SAS for data Management, Analysis, and Reporting

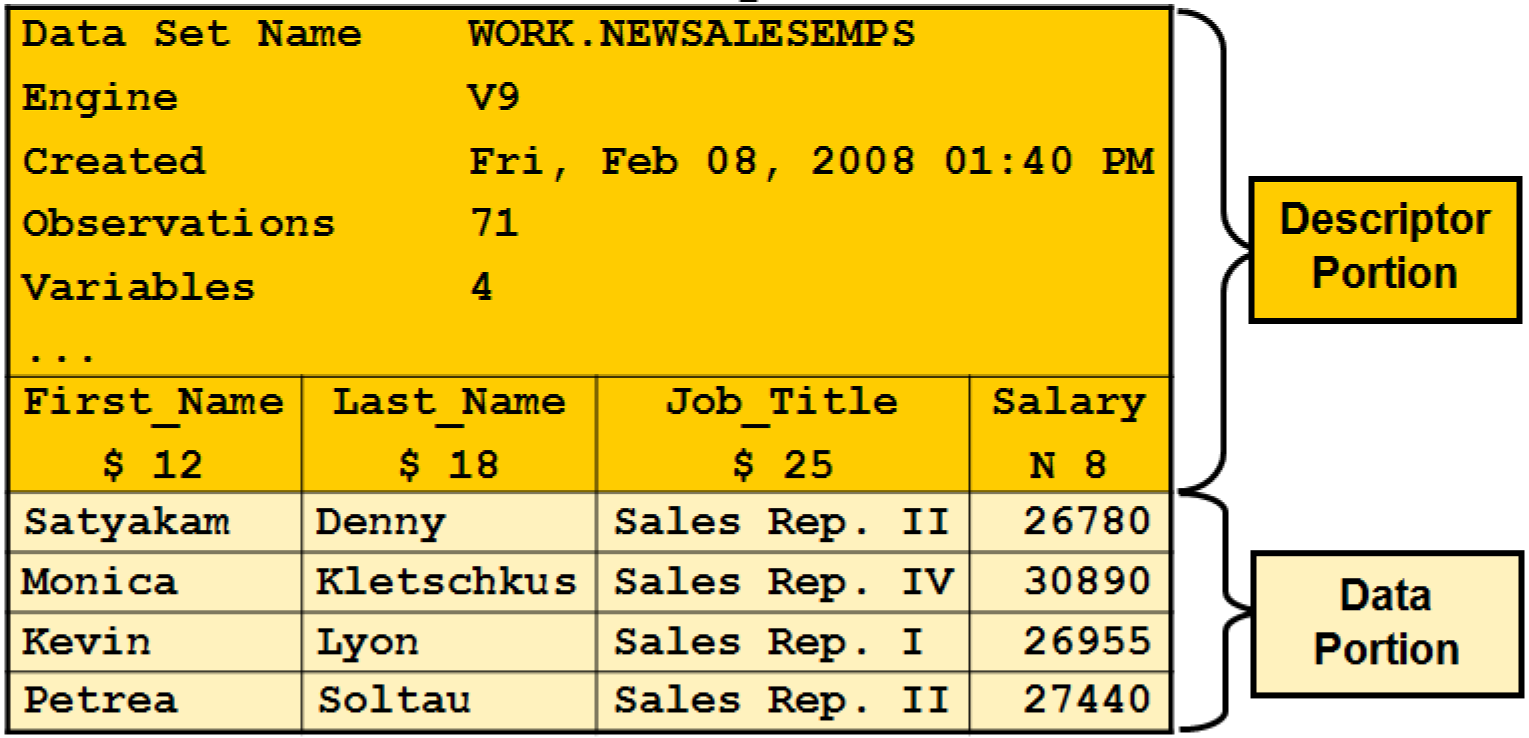
Lecture 2

Data Sets

SAS Data Set

- A *SAS data set* is a file that SAS creates and processes.

Partial Work.NewSalesEmps



```
Data Set Name       WORK.NEWSALESEMPS
Engine              V9
Created             Fri, Feb 08, 2008 01:40 PM
Observations        71
Variables           4
...
```

First_Name $ 12	Last_Name $ 18	Job_Title $ 25	Salary N 8
Satyakam	Denny	Sales Rep. II	26780
Monica	Kletschkus	Sales Rep. IV	30890
Kevin	Lyon	Sales Rep. I	26955
Petrea	Soltau	Sales Rep. II	27440

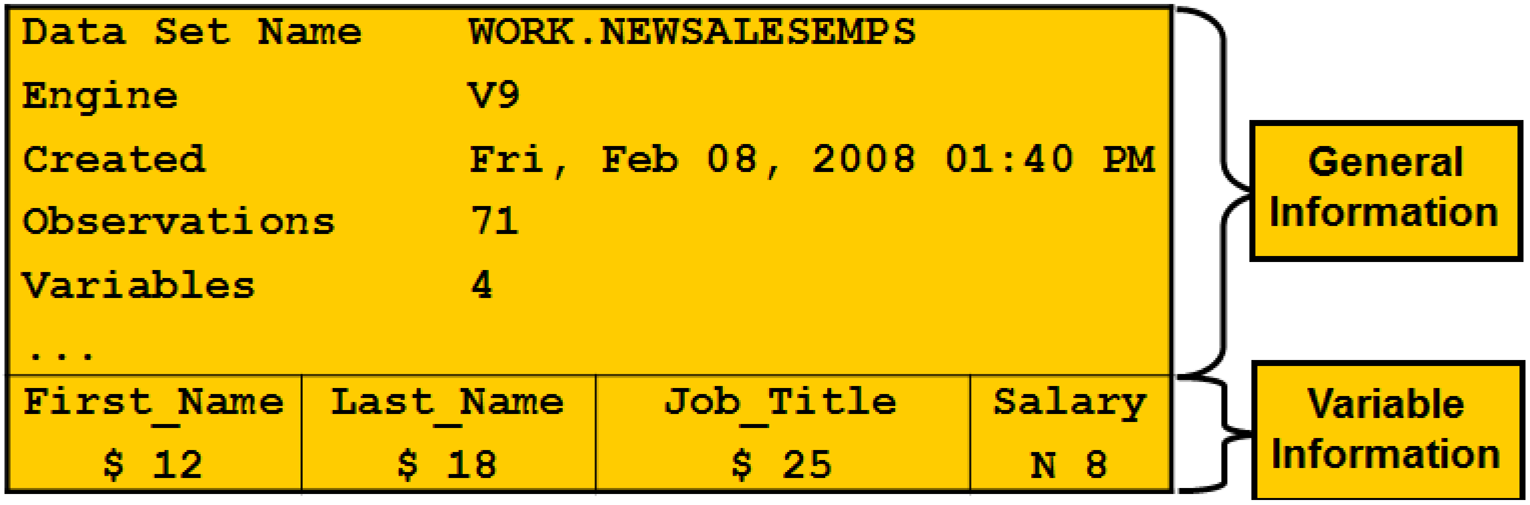
Descriptor Portion
Data Portion

Descriptor Portion

The *descriptor portion* of a SAS data set contains the following:

- general information about the SAS data set (such as data set name and number of observations)
- variable information (such as name, type, and length)

Partial Work.NewSalesEmps



```
Data Set Name     WORK.NEWSALESEMPS
Engine            V9
Created           Fri, Feb 08, 2008 01:40 PM
Observations      71
Variables         4
...
```

First_Name	Last_Name	Job_Title	Salary
$ 12	$ 18	$ 25	N 8

Browsing the Descriptor Portion

The *CONTENTS procedure* displays the descriptor portion of a SAS data set.

General form of the CONTENTS procedure:

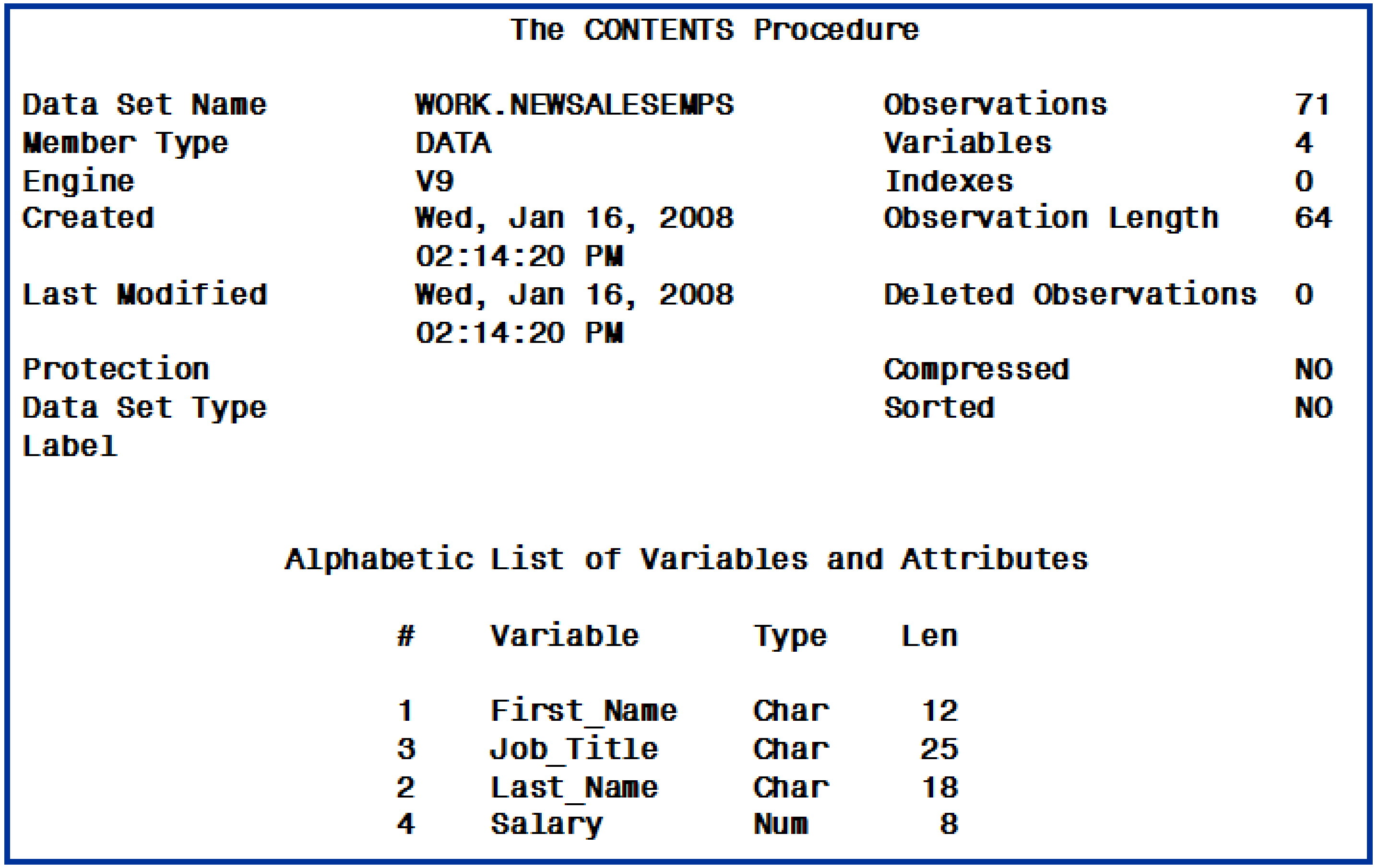
PROC CONTENTS DATA=*SAS-data-set*;
RUN;

