SAS Things That Come In Twos

Iowa User Group
13 May 2024

Charu Shankar
SAS Institute Inc

Bio

With a background in computer systems management. SAS Instructor Charu Shankar engages with logic, visuals, and analogies to spark critical thinking.

A SAS Instructor since 2007, she curates and delivers unique content via the SAS YouTube channel, SAS global forum, SAS Ask the Expert Series, SAS Training Post Blog, etc.

Charu loves to support users by teaching at conferences on topics related to SAS, SQL, Efficiencies, PERL, Macros, Python, Viya, etc.

When she's not coding, Charu is A Yoga Instructor who loves to explore Canadian trails with her husky Miko.

Agenda

- Nuts & Bolts
- 2 Steps
- 2 Processing Phases
- 2 Data Types
- 2 parts of a dataset
- 2 engines in SAS
- 2 types of SQL
- 2 row filters
- 2 column filters
- 2 binary variables (plus much much more)
- Handy Links

Computer Processing

You @ your terminal

Applications

O/S
Email
SAS

CPU

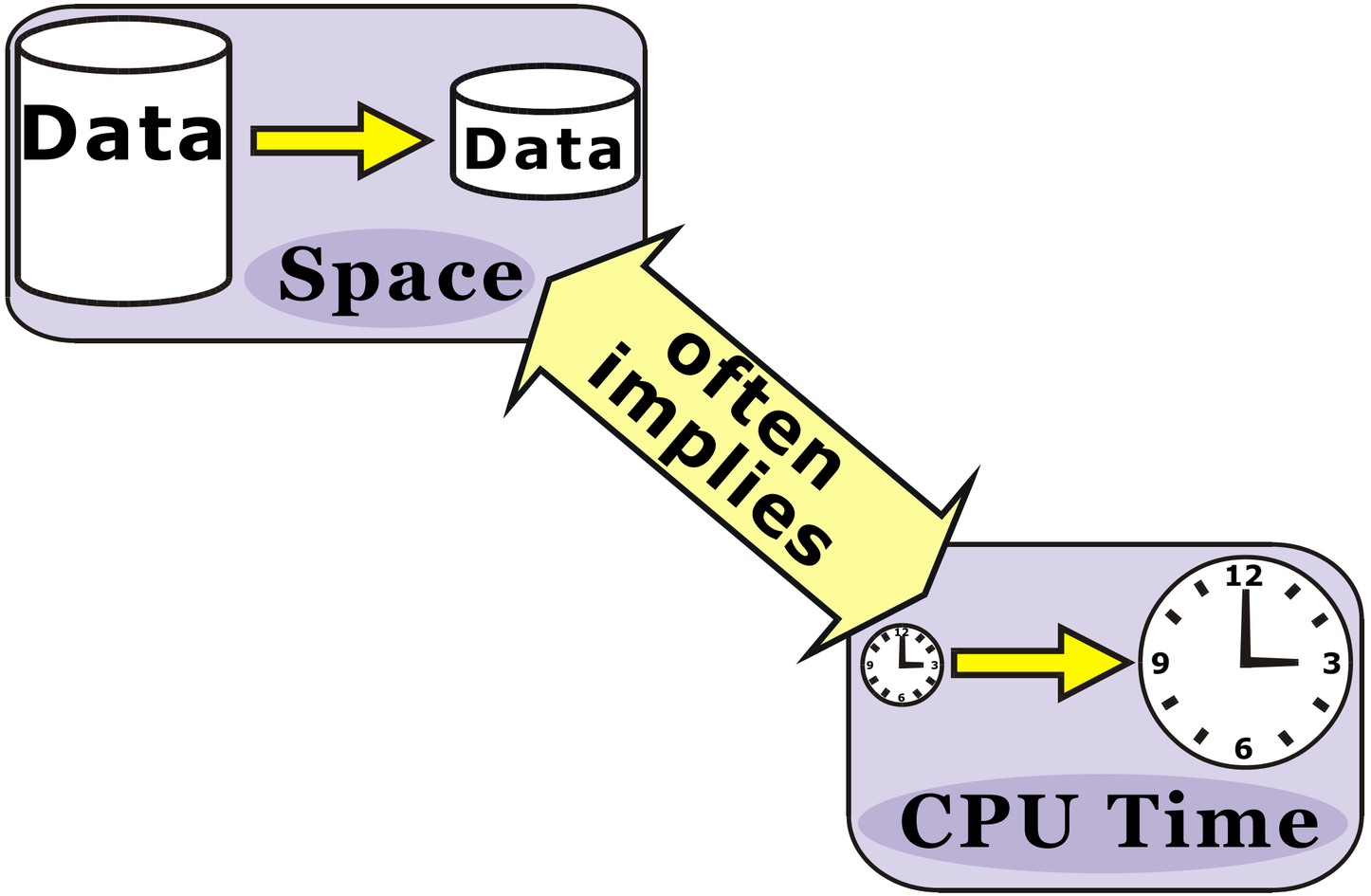
Operating System | Email Software | SAS Software | SAS data

