La columna con el número de observación se puede eliminar utilizando la opción NOOBS dentro de la sentencia PROC PRINT.

Ejemplo:

```
proc print data=SASDATA.LONPAR noobs;
run;
```

Resultado:



Se pueden seleccionar las variables y controlar el orden en el cual son impresas utilizando la sentencia VAR.

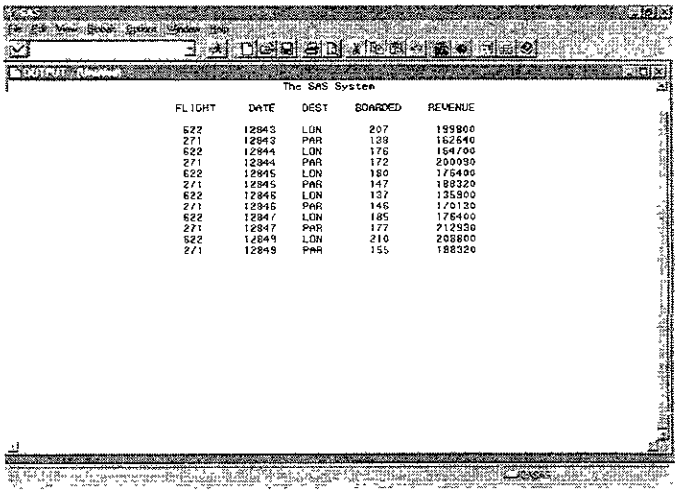
Forma General la sentencia VAR.

**VAR** *variables*;

Ejemplo:

```
proc print data=SASDATA.LONPAR noobs;
  var flight date dest boarded revenue;
run;
```

Resultado:




Para imprimir solo algunas observaciones, se puede utilizar la sentencia WHERE.

La sentencia WHERE

- Tiene efecto solo durante el PROC step.
- Puede ser utilizado en más pocedimientos SAS.

Forma General la sentencia WHERE.

**WHERE** *where_expression*;

La *where_expression* contiene operadores y operandos los cuales formaran un conjunto de instrucciones.

Los operandos pueden ser:

- Nombres de variables (carácter o numérica)
- Constantes

Hay varios tipos de operadores incluyendo:

- Comparación
- Lógicos
- Operadores Especiales

La expresión WHERE selecciona un subconjunto de observaciones comparando los valores de una variable con valores constantes o valores de otras variables.

Operadores de Comparación

Los operadores disponibles para realizar operaciones son:

| | | |
|---|---|---|
| **LT** | < | Menor que |
| **GT** | > | Mayor que |
| **EQ** | = | Igual |
| **LE** | < = | Menor o igual |
| **GE** | > = | Mayor o igual |
| **NE** | ^ = | No igual |
| **IN** | | Igual a uno en la lista |

**NOTA**: Se pueden utilizar las 2 letras de la abreviación o los símbolos.

Ejemplo.

```
WHERE DEST= 'LON'
WHERE DEST  IN ('LON','PAR')
WHERE FLIGHT = '219'
WHERE BOARDED LT 150
WHERE MAIL GT FREIGHT
```

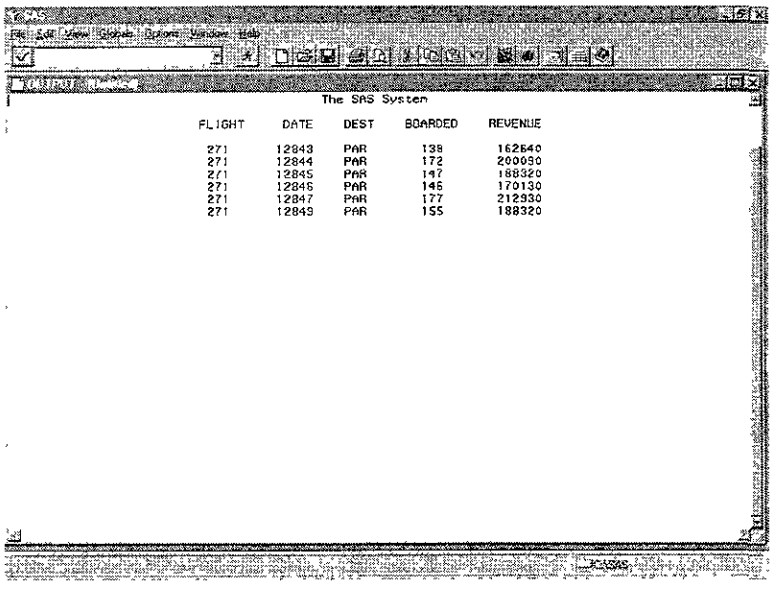
**NOTA**: Los valores de tipo carácter deben ser encerrados entre ' ' y coincidir en mayúsculas y minúsculas con los valores que se comparan.

Ejemplo: Imprimir solo las observaciones para el vuelo 271.

Programa:

```
proc print data=SASDATA.LONPAR noobs;
  var flight date dest boarded revenue;
  where flight = '271';
run;
```

Resultado:



The SAS System

| FLIGHT | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 271 | 12843 | PAR | 138 | 162840 |
| 271 | 12844 | PAR | 172 | 200090 |
| 271 | 12845 | PAR | 147 | 188320 |
| 271 | 12846 | PAR | 146 | 170130 |
| 271 | 12847 | PAR | 177 | 212930 |
| 271 | 12849 | PAR | 155 | 188320 |

**NOTA**: En este proceso el DATASET original no fue alterado.

Operadores Lógicos

En algunas aplicaciones se requiere seleccionar un subconjuto de observaciones basadas en múltiples condiciones.

Los operadores lógicos son utilizados para ligar secuencias de expresiones en una sola expresión.

| | |
|---|---|
| **OR** | Esta expresión será verdadera cuando alguna de las expresiones que la forman es verdadera. Cuando se comparan múltiples valores de la misma variable, se debe especificar el nombre de la variable en cada expresión. |

**NOTA**: Se puede utilizar el operador de comparación IN para ver si un valor es igual a uno de una lista de valores.

Ejemplo: WHERE DEST IN ('DFW','LAX');

| | |
|---|---|
| **AND** | Esta expresión será verdadera cuando todas las expresiones que la forman sean verdaderas. |

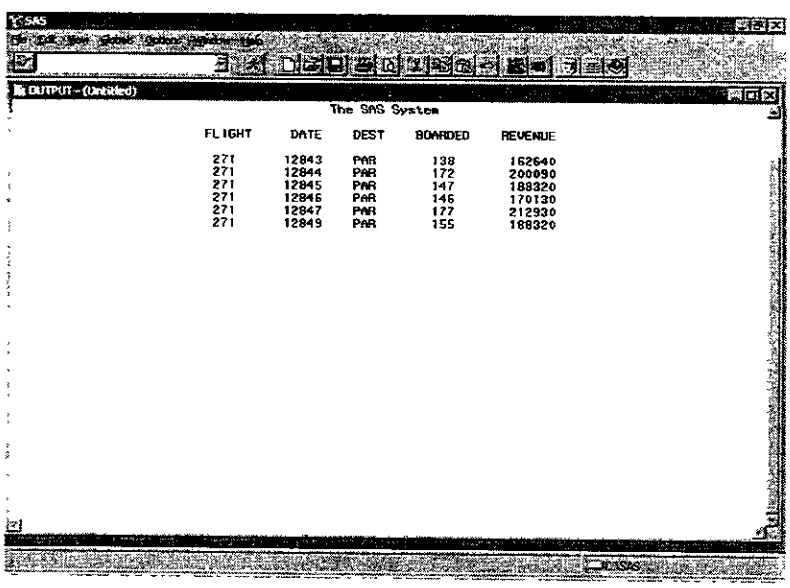
Ejemplo: WHERE FLIGHT = '217' AND BOARDED GT 180;

Ejemplo: Obtener una lista de todos los vuelos 271 donde la variable BOARDED tenga valores mayores a 100. Incluir las variables FLIGHT, DATE, DEST, BOARDED y REVENUE.

Programa:

```
proc print data=SASDATA.LONPAR noobs;
  var flight date dest boarded revenue;
  where flight = '271' and boarded gt 100;
run;
```

Resultado:



The SAS System

| FLIGHT | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 271 | 12843 | PAR | 138 | 162640 |
| 271 | 12844 | PAR | 172 | 200090 |
| 271 | 12845 | PAR | 147 | 188320 |
| 271 | 12846 | PAR | 146 | 170130 |
| 271 | 12847 | PAR | 177 | 212930 |
| 271 | 12849 | PAR | 155 | 188320 |

Operadores Especiales

Los operadores especiales están disponibles para simplificar la selección de observaciones.

Los operadores especiales son:

**BETWEEN - AND** Selecciona observaciones en las cuales los valores de las variables caen dentro de un rango de valores.

Ejemplo: WHERE BOARDED BETWEEN 50 AND 100;

**NOTA**: Se puede combinar el operador NOT con el operador BETWEEN – AND para seleccionar observaciones cuyos valores caen fuera del rango.

**CONTAINS** Selecciona observaciones que incluyen la cadena especificada en la expresión WHERE.

Ejemplo: WHERE JOBCODE CONTAINS 'FA';

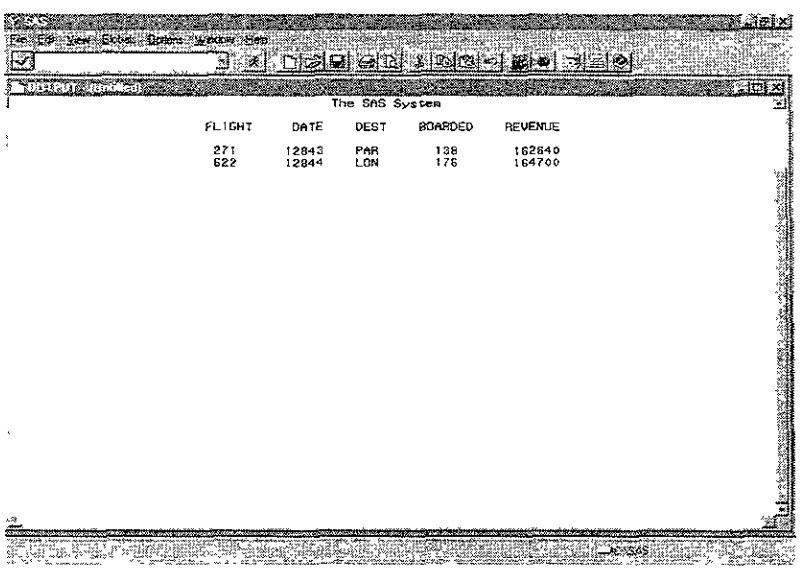
**NOTA**: Los operadores especiales WHERE son válidos solo en las expresiones WHERE.

Ejemplo: Obtener una lista de todos los vuelos que tiene un valor de la variable REVENUE entre 155000 y 170000. Incluir las variables FLIGHT, DATE, DEST, BOARDED y REVENUE.

Programa:

```
proc print data=SASDATA.LONPAR noobs;
  var flight date dest boarded revenue;
  where revenue between 155000 and 170000;
run;
```

Resultado:



Se pueden generar totales por columna utilizando la sentencia SUM.

Forma General la sentencia SUM.

**SUM** *variables*;

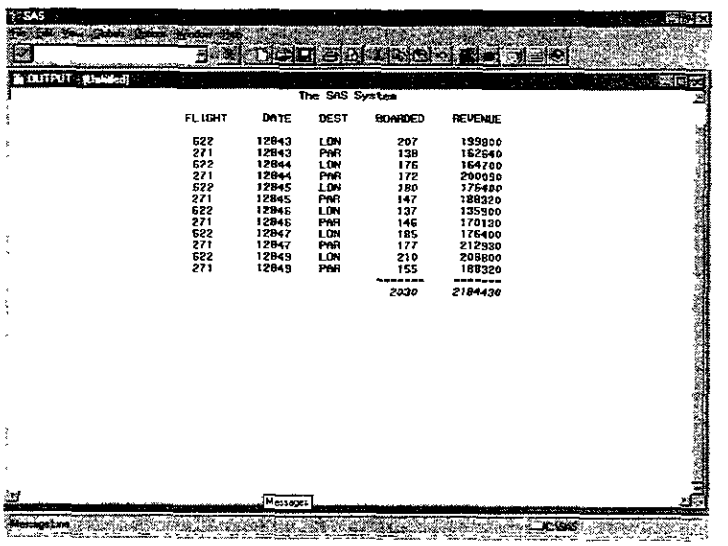
La sentencia SUM especifica las variables a ser totalizadas (o subtotales si las observaciones están listadas en subgrupos).