Example:

```
proc contents data=work.NewSalesEmps;
run;
```

Browsing the Descriptor Portion

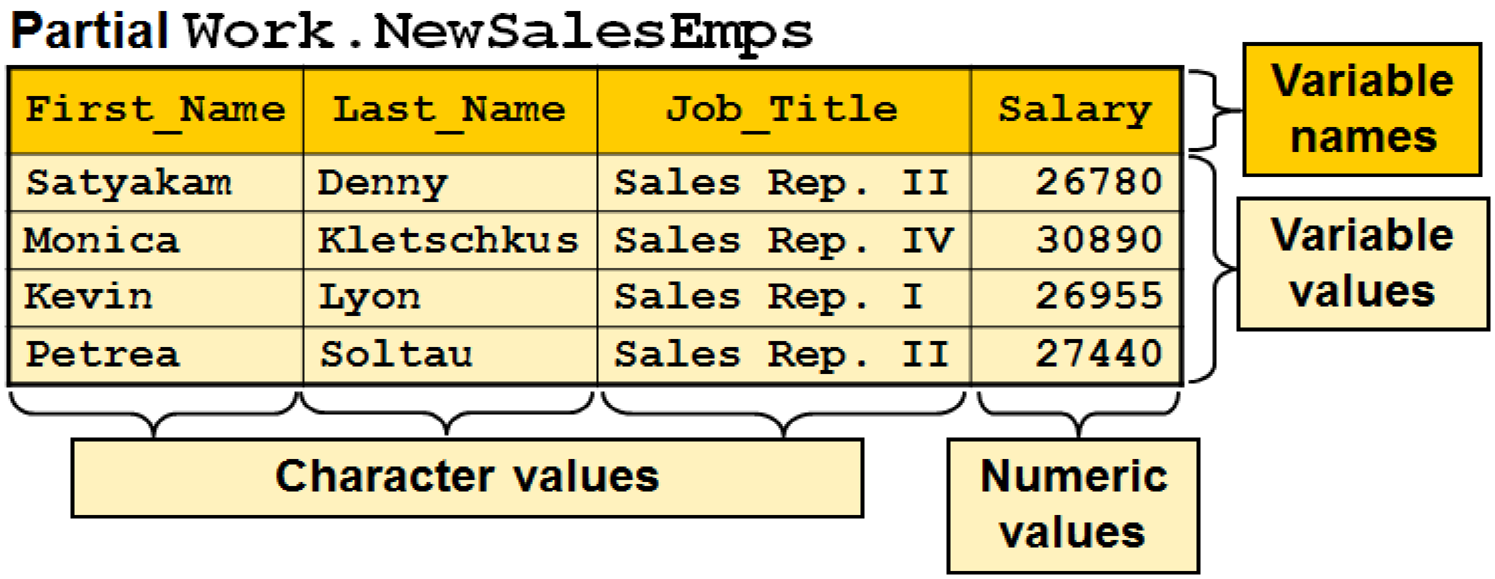
Partial PROC CONTENTS Output



```
                          The CONTENTS Procedure

Data Set Name          WORK.NEWSALESEMPS          Observations           71
Member Type            DATA                       Variables              4
Engine                 V9                         Indexes                0
Created                Wed, Jan 16, 2008          Observation Length     64
                       02:14:20 PM
Last Modified          Wed, Jan 16, 2008          Deleted Observations   0
                       02:14:20 PM
Protection                                        Compressed             NO
Data Set Type                                     Sorted                 NO
Label


              Alphabetic List of Variables and Attributes

                   #    Variable      Type    Len

                   1    First_Name    Char     12
                   3    Job_Title     Char     25
                   2    Last_Name     Char     18
                   4    Salary        Num       8
```

Data Portion

- The *data portion* of a SAS data set is a rectangular table of character and/or numeric data values.



Partial Work.NewSalesEmps

First_Name	Last_Name	Job_Title	Salary
Satyakam	Denny	Sales Rep. II	26780
Monica	Kletschkus	Sales Rep. IV	30890
Kevin	Lyon	Sales Rep. I	26955
Petrea	Soltau	Sales Rep. II	27440

Variable names
Variable values
Character values
Numeric values

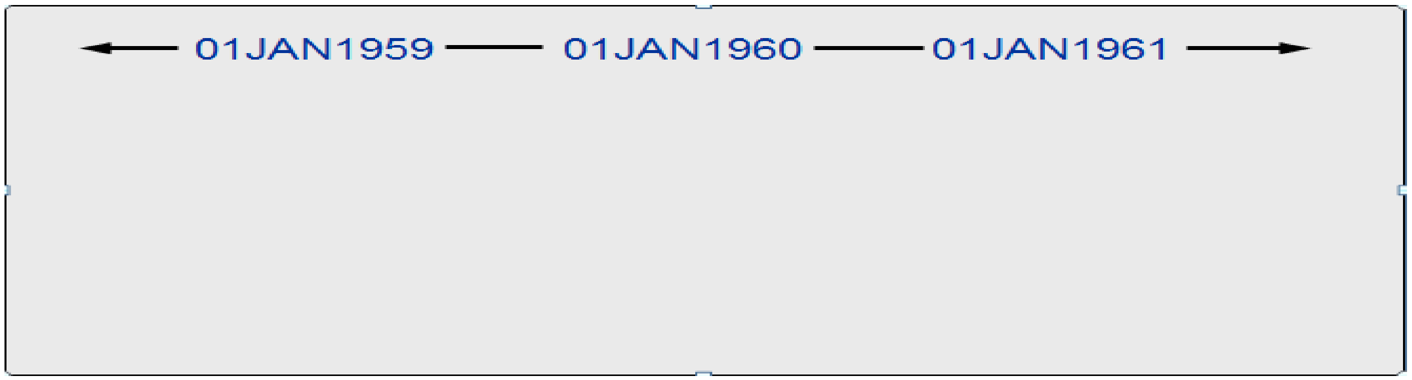
The data values are organized as a table of observations (rows) and variables (columns).

SAS Variable Values

- There are two types of variables:
  - Character
    - Contain any value: letters, numbers, special characters, and blanks.
    - Character values are stored with a length of 1 to 32,767 bytes. One byte equals one character.
  - Numeric
    - Stored as floating point numbers in 8 bytes of storage by default.
    - Eight bytes of floating point storage provide space for 16 or 17 significant digits. You are not restricted to 8 digits.

SAS Date Values

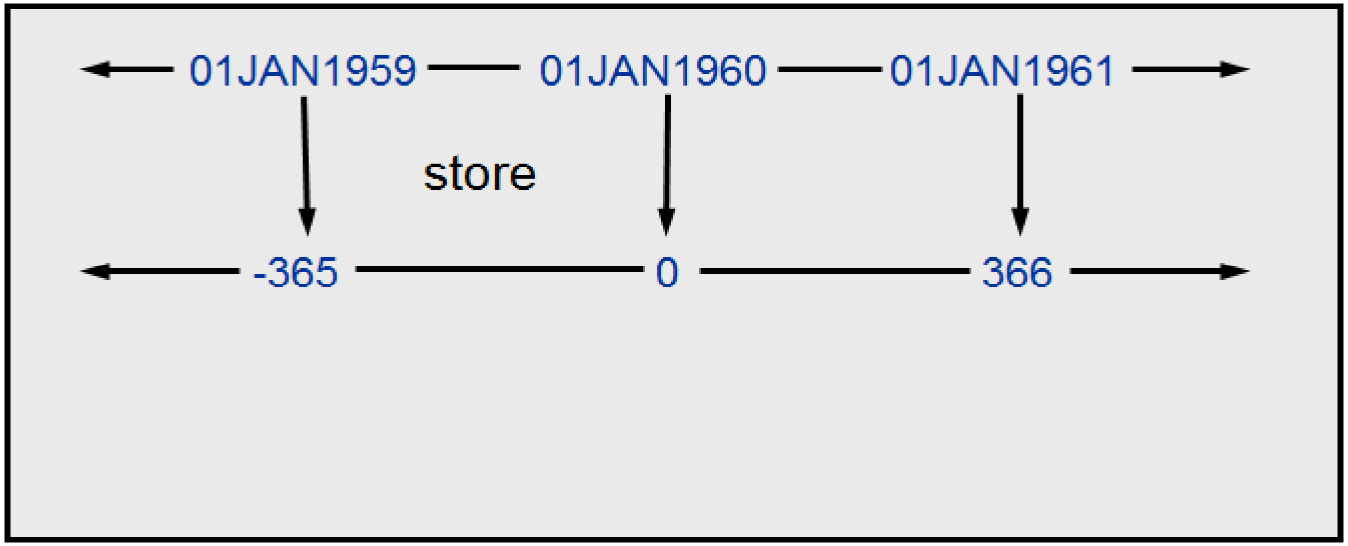
- SAS stores date values as numeric values.



- A *SAS date value* is stored as the number of days between January 1, 1960, and a specific date.

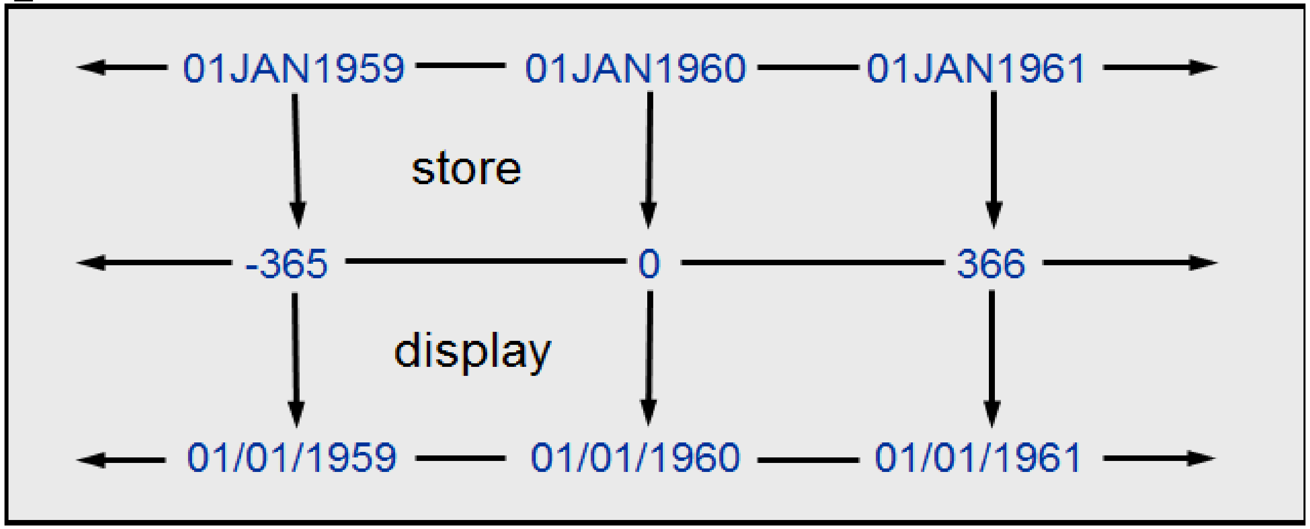
SAS Date Values

- SAS stores date values as numeric values.
- A *SAS date value* is stored as the number of days between January 1, 1960, and a specific date.