Memory

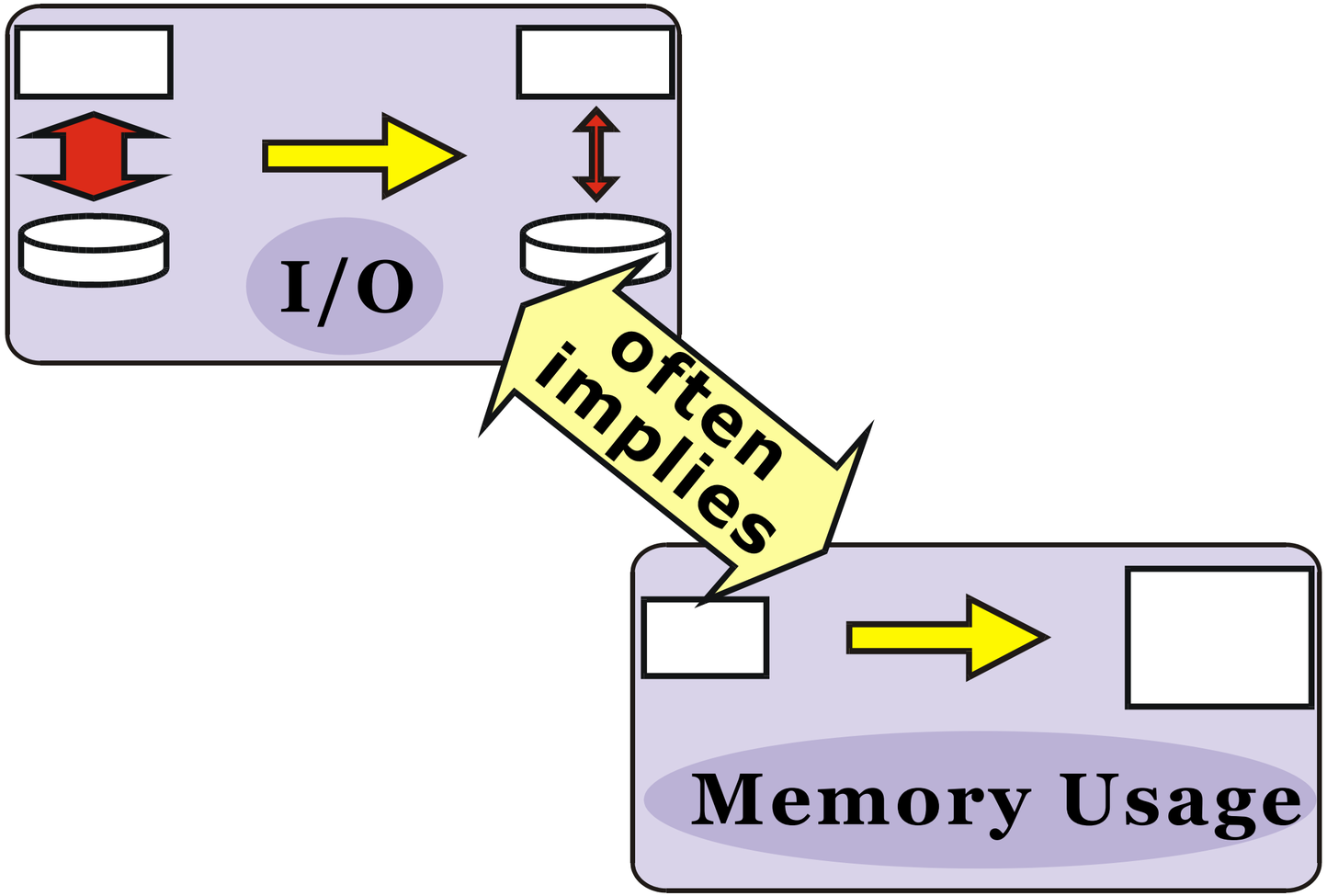
Data

Tables

Understanding Efficiency Trade -offs



Understanding Efficiency Trade -offs

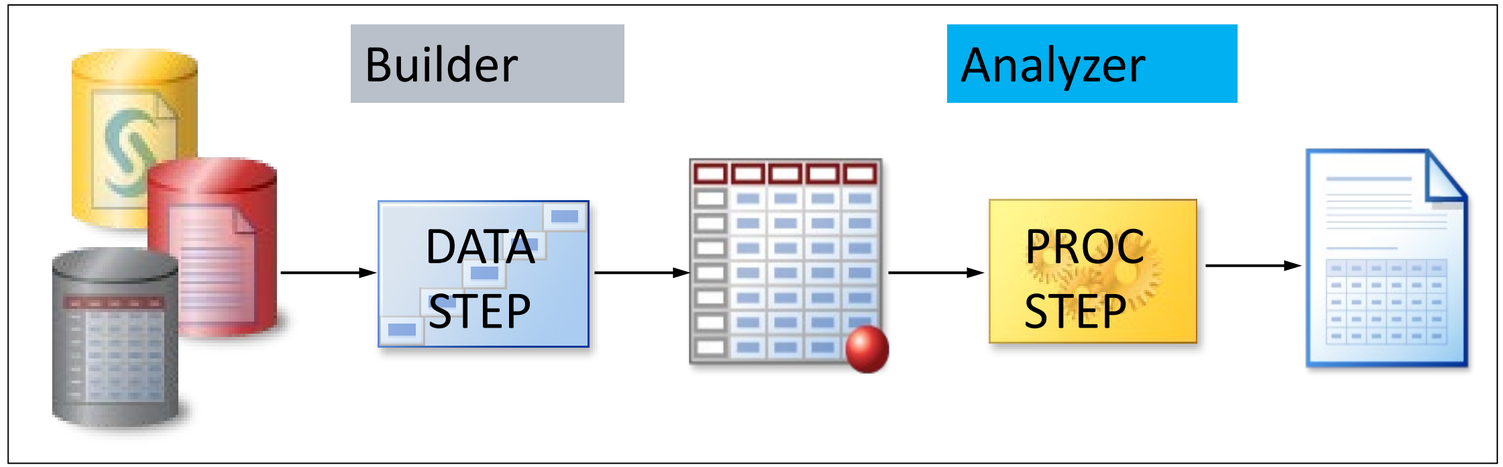


2 Steps

2 Steps - DATA step and PROC step

SAS Programs

A *SAS program* is a sequence of one or more steps.



- *DATA steps* typically build and manipulate SAS data sets.
- *PROC steps* typically process SAS data sets to generate reports and graphs, and to manage data.

Quiz

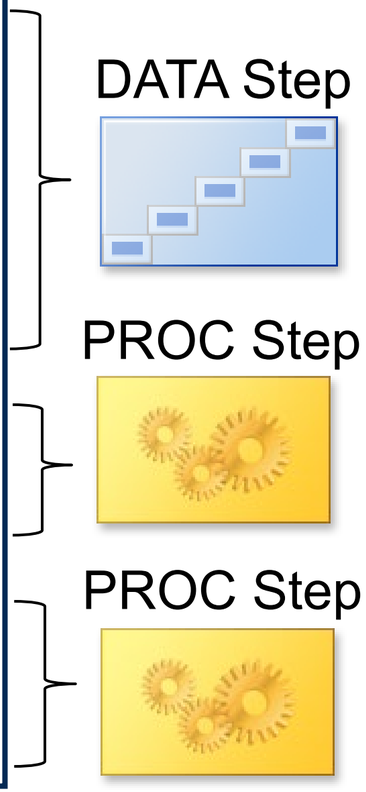
How many steps are in this program?

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```

Quiz - Correct Answer

How many steps are in this program? **three**

```
data work.newsalesemps;
   length First_Name $ 12
          Last_Name $ 18 Job_Title $ 25;
   infile "&path\newemps.csv" dlm=',';
   input First_Name $ Last_Name $
         Job_Title $ Salary;
run;

proc print data=work.newsalesemps;
run;

proc means data=work.newsalesemps;
   var Salary;
run;
```



p102d01

2 Processing Phases

DATA Step Processing

Compilation

establish data attributes and rules for execution

Execution

read, manipulate, and write data

DATA Step Processing: Compilation

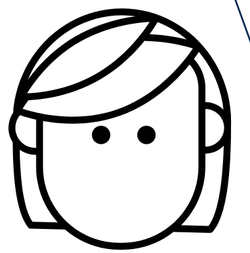
Compilation

1) Check for syntax errors.
2) Create the *program data vector (PDV)*, which includes all columns and attributes.
3) Establish the specifications for processing data in the PDV during execution.
4) Create the descriptor portion of the output table.

PDV

Season N 8	Name $ 25	StartDate N 8	Ocean $ 8

The PDV is the magic behind the DATA step's processing power!



DATA Step Processing: Compilation

```
data storm_complete;
     set pg2.storm_summary_small;
     length Ocean $ 8;
     drop EndDate;
     where Name is not missing;
     Basin=upcase(Basin);
     StormLength=EndDate-StartDate;
     if substr(Basin,2,1)="I" then Ocean="Indian";
     else if substr(Basin,2,1)="A" then Ocean="Atlantic";
     else Ocean="Pacific";
run;
```

Define the library and a name for the output table.

DATA Step Processing: Compilation

```
data storm_complete;
     set pg2.storm_summary_small;
     length Ocean $ 8;
     drop EndDate;
     where Name is not missing;
     Basin=upcase(Basin);
     StormLength=EndDate-StartDate;
     if substr(Basin,2,1)="I" then Ocean="Indian";
     else if substr(Basin,2,1)="A" then Ocean="Atlantic";
     else Ocean="Pacific";
run;
```

Columns are added to the PDV in the order in which they appear in the input table.

PDV

Name $ 15	Basin $ 2	MaxWind N 8	StartDate N 8	EndDate N 8

Attributes are inherited from the input table.

DATA Step Processing: Compilation

```
data storm_complete;
     set pg2.storm_summary_small;
     length Ocean $ 8;
     drop EndDate;
     where Name is not missing;
     Basin=upcase(Basin);
     StormLength=EndDate-StartDate;
     if substr(Basin,2,1)="I" then Ocean="Indian";
     else if substr(Basin,2,1)="A" then Ocean="Atlantic";
     else Ocean="Pacific";
run;
```

The remaining columns are added to the PDV in the order in which they appear in the DATA step.

Each column must have at least a name, type, and length.

PDV

Name $ 15	Basin $ 2	MaxWind N 8	StartDate N 8	EndDate N 8	Ocean $ 8	StormLength N 8

DATA Step Processing: Compilation

```
data storm_complete;
     set pg2.storm_summary_small;
     length Ocean $ 8;
     drop EndDate;
     where Name is not missing;
     Basin=upcase(Basin);
     StormLength=EndDate-StartDate;
     if substr(Basin,2,1)="I" then Ocean="Indian";
     else if substr(Basin,2,1)="A" then Ocean="Atlantic";
     else Ocean="Pacific";
run;
```

DROP or KEEP statements flag columns that will be excluded from the output table.