Ejemplo: Imprimir el número total de pasajeros y el total de la variable REVENUE.

Programa:

```
proc print data=SASDATA.LONPAR noobs;
  var flight date dest boarded revenue;
  sum boarded revenue;
run;
```

Resultado:



The SAS System

| FLIGHT | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 622 | 12843 | LON | 207 | 199936 |
| 271 | 12843 | PAR | 138 | 152640 |
| 622 | 12844 | LON | 176 | 164768 |
| 271 | 12844 | PAR | 172 | 200050 |
| 622 | 12845 | LON | 180 | 176480 |
| 271 | 12845 | PAR | 147 | 188320 |
| 622 | 12846 | LON | 137 | 135500 |
| 271 | 12846 | PAR | 146 | 170120 |
| 622 | 12847 | LON | 185 | 176400 |
| 271 | 12847 | PAR | 177 | 212930 |
| 622 | 12849 | LON | 210 | 208800 |
| 271 | 12849 | PAR | 155 | 188320 |
| | | | 2030 | 2184436 |

Generando Subtotales

Para generar un subgrupo de totales, es necesario que las observaciones dentro del DATASET estén agrupadas. Si las observaciones no están agrupadas, estas pueden ser ordenadas para generar grupos utilizando el procedimiento SORT.

El procedimiento SORT:

- Ordena las observaciones en un DATASET.
- Crea un nuevo DATASET que contiene las observaciones ordenadas.
- El DATASET original es remplazado por default.
- Puede ordenar utilizando varios criterios (variables).
- Puede ordenar de forma ascendente o descendente.
- No genera una salida impresa.
- Trata los valores nulos como los valores más pequeños.

☞ CUANDO ESTE PROCESO ES EJECUTADO, DEBE TOMARSE LA PRECAUCIÓN DE CONTAR CON AL MENOS TRES VECES EN ESPACIO LIBRE DISPONIBLE EL TAMAÑO DE LA TABLA QUE SE ESTA ORDENANDO.

ADEMAS DE ESTA OPCION DE ORDENAMIENTO TAMBIÉN SE PUEDEN GENERAR INDICES PARA CONSEGUIR UNA TABLA ORDENADA, PARA ESTE CASO, SE DEBE DE CONTEMPLAR UN ESPACIO EN DISCO PARA ALMACENARLO PARA MAYOR DETALLE AL RESPECTO DE INDICES CONSULTAR *' SAS Programming by Example"* INCLUIDO EN LA BIBLIOGRÁFIA

Forma General del procedimiento SORT.

**PROC SORT DATA** = *dataset1* **OUT**= *dataset2*;
**BY** *variables*;
**RUN;**

Ejemplo: Utilizar el PROC SORT para crear un nuevo dataset sin remplazar el dataset original

Programa:

```
proc sort data=SASDATA.LONPAR;
  by flight;
run;
```

Para ordenar las observaciones en forma descendente, especificar la opción DESCENDING en la sentencia BY.

```
proc sort data=SASDATA.LONPAR;
  by flight descending revenue;
run;
```

Una vez que el dataset esta ordenado y agrupado, se agregara la sentencia BY dentro del PROC PRINT para producir el reporte con totales por subgrupo. Los valores de la(s) variable(s) utilizada(s) en la sentencia BY determinan los subgrupos para los cuales los subtotales serán calculados.

Forma General de producir un reporte de tipo lista con subtotales.

**PROC SORT DATA** = *dataset1* **OUT**= *dataset2*;
**BY** *variables*;
**PROC PRINT DATA** = *dataset2*;
ID *variables;*
**VAR *variables;***
SUM *variables;*
**BY** *variables;*
**RUN;**

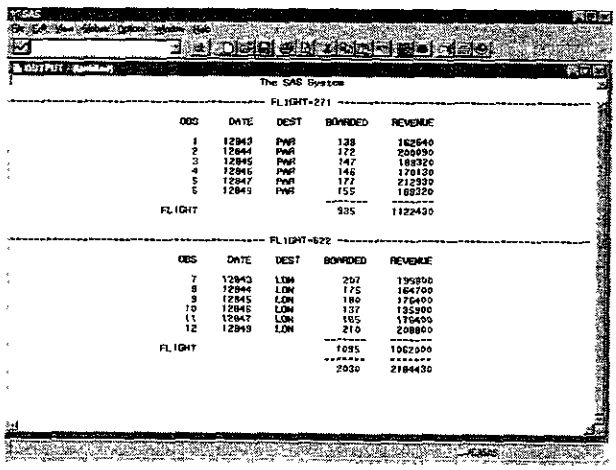
**NOTA:** Si el dataset esta ordenado previamente por las variables apropiadas el PROC SORT no es necesario.

Ejemplo: Imprimir subtotales para cada uno de los vuelos.

Programa:

```
proc sort data= SASDATA.LONPAR out=SORTED;
   by flight;
proc print data=SORTED;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
run;
```

Resultado:



The SAS System

FLIGHT=271

| OBS | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 1 | 12843 | PAR | 138 | 162540 |
| 2 | 12844 | PAR | 172 | 200090 |
| 3 | 12845 | PAR | 147 | 188320 |
| 4 | 12846 | PAR | 146 | 178130 |
| 5 | 12847 | PAR | 177 | 212330 |
| 6 | 12849 | PAR | 155 | 180320 |
| FLIGHT | | | 935 | 1122430 |

FLIGHT=622

| OBS | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 7 | 12843 | LON | 207 | 195800 |
| 8 | 12844 | LON | 175 | 164700 |
| 9 | 12845 | LON | 180 | 176400 |
| 10 | 12846 | LON | 137 | 135200 |
| 11 | 12847 | LON | 185 | 176400 |
| 12 | 12849 | LON | 210 | 208800 |
| FLIGHT | | | 1095 | 1062000 |
| | | | 2030 | 2184430 |

Con la sentencia ID, se puede especificar una variable cuyos valores se impriman en la columna más a la izquierda, remplazando los números de observaciones e identando los grupos de valores.

Cuando la sentencia ID especifica la misma variable que la sentencia BY,

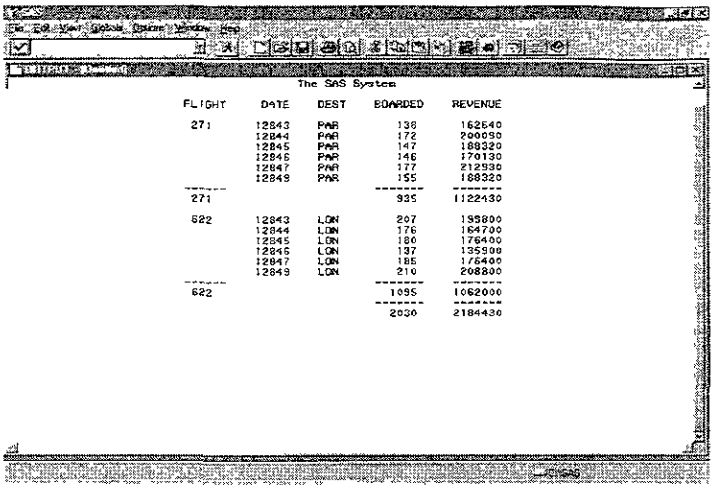
- La columna con el número de observación es suprimida.
- La variable ID/BY se imprime en la columna más a la izquierda.
- Cada valor ID/BY solo se imprime al principio de cada grupo generado por la sentencia BY y en la línea que contiene el subtotal de ese grupo.

Ejemplo: Especificar la variable **FLIGHT** como la variable ID y la variable BY.

Programa:

```
proc print data=SORTED;
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
run;
```

Resultado:



The SAS System

| FLIGHT | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 271 | 12843 | PAR | 138 | 162840 |
| | 12844 | PAR | 172 | 200050 |
| | 12845 | PAR | 147 | 188320 |
| | 12846 | PAR | 146 | 170130 |
| | 12847 | PAR | 177 | 212930 |
| | 12849 | PAR | 155 | 188320 |
| 271 | | | 935 | 1122430 |
| 622 | 12843 | LON | 207 | 155800 |
| | 12844 | LON | 176 | 164700 |
| | 12845 | LON | 180 | 176400 |
| | 12846 | LON | 137 | 135500 |
| | 12847 | LON | 185 | 176400 |
| | 12849 | LON | 210 | 208800 |
| 622 | | | 1095 | 1062000 |
| | | | 2030 | 2184430 |

☞ EL ORDEN EN EL CUAL SON UTILIZADAS LAS SENTENCIAS DENTRO DEL *PROC PRINT*, NO AFECTA EL RESULTADO

Adicionalmente se pueden generar paginas separadas para cada grupo de valores generados por la sentencia BY, utilizando la sentencia PAGEBY.

Forma General de utilizar la sentencia PAGEBY:

**PAGEBY** *variable-by*;

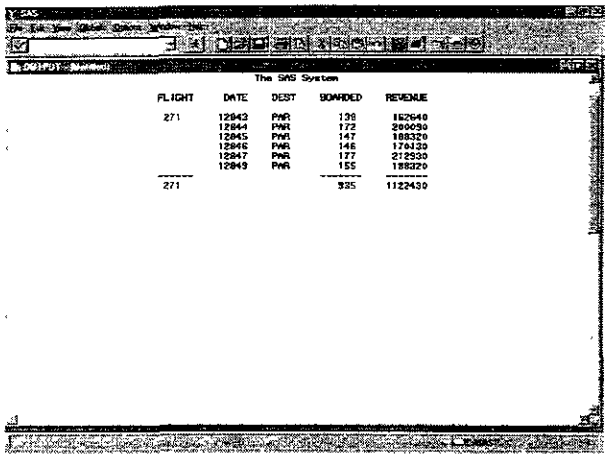
Ejemplo: Especifica la variable flight como la variable ID y la variable BY.

Programa:

```
proc print data=SORTED;
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
run;
```

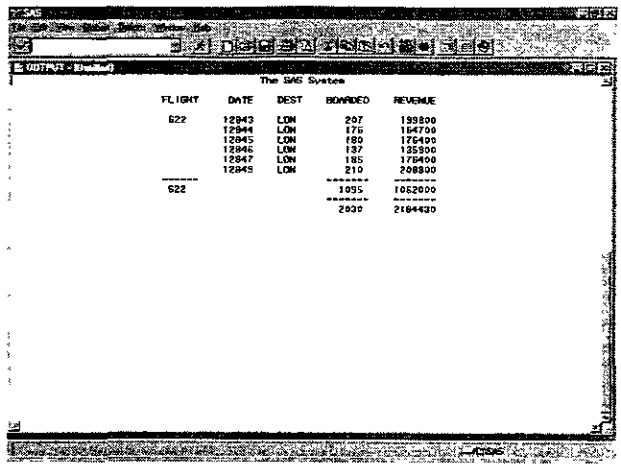
Resultado:

Página 1



| FLIGHT | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 271 | 12843 | PAR | 139 | 162648 |
| | 12844 | PAR | 172 | 200050 |
| | 12845 | PAR | 147 | 188320 |
| | 12846 | PAR | 146 | 174120 |
| | 12847 | PAR | 177 | 212530 |
| | 12849 | PAR | 155 | 188320 |
| 271 | | | 935 | 1122430 |

Página 2



| FLIGHT | DATE | DEST | BOARDED | REVENUE |
|---|---|---|---|---|
| 622 | 12843 | LON | 207 | 139800 |
| | 12844 | LON | 176 | 164700 |
| | 12845 | LON | 180 | 175400 |
| | 12846 | LON | 137 | 135900 |
| | 12847 | LON | 185 | 176400 |
| | 12849 | LON | 210 | 208800 |
| 622 | | | 1095 | 1052000 |
| | | | 2030 | 2104430 |

☞ LA VARIABLE ESPECIFICADA EN LA SENTENCIA *BY* DEBE SER LA MISMA QUE SE USE CON *PAGEBY/ID*, SI ESTA SENTENCIA NO ESTA DENTRO DEL PROGRAMA, NINGUNA DE LAS OTRAS PODRÁ SER USADA.