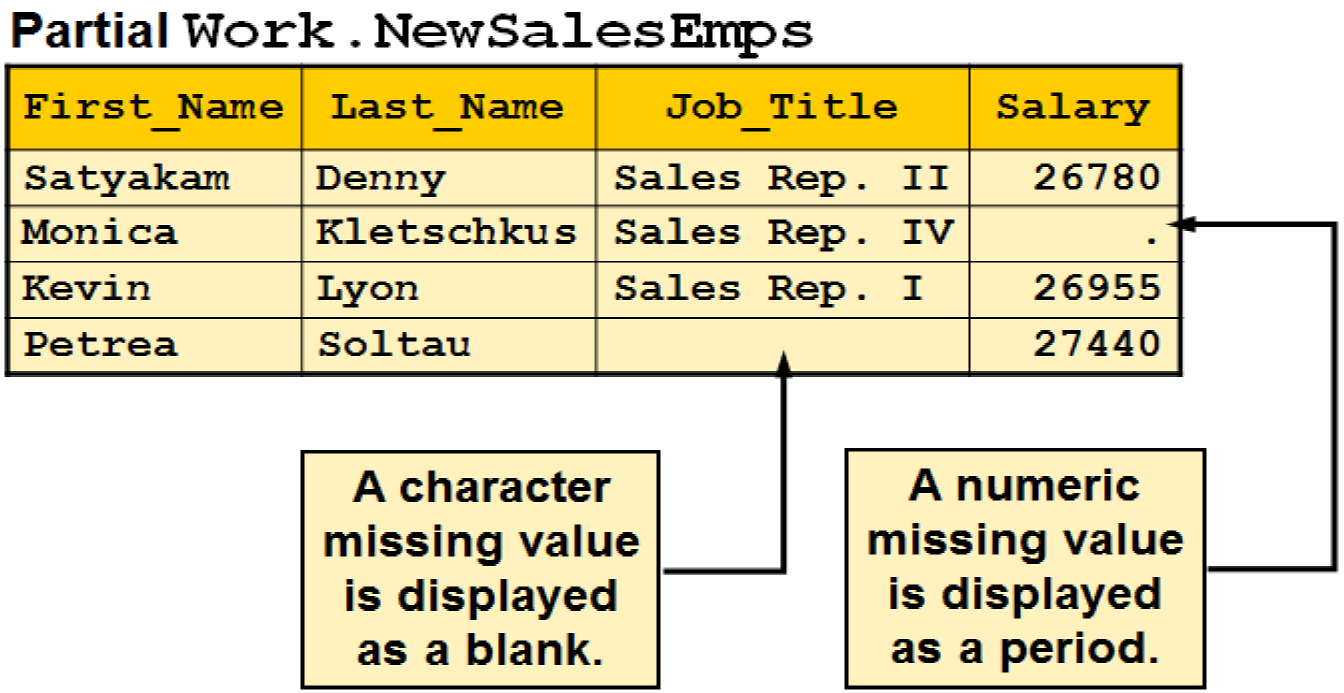
SAS Date Values

- SAS stores date values as numeric values.
- A *SAS date value* is stored as the number of days between January 1, 1960, and a specific date.



Missing Data Values

- A value must exist for every variable for each observation. Missing values are valid values in a SAS data set.



Partial `Work.NewSalesEmps`

First_Name	Last_Name	Job_Title	Salary
Satyakam	Denny	Sales Rep. II	26780
Monica	Kletschkus	Sales Rep. IV	.
Kevin	Lyon	Sales Rep. I	26955
Petrea	Soltau		27440

A character missing value is displayed as a blank.

A numeric missing value is displayed as a period.

SAS Data Set and Variable Names

SAS names have these characteristics:

- can be 32 characters long.
- must start with a letter or underscore. Subsequent characters can be letters, underscores, or numerals.
- can be uppercase, lowercase, or mixed case.
- are not case sensitive.

SAS Data Set Terminology

Comparable Terminology:

SAS Data Set	$\longleftrightarrow$	SAS Table
Observation	$\longleftrightarrow$	Row
Variable	$\longleftrightarrow$	Column

- The terminology of data set, observation, and variable is specific to SAS.
- The terminology of table, row, and column is common among databases.

Browsing the Data Portion

- The *PRINT procedure* displays the data portion of a SAS data set.
- By default, PROC PRINT displays the following:
  - all observations
  - all variables
  - an Obs column on the left side

Browsing the Data Portion

General form of the PRINT procedure:

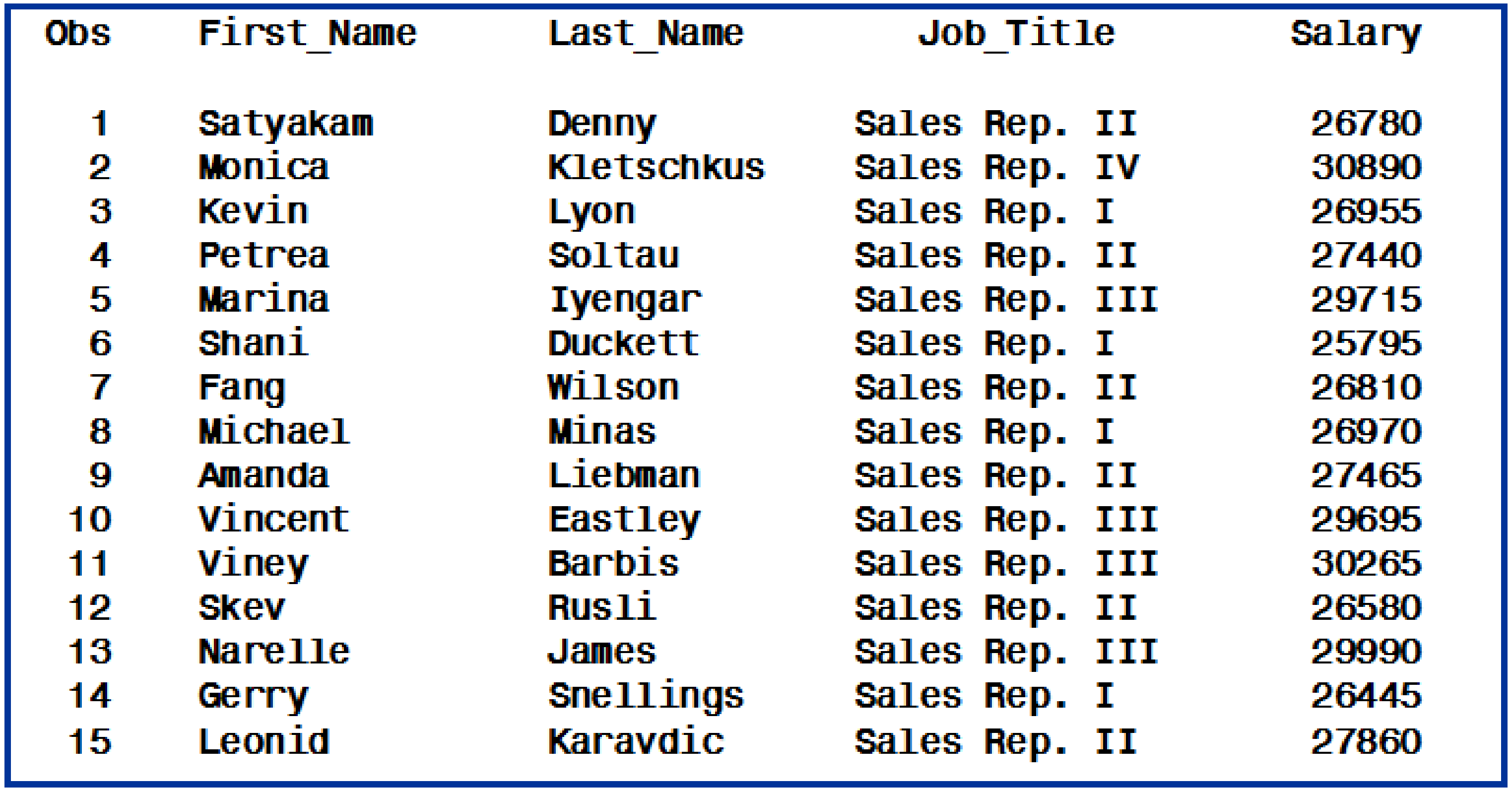
PROC PRINT DATA=*SAS-data-set*;
RUN;

Example:

```
proc print data=work.NewSalesEmps;
run;
```

Browsing the Data Portion

Partial PROC PRINT Output



```
Obs    First_Name      Last_Name        Job_Title            Salary

  1    Satyakam        Denny         Sales Rep. II           26780
  2    Monica          Kletschkus    Sales Rep. IV           30890
  3    Kevin           Lyon          Sales Rep. I            26955
  4    Petrea          Soltau        Sales Rep. II           27440
  5    Marina          Iyengar       Sales Rep. III          29715
  6    Shani           Duckett       Sales Rep. I            25795
  7    Fang            Wilson        Sales Rep. II           26810
  8    Michael         Minas         Sales Rep. I            26970
  9    Amanda          Liebman       Sales Rep. II           27465
 10    Vincent         Eastley       Sales Rep. III          29695
 11    Viney           Barbis        Sales Rep. III          30265
 12    Skev            Rusli         Sales Rep. II           26580
 13    Narelle         James         Sales Rep. III          29990
 14    Gerry           Snellings     Sales Rep. I            26445
 15    Leonid          Karavdic      Sales Rep. II           27860
```

Browsing the Data Portion

Options and statements can be added to the PRINT procedure.

```
PROC PRINT DATA=SAS-data-set NOOBS;
      VAR variable(s);
RUN;
```

- The NOOBS option suppresses the observation numbers on the left side of the report.
- The VAR statement selects variables that appear in the report and determines their order.

Browsing the Data Portion

```
proc print data=work.NewSalesEmps noobs;
   var Last_Name First_Name Salary;
run;
```

Partial PROC PRINT Output

```
Last_Name      First_Name      Salary

Denny          Satyakam         26780
Kletschkus     Monica           30890
Lyon           Kevin            26955
Soltau         Petrea           27440
Iyengar        Marina           29715
Duckett        Shani            25795
Wilson         Fang             26810
Minas          Michael          26970
Liebman        Amanda           27465
Eastley        Vincent          29695
```

SAS Data Sets

- SAS data sets are available in two varieties:
  - Temporary
  - Permanent
- SAS data set names have two level names such as work.distance, where the two levels are separated by a period.
  - The first level of a SAS data set name, work in this case, is called its libref (short for SAS data library reference).
  - A libref is like an arrow pointing to a particular location.