PDV

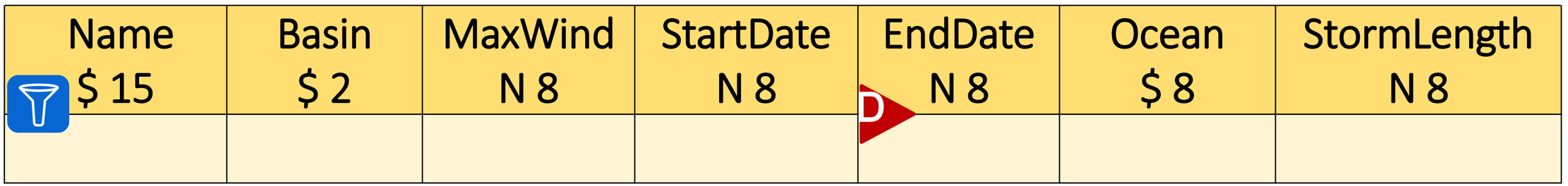
Name $ 15	Basin $ 2	MaxWind N 8	StartDate N 8	EndDate N 8	Ocean $ 8	StormLength N 8
				D		

DATA Step Processing: Compilation

```
data storm_complete;
     set pg2.storm_summary_small;
     length Ocean $ 8;
     drop EndDate;
     where Name is not missing;
     Basin=upcase(Basin);
     StormLength=EndDate-StartDate;
     if substr(Basin,2,1)="I" then Ocean="Indian";
     else if substr(Basin,2,1)="A" then Ocean="Atlantic";
     else Ocean="Pacific";
run;
```

The WHERE statement establishes conditions for which rows will be read from the input table into the PDV.

PDV



DATA Step Processing: Compilation

```
data storm_complete;
     set pg2.storm_summary_small;
     length Ocean $ 8;
     drop EndDate;
     where Name is not missing;
     Basin=upcase(Basin);
     StormLength=EndDate-StartDate;
     if substr(Basin,2,1)="I" then Ocean="Indian";
     else if substr(Basin,2,1)="A" then Ocean="Atlantic";
     else Ocean="Pacific";
run;
```

The descriptor portion is created for the output table.

work.storm_complete

Name $ 15	Basin $ 2	MaxWind N 8	StartDate N 8	Ocean $ 8	StormLength N 8

DATA Step Processing: Execution

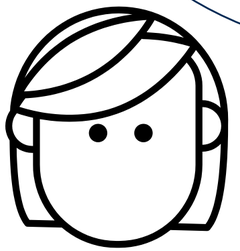
Execution

1) Initialize the PDV.
2) Read a row from the input table into the PDV.
3) Sequentially process statements and update values in the PDV.
4) At the end of the step, write the contents of the PDV to the output table.
5) Return to the top of the DATA step.

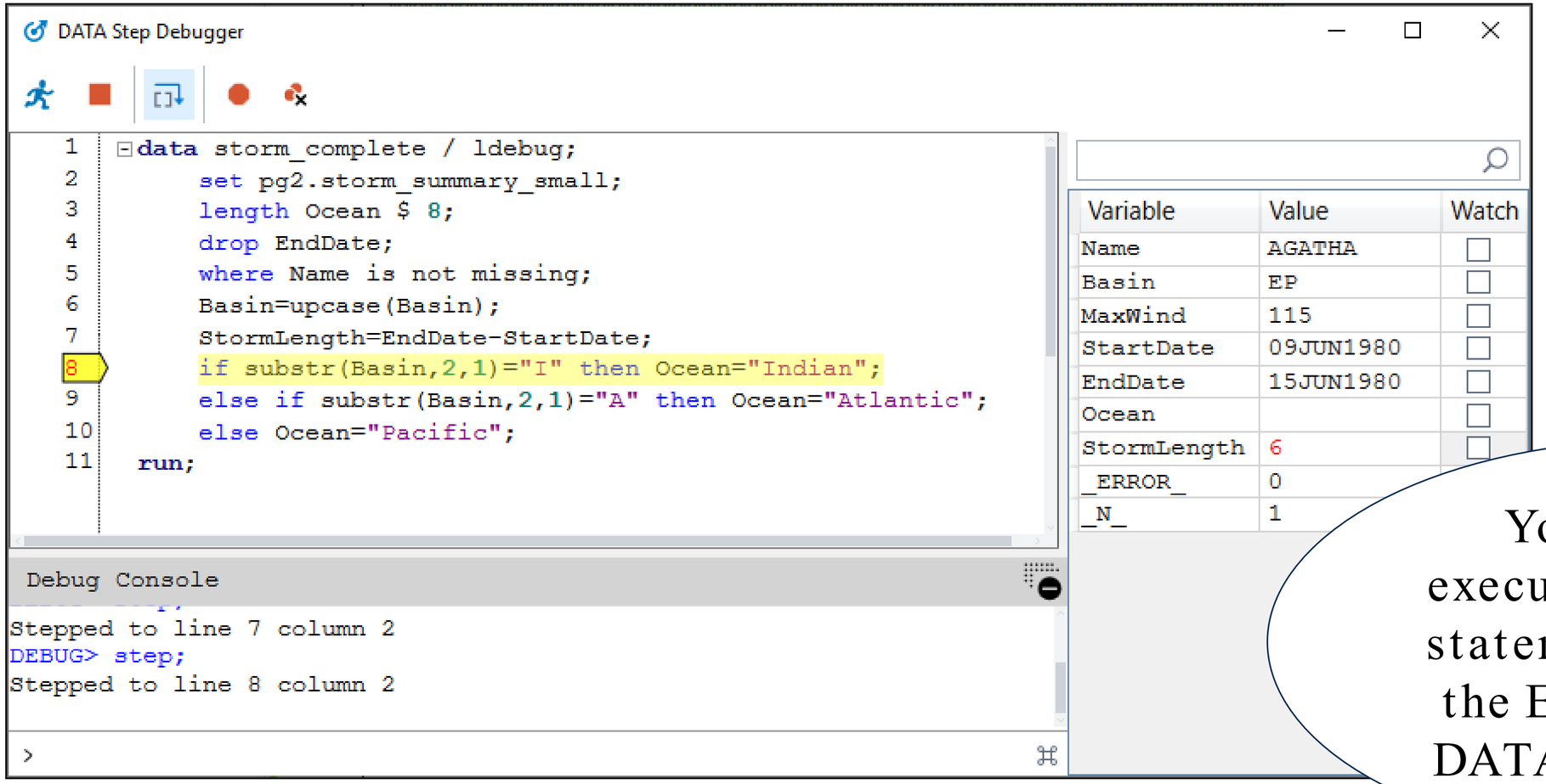
```
data output-table;
     set input-table;
     ...other statements...
run;
```

