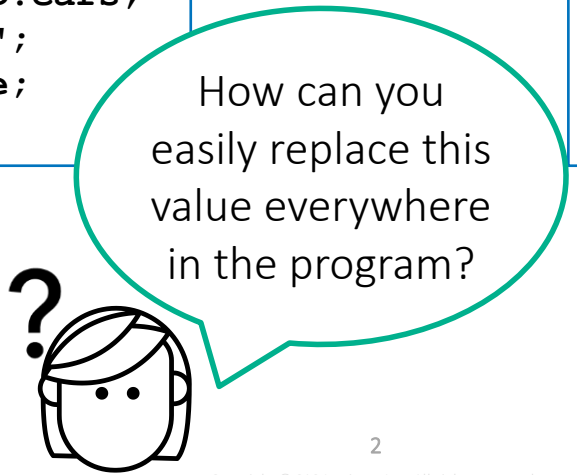
# SAS Lesson 06

# Efficiently Changing the Filter Value

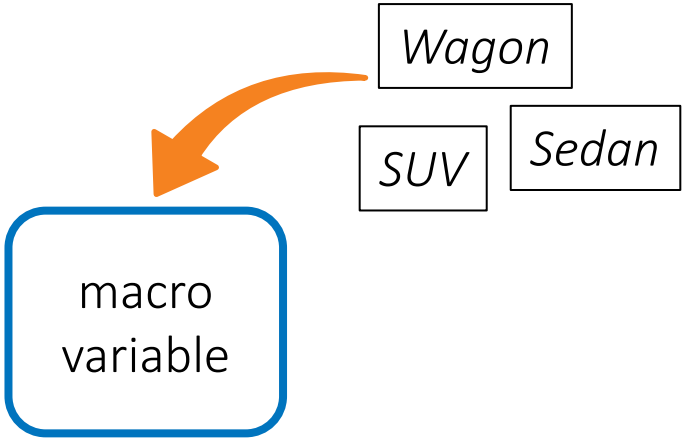
```
proc print data=sashelp.cars;
    where Type="Wagon";
    var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
    where Type="Wagon";
    var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
    where Type="Wagon";
    tables Origin Make;
run;
```

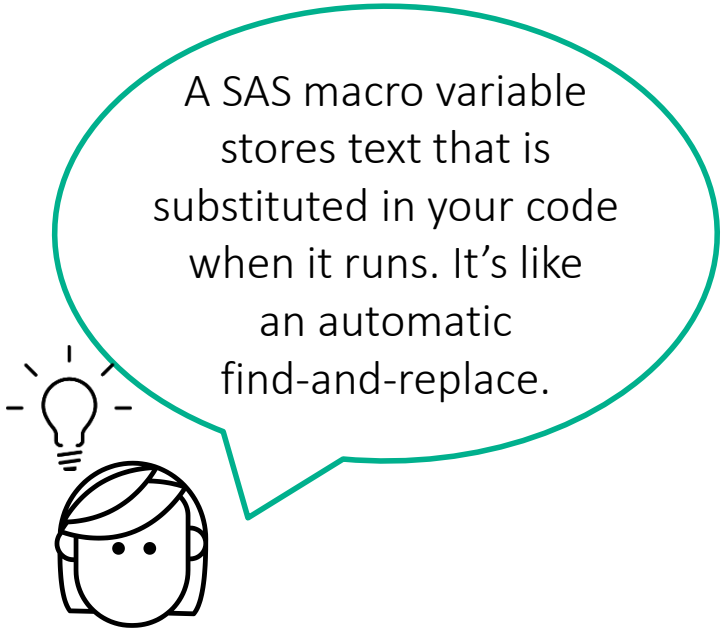
Wagon ⇨ SUV

```
proc print data=sashelp.cars;
    where Type="SUV";
    var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
    where Type="SUV";
    var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
    where Type="SUV";
    tables Origin Make;
run;
```



# Efficiently Changing the Filter Value



# Creating and Using SAS Macro Variables

create the macro variable

```
%let CarType=Wagon;

proc print data=sashelp.cars;
    where Type="Wagon";
    var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
    where Type="Wagon";
    var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
    where Type="Wagon";
    tables Origin Make;
run;
```

%LET *macro-variable*=*value*;

creates a macro variable named **CarType** that stores the text **Wagon**

# Creating and Using SAS Macro Variables

use the macro variable

```
%let CarType=Wagon;

proc print data=sashelp.cars;
    where Type="&CarType";
    var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
    where Type="&CarType";
    var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
    where Type="&CarType";
    tables Origin Make;
run;
```

&*macro-var*

Use the macro variable in place of the value in the program.