*AGREGANDO TÍTULOS Y PIES DE PÁGINA*

Se pueden agregar títulos y pies de página para mejorar la apariencia de los reportes.

Forma General de las sentencias TITLE y FOOTNOTE.

**TITLE*n*** *'text'*;
**FOOTNOTE*n*** *'text'*;

- El valor de ***n* puede ser de 1 hasta 10.**
- Los títulos aparecen al iniciode la página.
- Los pies de página aparecen al final de la página.
- Si no se especifica un título, el valor default es **"THE SAS SYSTEM"**.
- Un pie de página no es impreso a menos que se especifique uno.
- Un TITLE o FOOTNOTE sin número (*n*) es equivalente a TITLE1 o FOOTNOTE1.

**NOTA**: Al generar títulos y pies de página, hay que asegúrese de abrir y cerrar las comillas correspondientes.

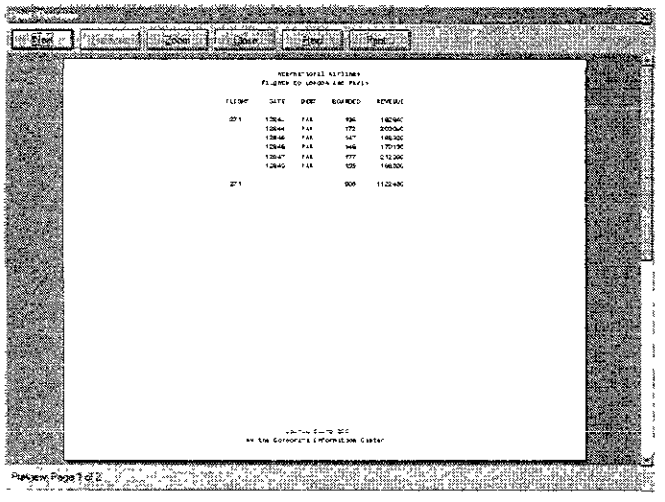
Se pueden utilizar las ventanas de TITLES y FOOTNOTES disponibles en el menú para definir los títulos y pies de página.

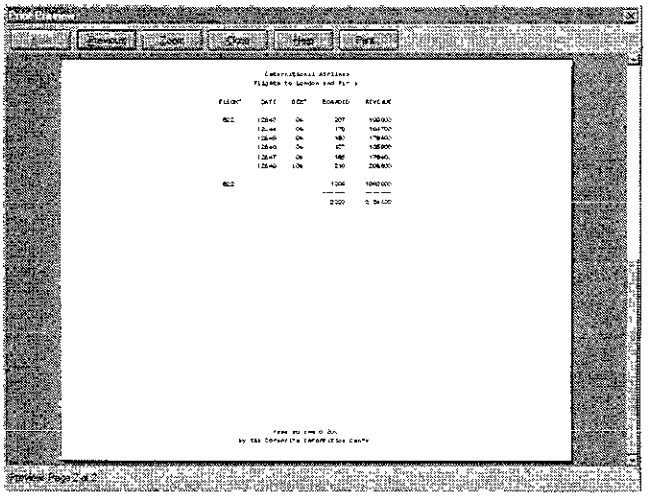
Ejemplo: Agregar títulos y pies de página al reporte anterior, asegúrese de abrir y cerrar las comillas.

Programa:

```
proc print data=SORTED;
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
  title 'International Airlines';
  title2 'Flights to London and Paris';
  footnote 'Prepered Enero 2001';
  footnote2 'by the Corporate Information Center';
run;
```

Resultado:





Los títulos y pies de página mantendrán su efecto hasta que estos sean cambiados, cancelados o hasta que la sesión de SAS sea cerrada.

Las sentencias:

TITLE*n* o FOOTNOTE*n*

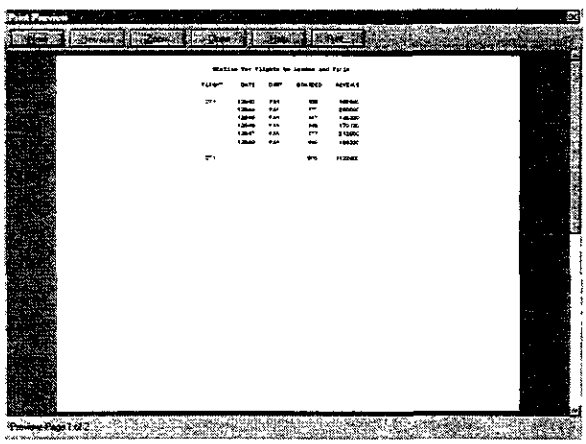
- Reemplazan el previo título o pie de página con el mismo número.
- Cancelan títulos o pies de página con números mayores.

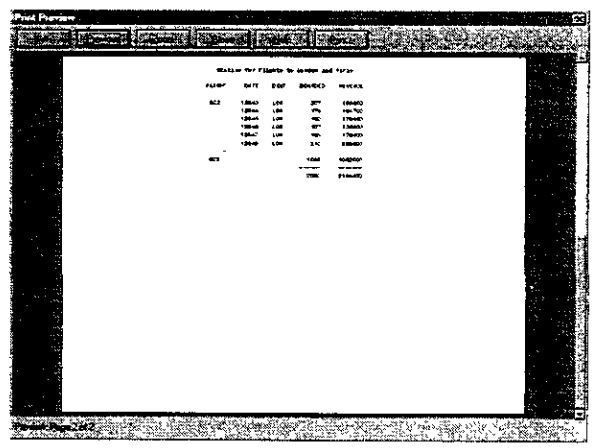
Ejemplo: Agregar títulos y pies de página al reporte anterior, asegúrese de abrir y cerrar las comillas.

Programa:

```
proc print data=SORTED;
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
  title1 'Statics for Flights to London and Paris';
  footnote1;
run;
```

Resultado:





Se pueden cancelar todos los títulos y pies de página previamente definidos,

- Ejecutando las sentencias

  TITLE;
  FOOTNOTE;

- Utilizando las ventanas TITLES y FOOTNOTES del menú principal para cancelar los valores de títulos y pies de página existentes.

*AGREGANDO ENCABEZADOS A LAS COLUMNAS*

Un reporte puede ser mejorado etiquetando las columnas con textos más descriptivos. Para asignar etiquetas descriptivas a una variables se utilizara la sentencia LABEL.

Forma General de utilizar la sentencia LABEL.

**LABEL** *variable1='label'*
*Variable2='label';*

Las etiquetas

- Pueden tener hasta 40 caracteres de longitud
- Pueden ser utilizadas en varios procedimientos
- Requieren de las sentencias LABEL o SPLIT= en la sentencia PROC PRINT para ser utilizadas por el procedimiento.
- Existen solo durante el PROC step cuando son utilizadas dentro de un procedimiento.
- Son asignadas permanentemente si se utilizan dentro de un DATA step.

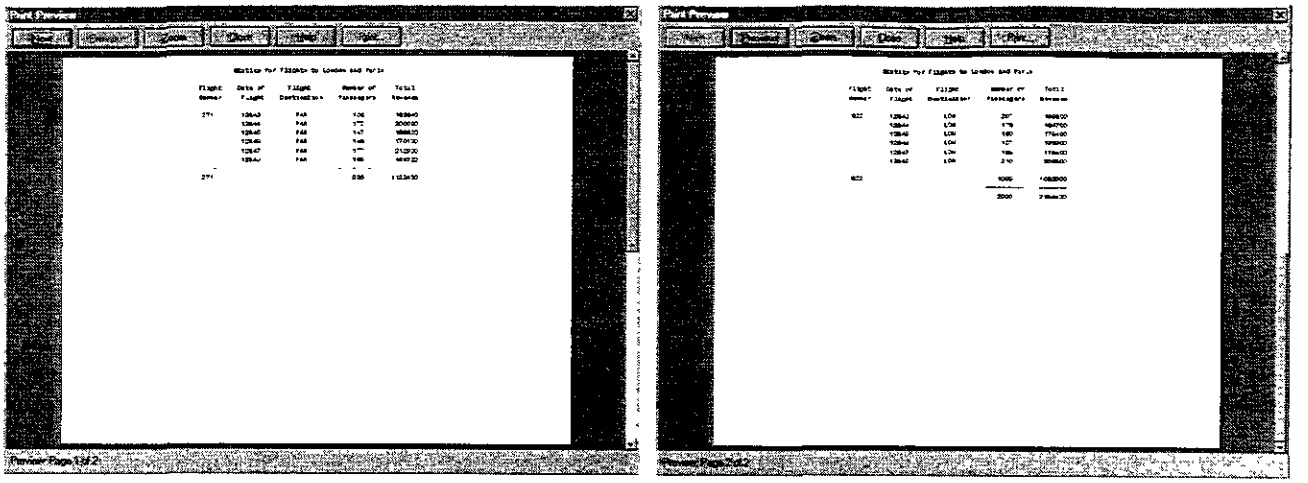
Ejemplo: Agregar etiquetas descriptivas al reporte anterior. Utilizar la opción LABEL en la sentencia PROC PRINT para desplegar las etiquetas de las variables como encabezados de las columnas.

Programa:

```
proc print data=SORTED label;
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
  title1 'Statics for Flights to London and Paris';
  label  flight = 'Flight Number'
     date = 'Date of Flight'
     dest = 'Flight Destination'
     boarded = 'Number of Passangers'
     revenue = 'Total Revenue';
run;
```

☞ SI LA OPCION *LABEL* NO ES COLOCADA DESPUÉS DE LA SENTENCIA *PROC PRINT* LAS ETIQUETAS ASIGNADAS A LAS VARIABLES NO SERAN DESPLEGADAS

Resultado:

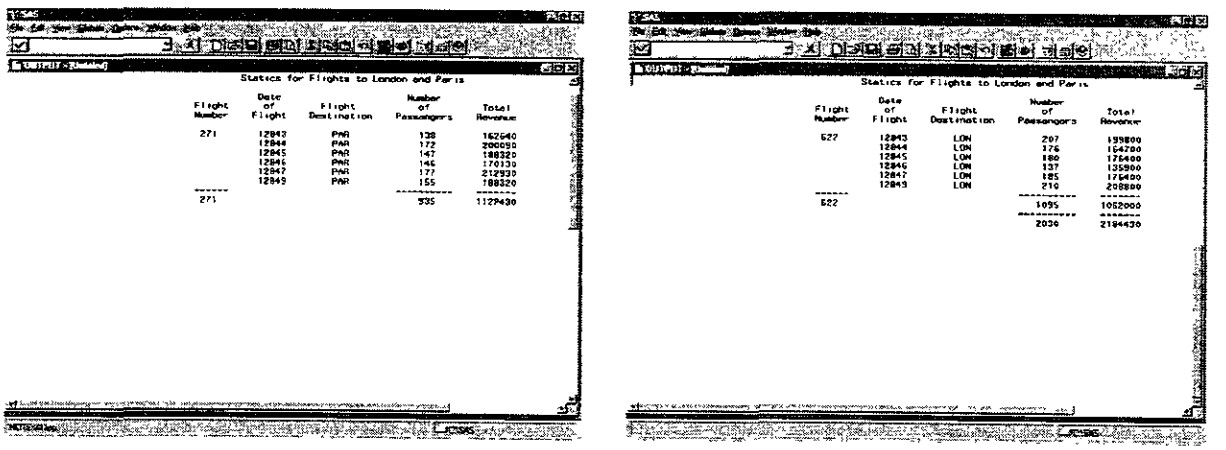


Ejemplo: Utilizar la opción SPLIT= para controlar como son centradas las etiquetas de las columnas.

Programa:

```
proc print data=SORTED split='*';
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
  title1 'Statics for Flights to London and Paris';
  label  flight = 'Flight*Number'
     date = 'Date*of*Flight'
     dest = 'Flight*Destination'
     boarded = 'Number*of*Passangers'
     revenue = 'Total*Revenue';
run;
```

Resultado:



**NOTA**: Cualquier carácter puede ser utilizado como el carácter SPLIT= , es recomendable no utilizar letras.

*FORMATEANDO VALORES*

Un reporte puede mejorar su apariencia formateando los valores de las variables con formatos SAS. Para utilizar un formato dentro de un procedimiento, se usara la sentencia FORMAT.

Forma General de la sentencia FORMAT.