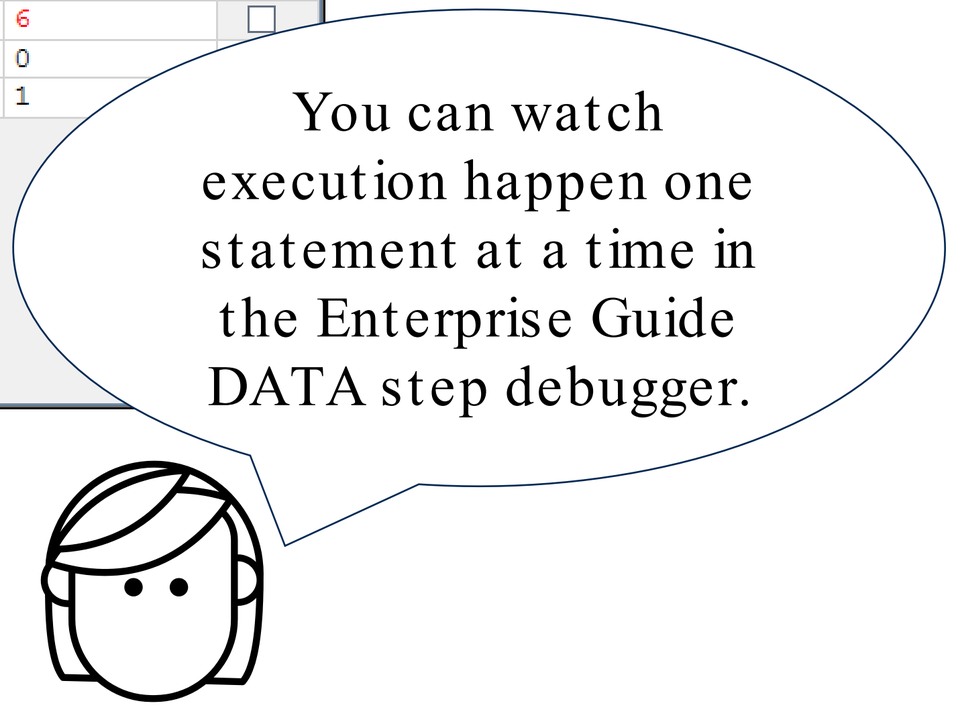
Implicit OUTPUT;
Implicit RETURN;

Automatic looping makes processing data easy!



DATA Step Processing in Action





2 Data Types

2 Data Types

A SAS data set supports two types of variables.

Character variables

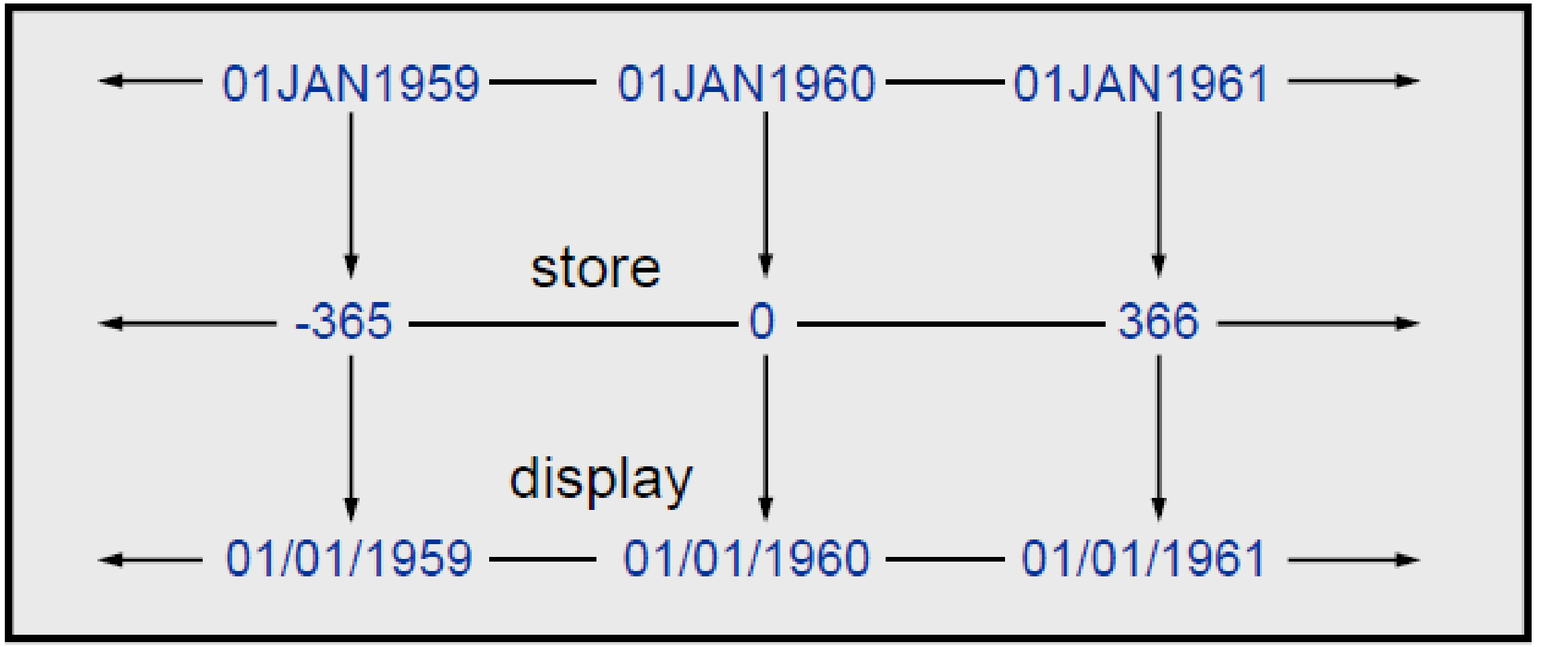
- can contain any value: letters, numerals, special characters, and blanks
- range from 1 to 32,767 characters in length
- have 1 byte per character.

Numeric variables

- store numeric values using floating point or binary representation
- have 8 bytes of storage by default
- can store 16 or 17 significant digits.

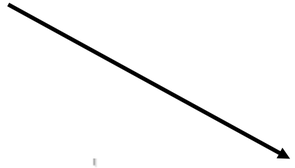
SAS Date Values

SAS stores calendar dates as numeric values.



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

What type should my data be - Character or numeric?



	Customer_ID	Customer_Na...	Customer_Ty...	Product_ID	Month	Year	Total_Cases	Total_Sales	productnum
1	100	All-Mart	BigBoxRetail	101001	1	1999	36	$280.80	101001
2	100	All-Mart	BigBoxRetail	101001	2	1999	103	$803.40	101001
3	100	All-Mart	BigBoxRetail	101001	4	1999	15	$117.00	101001
4	100	All-Mart	BigBoxRetail	101001	5	1999	19	$148.20	101001
5	100	All-Mart	BigBoxRetail	101001	6	1999	29	$226.20	101001
6	100	All-Mart	BigBoxRetail	101001	7	1999	22	$171.60	101001
7	100	All-Mart	BigBoxRetail	101001	8	1999	22	$171.60	101001
8	100	All-Mart	BigBoxRetail	101001	9	1999	74	$577.20	101001
9	100	All-Mart	BigBoxRetail	101001	10	1999	39	$304.20	101001
10	100	All-Mart	BigBoxRetail	101001	11	1999	51	$397.80	101001
11	100	All-Mart	BigBoxRetail	101001	12	1999	77	$600.60	101001
12	100	All-Mart	BigBoxRetail	101001	1	2000	34	$265.20	101001
13	100	All-Mart	BigBoxRetail	101001	2	2000	70	$546.00	101001
14	100	All-Mart	BigBoxRetail	101001	3	2000	13	$101.40	101001
15	100	All-Mart	BigBoxRetail	101001	4	2000	7	$54.60	101001

1. Storage Considerations

How much space does each of these columns use? Which type is more efficient for saving space

	Character	Numeric
Saving Space	Yes	No

Product_ID	productnum
101001	101001
101001	101001
101001	101001
101001	101001

2. Data Manipulation Considerations

The first 2 characters of Product ID indicate a tier level. What type should Product_Id be?

	Character	Numeric
Saving Space	Yes	No
Manipulation	Yes	No

Product_ID	productnum
101001	101001
101001	101001
101001	101001
101001	101001

3. Data Calculation Considerations

We would like to see the minimum of product_ID values. What type should you consider for Product_ID?