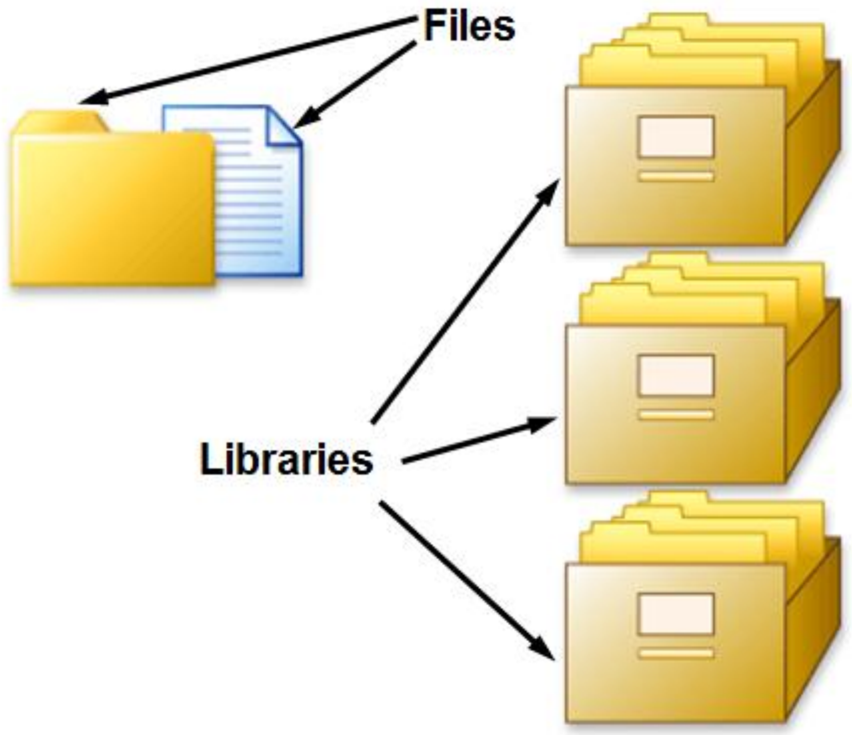
SAS Data Libraries

- A *SAS data library* is a collection of SAS files that are recognized as a unit by SAS.

Directory-based System	A SAS data library is a directory.
Windows Example:	`s:\workshop`

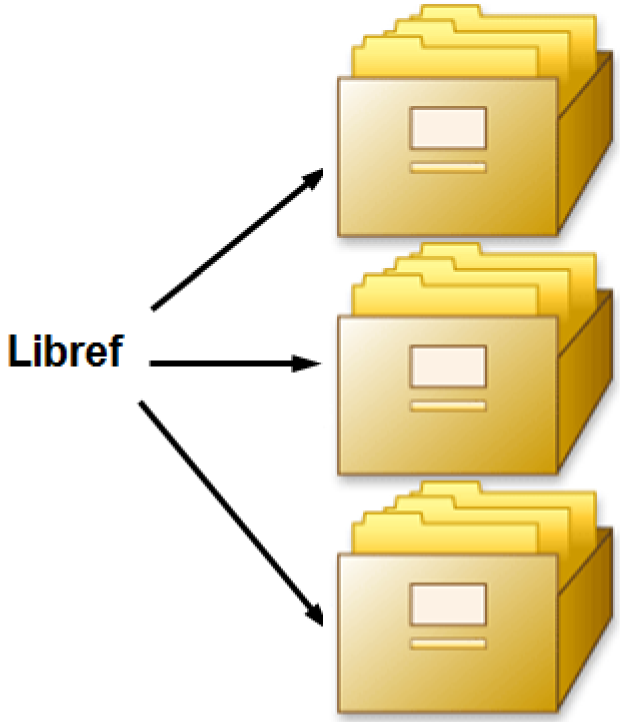
SAS Data Libraries

You can think of a SAS data library as a drawer in a filing cabinet and a SAS data set as one of the file folders in the drawer.



Assigning LIBREF

Regardless of which host operating system you use, you identify SAS data libraries by assigning a *library reference name (libref)* to each library.

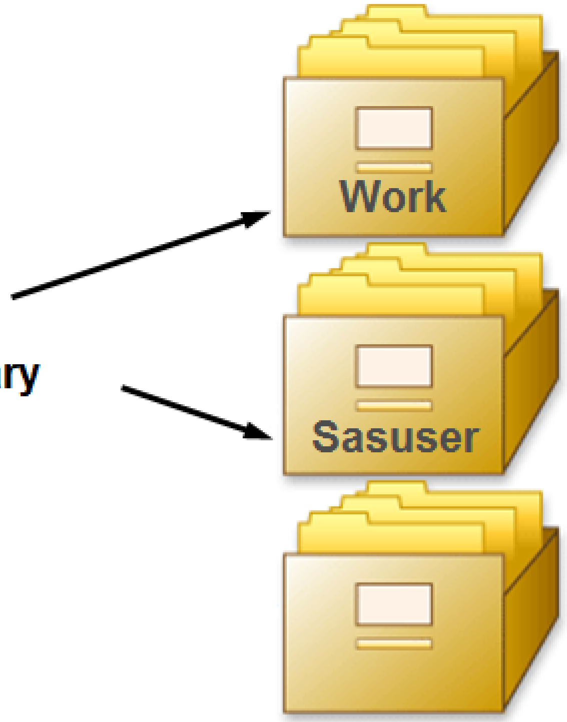


SAS Data Libraries

When a SAS session starts, SAS automatically creates one temporary and at least one permanent SAS data library that you can access.

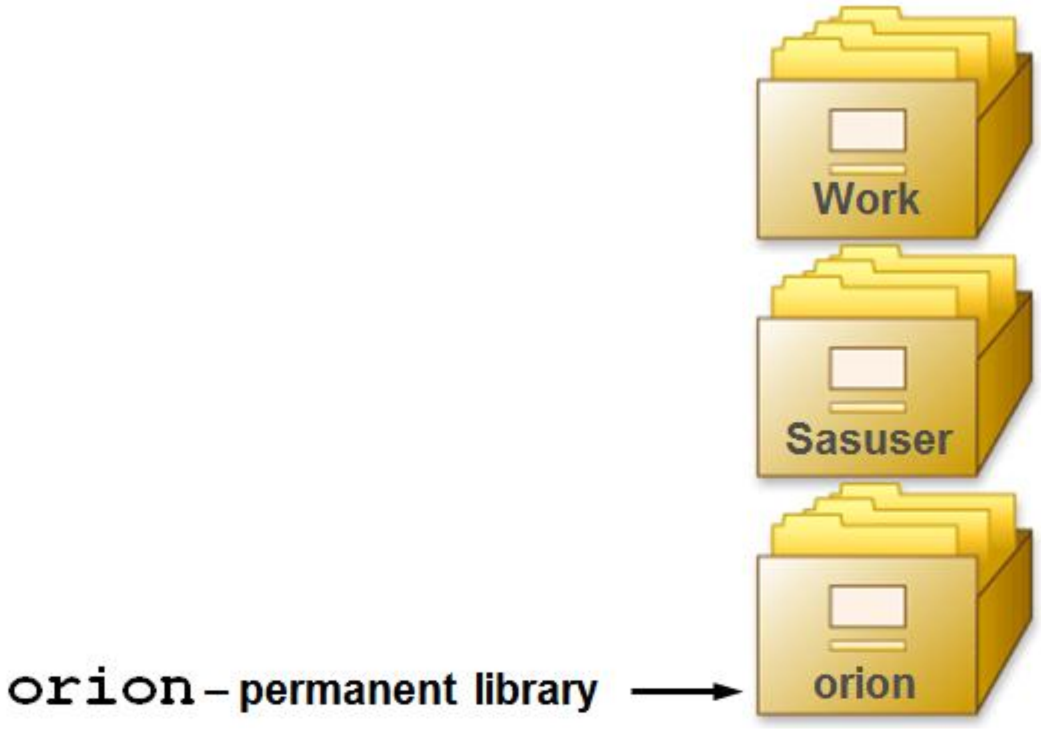
Work – **temporary library**

Sasuser – **permanent library**



SAS Data Libraries

You can also create and access your own permanent libraries.



Assigning a LIBREF

- You can use the *LIBNAME statement* to assign a library reference name (libref) to a SAS data library.
- General form of the LIBNAME statement:

LIBNAME *libref 'SAS-data-library' <options>*;

- Rules for naming a libref:
  - The name must be 8 characters or less.
  - The name must begin with a letter or underscore.
  - The remaining characters must be letters, numerals, or underscores.

Assigning a LIBREF

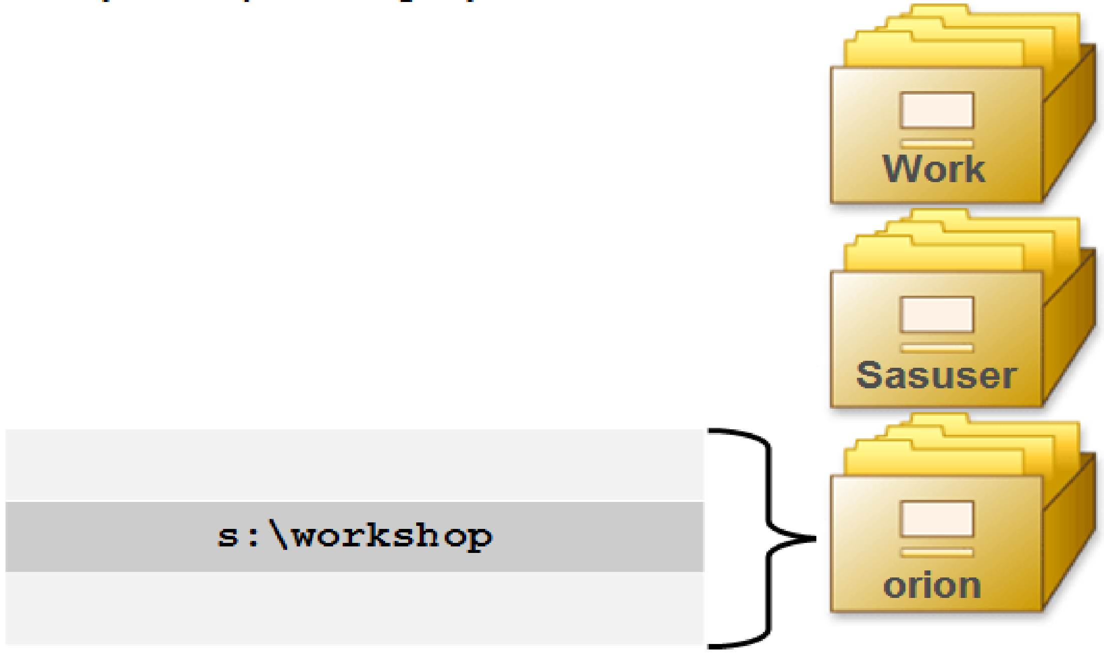
- Examples:
  - **Windows**

```
libname orion 's:\workshop';
```

Assigning a LIBREF

- Making the connection

When you submit the LIBNAME statement, a connection is made between a libref in SAS and the physical location of files on your operating system.