**FORMAT** *variable format;*

Los formatos asociados a variables en un,

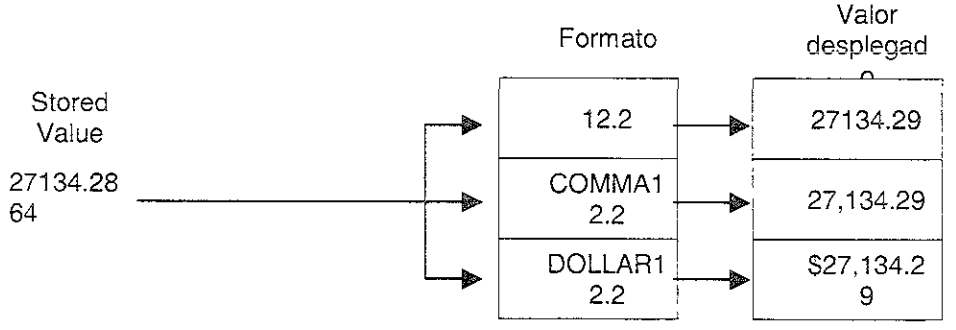
- PROC step solo funcionan durante la ejecución de este
- DATA step son permanentemente asignados a la variable

Algunos formatos SAS[1]:

*w.d* Formato estándar numérico
$*w.* Formato estándar carácter
COMMA *w.d* Comas en un número
DOLLAR *w.d* Signos de dólar y comas en un número
DATE *w.* Para escribir fechas de la forma 19OCT92 (DATE7.) o 19OCT1992 (DATE9.)
MMDDYY *w.* Para escribir fechas en la forma 101692 (MMDDYY6.) o 10/16/92 (MMDDYY8.)

En un formato SAS

*W* Indica el total de caracteres utilizados para la salida. Cada formato tiene un ancho definido.
. Es el delimitador requerido.
*D* Indica el número de decimales (solo para valores numéricos). El valor default es cero.



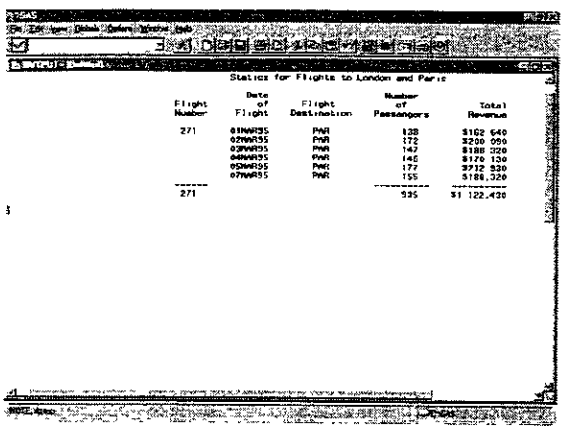

[1] Para mayor información referente a los formatos disponibles en SAS consultar la ayuda del Sistema SAS.

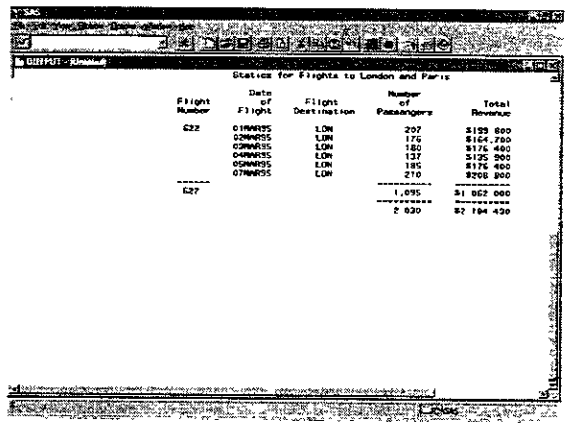
Ejemplo: Formatear la variable DATE utilizando el formato DATE7. , la variable BOARDED con comas y la variable REVENUE con signo de pesos, comas y sin decimales.

Programa:

```
proc print data=SORTED split='*';
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
  title1 'Statics for Flights to London and Paris';
  label  flight = 'Flight*Number'
     date = 'Date*of*Flight'
     dest = 'Flight*Destination'
     boarded = 'Number*of*Passangers'
     revenue = 'Total*Revenue';
  format date date7. Boarded comma6. Revenue dollar10.;
run;
```

Resultado:





*CREANDO FORMATOS DEFINIDOS POR EL USUARIO*

El procedimiento FORMAT (PROC FORMAT) se puede utilizar para definir formatos que asignen etiquetas descriptivas a los valores de las variables.

Forma General del PROC FORMAT.

**PROC FORMAT**;
  **VALUE** *nombre del formato rango1='etiqueta1'*
  *rango2='etiqueta2'*
  . . . ;
**RUN**;

Ejemplo: Utilizar el PROC FORMAT para crear un formato que asigne los nombres de las ciudades a los códigos.

Programa:

```
proc format;
   value $destfmt  'LON' = 'London'
       'PAR' = 'Paris';
run;
```

Los nombres de los formatos,

- No deben tener un nombre mayor a 8 caracteres.
- Debe ser único
- Puede comenzar con una letra o un guión bajo para variables numéricas, para variables carácter con un signo de pesos.
- No pueden terminar con número

En una sentencia VALUE las etiquetas pueden ser asignadas como

- Números sencillos

```
value sexfmt  1 = 'Female'
       2 = 'Male'
       other = 'Miscoded';
```

- Rangos de números

```
value boardfmtlow-49 = 'Below Average'
        50-99 = 'Average'
        100-high = 'Above Average';
```

- Valores carácter y rangos de valores caracter

```
value $grade       'A' = 'Good'
       'B'-'D' = 'Fair'
    'F' = 'Good'
       'I','U' = 'Fair'
         other = 'Miscoded';
```

Las etiquetas deben ser,

- De una longitud no mayor a 40 caracteres.
- Encerradas entre comillas

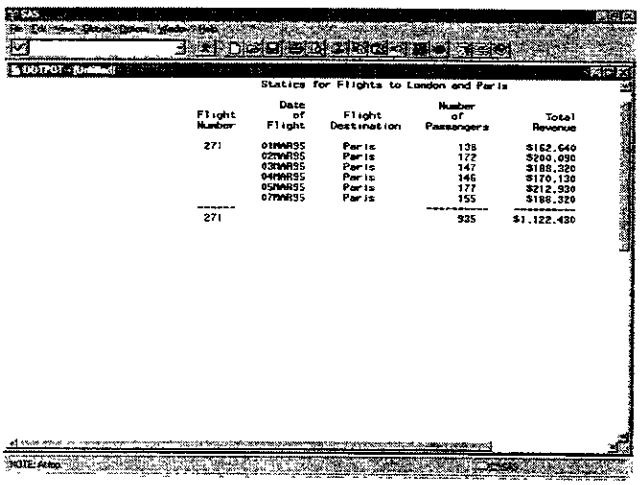
Una vez que un formato es definido con un PROC FORMAT, el formato puede ser asignado a la variable utilizando la sentencia FORMAT.

Ejemplo: Formatear la variable DATE utilizando el formato DATE7. , la variable BOARDED con comas y la variable REVENUE con signo de pesos, comas y sin decimales.

Programa:

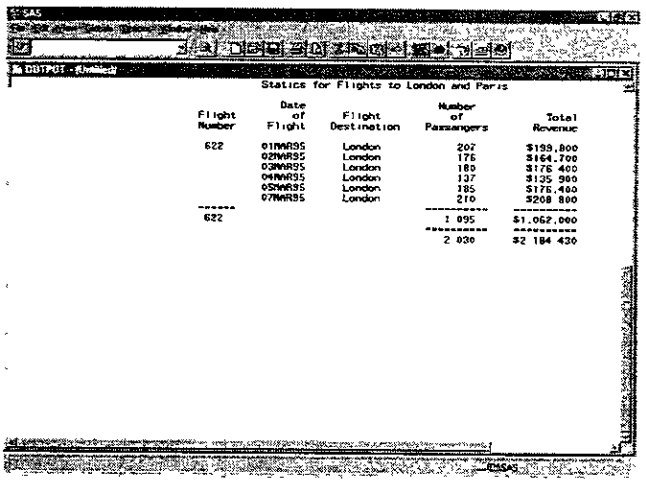
```
proc format;
   value $destfmt  'LON' = 'London'
       'PAR' = 'Paris';
proc print data=SORTED split='*';
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
  title1 'Statics for Flights to London and Paris';
  label  flight = 'Flight*Number'
     date = 'Date*of*Flight'
     dest = 'Flight*Destination'
     boarded = 'Number*of*Passangers'
     revenue = 'Total*Revenue';
  format date date7. Boarded comma6. Revenue dollar10.
         dest $destfmt.;
run;
```

Resultado:



Statics for Flights to London and Paris

| Flight Number | Date of Flight | Flight Destination | Number of Passangers | Total Revenue |
|---|---|---|---|---|
| 271 | 01MAR95 | Paris | 138 | $162,640 |
| | 02MAR95 | Paris | 172 | $200,090 |
| | 03MAR95 | Paris | 147 | $188,320 |
| | 04MAR95 | Paris | 146 | $170,130 |
| | 05MAR95 | Paris | 177 | $212,930 |
| | 07MAR95 | Paris | 155 | $188,320 |
| 271 | | | 935 | $1,122,430 |



Statics for Flights to London and Paris

| Flight Number | Date of Flight | Flight Destination | Number of Passangers | Total Revenue |
|---|---|---|---|---|
| 622 | 01MAR95 | London | 207 | $199,800 |
| | 02MAR95 | London | 176 | $164,700 |
| | 03MAR95 | London | 189 | $176,400 |
| | 04MAR95 | London | 137 | $135,900 |
| | 05MAR95 | London | 185 | $176,400 |
| | 07MAR95 | London | 210 | $208,800 |
| 622 | | | 1,095 | $1,062,000 |
| | | | 2,030 | $2,184,430 |

☞ LOS FORMATOS SE PUEDEN ALMACENAR DE FORMA PERMANENTE O EN MEMORIA, POR LO CUAL SON ÚTILES PARA OPTIMIZAR PROCESOS Y AHORRAR ESPACIO DE ALMACENAMIENTO YA QUE PUEDEN SER UTILIZADOS COMO CATALOGOS EN MEMORIA PARA MÁS DETALLES CONSULTAR *"Professional SAS Programming Secrets"* Y/O *"SAS Fundamentals: A Programming Approach"*.

*UTILIZANDO OPCIONES DE SISTEMA*

Las opciones de sistema pueden cambiar la apariencia de un reporte.

Forma General de la sentencia OPTIONS:

**OPTIONS** *option . . . ;*

Algunas opciones de sistema:

CENTER|NOCENTER LINESIZE = *ancho*

| | |
|---|---|
| DATE\|NODATE | PAGESIZE = *n* |
| NUMBER\|NONUMBER | PAGENO = *n* |

Las opciones de sistema,

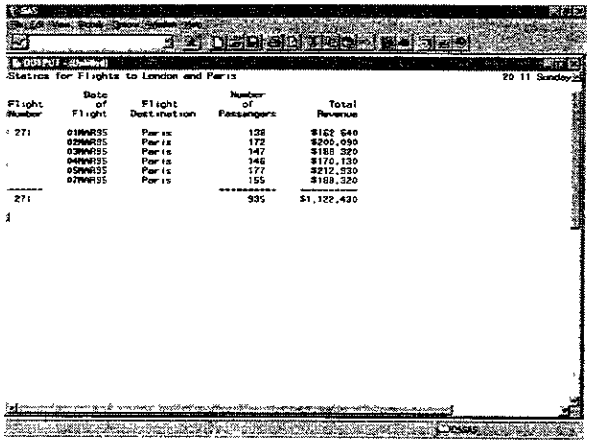
- Tienen efecto durante la sesión de SAS activa o hasta que son modificados
- Pueden ser definidos con la sentencia OPTIONS o en la ventana OPTIONS

Ejemplo: Formatear la variable DATE utilizando el formato DATE7. , la variable BOARDED con comas y la variable REVENUE con signo de pesos, comas y sin decimales.