Considerations	Character	Numeric
Saving space	Yes	No
Manipulation	Yes	No
Calculation	No	Yes

Product_ID	productnum
101001	101001
101001	101001
101001	101001
101001	101001

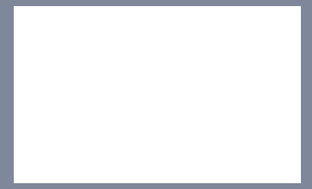
4. Data Type Conversion

```
/*Saving Space - convert numeric data to
character data using the PUT function*/
data choc.cechardata;
     set choc.cenumdata;
     productchar=put(productnum, $6.);
run;
```

productnum	productchar
101001	101001
101001	101001
101001	101001
101001	101001
101001	101001
101001	101001

Function	What it does
INPUT(*source*, *informat*)	Converts character values to numeric values using a specified informat
PUT(*source*, *format*)	Converts numeric or character values to character values using a specified format

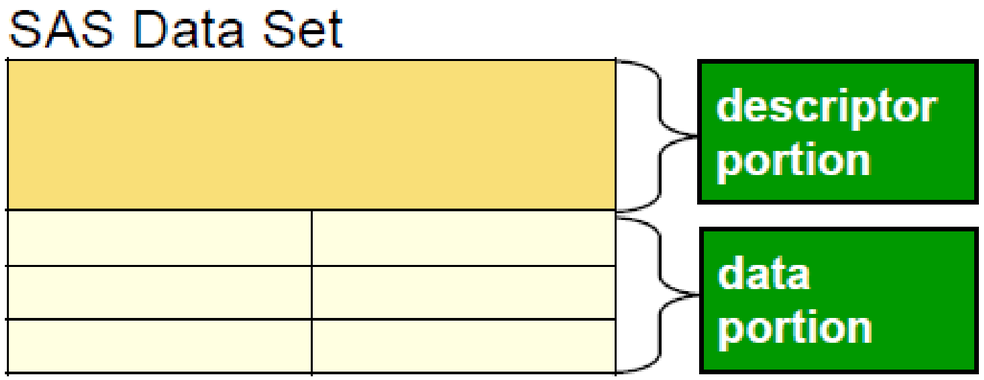
2 Parts of a Dataset



2 Parts of a Dataset

SAS Data Set Terminology

A SAS data set contains a descriptor portion and a data portion.

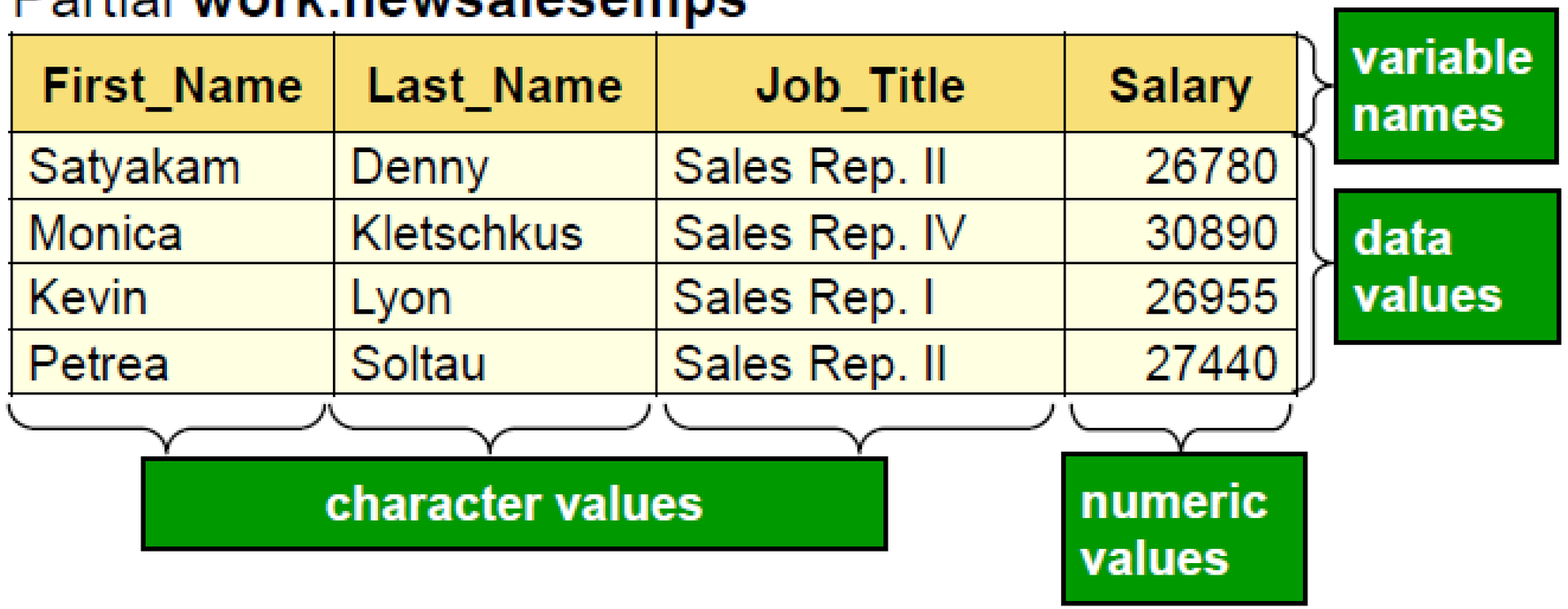


Data

The *data portion* of a SAS data set contains the data values, which are either character or numeric.

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



Browsing the Data Portion

Use *PROC PRINT* to display the data portion of a SAS data set.

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

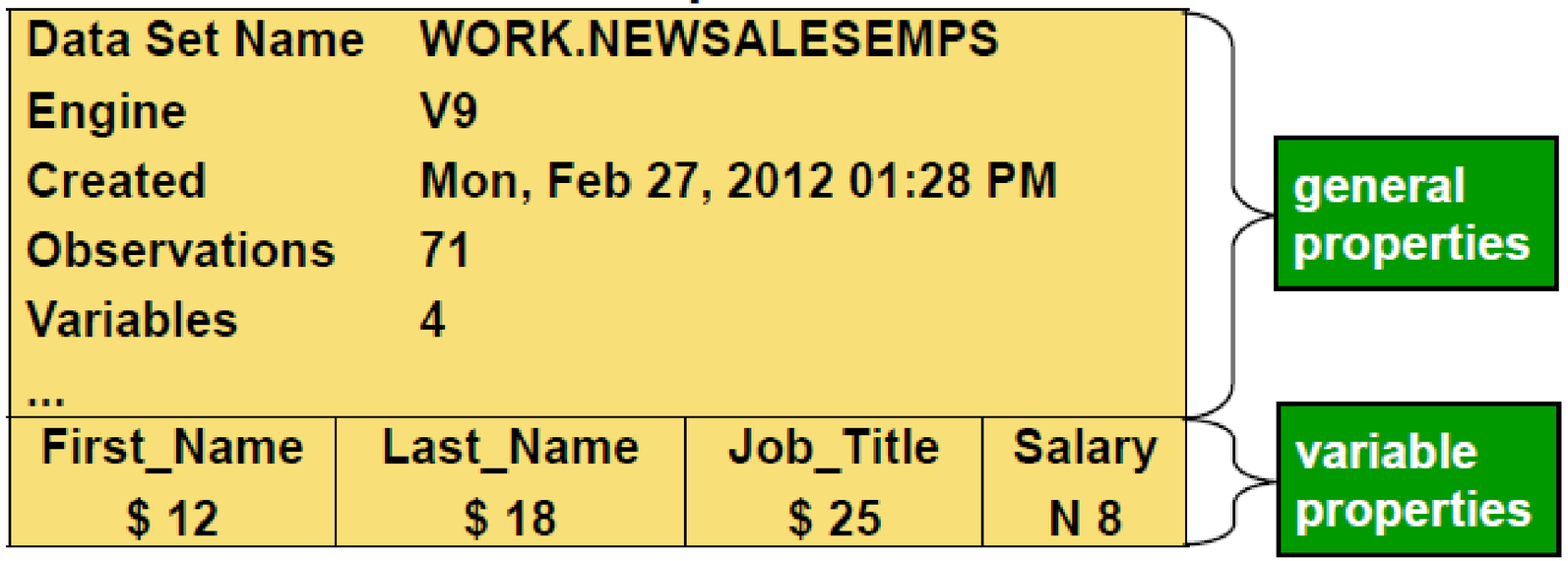
PROC PRINT DATA=*SAS-data-set*;
RUN;

Descriptor

The *descriptor portion* contains the following metadata:

- general properties (such as data set name and number of observations)
- variable properties (such as name, type, and length)

Partial **work.newsalesemps**



Browsing the Descriptor Portion

Use *PROC CONTENTS* to display the descriptor portion of a SAS data set.