# Creating and Using SAS Macro Variables

use the macro variable

```
%let CarType=Wagon;

proc print data=sashelp.cars;
    where Type="Wagon";
    var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
    where Type="Wagon";
    var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
    where Type="Wagon";
    tables Origin Make;
run;
```

SAS replaces **&CarType** with **Wagon** when the program runs.

# Creating and Using SAS Macro Variables

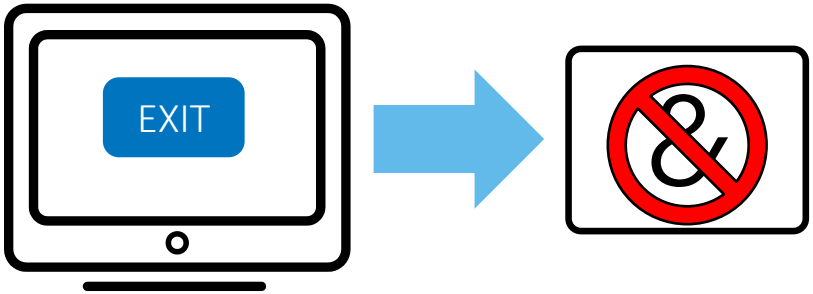
use the macro variable

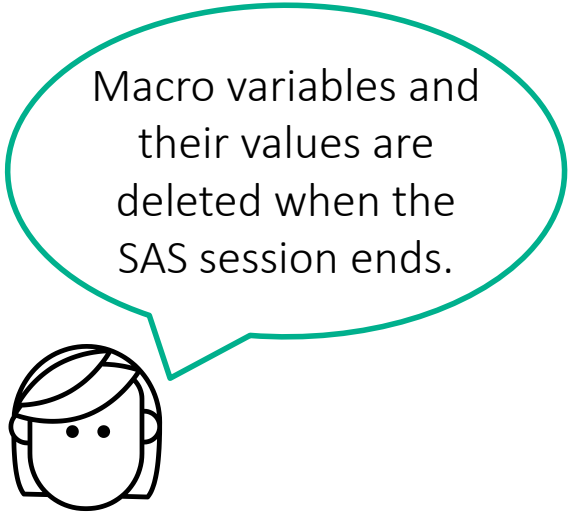
```
%let CarType=SUV;

proc print data=sashelp.cars;
    where Type="SUV";
    var Type Make Model MSRP;
run;
proc means data=sashelp.cars;
    where Type="SUV";
    var MSRP MPG_Highway;
run;
proc freq data=sashelp.cars;
    where Type="SUV";
    tables Origin Make;
run;
```

You must change the value only in the %LET statement to change the filter value in all three procedures!

# Creating and Using SAS Macro Variables



# Filtering Rows Using Macro Variables

This demonstration illustrates modifying a program to use SAS macro variables to filter data in multiple steps.

# DATA Step Processing

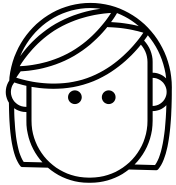
## Filtering Columns

# Subsetting Columns in the DATA Step

**DROP** *col-name <col-name>*;

**KEEP** *col-name <col-name>*;

Choose the statement based on the number of columns that you want to specify.



# Subsetting Columns in the DATA Step

```
data myclass;
    set sashelp.class;
    keep name age height;
    drop sex weight;
run;
```

or

These statements have the same result in the output table.

| Name | Age | Height |
|---|---|---|
| Alfred | 14 | 69 |
| Alice | 13 | 56.5 |
| Barbara | 13 | 65.3 |
| Carol | 14 | 62.8 |
| Henry | 14 | 63.5 |

# Preparing Data

## Computing New Columns

# Using Expressions to Create New Columns

assignment statement

```
DATA output-table;
    SET input-table;
    new-column = expression;
RUN;
```

arithmetic expression or constant

The assignment statement can create or update a column.