Programa:

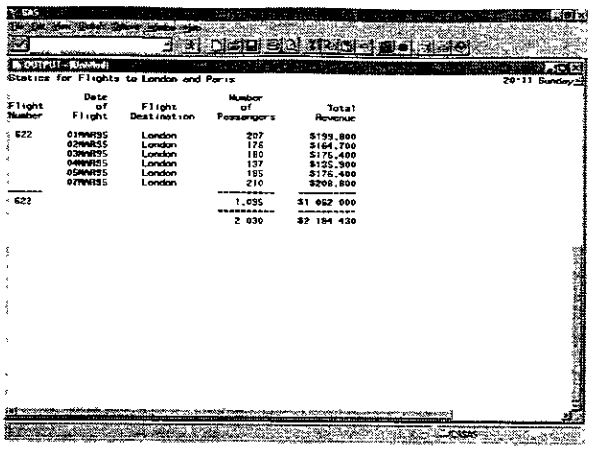
```
Options nocenter date number;
proc print data=SORTED split='*';
  id flight;
  var date dest boarded revenue;
  sum boarded revenue;
  by flight;
  pageby flight;
  title1 'Statics for Flights to London and Paris';
  label  flight = 'Flight*Number'
     date = 'Date*of*Flight'
     dest = 'Flight*Destination'
     boarded = 'Number*of*Passangers'
     revenue = 'Total*Revenue';
  format date date7. Boarded comma6. Revenue dollar10.
         dest $destfmt.;
run;
```

Resultado:



Statics for Flights to London and Paris — 20 11 Sunday

| Flight Number | Date of Flight | Flight Destination | Number of Passengers | Total Revenue |
|---|---|---|---|---|
| 271 | 01MAR95 | Paris | 128 | $162 560 |
| | 02MAR95 | Paris | 172 | $200,090 |
| | 03MAR95 | Paris | 147 | $188 320 |
| | 04MAR95 | Paris | 146 | $170,130 |
| | 05MAR95 | Paris | 177 | $212,930 |
| | 07MAR95 | Paris | 155 | $188,320 |
| 271 | | | 935 | $1,122,430 |



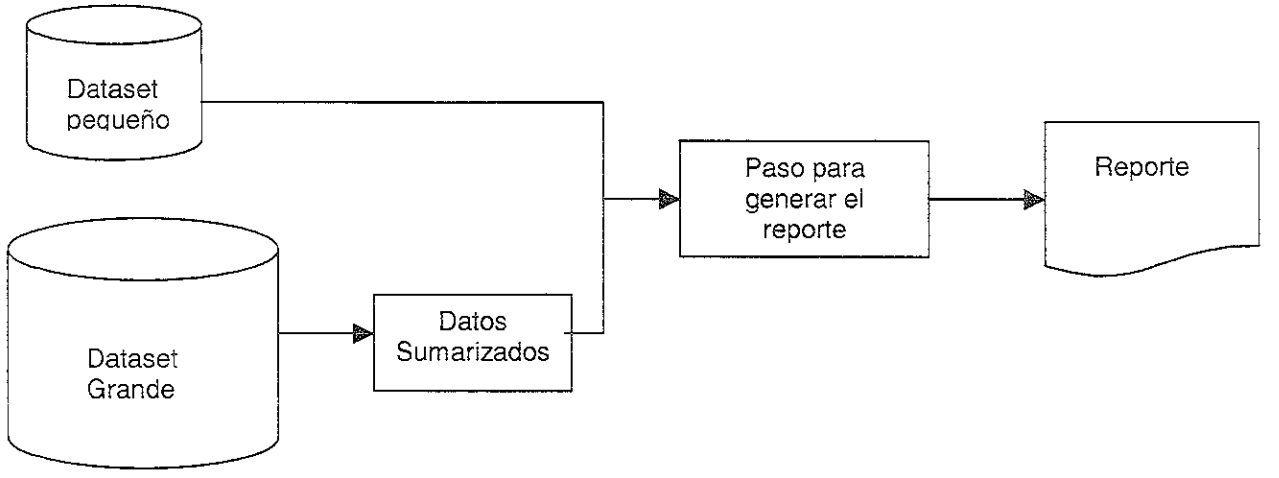
Statics for Flights to London and Paris — 20:11 Sunday

| Flight Number | Date of Flight | Flight Destination | Number of Passengers | Total Revenue |
|---|---|---|---|---|
| 622 | 01MAR95 | London | 207 | $193,800 |
| | 02MAR95 | London | 176 | $164,700 |
| | 03MAR95 | London | 180 | $175,400 |
| | 04MAR95 | London | 137 | $135,300 |
| | 05MAR95 | London | 185 | $175,400 |
| | 07MAR95 | London | 210 | $208,800 |
| 622 | | | 1,095 | $1 052 900 |
| | | | 2 030 | $2 184 430 |

## 6.3 PROC TABULATE

Reportes Sumarizados

Un reporte de tipo lista es muy útil cuando se necesita interpretar o consultar datos contenidos en un dataset pequeño, sin embargo los datasets grandes pueden ser más fáciles de interpretar utilizando reportes sumarizados.




En general, un reporte sumarizado es generado para:

- Organizar los datos en clases o categorías para análisis
- Calcular estadísticas como son cuentas, sumas, media y porcentajes sobre las clases.

**NOTA**: Los datos pueden ser sumarizados y presentados en un reporte en un paso o en varios pasos. El PROC TABULATE permite realizar ambos en un paso.

Antes de generar un reporte sumarizado, se deben definir cuales serán las variables de clasificación y cuales las variables de análisis.

Las variables de clasificación,

- Pueden ser numéricas o carácter
- Si son numéricas deben representar valores discretos
- Identificar clases o categorías en donde los cálculos son hechos

Las variables de Análisis

- Son numéricas
- Tienden a ser continuas
- Son apropiadas para calcular promedios, sumas u otras estadísticas

La variable carácter JOBCODE

- Categoriza el nivel de trabajo de los sobrecargos.
- Es utilizada como una variable de clasificación para producir un reporte que despliegue información acerca de los sobrecargos por categoría.

La variable numérica SALARY

- Contiene el salario de cada uno de los sobrecargos
- Es utilizada como variable de Análisis para producir un reporte que examine el valor máximo y el promedio de salario por niveles.

El procedimiento TABULATE permite calcular varias estadísticas para sumarizar datos y para controlar el formato del reporte.

Algunas de las estadísticas que incluye son:

- Frecuencia
- Media
- Desviación estándar
- Mínimo
- Máximo
- Rango
- Suma
- Porcentaje

El formato del reporte incluye:

- Control de la construcción de la tabla
- Formato a los valores
- Múltiples variables por cada dimensión en la tabla
- Una variable de clasificación ALL para totalizar
- Etiquetas para la variables y para las estadísticas

Un PROC TABULATE especifica

- Las variables de clasificación
- Las variables de análisis
- La estructura de la tabla y el formato

Forma General del PROC TABULATE.

**PROC TABULATE DATA=** *dataset*;
  **CLASS** *variables de clasificación;*
  **VAR** *variables de análisis;*
  **TABLE** *página,renglón, columna;*
**RUN**;

Dentro de la sentencia TABLE se pueden especificar hasta tres dimensiones:

- Una dimensión define la columna
- Una segunda dimensión define los renglones
- Una tercera dimensión define las páginas

Se pude especificar el formato de la tabla y las estadísticas deseadas en la sentencia TABLE con *expresiones*,

Las *expresiones* consisten de *elementos* y *operadores*

Los *elementos* pueden ser

- Variables
- Estadísticas

Los *operadores* que controlan el formato de la tabla dentro de la sentencia TABLE

| OPERADOR | | ACCIÓN |
|---|---|---|
| COMA | , | Genera una nueva dimensión |
| BLANCO | | Concatena Tablas |
| ASTEISCO | * | Genera cruces o subgrupos |

Ejemplo: Utilizando las tres variables de clase FLIGHT, MONTH y DAY en una dataset llamado FLIGHTS, se pueden construir una variedad de tablas.

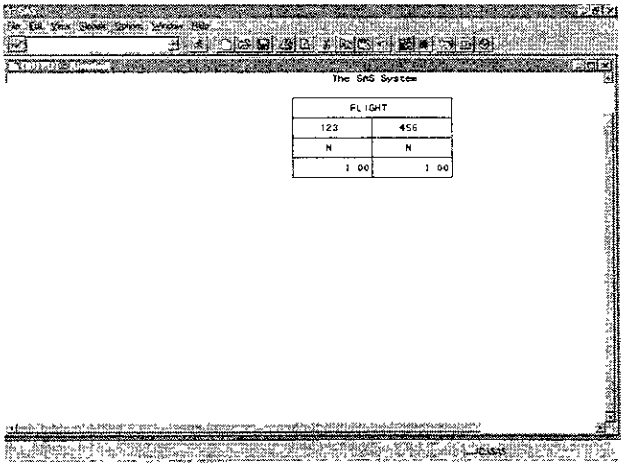
| VARIABLE | VALORES |
|---|---|
| FLIGHT | 123, 456 |
| MONTH | FEB, MAR |
| DAY | SAT, SUN |

Una sentencia TABLE sin operadores genera una columna por cada valor de la variable.

Programa:

```
proc tabulate data=flights;
   class flight;
   table flight;
run;
```

Resultado:

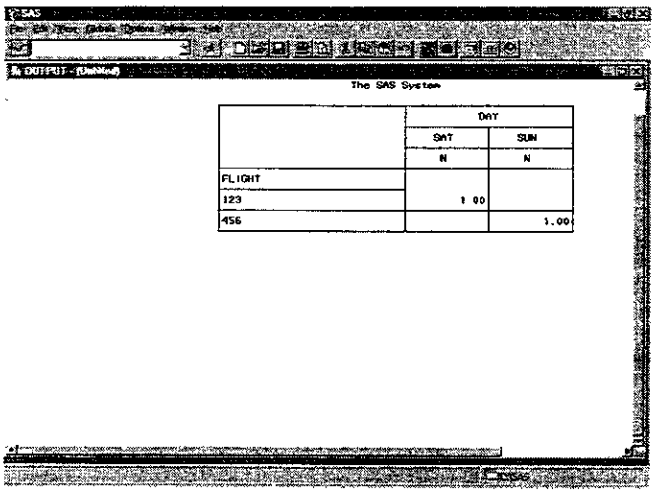



Dos expresiones separadas por un operador coma (,) generan una tabla de dos dimensiones. La primer dimensión define los renglones, la segunda define las columnas.

Programa:

```
proc tabulate data=flights;
   class flight day;
   table flight, day;
run;
```

Resultado:



The SAS System

| | DAY | |
|---|---|---|
| | SAT | SUN |
| | N | N |
| FLIGHT | | |
| 123 | 1.00 | |
| 456 | | 1.00 |

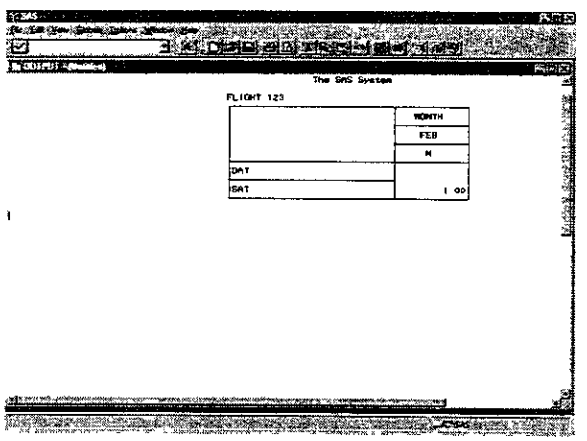
Tres expresiones separadas por comas generan una tabla tridimensional. La primera expresión define las páginas, la segunda define los renglones y las tercera define las columnas.