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

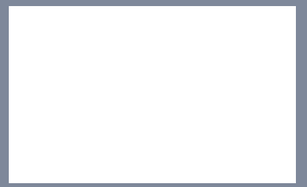
PROC CONTENTS DATA=*SAS-data-set*;
RUN;

Browsing the Descriptor Portion

```
proc sql outobs=10;
 select name, memname, type, length
  from dictionary.columns
   where libname =SASHELP'
    group by name
     having count(name) > 1
        order by name;
quit;
```

Column Name	Member Name	Column Type	Column Length
ACTUAL	PRDSAL2	num	8
ACTUAL	PRDSAL3	num	8
ACTUAL	PRDSALE	num	8
AIR	AIR	num	8
AIR	AIRLINE	num	8
ALIAS_CITY	ZIPMIL	char	300
ALIAS_CITY	ZIPCODE	char	300
ALIAS_CITYN	ZIPMIL	char	300
ALIAS_CITYN	ZIPCODE	char	300
AMOUNT	NVST4	num	8

2 Engines In SAS



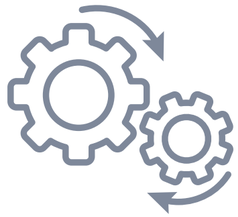
2 SAS Viya Engines

SAS Viya

Traditional SAS processing engine

Executes traditional SAS®9 code

SAS Compute Server

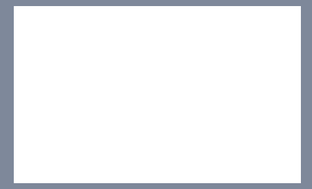


Next-gen SAS processing engine

SAS Cloud Analytic Services (CAS)

Executes CAS-enabled code in parallel on in-memory data

2 Types of SQL



PROC SQL vs. PROC FEDSQL

	PROC SQL	PROC FEDSQL	
SAS SQL Implementation	✓		
SAS Enhancements & Function	✓		
ANSI Standard	ANSI Standard 2	✓	Vendor neutral ANSI SQL 3
Database Data Types	Only char & num	✓	17 ANSI data types
Dictionary Tables	Session & table info		Enhanced DBMS table info using dictionaires
Multi-threaded for sorting & indexing on the SAS platform	Limited -for sorting & indexing on the SAS platform		Fully multi-threaded on the SAS platform
Many Non-ANSI enhancements	✓		
Executes in CAS(SAS Viya)		✓	

2 Types of SQL

```
proc sql;
select * from sashelp.cars
    where origin = 'Asia';
quit;
```

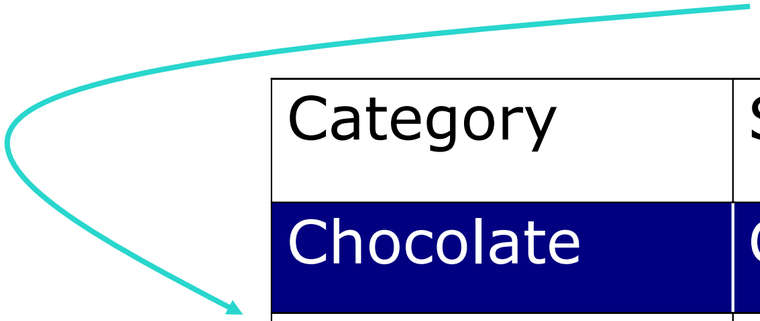
```
proc fedsql;
connect to oracle (user=testuser
password=testpass);

create table test_table as
    select * from connection to oracle
        (select * from employees
            where department = 'Sales');

disconnect from oracle;
quit;
```

2 Row Filters

Efficiently using resources
Miko, my beautiful northern rescue dog had a hip fracture in the wild. While that self healed, his right hind leg became shorter. He uses his front legs to drive him forward!

Selecting Observations

We want to subset for **category**= “Chocolate“



Category	SubCategory	Product_Name
Chocolate	Chocolate Dark	SM Dark Choc Bar
Gummy	Gummy Sour	Gummy Lions Bag
Hard	Hard Sweet	Butterscotch Disks Bag
Sugar-Free	SF Chocolate	SF Jelly Beans Bag

Where or IF – that is the question?

Subsetting IF Or The Where Clause?

Create a subset of the cesales_analysis dataset that contains data for Chocolate.

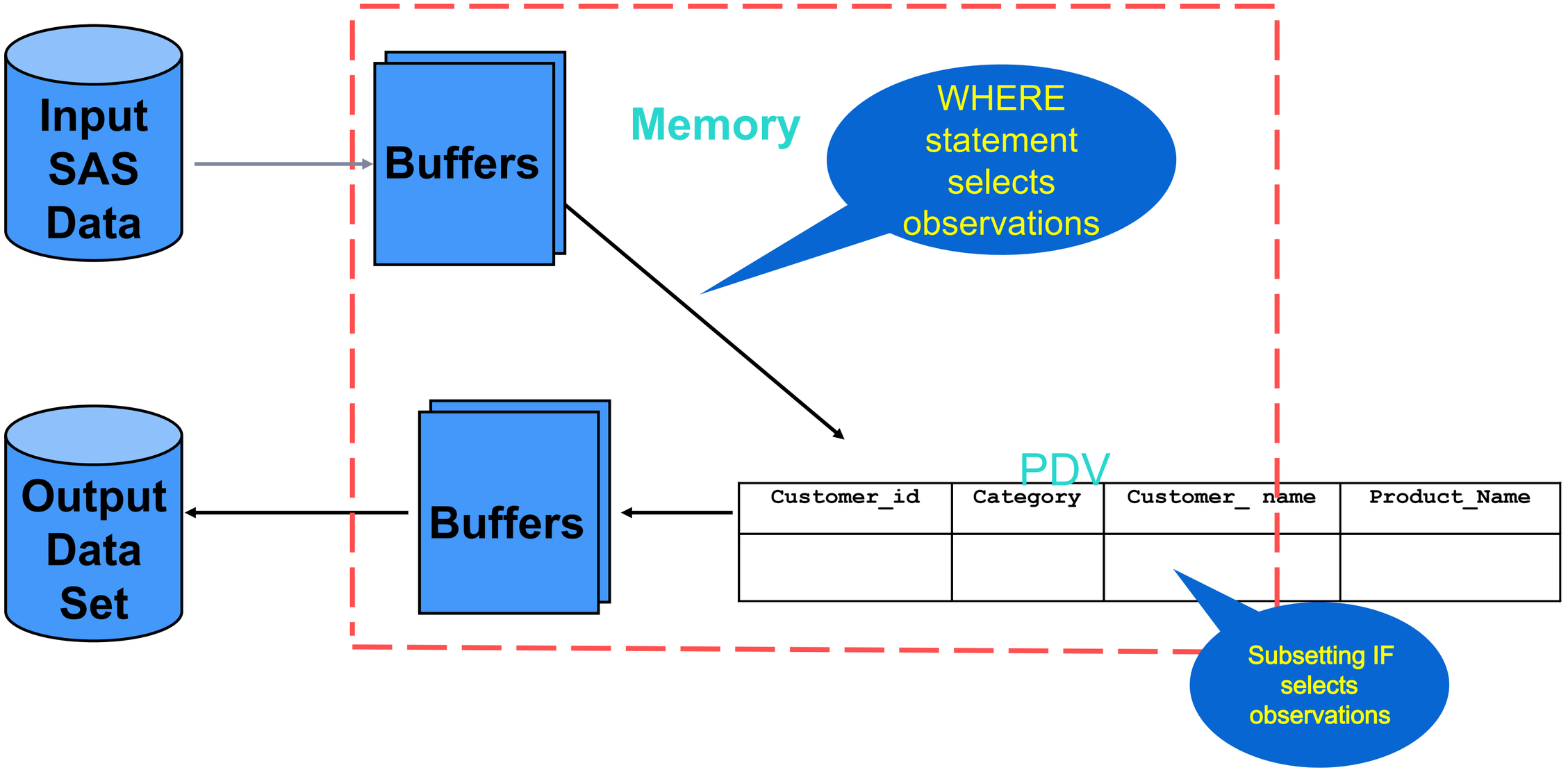
```
data chocolate;
    set choc.cesales_analysis;
    if category='Chocolate';
Run;
```

```
NOTE: There were 115928 observations read from the
data set CHOC.CESALES_ANALYSIS.
NOTE: The data set WORK.CHOCOLATE has 50368
observations and 11 variables.
NOTE: DATA statement used (Total process time):
real time           2.83 seconds
cpu time            0.04 seconds
```