# Using Expressions to Create New Columns

```
data cars_new;
     set sashelp.cars;
     where Origin ne "USA";
     Profit = MSRP-Invoice;
     Source = "Non-US Cars";
keep Make Model MSRP Invoice Profit Source;
run;
```

| Make | Model | MSRP | Invoice | Profit | Source |
|---|---|---|---|---|---|
| Acura | MDX | $36,945 | $33,337 | $3,608 | Non-US Cars |
| Acura | RSX Type S 2dr | $23,820 | $21,761 | $2,059 | Non-US Cars |
| Acura | TSX 4dr | $26,990 | $24,647 | $2,343 | Non-US Cars |
| Acura | TL 4dr | $33,195 | $30,299 | $2,896 | Non-US Cars |
| Acura | 3.5 RL 4dr | $43,755 | $39,014 | $4,741 | Non-US Cars |
| Acura | 3.5 RL w/Navi... | $46,100 | $41,100 | $5,000 | Non-US Cars |
| Acura | NSX coupe 2d... | $89,765 | $79,978 | $9,787 | Non-US Cars |

The column name is stored in the case that you use to create it.

# Operands

Operands are constants (character, numeric, or date) and variables (character or numeric).

Examples:

```
Bonus = 500;
```

numeric constant

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

character constant

```
Hire_Date = '01APR2008'd;
```

date constant

```
NewSalary = 1.1 * Salary;
```

variable

# Two Digit Years

What happens if you enter this:

```
Payoff_Date = '01APR20'd;
```

date constant

| Obs | payoff_date |
|---|---|
| 1 | 22006 |

YEARCUTOFF

- System option that specifies first year of 100 year span for interpreting two-digit years
- Default in 9.4 = 1926
- Use four-digit years to avoid misinterpretation

# Operators

Operators are symbols that represent an arithmetic calculation and SAS functions.

Examples:

```
Revenue = Quantity * Price;
```

```
NewCountry = upcase(Country);
```

# Arithmetic Operators

*Arithmetic operators* indicate that an arithmetic calculation is performed.

| Symbol | Definition | Priority |
|---|---|---|
| ** | Exponentiation | 1 |
| - | negative prefix | 1 |
| * | multiplication | 2 |
| / | division | 2 |
| + | addition | 3 |
| - | subtraction | 3 |

right to left within group 1

left to right within same priority group

✎ If a missing value is an operand for an arithmetic operator, the result is a missing value.

# Poll

# Quiz

# Quiz

What is the result of the assignment statement?

```
num = 4 + 10 / 2;
```

a. . (missing)
b. 0
c. 7
d. 9

# Quiz – Correct Answer

What is the result of the assignment statement?

a. . (missing)

b. 0

c. 7

d. 9

```
num = 4 + 10 / 2;
```

**The order of operations from left to right is division and multiplication followed by addition and subtraction.**

**Parentheses can be used to control the order of operations.**

```
num = (4 + 10) / 2;
```

# Quiz

What is the result of the assignment statement given the values of **`var1`** and **`var2`**?

a. . (missing)

b. 0

c. 5

d. 10

```
num = var1 + var2 / 2;
```

| Var1 | var2 |
|---|---|
| . | 10 |

# Quiz – Correct Answer

What is the result of the assignment statement given the values of **`var1`** and **`var2`**?

a. . (missing)
b. 0
c. 5
d. 10

```
num = var1 + var2 / 2;
```

| var1 | var2 |
|---|---|
| . | 10 |

If an operand is missing for an arithmetic operator, the result is missing.

# Preparing Data

## Conditional Processing

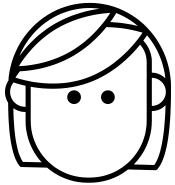
# Conditional Processing with IF-THEN

IF *expression* THEN *statement*;

defines a condition

any single SAS action

If *expression* is true,
then execute
the *statement*.