Programa:

```
proc tabulate data=flights;
   class flight day month;
   table flight, day, month;
run;
```

Resultado:

**Pág. 1**

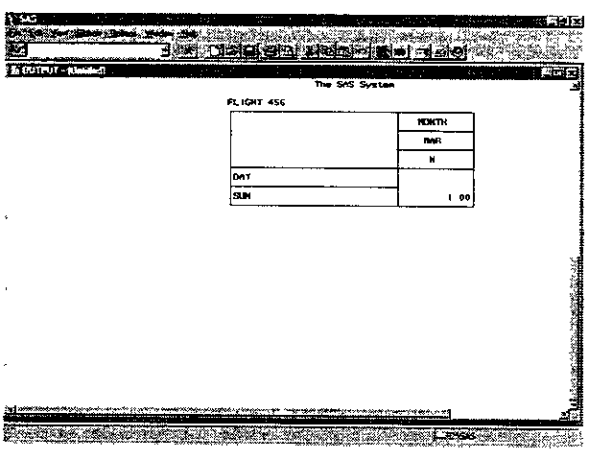


The SAS System

FLIGHT 123

| | MONTH |
|---|---|
| | FEB |
| | N |
| DAY | |
| SAT | 1.00 |

**Pág. 2**



The SAS System

FLIGHT 456

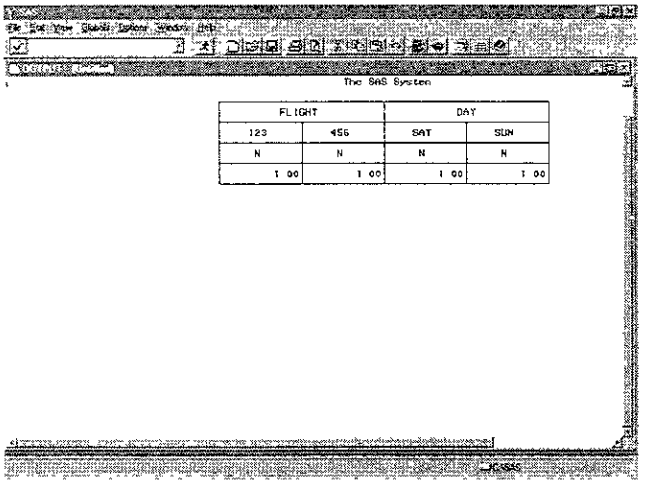
| | MONTH |
|---|---|
| | MAR |
| | N |
| DAY | |
| SUN | 1.00 |

Dos elementos separados por un operador blanco ( ) concatenan tablas.

Programa:

```
proc tabulate data=flights;
   class flight day;
   table flight day;
run;
```

Resultado:



The SAS System

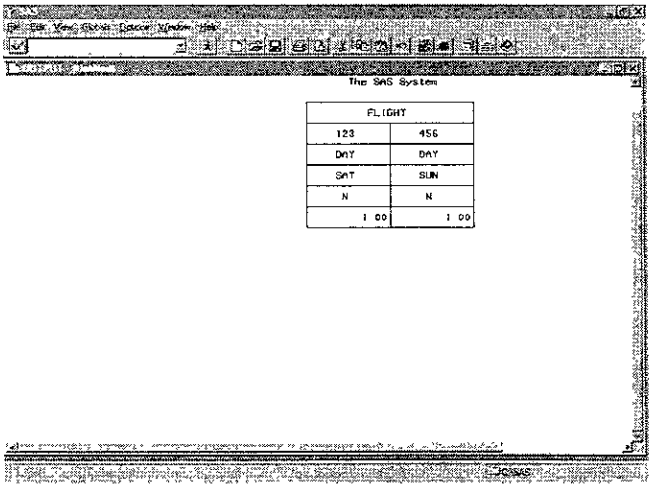
| FLIGHT | | DAY | |
|---|---|---|---|
| 123 | 456 | SAT | SUN |
| N | N | N | N |
| 1 00 | 1 00 | 1 00 | 1 00 |

Dos elementos separados por un operador asterisco ( * ) cruzan elementos y producen tablas jerárquicas.

Programa:

```
proc tabulate data=flights;
   class flight day;
   table flight*day;
run;
```

Resultado:



The SAS System

| FLIGHT | |
|---|---|
| 123 | 456 |
| DAY | DAY |
| SAT | SUN |
| N | N |
| 1 00 | 1 00 |

*CONTROLANDO DIMENSIONES EN LAS TABLAS*

TABLAS DE UNA DIMENSIÓN

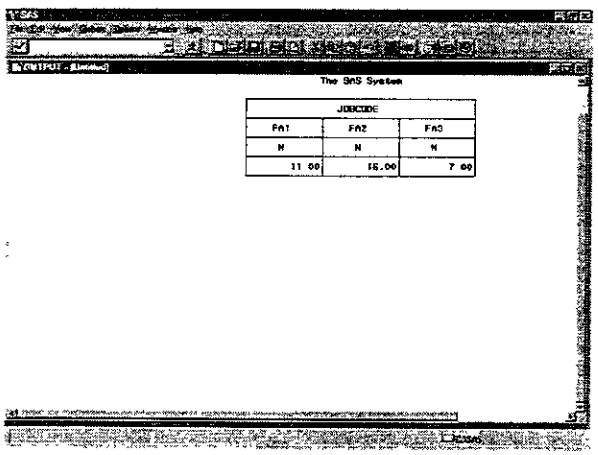
Ejemplo: Utiliza el dataset SASDATA.FLTATEN para generar una tabla unidimensional que despliegue los números de vuelos atendidos por cada grupo de empleados definidos por la variable JOBCODE.

Una sentencia TABLE sin comas genera una tabla unidimensional.

Programa:

```
proc tabulate data=sasdata.fltaten;
   class jobcode;
   table jobcode*n;
run;
```

Resultado:



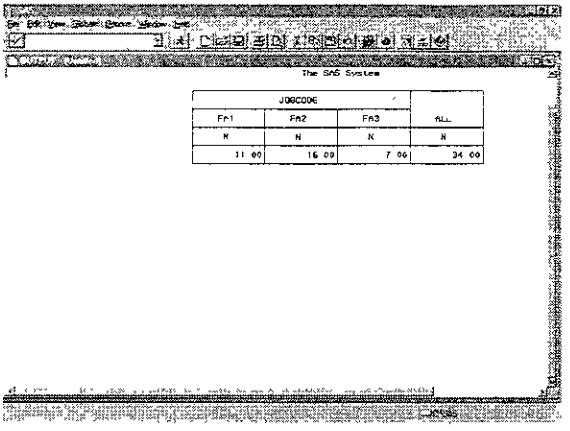

- Para especificar la estadística que se va a desplegar, se utilizara un asterisco y el nombre de la estadística.
- Si no se especifica una estadística en la sentencia TABLE, la estadística por default es N (frecuencia). El número de observaciones en una clase en particular.
- Las variables utilizadas en la sentencia TABLE, deben ser especificadas en la sentencias CLASS o VAR dependiendo del caso.

Ejemplo: Agregar una columna con el total de la tabla utilizando la variable de clase ALL en la sentencia TABLE.

Programa:

```
proc tabulate data=sasdata.fltaten;
   class jobcode;
   table jobcode*n all*n;
run;
```

Resultado:




En el ejemplo, el operador en blanco concatena la columna ALL con las columnas generadas por la variable JOBCODE. La variable de clase ALL es una opción del PROC TABULATE, que es utilizada para generar totales.

Dentro del PROC TABULATE, se puede

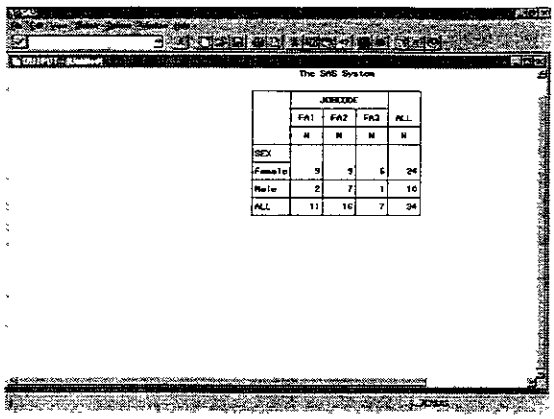
- Controlar el ancho de la celda y dar formato a las estadísticas utilizando la opción FORMAT= en la sentencia PROC TABULTE.
- Definir el título de los renglones espaciando con la opción RTS= en la sentencia TABLE.
- Formatear los valores de las variables de clase con la sentencia FORMAT.

Ejemplo: Generar la misma tabla con un formato de celda más pequeño y títulos espaciados en los renglones. Substituye las cadenas de texto por los valores de la variable SEX.

Programa:

```
proc format;
  value $sexfmt  'F'='Female'
                 'M'='Male';
proc tabulate data=sasdata.fltaten format=5.0;
   class sex jobcode;
   table sex all,jobcode*n all*n /rts=8;
   format sex $sexfmt.;
run;
```

Resultado:

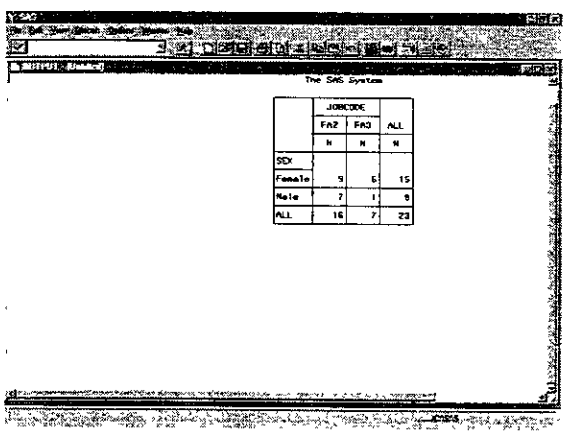


Ejemplo: Generar la misma tabla desplegando solo los valores de JOBCODE FA2 y FA3.

Programa:

```
proc tabulate data=sasdata.fltaten format=5.0;
   where jobcode in ('FA2','FA3');
   class sex jobcode;
   table sex all,jobcode*n all*n /rts=8;
   format sex $sexfmt.;
run;
```

Resultado:



Algunas de las estadísticas que se pueden generar con el PROC TABULATE son.

| | |
|---|---|
| N | Número de observaciones con valores no nulos |
| NMISS | Número de observaciones nulos |
| MEAN | Media aritmética (promedio) |
| STD | Desviación Estándar |
| MIN | Valor mínimo |
| MAX | Valor Máximo |
| RANGE | Rango de valores |
| SUM | Suma |

☞ SE PUEDEN ESPECIFICAR CUALQUIER NÚMERO DE ESTADÍSTICAS EN CUALQUIER DIMENSIÓN, TODAS ÉSTAS DEBEN SER ESPECIFICADAS EN LA MISMA DIMENSIÓN UTILIZANDO EL OPERADOR * Y UNA VARIABLE DE ANALISIS A LA CUAL SE APLICA EL CALCULO

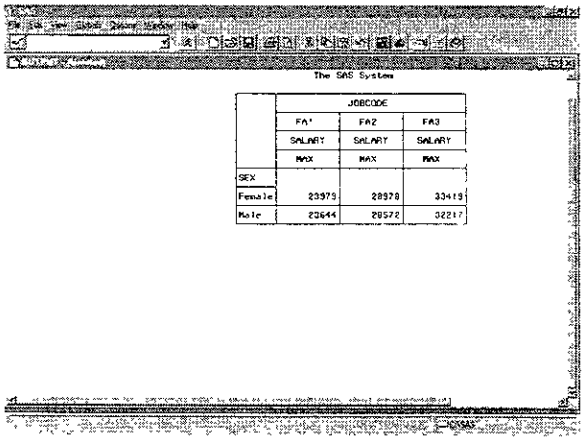
*UTILIZANDO VARIABLES DE ANÁLISIS*

Ejemplo: Generar una tabla que compare el valor máximo de salario entre los distintos niveles de la variable JOBCODE para cada sexo.

Programa:

```
proc tabulate data=sasdata.fltaten format=10.0;
   class sex jobcode;
   var salary;
   table sex,jobcode*salary*max /rts=8;
   format sex $sexfmt.;
run;
```

Resultado:



The SAS System

| | JOBCODE | | |
|---|---|---|---|
| | FA1 | FA2 | FA3 |
| | SALARY | SALARY | SALARY |
| | MAX | MAX | MAX |
| SEX | | | |
| Female | 23979 | 28978 | 33419 |
| Male | 23644 | 28572 | 32217 |

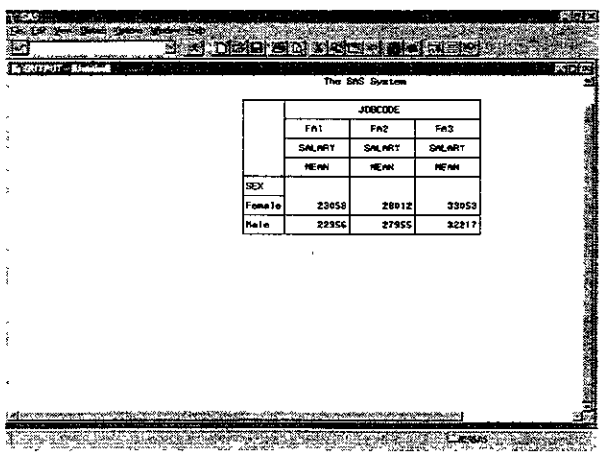
☞ CUANDO SE ESPECIFICA UNA VARIABLE DE ANÁLISIS EN LA SENTENCIA TABLE, LA ESTADÍSTICA POR DEFAULT ES LA SUMA

Ejemplo: Crear una tabla que compare los promedios de salarios entre los distintos niveles de JOBCODE para cada sexo.