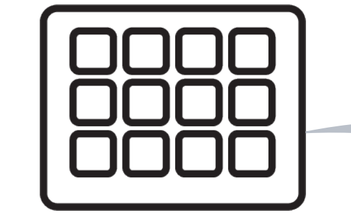
```
data chocolate;
    set choc.cesales_analysis;
    where category='Chocolate';
Run;
```

```
NOTE: There were 50368 observations read from the
data set choc.CESALES_ANALYSIS.
WHERE category='Chocolate';
NOTE: The data set WORK.CHOCOLATE has 50368
observations and 11 variables.
NOTE: DATA statement used (Total process time):
real time           2.26 seconds
cpu time            0.06 seconds
```

The Subsetting IF and WHERE Statements



Subsetting Rows in Execution

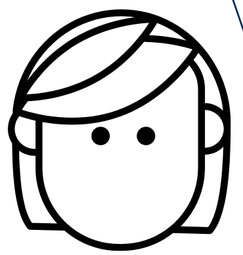


input table

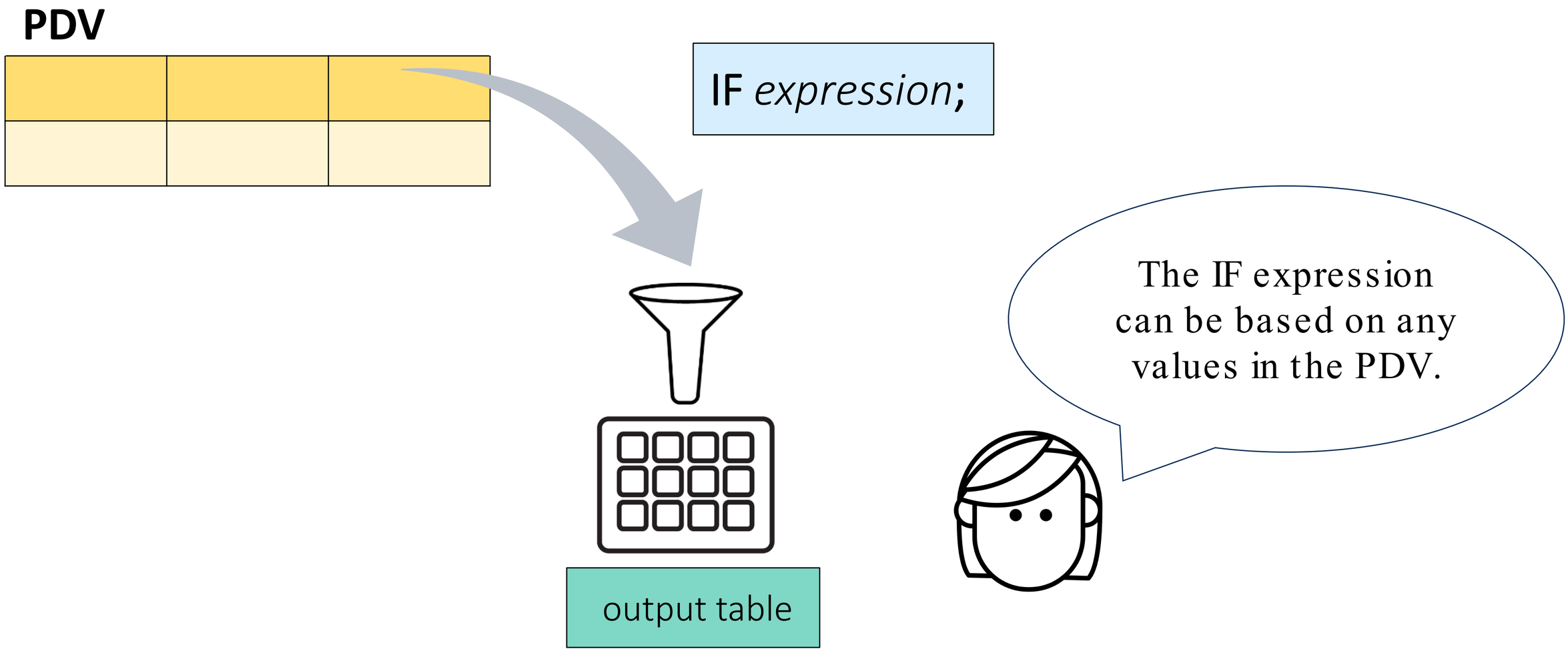
WHERE *expression*;

PDV

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

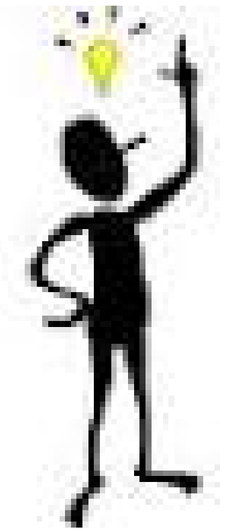


Subsetting Rows in Execution



Consider- When To Use Which One?

The WHERE clause Or The Subsetting IF
The answer lies in this question - do you want to subset existing rows or newly created rows?



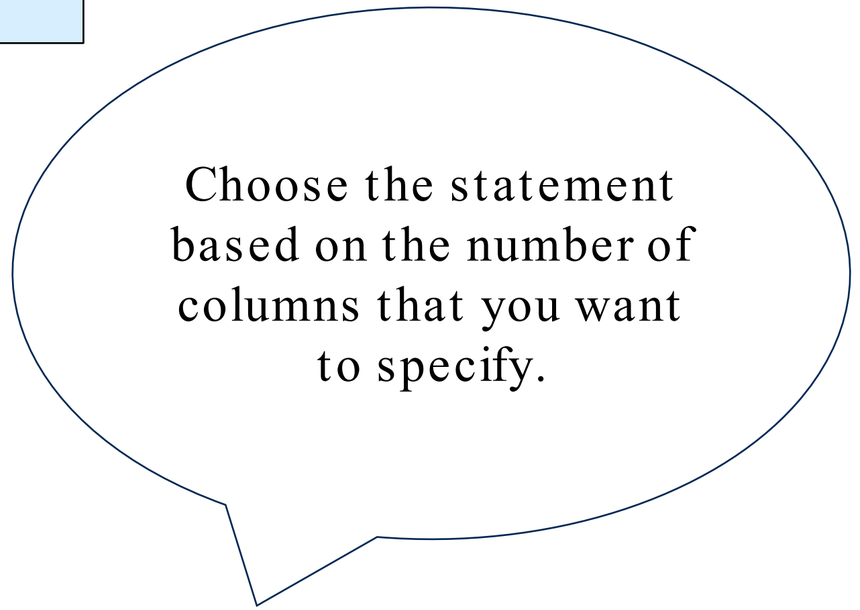
Did you know ? The WHERE clause is the same one used in SQL. If you want to subset existing obs use the WHERE. The powerful WHERE acts on obs before moving it to the PDV. The IF statement works on newly created var but has to read in row by row into the PDV thus slower in comparison

2 Column Filters

Filtering Columns in the DATA Step

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

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



p104d01

Filtering Columns in the DATA Step

These statements have the same result in the output table.