# Conditional Processing with IF-THEN

```
data cars2;
    set sashelp.cars;
    if MSRP<30000 then Cost_Group=1;
    if MSRP>=30000 then Cost_Group=2;
    keep Make Model Type MSRP Cost_Group;
run;
```

| Make | Model | Type | MSRP | Cost_Group |
|---|---|---|---|---|
| Acura | MDX | SUV | $36,945 | 2 |
| Acura | RSX Type S 2dr | Sedan | $23,820 | 1 |
| Acura | TSX 4dr | Sedan | $26,990 | 1 |
| Acura | TL 4dr | Sedan | $33,195 | 2 |
| Acura | 3.5 RL 4dr | Sedan | $43,755 | 2 |
| Acura | 3.5 RL w/Navi... | Sedan | $46,100 | 2 |
| Acura | NSX coupe 2d... | Sports | $89,765 | 2 |
| Audi | A4 1.8T 4dr | Sedan | $25,940 | 1 |





p104d05

# Conditional Test for Missing

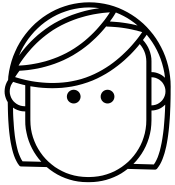
| Expression | What It Does |
|---|---|
| *N* = . | Returns TRUE if numeric value is missing |
| *C* = ‘ ‘ | Returns TRUE if character value is missing |
| MISSING (*variable*) | Returns TRUE if the numeric or character argument is missing |
| | |

# Conditional Processing with IF-THEN/ELSE

IF *expression* THEN *statement*;
<ELSE IF *expression* THEN *statement*;>
<ELSE IF *expression* THEN *statement*;>
ELSE *statement*;

If the others are not true, execute this statement.

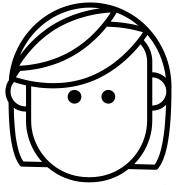
If *expression* is true, then execute the *statement* and skip the rest.



# Conditional Processing with IF-THEN/ELSE

```
data cars2;
    set sashelp.cars;
    if MSRP<20000 then Cost_Group=1;
    else if MSRP<40000 then Cost_Group=2;
    else if MSRP<60000 then Cost_Group=3;
    else Cost_Group=4;
    keep Make Model Type MSRP Cost_Group;
run;
```

For efficiency, place the most frequently occurring conditions at the top.



# Conditional Processing with IF-THEN/ELSE

Example: MSRP=35000

false

```
data cars2;
    set sashelp.cars;
    if MSRP<20000 then Cost_Group=1;
    else if MSRP<40000 then Cost_Group=2;
    else if MSRP<60000 then Cost_Group=3;
    else Cost_Group=4;
    keep Make Model Type MSRP Cost_Group;
run;
```

# Conditional Processing with IF-THEN/ELSE

Example: MSRP=35000

true

```
data cars2;
    set sashelp.cars;
    if MSRP<20000 then Cost_Group=1;
    else if MSRP<40000 then Cost_Group=2;
    else if MSRP<60000 then Cost_Group=3;
    else Cost_Group=4;
    keep Make Model Type MSRP Cost_Group;
run;
```

execute

# Conditional Processing with IF-THEN/ELSE

Example: MSRP=35000

```
data cars2;
    set sashelp.cars;
    if MSRP<20000 then Cost_Group=1;
    else if MSRP<40000 then Cost_Group=2;
    else if MSRP<60000 then Cost_Group=3;
    else Cost_Group=4;
    keep Make Model Type MSRP Cost_Group;
run;
```

skip

# Conditional Processing with IF-THEN/ELSE

Example: MSRP=75000

```
data cars2;
    set sashelp.cars;
    if MSRP<20000 then Cost_Group=1;
    else if MSRP<40000 then Cost_Group=2;
    else if MSRP<60000 then Cost_Group=3;
    else Cost_Group=4;
    keep Make Model Type MSRP Cost_Group;
run;
```

false

execute

The final ELSE statement executes if all previous conditions were false.

# Creating Character Columns with IF-THEN/ELSE

```
data cars2;
    set sashelp.cars;
    if MSRP<60000 then CarType="Basic";
    else CarType="Luxury";
    keep Make Model MSRP CarType;
run;
```

Based on the value of **MSRP**, assign a value to the new character column **CarType**.

| Make | Model | MSRP | CarType |
|---|---|---|---|
| Acura | MDX | $36,945 | Basic |
| Acura | RSX Type S 2dr | $23,820 | Basic |
| Acura | TSX 4dr | $26,990 | Basic |
| Acura | TL 4dr | $33,195 | Basic |
| Acura | 3.5 RL 4dr | $43,755 | Basic |
| Acura | 3.5 RL w/Navi... | $46,100 | Basic |
| Acura | NSX coupe 2d... | $89,765 | Luxur |
| Audi | A4 1.8T 4dr | $25,940 | Basic |