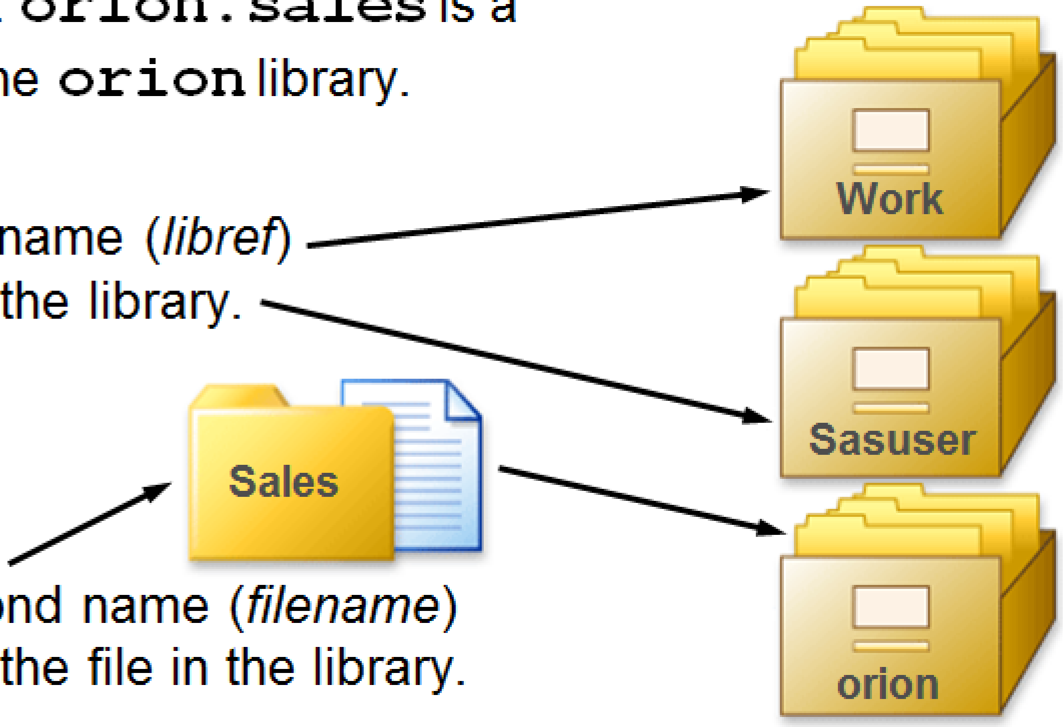


Two-Level SAS File Names

Every SAS file has a two-level name: *libref.filename*

The data set **`orion.sales`** is a SAS file in the **`orion`** library.

- The first name (*libref*) refers to the library.
- The second name (*filename*) refers to the file in the library.



Temporary SAS File Name

The default libref is Work if the libref is omitted.

`distance` ⟷ `work.distance`

```
data distance;
     Miles =26.22;
     Kilometers =1.61*Miles
run;

proc print data=distance;;
run;
```

Temporary SAS File Name

- The following program creates and then prints a temporary SAS data set named distance.

The default libref is Work if the libref is omitted.

`distance` ⟷ `work.distance`

```
data distance;
         Miles =26.22;
         Kilometers =1.61*Miles
run;

proc print data=distance;;
run;
```

Permanent SAS File Name

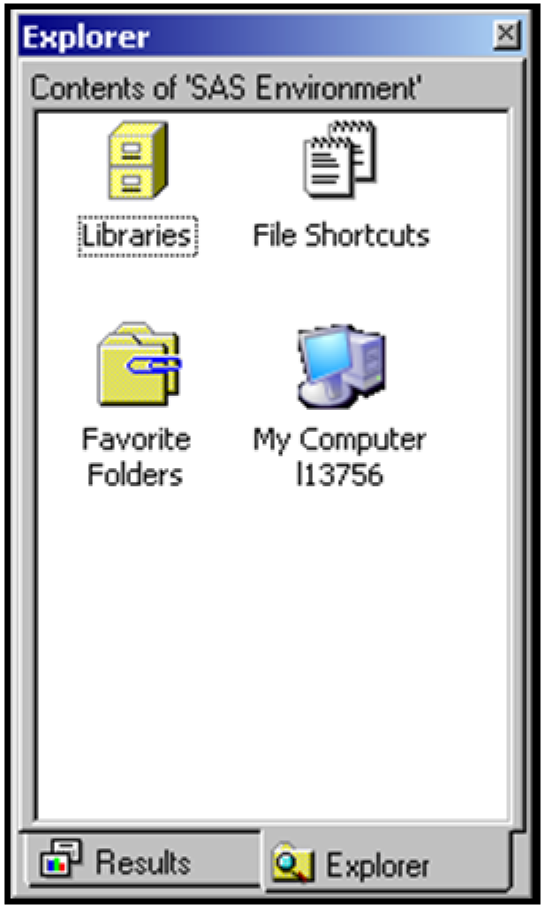
- The following program is the same as the preceding one except that it creates a permanent SAS data set.
  - Notice that the two –level name appears in the DATA statement.

```
data bikes.distance;
     Miles =26.22;
     Kilometers =1.61*Miles
run;

proc print data=bikes.distance;;
run;
```

Browsing A SAS Data Library

- Once the libref is assigned, you can view the contents of the data library by using the SAS Explorer window.

Browsing a SAS Data Library



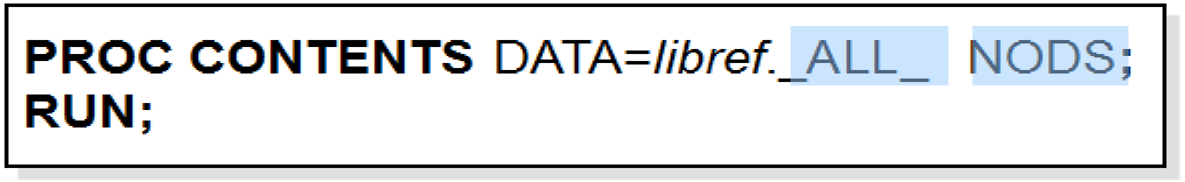
The *SAS Explorer* enables you to manage your files in the windowing environment.

In the SAS Explorer,
you can do the following:

- view a list of all the libraries available during your current SAS session
- navigate to see all members of a specific library
- display the descriptor portion of a SAS data set

Browsing a SAS Data Library

- The CONTENTS procedure with the _ALL_ keyword produces a list of all the SAS files in the data library.



```
PROC CONTENTS DATA=libref._ALL_ NODS;
RUN;
```

- The NODS option suppresses the descriptor portions of the data sets.
- NODS is only used in conjunction with the keyword _ALL_.

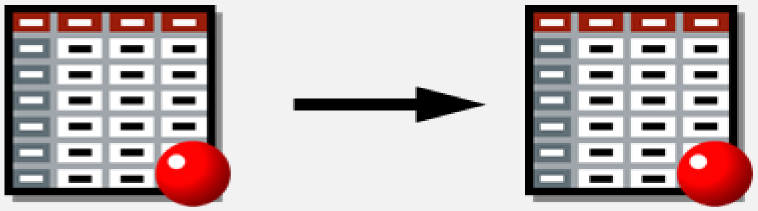
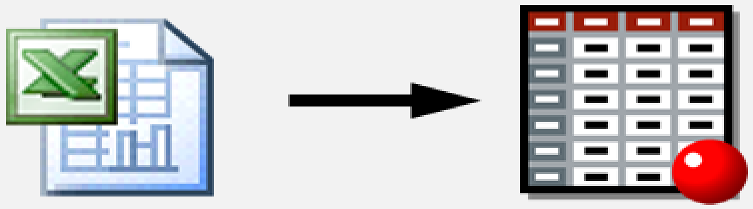
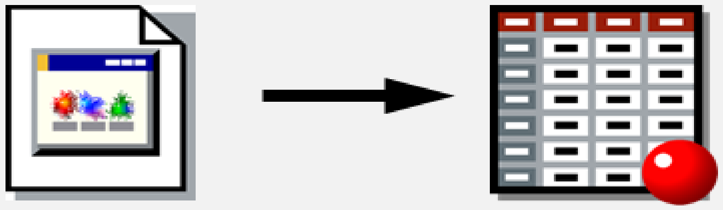
External Data Files

- Reading From a Data Source to Create a SAS Data Set .
  - If you collect a set of data that has many observations, you may want to put it in a separate file that you can easily access.
  - It would be a waste of time and effort if you had to retype the data in your SAS program whenever you need to use it.

External Data Files

- We will look at three different data sources to create a new SAS data set.
  - SAS data sets
  - Excel workbooks
  - Raw data files

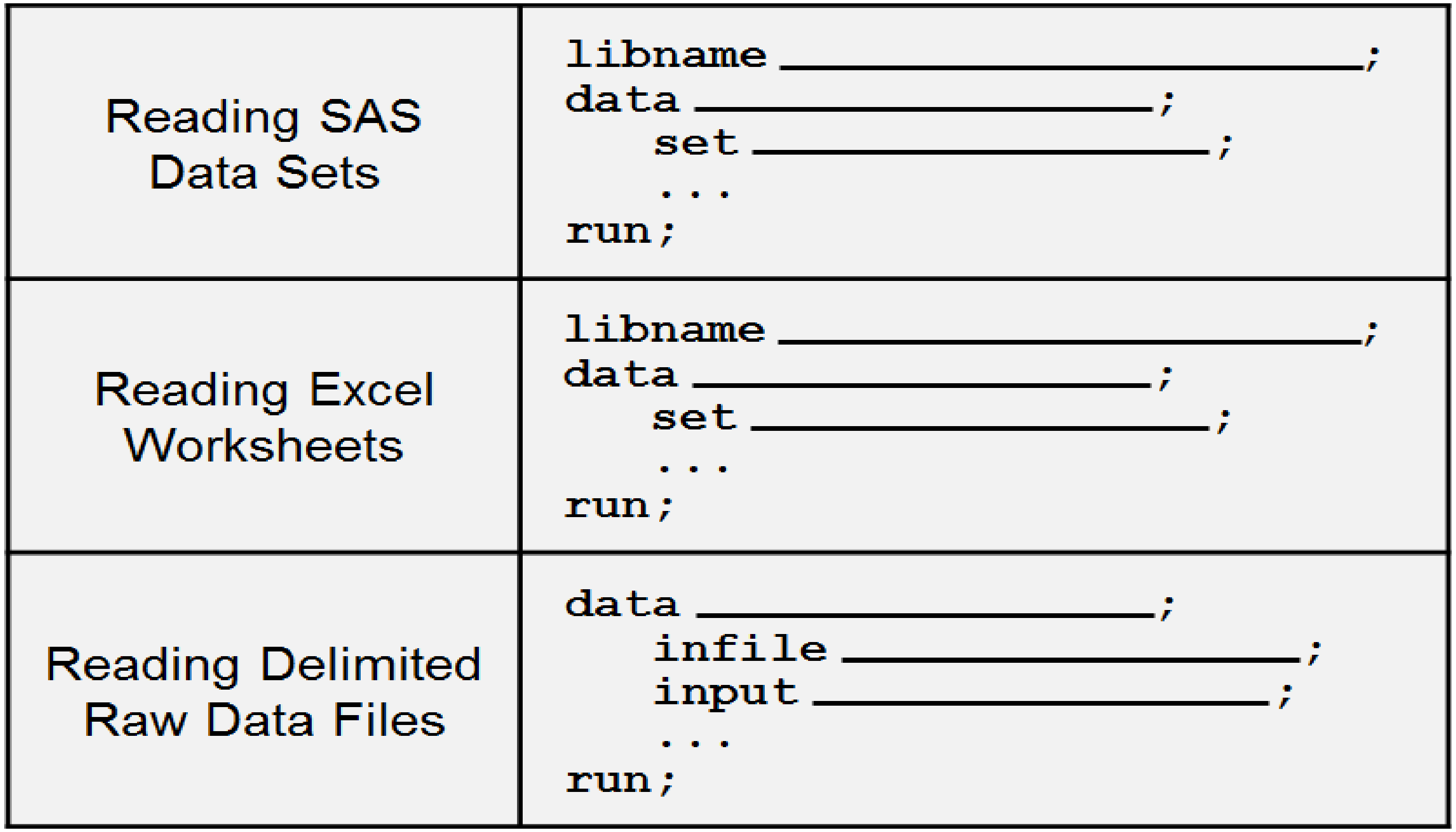
External Data Files

Reading SAS Data Sets	
Reading Excel Worksheets	
Reading Delimited Raw Data Files	