Programa:

```
proc tabulate data=sasdata.fltaten format=10.0;
   class sex jobcode;
   var salary;
   table sex,jobcode*salary*mean /rts=8;
   format sex $sexfmt.;
run;
```

Resultado:



| | JOBCODE | | |
|---|---|---|---|
| | FA1 | FA2 | FA3 |
| | SALARY | SALARY | SALARY |
| | MEAN | MEAN | MEAN |
| SEX | | | |
| Female | 23058 | 28012 | 33053 |
| Male | 22956 | 27955 | 32817 |

Etiquetando la tabla

Dentro de un PROC TABULATE es posible,

- Mejorar la apariencia del reporte utilizando sentencias FORMAT, LABEL, TITLE y FOOTNOTE (descritas anteriormente).
- Asignar etiquetas descriptivas a la variable de clase ALL y todas las estadísticas utilizando la sentencia KEYLABEL.

Forma General de la sentencia KEYLABEL,

**KEYLABEL** *keyword='label'*;

*Keyword* Pueden ser la variable de clase ALL o cualquier descripción de estadísticas.

*Label* Puede ser hasta de 40 caracteres de longitud.

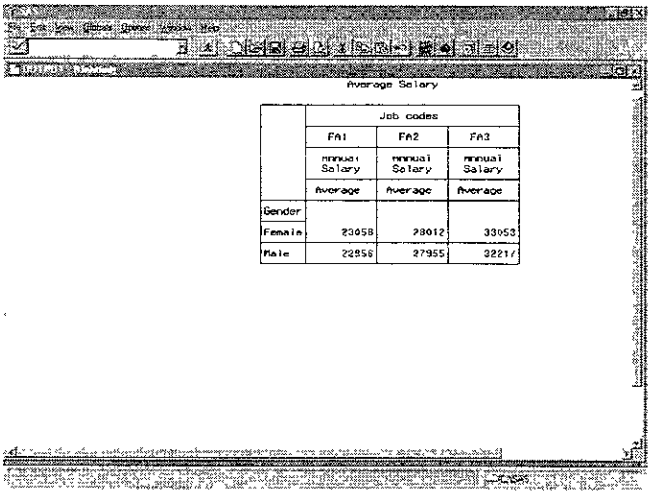
**NOTA:** El PROC TABULATE asigna automáticamente a las variables utilizadas, las etiquetas incluidas dentro de la parte descriptora del DATASET sin necesidad de utilizar la opción LABEL.

Ejemplo: Agregar al reporte anterior una columna que despliegue el salario promedio de todos los niveles de la variable JOBCODE. Etiquetar las estadísticas por nivel, el total y la variables. Agregar títulos y pies de página adecuados.

Programa:

```
proc format;
   value $sexfmt 'F' ='Female'
                 'M' = 'Male';
proc tabulate data=sasdata.fltaten format=10.0;
   class sex jobcode;
   var salary;
   table sex,jobcode*salary*mean /rts=8;
   format sex $sexfmt.;
   keylabel all='Overall'
            mean='Average';
   label sex='Gender'
     jobcode='Job codes'
      salary='Annual Salary';
   title 'Average Salary';
   footnote 'by job code and gender';
run;
```

Resultado:



Average Salary

| | Job codes | | |
|---|---|---|---|
| | FA1 | FA2 | FA3 |
| | annual Salary | annual Salary | annual Salary |
| | Average | Average | Average |
| Gender | | | |
| Female | 23058 | 28012 | 33053 |
| Male | 22856 | 27955 | 32217 |

by job code and gender

## 6.4 PROC GCHART

Se pueden utilizar gráficas de pie o de barras para desplegar los siguientes tipos de información:

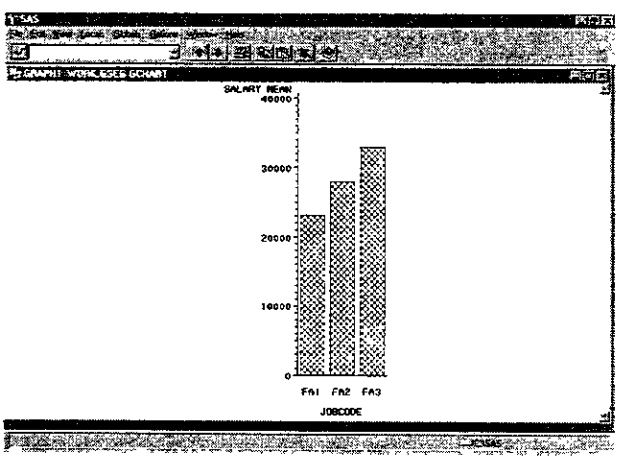
- La distribución de los valores de las variables
- La relación entre dos o más variables
- Valores promedio de algunas mediciones de diferentes categorías
- Valores totales de algunas mediciones para diferentes categorías

El procedimiento GCHART es utilizado para generar este tipo de gráficas.

Dentro de un PROC GCHART ,

- Se debe especificar la forma física que se desea tenga la gráfica (pie, vertical, horizontal)
- Se debe determinar una variables que determine el número de barras o rebanadas de pie que serán creadas.
- Opcionalmente se identificara una variable de análisis que será utilizada para calcular estadísticas que determinen ancho (o largo) de la barra o el tamaño de la rebanada. Por default estos valores son representados contando la frecuencia.

| VARIABLE A GRAFICAR | VARIABLE DE ANÁLISIS | |
|---|---|---|
| JOBCODE | SALARY | |
| FA1 | 23177 | } SALARIO PROMEDIO |
| FA2 | 28572 | |
| FA3 | 32886 | |



Forma General de la sentencia PROC GCHART

**PROC GHART DATA=** *dataset*;

Utilizando una sentencia HBAR, VBAR o PIE para especificar el tipo de gráfica que se requiere.

**HBAR** *variable a gráficar /opciones*;
**VBAR** *variable a gráficar /opciones*;
**PIE** *variable a gráficar /opciones*:

La *variable a gráficar*

- Identifica la variable que se va a gráficar, la cual determina el número de barras o rebanadas que serán generadas.
- Puede ser carácter o numérica

Algunas de las *opciones*

**SUMVAR** = *variable de análisis* Identifica la variable de análisis que se utilizara para los cálculos de sumas y promedios.

**TYPE** = MEAN | SUM Especifica que el ancho o largo de la barra o el tamaño de la rebanada representa la suma o promedio de los valores de la *variable de análisis*. El valor por default es la suma.

**NOTA**: Si no se especifica una variable de análisis el valor default de TYPE es FREQ. Esto es la frecuencia con la que se presenta cada uno de los distintos valores de la variable que sé esta graficando.
Este procedimiento debe ser cerrado utilizando la sentencia QUIT.

Cuando la *variable a graficar* es una variable numérica,

- La variable se asume como continua a menos de que se especifique otra cosa.
- Los intervalos son automáticamente calculados e identificados por puntos medios.
- Una barra o rebanada es construida para cada punto medio.

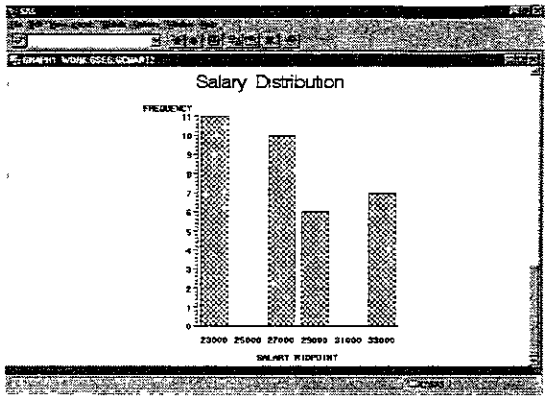
☞ LAS CARACTERISTICAS DESCRITAS ARRIBA, SON ESPECIFICA S DEL *PROC GCHART*, SIN EMBARGO EXISTEN MUCHOS OTROS PROCEDIMIENTOS DENTRO DE SAS PARA GENERAR GRÁFICAS (INCLUIDAS DE TRES DIMENSIONES), LOS CUALES NO SON MOTIVO DE ESTE TRABAJO, PERO PUEDEN SER CONSULTADOS EN EL LIBRO " *SAS Language and Procedures* " INCLUIDO EN LA BIBLIOGRAFIA

Ejemplo: Generar una gráfica de barras mostrando la distribución de la variable SALARY.

Programa:

```
proc gchart data=sasdata.numeric;
   vbar salary;
   title1 'Salary Distribution';
run;
quit;
```

Resultado:



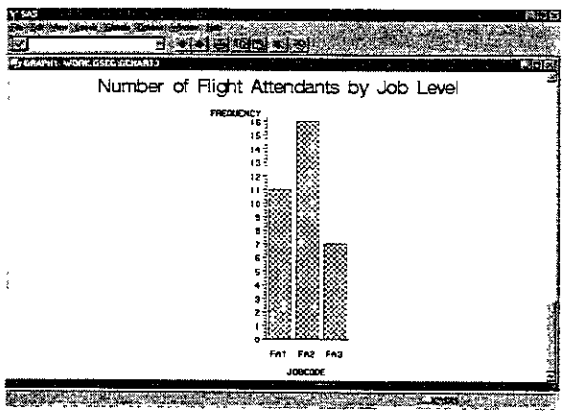

Cuando la *variable a graficar* es una variable carácter, una barra o rebanada es construida por cada valor único de la variable.

Ejemplo: Utilizar el dataset SASDATA.FLTATEN que contiene solo los siguientes valores de la variable JOBCODE (FA1,FA2,FA3). Generar una gráfica de barras verticales que despliegue el número de empleados en cada uno de los tres niveles.

Programa:

```
proc gchart data=sasdata.fltaten;
   vbar jobcode;
   title1 'Number of Flight Attendants by Job Level';
run;
quit;
```

Resultado:

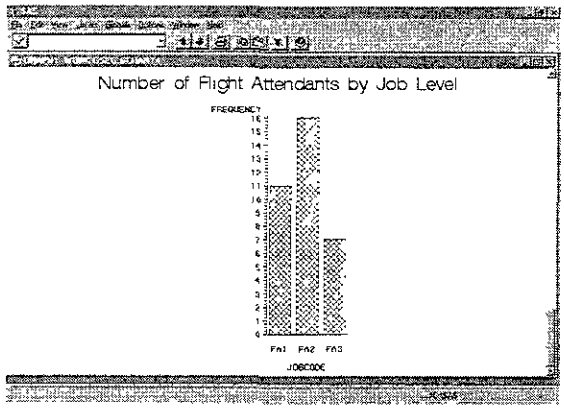



Ejemplo: Generar una gráfica de barras verticales que despliegue los promedios de salario por empleado en cada uno de los tres niveles de empleo.

Programa:

```
proc gchart data=sasdata.fltaten;
   vbar jobcode;
   title1 'Number of Flight Attendants by Job Level';
run;
quit;
```

Resultado:



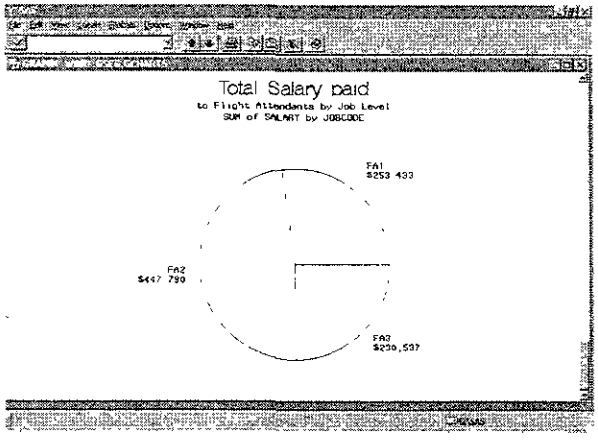

**GRÁFICAS DE PIE**

Ejemplo: Utilizar el data set SASDATA.FLTATEN para generar una gráfica de PIE comparando el total de salario de los empleados para cada uno de los valores de la variable JOB.