# Creating Character Columns with IF-THEN/ELSE

```
data cars2;
    set sashelp.cars;
    if MSRP<60000 then CarType="Basic";
    else CarType="Luxury";
    keep Make Model MSRP CarType;
run;
```

creates a new character column with a length of 5

| Make | Model | MSRP | CarType |
|---|---|---|---|
| Acura | MDX | $36,945 | Basic |
| Acura | RSX Type S 2dr | $23,820 | Basic |
| Acura | TSX 4dr | $26,990 | Basic |
| Acura | TL 4dr | $33,195 | Basic |
| Acura | 3.5 RL 4dr | $43,755 | Basic |
| Acura | 3.5 RL w/Navi... | $46,100 | Basic |
| Acura | NSX coupe 2d... | $89,765 | Luxur |
| Audi | A4 1.8T 4dr | $25,940 | Basic |

The first **mention** of a column in the DATA step defines the name, type, and length.

# Creating Character Columns with IF-THEN/ELSE

**LENGTH** *char-column* **$** *length*;

number of bytes or characters

column name in the case that you specify

indicator for a character column

Using the LENGTH statement leaves nothing to chance.

# Creating Character Columns with IF-THEN/ELSE

```
data cars2;
    set sashelp.cars;
    length CarType $ 6;
    if MSRP<60000 then CarType="Basic";
    else CarType="Luxury";
    keep Make Model MSRP CarType;
run;
```

explicitly creates
a new character column
with a length of 6

| Make | Model | MSRP | CarType |
|---|---|---|---|
| Acura | MDX | $36,945 | Basic |
| Acura | RSX Type S 2dr | $23,820 | Basic |
| Acura | TSX 4dr | $26,990 | Basic |
| Acura | TL 4dr | $33,195 | Basic |
| Acura | 3.5 RL 4dr | $43,755 | Basic |
| Acura | 3.5 RL w/Navi... | $46,100 | Basic |
| Acura | NSX coupe 2d... | $89,765 | Luxury |
| Audi | A4 1.8T 4dr | $25,940 | Basic |

# Conditional Processing with IF-THEN

This demonstration illustrates using IF-THEN syntax to assign values conditionally to a new column.

# Using Compound Conditions with IF-THEN/ELSE

```
data cars2;
    set sashelp.cars;
    if MPG_City>26 and MPG_Highway>30 then Efficiency=1;
    else if MPG_City>20 and MPG_Highway>25 then Efficiency=2;
    else Efficiency=3;
    keep Make Model MPG_City MPG_Highway Efficiency;
run;
```

AND — Both conditions must be true.

OR — One condition must be true.

| Make | Model | MPG_City | MPG_Highway | Efficiency |
|---|---|---|---|---|
| Chevrolet | Tracker | 19 | 22 | 3 |
| Chevrolet | Aveo 4dr | 28 | 34 | 1 |
| Chevrolet | Aveo LS 4dr hatch | 28 | 34 | 1 |
| Chevrolet | Cavalier 2dr | 26 | 37 | 2 |
| Chevrolet | Cavalier 4dr | 26 | 37 | 2 |

# Processing Multiple Statements

compound statements not allowed after then.

```
data cars2;
    set sashelp.cars;
    length Cost_Type $ 4;
    if MSRP<20000 then Cost_Group=1 and Cost_Type="Low";
    else if MSRP<40000 then Cost_Group=2 and Cost_Type="Mid";
    else Cost_Group=3 and Cost_Type="High";
run;
```

Compound statements are not allowed.

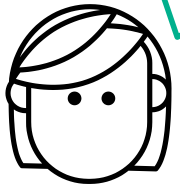
This program doesn't work because only one statement is permitted after THEN.

# Processing Multiple Statements with IF-THEN/DO

```
IF expression THEN DO;
   <executable statements>
END;
ELSE IF expression THEN DO;
   <executable statements>
END;
ELSE DO;
   <executable statements>
END;
```

need end statement for block

If *expression* is true, then execute all the *statements* between DO and END.