External Data Files

- The Data step is used to accomplish the scenario regardless of the input data source
- Additional statements are added to the Data Step to complete all of the requirements
- The libname statement references a SAS data library when reading a SAS data set, and an Excel workbook when reading an Excel worksheet.

External Data Files



Reading SAS Data Sets

```
libname ______________________________;
data _______________________;
     set _______________________;
     ...
run;
```

Reading Excel Worksheets

```
libname ______________________________;
data _______________________;
     set _______________________;
     ...
run;
```

Reading Delimited Raw Data Files

```
data _______________________;
     infile _______________________;
     input _______________________;
     ...
run;
```

Business Scenario

- An existing data source contains information on Orion Star sales employees from Australia and the United States.
- A new SAS data set needs to be created that contains a subset of this existing data source.
- This new SAS data set must contain the following:
  - only the employees from Australia who are Sales Representatives
  - the employee's first name, last name, salary, job title, and hired date
  - labels and formats in the descriptor portion

Reading SAS Data Sets

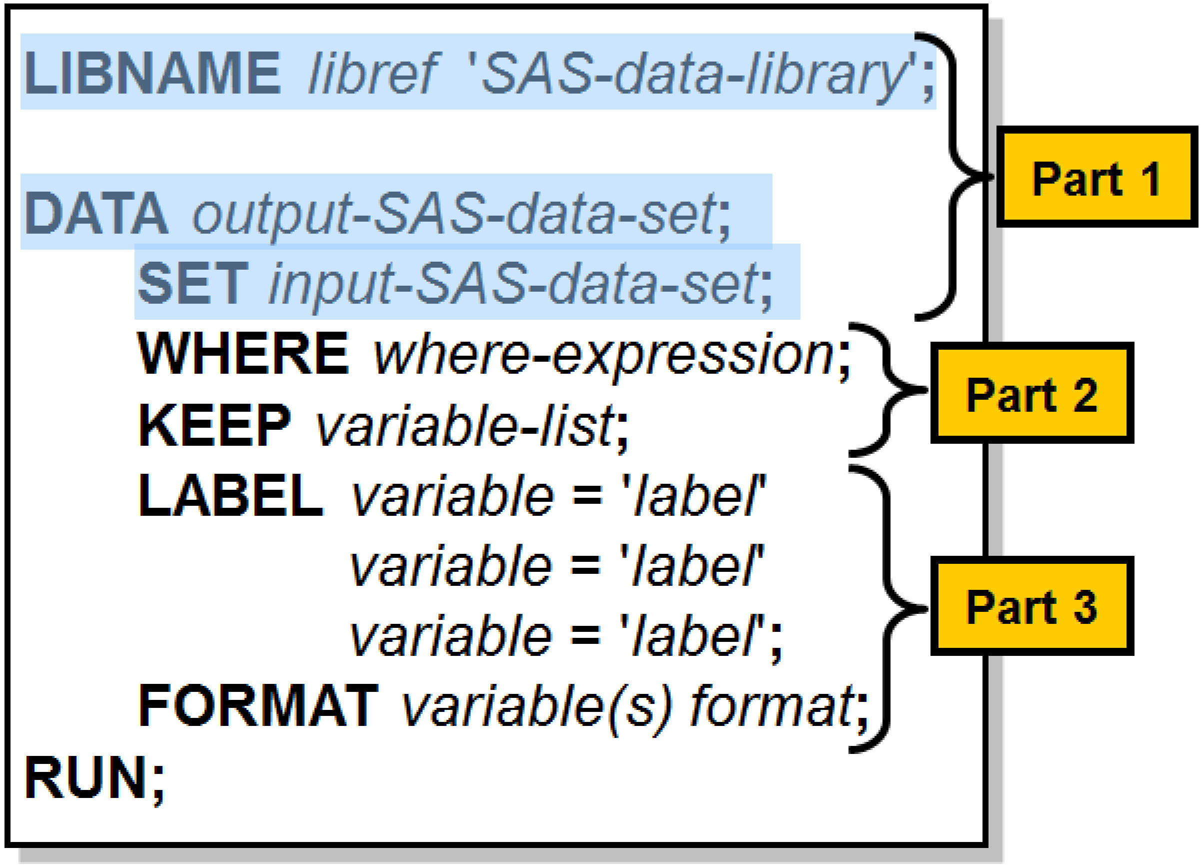
- Use the following statements to perform the task.

```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
            variable = 'label'
            variable = 'label';
      FORMAT variable(s) format;
RUN;
```

Reading SAS Data Sets

- Instead of writing the program all at once, break the program into three parts. Test each part before you move to the next part.

Reading SAS Data Sets

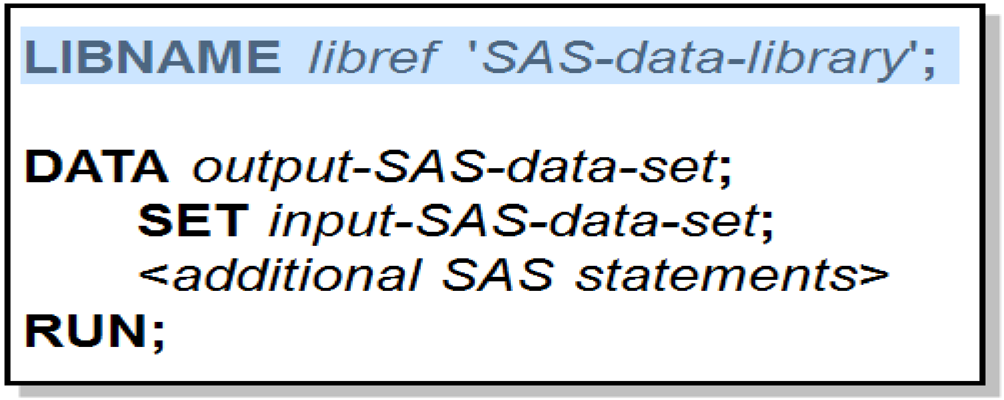


```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
            variable = 'label'
            variable = 'label';
      FORMAT variable(s) format;
RUN;
```

Part 1
Part 2
Part 3

Libname Statement Review

- A library reference name (libref) is needed if a permanent data set is being read or created.

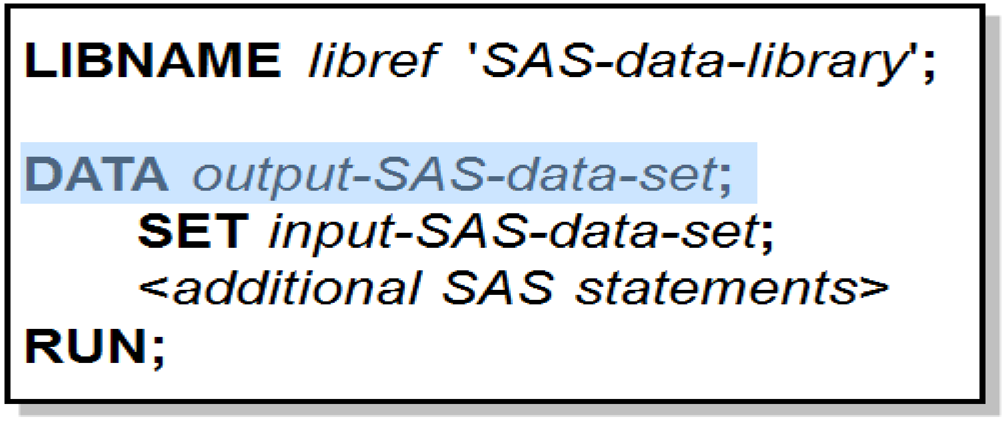


```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      <additional SAS statements>
RUN;
```

- The *LIBNAME statement* assigns a libref to a SAS data library.

The Data Statement

- The *DATA statement* begins a DATA step and provides the name of the SAS data set being created.

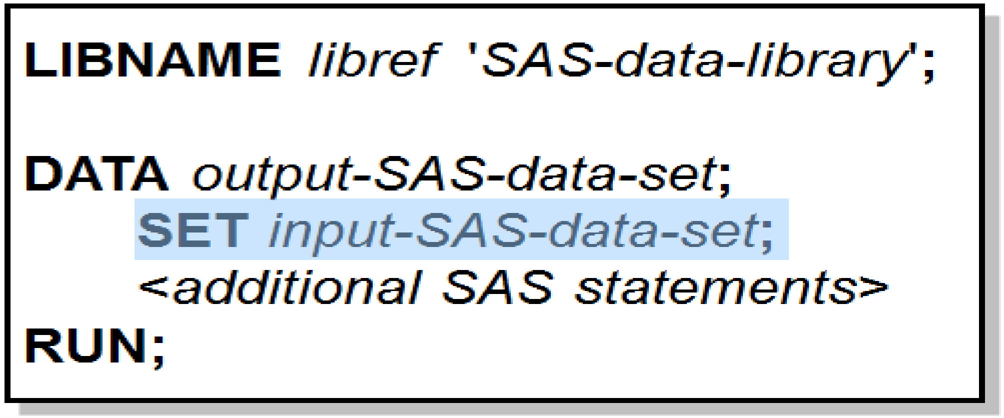


```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      <additional SAS statements>
RUN;
```

- The DATA statement can create temporary or permanent data sets.