Programa:

```
proc gchart data=SASDATA.FLTATEN;
  pie jobcode /sumvar=salary type=sum;
  title 'Total Salary paid';
  title2 'to Flight Attendants by Job Level';
  format salary dollar8.;
run;
quit;
```

Resultado:



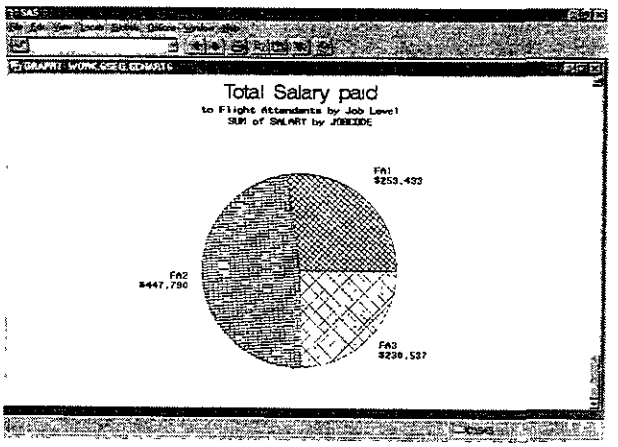
La opción FILL= puede ser utilizada para especificar si se desea algún tipo de relleno para las rebanadas de la gráfica de PIE. Los valores posibles son: FILL=S (para un relleno sólido) y FILL=X (para un relleno cruzado).

Ejemplo: Generar una gráfica de PIE cuyas rebanadas se muestren rellenas de forma cruzada.

Programa:

```
proc gchart data=SASDATA.FLTATEN;
  pie jobcode /sumvar=salary type=sum fill=x;
  title 'Total Salary paid';
  title2 'to Flight Attendants by Job Level';
  format salary dollar8.;
run;
quit;
```

Resultado:



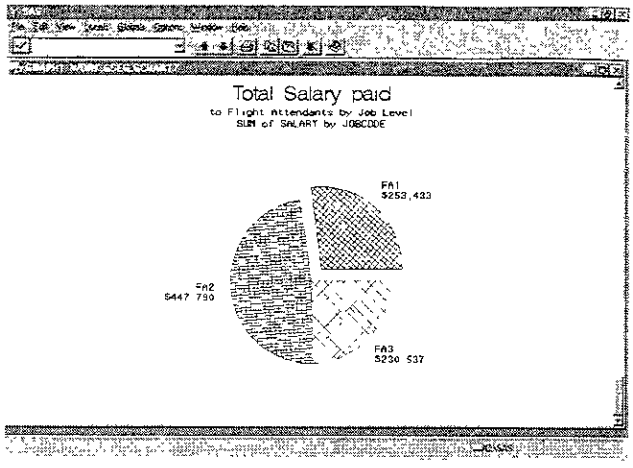

Además de la opción anterior, es posible hacer que una de las rebanadas de la gráfica sobresalga, utilizando la opción EXPLODE= . El valor que recibirá esta opción, será el valor de la variable que corresponde a la rebanada que se desea sobresaltar.

Ejemplo: Sobresaltar la rebana correspondiente al valor 'FA1' de la gráfica anterior.

Programa:

```
proc gchart data=SASDATA.FLTATEN;
  pie jobcode /sumvar=salary type=sum fill=x EXPLODE='FA1';
  title 'Total Salary paid';
  title2 'to Flight Attendants by Job Level';
  format salary dollar8.;
run;
quit;
```

Resultado:


Total Salary paid
to Flight Attendants by Job Level
SUM of SALART by JOBCODE
FA1
$253,433
FA2
$447 790
FA3
$230 537

## CONCLUSIONES

Durante más de 5 años de experiencia con el sistema SAS y 3 años trabajando para una de las empresas más importantes en la industria del Software, he tenido la oportunidad de conocer las necesidades tecnológicas que están surgiendo dentro del mercado, donde los egresados de la carrera de Matemática Aplicadas y Computación se están integrando como profesionistas eficientes, apoyando a dichas empresas a mantener su operación, desarrollar nuevas tecnologías y sobre todo aportando su conocimiento en la de toma de decisiones.

A lo largo de estos años, he percibido también la creciente necesidad por parte de las empresas financieras, de gobierno, de telecomunicación, aseguradoras, manufactureras, farmacéuticas, de investigación de Mercado y muchas otras que están trabajando con SAS, de contar con profesionistas que sean capaces de implementar las múltiples soluciones de toma de decisiones que esta herramienta les ofrece. Es aquí donde los egresados de la carrera de Matemática Aplicadas y Computación tenemos una de tantas oportunidades de desarrollar nuestro potencial y conocimiento.

Hoy en día las organizaciones cuentan con sistemas transaccionales. que además de ayudar a controlar la operación diaria de la empresa, generan más y más información. Lamentablemente esta información no se aprovecha al máximo en la toma de decisiones, ya que es difícil integrar datos provenientes de distintas fuentes y plataformas. Aunado a esto, el auge de la trasferencia de información y comunicaciones a través de Internet a provocado que las empresas deban tomar decisiones eficaces en un lapso de tiempo muy corto, conseguir este objetivo, requiere contar con la información necesaria, en el momento oportuno.

Por lo anterior, herramientas como SAS que ofrecen entrega de información para la toma de decisiones, se vuelven día a día indispensables dentro de cualquier empresa. Sin embargo, no todo el trabajo, viene de la herramienta, además de esto, se requiere del conocimiento necesario para extraer la información adecuada y organizarla dentro de un repositorio de toma de decisiones, de donde posteriormente es tomada para ser analizada. Es durante este proceso donde el conocimiento matemático e informático que los egresados de MAC adquieren a lo largo de la carrera, se convierte en un factor de éxito para las organizaciones.

El lenguaje 4GL BASE SAS es una excelente opción para la extracción, transformación y presentación de información proveniente de cualquier fuente de datos, esto debido a que es sencillo de utilizar, una vez que se cuenta con el conocimiento lógico y tecnológico proporcionado por las distintas asignaturas de la carrera de Matemática Aplicadas y Computación. Con este lenguaje es posible acceder y escribir en la mayoría de las bases de datos, además de que el código es totalmente transportable ya que SAS corre casi en todas las plataformas conocidas dentro del mercado, esto proporciona la capacidad de ejecutar procesos locales y remotos que aprovechan las capacidades de los distintos equipos, los cuales en la mayoría de las veces son los mismos a los que los alumnos de la carrera de Matemática Aplicadas y Computación tienen acceso en el Centro de Cómputo de la escuela. En este punto es importante hacer la recomendación de reforzar más el conocimiento en cuanto a plataformas y más aún aprovechar al máximo todas las capacidades y oportunidades que la Universidad ofrece.

## BIBLIOGRAFIA


"SAS Language and Procedures"
SAS Institute Inc.
SAS Institute Inc.
1991

" SAS Programming by Example"
Ron Cody & Ray Pass
SAS Institute Inc.
1995

"Professional SAS programming Secrets"
Rick Aster & Rhena Seidman
Windcrest/ McGraw Hill
1991

"SAS Language:Reference (Version 6, First Edition)"
SAS Institute Inc.
SAS Institute Inc.
1990

"Suministro de Información con el Sistema SAS"
Lisa Evans
SAS Institute Inc.
1997

"SAS Programming"
Jeff Cartier
SAS Institute Inc.
1992

"Fundamentals of the SAS System (Versión 6)"
J. Larry Stewart
SAS Institue Inc.
1997

"SAS Fundamentals: A programming Approach"
Donnalee Cowden
SAS Institue Inc.
1995

"Aplicaciones Estadísticas y Econométricas con SAS"
Carrascal U
Ra-ma
1997

Web Site
www.sas.com

## GLOSARIO

| | |
|---|---|
| Argumento | Elemento dentro de una función. |
| Arima | Por sus siglas en ingles (**A**uto**R**egressive **I**ntegrated **M**oving-**A**verage ). Un modelo ARIMA predice un valor en una respuesta dentro de una serie de tiempo, como una combinación lineal, se sus propios valores y errores pasados además, de valores actuales y pasados de otras series de tiempo. |
| Buffer | Dentro de SAS. se refiere al espacio de trabajo temporal asignado en memoria. |
| Campo llave | Campo/Columna/Variable cuyos valores no puede repetirse dentro de una tabla. |
| Catalog | Archivo de almacenamiento de pantallas, consultas y códigos SAS |
| Cliente / Servidor | Ambiente de trabajo de red, que consiste en instalar aplicaciones en la máquina con mayores capacidades (servidor), para ser ejecutadas desde equipos remotos (Clientes). |
| Valores Continuos | Aquellos con un rango de valores infinitos. |
| Dara Warehouse | Proceso de construcción de un repositorio para la toma de decisiones. |
| Datamation | Revista Informática, especializada en Data Warehouse. |
| Valores Discretos | Aquellos cuyos valores son finitos, pueden ser contabilizados (se utilizan para agrupas y/o clasificar información). |
| Distribución | Se obtiene al considerar el número de veces que ocurre un suceso al repetir una experiencia. |
| Drill Down | Funcionalidad característica de los Sistemas de Información Ejecutiva (EIS), para ver la información en distintos niveles de sumarización. |
| EIS | Sistema de Información Ejecutiva |
| Enterprise Miner | Producto de SAS para Minería de datos |
| Estructura de tabla | Que esta formado por renglones y columnas definidas. |
| Expresión | Declaración de una cosa para darla a entender.<br>Dentro de SAS, grupo de sentencias que representan una cantidad o sentencia. |
| Formato | Mascara/ Forma de presentar información. |
| Frecuencia | Número de veces que se presenta un valor en una columna. |
| Función | Programa que realiza una tarea especifica. Ej: SUM (Suma de Valores).<br>Cantidad que depende de otras cantidades variables. |
| Gartner Group | Consultora Estadounidense. que se dedica a evaluar el desempeño de las empresas alrededor del mundo. |
| Índice | Lista o numeración breve y por orden.<br>Dentro de SAS, archivo que contiene referencias y apuntadores ordenados, a una tabla (data set). |
| Informat | Mascara/ Forma de leer información. |
| Interactivos | Que permiten ejercer una acción recíprocamente, entre dos o más objetos, agentes, fuerzas, etc. |

| | |
|---|---|
| Lenguajes 4GL | Lenguajes de ordenes que expresan lo que hay que hacer en lugar de cómo hacerlo. Lenguajes deductivos. |
| Librería | Biblioteca local, donde se tienen libros o conjunto de éstos.<br>Dentro de SAS, forma lógica de nombrar un espacio físico de almacenamiento de información. |
| Media | Promedio aritmético. |
| Minería de datos | Conocimiento y análisis de patrones de comportamiento de datos. |
| Missing | Valor ausente. |
| Multidimensional | Que se puede consultar en distintas dimensiones. |
| Nulo | Valor inexistente. |
| Números Aleatorios | Aquellos que son generados de forma aleatoria (por azar), regidos por leyes de probabilidad. |
| OLAP | Procesos Analíticos en Línea |
| POPMENU | Menú que se activa, al presionar el botón izquierdo del mouse. |
| PROC | Abreviación de procedimiento. Es usado como palabra reservada dentro de SAS. Sentencia que indica el nombre del proceso que se va a utilizar. |
| Program Data Vector (PDV) | Espacio en memoria, donde se almacena temporalmente el registro que se esta procesando en el paso DATA. |
| Protocolo de Comunicación | Forma ordenada y estructurada de transmisión de información. |
| Rango | Diferencia entre el limite Superior y el limite inferior en un grupo de valores. |
| Registro | Línea de datos en una tabla. |
| Sentencia | Instrucción de un programa. Es una frase que puede desglosarse en varias operaciones elementales reconocibles y ejecutables por la Unidad Central de Procesamiento (CPU). |
| Sintaxis | Descripción del uso adecuado de sentencias dentro de un lenguaje de programación. |
| Sites | Localidad, lugar. |
| Software | Conjunto de Programas que puede ejecutar una computadora. |
| SQL | Por sus siglas en inglés (Structure Query Language), Lenguaje de Consulta Estructurado. |
| Target | Blanco, Objeto. |
| Value | Valor |
| What - If | "Qué-si" .Análisis derivado de las Series de Tiempo y/o pronósticos, el cual consiste en modificar valores y ver como estos afectan el comportamiento y el resultado. |