# Processing Multiple Statements with IF-THEN/DO

```
data under40 over40;
    set sashelp.cars;
    keep Make Model MSRP Cost_Group;
    if MSRP<20000 then do;
       Cost_Group=1;
       output under40;
    end;
    else if MSRP<40000 then do;
       Cost_Group=2;
       output under40;
    end;
    else do;
       Cost_Group=3;
       output over40;
    end;
run;
```

create two tables

DATA *table1 table2...*

conditionally output to one table

OUTPUT *table*;

# QUIZ

Why does the program fail?

```
data girls boys;
    set sashelp.class;
    if sex="F" then do;
        Gender="Female";
        output girls;
    else do;
        Gender="Male";
        output boys;
run;
```

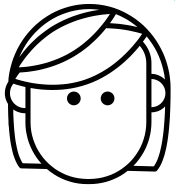
# QUIZ– Correct Answer

Why does the program fail?

```
ERROR 117-185: There were 2 unclosed DO blocks.
```

```
data girls boys;
    set sashelp.class;
    if sex="F" then do;
        Gender="Female";
        output girls;
    end;
    else do;
        Gender="Male";
        output boys;
    end;
run;
```

Using the Format Code feature in Enterprise Guide and SAS Studio helps you identify the DO blocks.



# Processing Multiple Statements with IF-THEN/DO

This demonstration illustrates using IF-THEN/DO syntax to execute multiple statements for each condition.

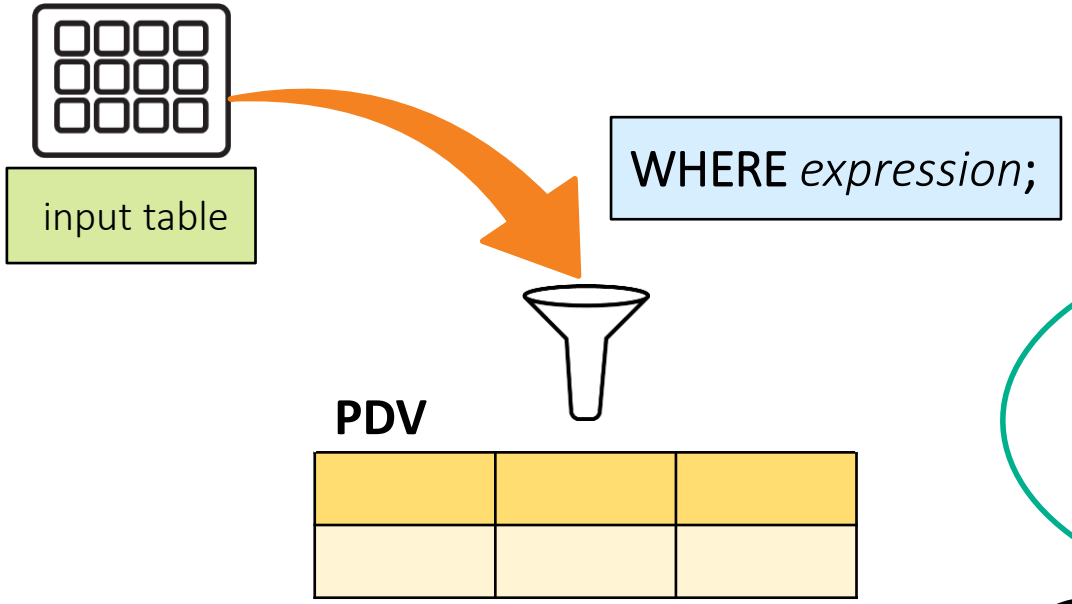
# Subsetting Rows in Execution

What action is taken here when the IF condition is true?

true

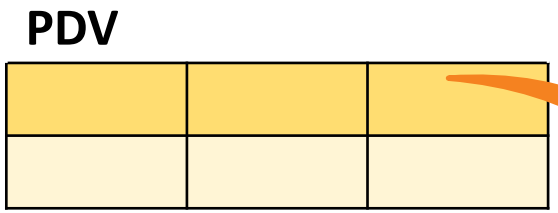
```
data cheapcars;
     set sashelp.cars;
     if MSRP < 30000;
run;
```

# Subsetting Rows in Execution

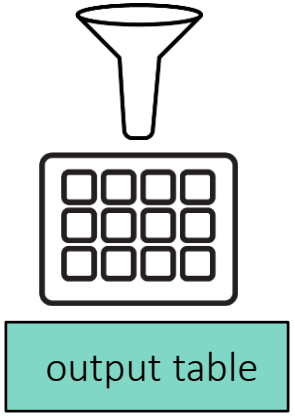


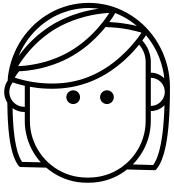

The WHERE expression must be based on columns in the input table.