The Set Statement

- The *SET statement* reads observations from a SAS data set for further processing in the DATA step.



```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
     SET input-SAS-data-set;
     <additional SAS statements>
RUN;
```

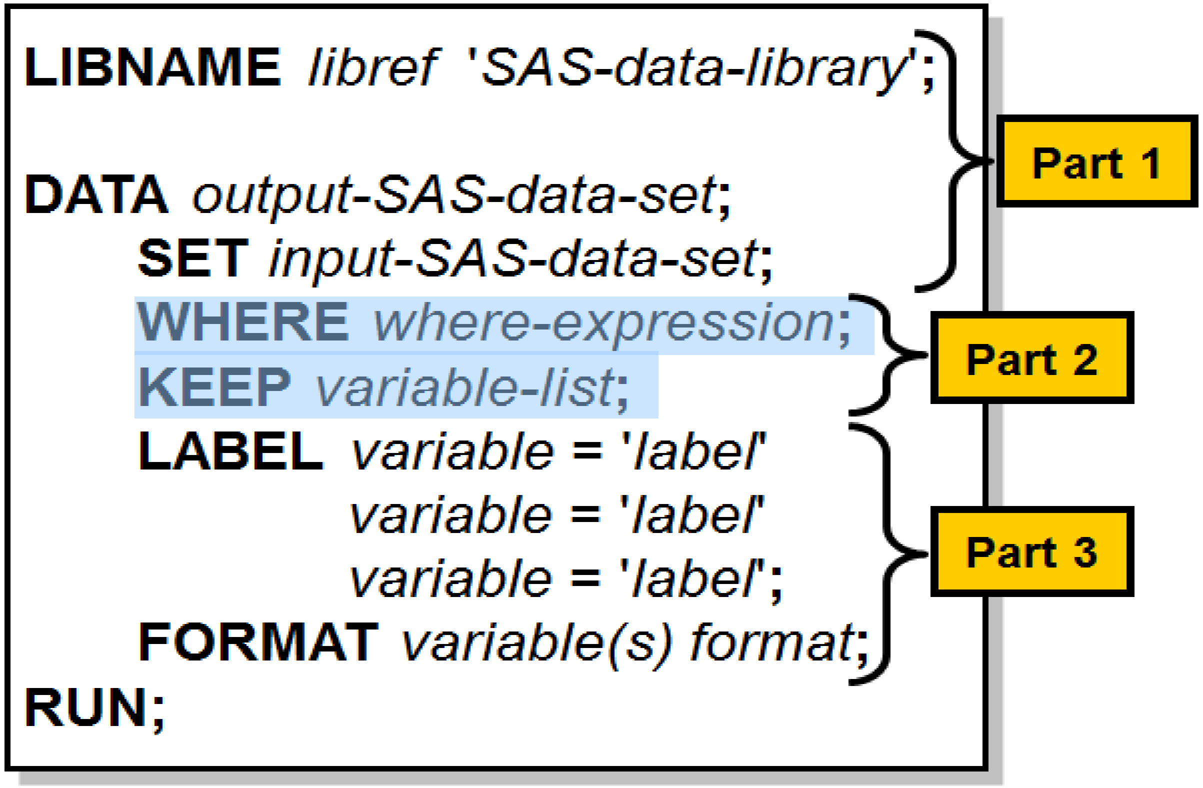
- By default, the SET statement reads all observations and all variables from the input data set.
- The SET statement can read temporary or permanent data sets.

Subsetting Observations and Variables

- Subset Observations by using the WHERE statement.
- Subset variables by using the DROP and Keep statements.

Subsetting Observations and Variables

- Use the following to perform the task.



```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
            variable = 'label'
            variable = 'label';
      FORMAT variable(s) format;
RUN;
```

Subsetting Observations and Variables

- By default, the SET statement reads **all observations** and **all variables** from the input data set.
- By adding statements to the DATA step, the number of observations and variables can be reduced.

The Where Statement

- The *WHERE statement* subsets observations that meet a particular condition.
- General form of the WHERE statement:

```
WHERE where-expression ;
```

- The *where-expression* is a sequence of operands and operators that form a set of instructions that define a condition for selecting observations.
  - Operands include constants and variables.
  - Operators are symbols that request a comparison, arithmetic calculation, or logical operation.

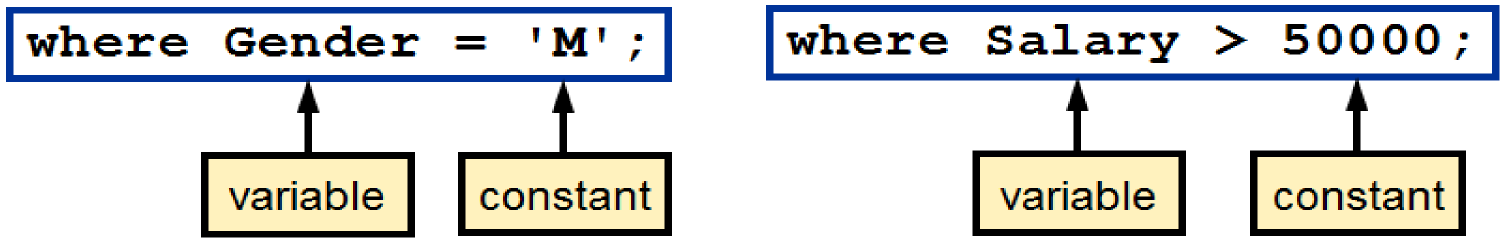
Operand

A *constant operand* is a fixed value.

- Character values must be enclosed in quotation marks and are case sensitive.
- Numeric values do not use quotation marks.

A *variable operand* must be a variable coming from an input data set.

Examples:



```
where Gender = 'M';
```

```
where Salary > 50000;
```

Comparison Operators

Comparison operators compare a variable with a value or with another variable.

Symbol	Mnemonic	Definition
=	EQ	equal to
^= ¬= ~=	NE	not equal to
>	GT	greater than
<	LT	less than
>=	GE	greater than or equal
<=	LE	less than or equal
	IN	equal to one of a list

Comparison Operators

Examples:

```
where Gender = 'M';
```

```
where Gender eq ' ';
```

```
where Salary ne .;
```

```
where Salary >= 50000;
```

```
where Country in ('AU','US');
```

```
where Country in ('AU' 'US');
```

Values must be separated by commas or blanks.

Arithmetic Operators

Arithmetic operators indicate that an arithmetic calculation is performed.

Symbol	Definition
**	exponentiation
*	multiplication
/	division
+	addition
-	subtraction

Arithmetic Operators

Examples:

```
where Salary / 12 < 6000;
```

```
where Salary / 12 * 1.10 >= 7500;
```

```
where (Salary / 12 ) * 1.10 >= 7500;
```

```
where Salary + Bonus <= 10000;
```

Logical Operators

Logical operators combine or modify expressions.

Symbol	Mnemonic	Definition
&	AND	logical and
|	OR	logical or
^ ¬ ~	NOT	logical not

Logical Operators

Examples:

```
where Gender ne 'M' and Salary >=50000;
```

```
where Gender ne 'M' or Salary >= 50000;
```

```
where Country = 'AU' or Country = 'US';
```

```
where Country not in ('AU' 'US');
```

Special Where Operators

Special WHERE operators are operators that can only be used in a where-expression.

Symbol	Mnemonic	Definition
	BETWEEN-AND	an inclusive range
	IS NULL	missing value
	IS MISSING	missing value
?	CONTAINS	a character string
	LIKE	a character pattern

BETWEEN- AND Operator

The *BETWEEN-AND operator* selects observations in which the value of a variable falls within an inclusive range of values.

Examples:

```
where salary between 50000 and 100000;
```

```
where salary not between 50000 and 100000;
```

Equivalent Expressions:

```
where salary between 50000 and 100000;
```

```
where 50000 <= salary <= 100000;
```

IS NULL and IS MISSING Operator

The *IS NULL* and *IS MISSING operators* select observations in which the value of a variable is missing.

- The operator can be used for both character and numeric variables.
- You can combine the NOT logical operator with IS NULL or IS MISSING to select nonmissing values.

Examples:

```
where Employee_ID is null;
```

```
where Employee_ID is missing;
```

CONTAIN Operator

The *CONTAINS (?) operator* selects observations that include the specified substring.

- The position of the substring within the variable's values is not important.
- The operator is case sensitive when you make comparisons.

Example:

```
where Job_Title contains 'Rep';
```

LIKE Operator

The *LIKE operator* selects observations by comparing character values to specified patterns.

There are two special characters available for specifying a pattern:

- A percent sign (%) replaces any number of characters.
- An underscore (_) replaces one character.

Consecutive underscores can be specified.

A percent sign and an underscore can be specified in the same pattern.

The operator is case sensitive.

LIKE Operator

Examples:

```
where Name like '%N';
```

This WHERE statement selects observations that begin with any number of characters and end with an N.

```
where Name like 'T_M%';
```

This WHERE statement selects observations that begin with a T, followed by a single character, followed by an M, followed by any number of characters.

The DROP and KEEP Statements

The *DROP statement* specifies the names of the variables to omit from the output data set(s).

```
DROP variable-list;
```

The *KEEP statement* specifies the names of the variables to write to the output data set(s).

```
KEEP variable-list;
```

The *variable-list* specifies the variables to drop or keep, respectively, in the output data set.

The DROP and KEEP Statements

Examples:

```
drop Employee_ID Gender
     Country Birth_Date;
```

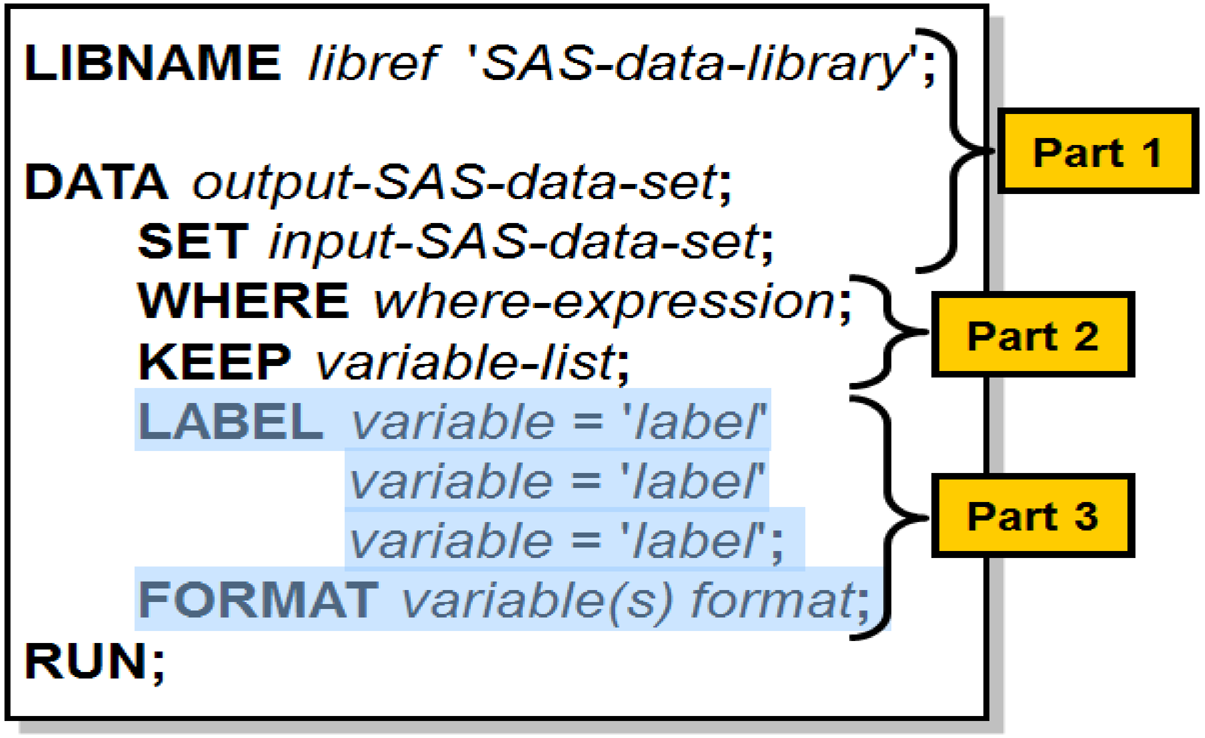
```
keep First_Name Last_Name
     Salary Job_Title
     Hire_Date;
```

Adding Permanent Attributes

- Add labels to the descriptor portion of a SAS data set by using LABEL statement.
- Add format to the descriptor portion of a SAS data set by using FORMAT statement.