OR

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

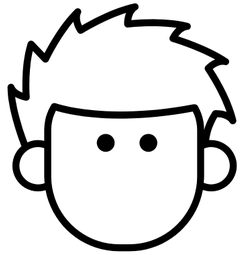
Name	Age	Height
Alfred	14	69
Alice	13	56.5
Barbara	13	65.3
Carol	14	62.8
Henry	14	63.5

2 Binary Variables

Processing Data in Groups

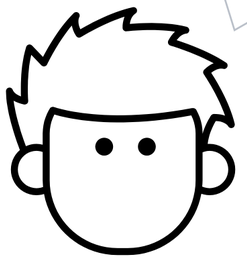
Basin	Name	MaxWindMPH	StartDate
NA	NATE	90	04OCT2017
NA	OPHELIA	115	09OCT2017
NA	PHILIPPE	60	28OCT2017
NA	RINA	60	06NOV2017
NI	MAARUTHA	45	15APR2017
NI	MORA	70	28MAY2017
NI	OCKHI	100	29NOV2017
SI	ALFRED	50	16FEB2017
SI	BLANCHE	65	02MAR2017
SI	CALEB	50	23MAR2017
SI	ERNIE	140	05APR2017
SI	FRANCES	75	21APR2017
SI	GREG	40	29APR2017
SI	CEMPAKA	40	22NOV2017
SI	DAHLIA	60	24NOV2017
SI	HILDA	60	24DEC2017
SP	DEBBIE	120	23MAR2017
SP	BART	45	19FEB2017
SP	COOK	100	06APR2017
SP	DONNA	125	01MAY2017
SP	ELLA	70	07MAY2017

If your data is sorted into groups, the DATA step can identify when each group begins and ends.

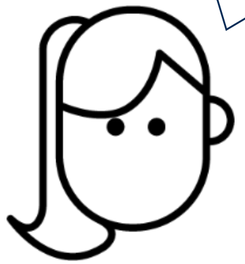


Processing Data in Groups

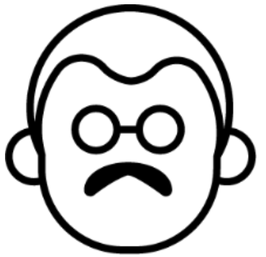
What is the maximum wind measurement for each storm?



When did the first storm occur in each basin?



Which storm names are used more than once within a season?



Processing Data in Groups

```
PROC SORT DATA=input-table
          <OUT=sorted-output-table>;
          BY <DESCENDING> col-
name(s);
RUN;
```

sorts the table into groups

```
DATA output-table;
          SET sorted-output-table;
          BY <DESCENDING> col-
name(s);
RUN;
```

processes the data in the sorted table by groups

Processing Data in Groups

```
data storm2017_max;
     set storm2017_sort;
     by Basin;
run;
```

First.*BY-column*

Last.*BY-column*

The BY statement creates **First./ Last.** variables in the PDV that can be used to identify when each BY group begins and ends.

PDV

...other columns...	Basin	First.Basin	Last.Basin
		D	D

2 Binary Variables – Practical Use

PDV

…other columns…	Basin	First.Basin	Last.Basin
	NA	1	0

first row where **Basin** is *NA*

PDV

…other columns…	Basin	First.Basin	Last.Basin
	NA	0	0

subsequent rows where **Basin** is *NA*

PDV

…other columns…	Basin	First.Basin	Last.Basin
	NA	0	1

last row where **Basin** is *NA*

2 Binary Variables – Practical Use

```
proc sort data=pg2.storm_2017out=storm2017_sort;
    by Basin;
run;

data storm2017_max;
     set storm2017_sort;
     by Basin;
     if last.Basin=1;
     StormLength=EndDate-StartDate;
     MaxWindKM=MaxWindMPH*1.60934;
run;
```

Year	Basin	Name	StartDate	EndDate	MaxWindMPH	MinPressure	
2017	EP	ADRIAN	09MAY2017	10MAY2017	45	1004	None
2017	EP	BEATRIZ	31MAY2017	02JUN2017	45	1001	Southwestern Mexico
2017	EP	CALVIN	11JUN2017	13JUN2017	45	1004	Southwestern Mexico, Guatemala
2017	EP	DORA	25JUN2017	28JUN2017	105	974	Southwestern Mexico, Revillagigedo
2017	EP	EUGENE	07JUL2017	12JUL2017	115	966	Baja California Peninsula, California
2017	EP	FERNANDA	12JUL2017	22JUL2017	145	947	Hawaii
2017	EP	GREG	17JUL2017	26JUL2017	60	1001	None
2017	EP	EIGHT-E	18JUL2017	20JUL2017	35	1007	None
2017	EP	HILARY	21JUL2017	31JUL2017	105	972	Southwestern Mexico
2017	EP	IRWIN	22JUL2017	01AUG2017	90	980	None
2017	EP	ELEVEN-E	04AUG2017	05AUG2017	35	1006	Revillagigedo Islands
2017	EP	JOVA	12AUG2017	14AUG2017	40	1003	Western Mexico, Revillagigedo Island
2017	EP	KENNETH	18AUG2017	23AUG2017	130	952	None
2017	EP	LIDIA	31AUG2017	03SEP2017	65	986	Revillagigedo Islands, Western Mexic
2017	EP	OTIS	11SEP2017	19SEP2017	115	965	None
2017	EP	MAX	13SEP2017	15SEP2017	85	985	Southern Mexico

Year	Basin	Name	StartDate	EndDate	MaxWindMPH	MinPressure	Location	StormLength	MaxWindKM
2017	EP	SELMA	27OCT2017	28OCT2017	40	1005	Nicaragua, Costa Rica, El Salvador, Guatemala, Honduras	1	64.3736
2017	NA	RINA	06NOV2017	09NOV2017	60	995	United Kingdom, Ireland	3	96.5604
2017	NI	OCKHI	29NOV2017	06DEC2017	100	976	Sri Lanka, India, Maldives	7	160.934
2017	SI	HILDA	24DEC2017	30DEC2017	60	980	Timor, Western Australia	6	96.5604
2017	SP	ELLA	07MAY2017	15MAY2017	70	977	Samoan Islands, Tonga, Wallis and Futuna	8	112.6538

Handy Links

- Read what you need
- Intro to Coding in SAS Viya
- Retail therapy the SAS Way
- Top 10 SAS Coding Efficiencies
- FIRST. and LAST. DATA Step Variables
- "SAS variable lists". Support.sas.com website
- The Power of SAS SQL – SAS YouTube Video
- Ask The Expert Webinar – Top 5 Handy PROC SQL Tips
- Proc SQL or Proc FEDSQL: Which Should a Programmer Use
- SAS Tutorial | Step-by-Step PROC SQL – SAS YouTube Video
- "Techniques for Optimizing Memory Usage". Support.sas.com website.
- The difference between the Subsetting IF and the IF—THEN—ELSE—IF statement
- Shankar, Charu. "Know Thy Data: Techniques for Data Exploration". Pharmasug 2018,
- Ask the Expert Webinar - Why choose between SAS data Step & PROC SQL When You Can Have Both
- Hadden, Louise; Shankar, Charu. "Put on the SAS®Sorting Hat and Discover Which Sort is Best for You!". PharmaSUG 2019

Thank You

✓ Did you enjoy this session, Let us know in the **evaluation**

Charu Shankar
SAS Institute Toronto

EMAIL Charu.shankar@sas.com
BLOG https://blogs.sas.com/content/author/charushankar/
TWITTER CharuYogaCan
LINKEDIN https://www.linkedin.com/in/charushankar/