# Subsetting Rows in Execution




IF *expression*;



The IF expression can be based on any values in the PDV.

(It can be based on created variables in data step)

# Subsetting Rows in Execution

true

```
data cheapcars;
     set sashelp.cars;
     if MSRP < 30000;
     other code lines
run;
```

Implicit OUTPUT;
Implicit RETURN;

When the IF condition is true, SAS continues processing statements for that row.

# Subsetting in Execution

Implicit RETURN;

```
data cheapcars;
    set sashelp.cars;
    if MSRP < 30000;
    other code lines
run;
```

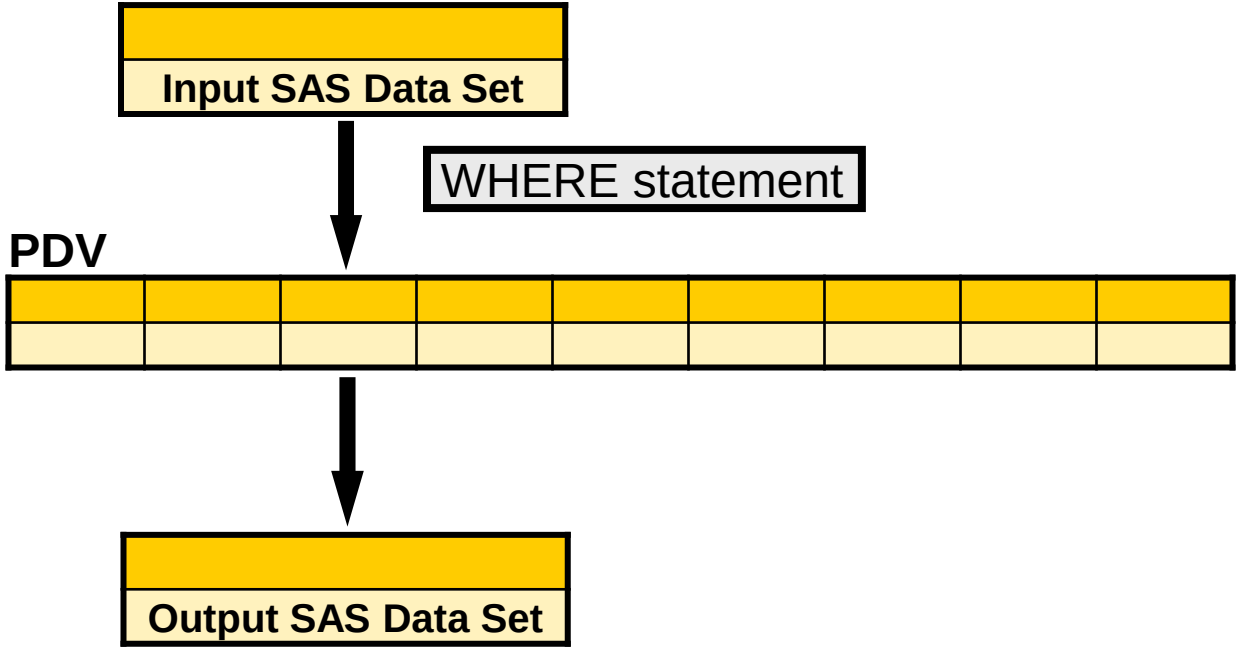
false

Implicit OUTPUT;

When the IF condition is false, SAS stops processing statements for that row and returns to the top of the DATA step.

# Processing the WHERE Statement

The WHERE statement selects observations **before** they are brought into the program data vector.