Adding Permanent Attributes

- Keep the following statements to perform the task.



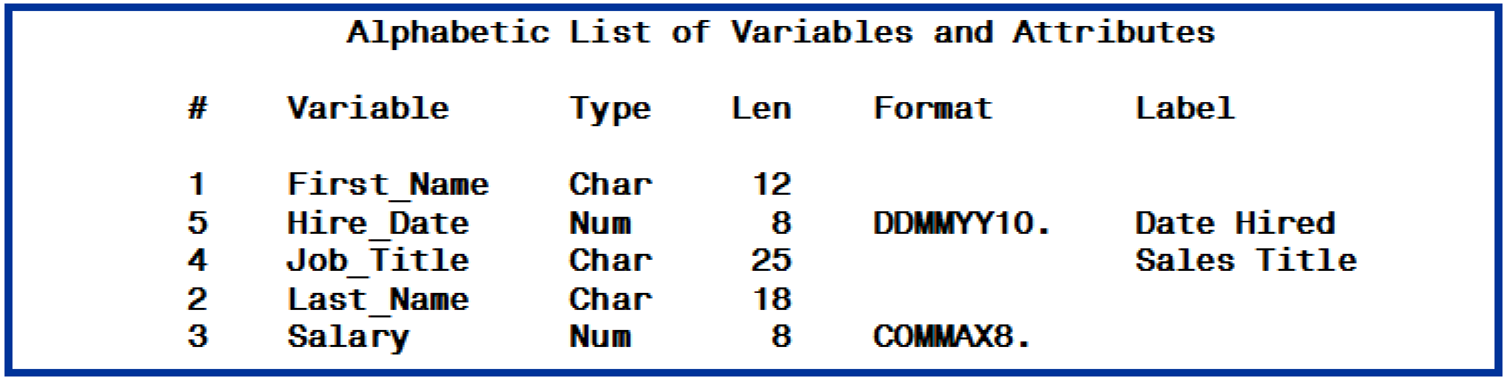
```
LIBNAME libref 'SAS-data-library';

DATA output-SAS-data-set;
      SET input-SAS-data-set;
      WHERE where-expression;
      KEEP variable-list;
      LABEL variable = 'label'
            variable = 'label'
            variable = 'label';
      FORMAT variable(s) format;
RUN;
```

Adding Permanent Attributes

The descriptor portion of the SAS data set stores variable attributes including the name, type (character or numeric), and length of the variable.

Labels and formats can also be stored in the descriptor portion.

Partial PROC CONTENTS Output



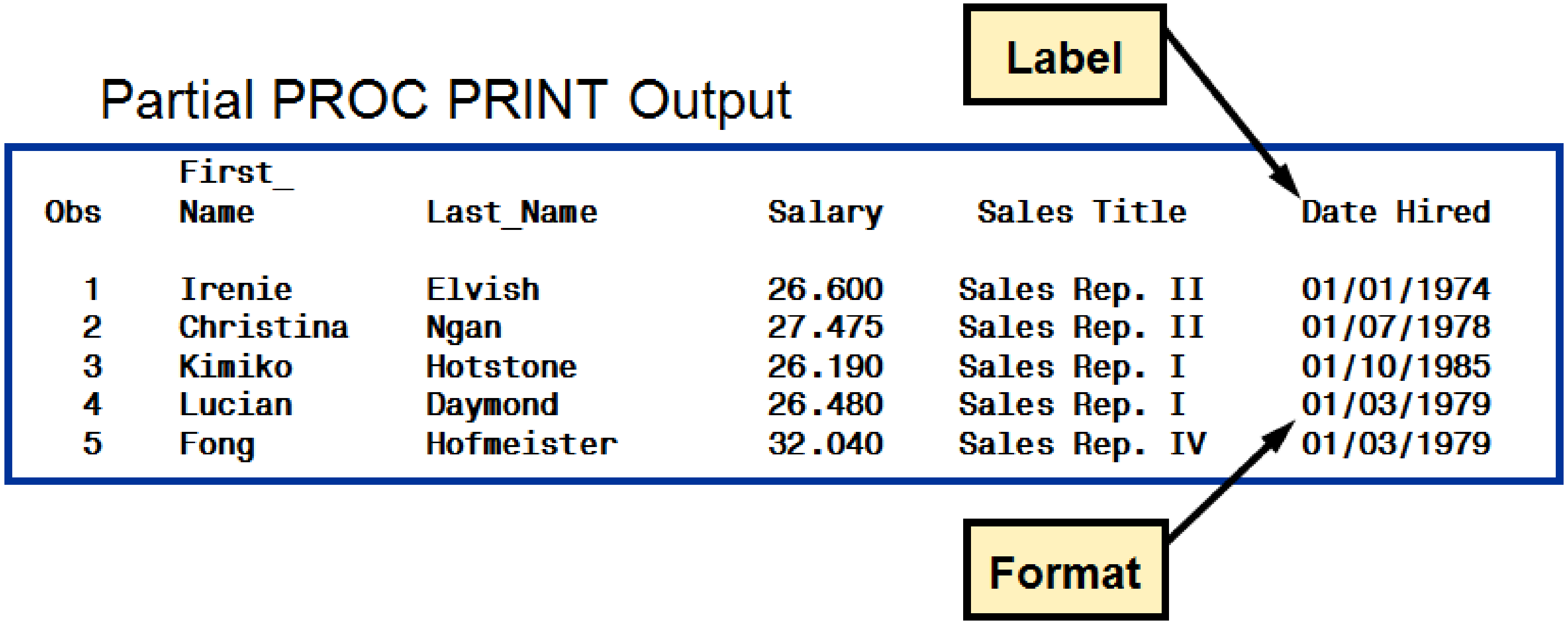
```
                  Alphabetic List of Variables and Attributes

        #     Variable       Type     Len     Format        Label

        1     First_Name     Char      12
        5     Hire_Date      Num        8     DDMMYY10.     Date Hired
        4     Job_Title      Char      25                   Sales Title
        2     Last_Name      Char      18
        3     Salary         Num        8     COMMAX8.
```

Adding Permanent Attributes

When displaying reports,

- a *label* changes the appearance of a variable name
- a *format* changes the appearance of variable value.

Partial PROC PRINT Output



```
        First_
Obs     Name          Last_Name          Salary     Sales Title       Date Hired

  1     Irenie        Elvish             26.600    Sales Rep. II     01/01/1974
  2     Christina     Ngan               27.475    Sales Rep. II     01/07/1978
  3     Kimiko        Hotstone           26.190    Sales Rep. I      01/10/1985
  4     Lucian        Daymond            26.480    Sales Rep. I      01/03/1979
  5     Fong          Hofmeister         32.040    Sales Rep. IV     01/03/1979
```

The Label Statement

The *LABEL statement* assigns descriptive labels to variable names.

General form of the LABEL statement:

```
LABEL variable = 'label'
      variable = 'label'
      variable = 'label';
```

- A label can have as many as 256 characters.
- Any number of variables can be associated with labels in a single LABEL statement.
- Using a LABEL statement in a DATA step permanently associates labels with variables by storing the label in the descriptor portion of the SAS data set.

The FORMAT Statement

The *FORMAT statement* assigns formats to variable values.

General form of the FORMAT statement:

FORMAT *variable(s) format*;

- A *format* is an instruction that SAS uses to write data values.
- Using a FORMAT statement in a DATA step permanently associates formats with variables by storing the format in the descriptor portion of the SAS data set.

SAS Formats

SAS formats have the following form:

```
<$>format<w>.<d>
```

$	indicates a character format.
format	names the SAS format or user-defined format.
w	specifies the total format width including decimal places and special characters.
.	is a required delimiter.
d	specifies the number of decimal places in numeric formats.

SAS Formats

Selected SAS formats:

Format	Definition
$*w*.	writes standard character data.
w.d	writes standard numeric data.
COMMA*w.d*	writes numeric values with a comma that separates every three digits and a period that separates the decimal fraction.
COMMAX*w.d*	writes numeric values with a period that separates every three digits and a comma that separates the decimal fraction.
DOLLAR*w.d*	writes numeric values with a leading dollar sign, a comma that separates every three digits, and a period that separates the decimal fraction.
EUROX*w.d*	writes numeric values with a leading euro symbol (€), a period that separates every three digits, and a comma that separates the decimal fraction.

SAS Formats

Selected SAS formats:

Format	Stored Value	Displayed Value
$4.	Programming	Prog
12.	27134.2864	27134
12.2	27134.2864	27134.29
COMMA12.2	27134.2864	27,134.29
COMMAX12.2	27134.2864	27.134,29
DOLLAR12.2	27134.2864	$27,134.29
EUROX12.2	27134.2864	€27.134,29

SAS Formats

If you do not specify a format width that is large enough to accommodate a numeric value, the displayed value is automatically adjusted to fit into the width.

Format	Stored Value	Displayed Value
DOLLAR12.2	27134.2864	$27,134.29
DOLLAR9.2	27134.2864	$27134.29
DOLLAR8.2	27134.2864	27134.29
DOLLAR5.2	27134.2864	27134
DOLLAR4.2	27134.2864	27E3

SAS Date Formats

SAS date formats display SAS date values in standard date forms.

Format	Stored Value	Displayed Value
MMDDYY6.	0	010160
MMDDYY8.	0	01/01/60
MMDDYY10.	0	01/01/1960
DDMMYY6.	365	311260
DDMMYY8.	365	31/12/60
DDMMYY10.	365	31/12/1960

SAS Date Formats

Additional date formats:

Format	Stored Value	Displayed Value
DATE7.	-1	31DEC59
DATE9.	-1	31DEC1959
WORDDATE.	0	January 1, 1960
WEEKDATE.	0	Friday, January 1, 1960
MONYY7.	0	JAN1960
YEAR4.	0	1960