# Quiz

Why does the WHERE statement not work in this DATA step?

```
data work.december;
   set orion.sales;
   BonusMonth=month(Hire_Date);
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   where Country='AU' and BonusMonth=12;
run;
```

# Quiz – Correct Answer

Why does the WHERE statement not work in this DATA step?

```
data work.december;
   set orion.sales;
   BonusMonth=month(Hire_Date);
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   where Country='AU' and BonusMonth=12;
run;
```

```
ERROR: Variable BonusMonth is not on file ORION.SALES.
```

The WHERE statement can only subset variables that are coming from an existing data set.

# The Subsetting IF Statement

Examples:

```
if Salary > 50000;
```

```
if Last_Name='Smith' and First_Name='Joe';
```

```
if Country not in ('GB', 'FR', 'NL');
```

```
if Hire_Date = '15APR2008'd;
```

```
if not missing(end_date);
```

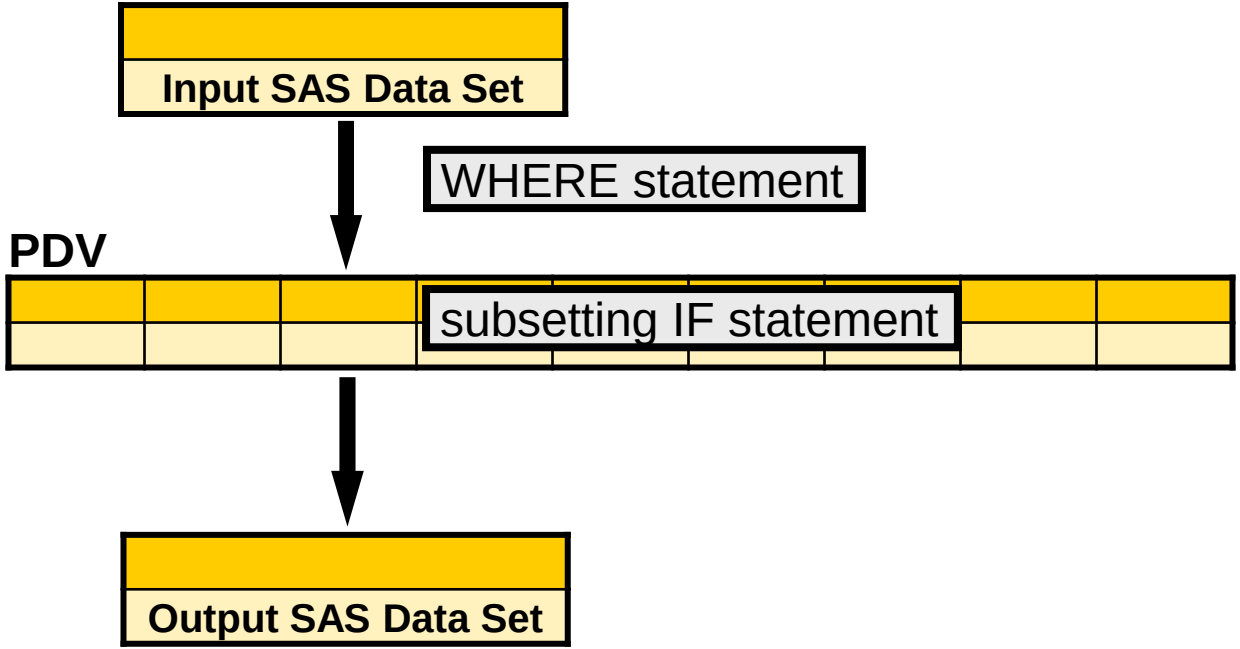
```
if upcase(Gender)='M';
```

```
if 40000 <= Compensation <= 80000;
```

```
if Salary+Bonus < 43000;
```

# Processing the Subsetting IF Statement

The subsetting IF statement determines if observations continue being processed in the program data vector.



# Business Scenario

Include only the employees from Australia who have a bonus month in December.

```
data work.december;
    set orion.sales;
    where Country='AU';
    BonusMonth=month(Hire_Date);
    if BonusMonth=12;
    Bonus=500;

Compensation=sum(Salary,Bonus);
run;
```

for efficiency use where when possible

Partial SAS Log

```
NOTE: There were 63 observations read from the data set ORION.SALES.
      WHERE Country='AU';
NOTE: The data set WORK.DECEMBER has 3 observations and 12 variables.
```