SAS presentation

Introduction

»**SAS** stands for **Statistical Analysis Software**.
»It was created in the year 1960 by the SAS Institute.
»From 1st January 1960, SAS was used for data management, business intelligence, Predictive Analysis, Descriptive and Prescriptive Analysis etc.
»Since then, many new statistical procedures and components were introduced in the software.

»SAS is platform independent which means you can run SAS on any operating system either Linux or Windows.
» SAS is driven by SAS programmers who use several sequences of operations on the SAS datasets to make proper reports for data analysis.

Why we use SAS

»**SAS** is basically worked on large datasets. With the help of SAS software you can perform various operations on the data like:

- Data Management
- Statistical Analysis
- Report formation with perfect graphics
- Business Planning
- Operations Research and project Management
- Quality Improvement
- Application Development
- Data extraction
- Data transformation
- Data update and modification

Installation

Start by choosing your operating system below:

Windows | OS X | LINUX

Link : https://www.sas.com/en_us/software/university-edition.html

Installation

Get virtualization software.

First, download and install virtualization software on your computer. We recommend Oracle VirtualBox for Windows, which is free.

Download VirtualBox for Windows

Note: SAS University Edition also works with VMware Workstation Player, which you can download here: VMware Workstation Player download page. Charges may apply.

Create a folder for your SAS files.

While you wait for VirtualBox to install:

1. Create a folder named *SASUniversityEdition* (no spaces) on your local computer in a location you will remember.
2. Then create a subfolder within the *SASUniversityEdition* folder called *myfolders* (no spaces). This is where you'll save all your SAS University Edition files.

Installation

Download the SAS University Edition vApp.

1. Click the **Get SAS University Edition** button below. You'll be prompted to create a SAS Profile, or sign in if you already have one.
2. After you're signed in to your SAS Profile, accept the license agreement terms and conditions.
3. On the Order Summary Page, click the **Download** link, and the download will begin. (If your browser prompts you to save or run the file, select **Save** to save the file in your *Downloads* directory.)

Note: The file is more than 1.7GB. Depending on your internet connection, it might take a while to download. Grab a snack, call a friend, read a book – it will be done before you know it. And remember – you're getting the world's most powerful analytics software. It's worth the wait!

Get SAS University Edition

Import SAS University Edition into VirtualBox.

1. Launch VirtualBox and select **File > Import Appliance**.
2. In the Import Virtual Appliance window, click the folder icon to the right of the field.
3. In the file browser window, select the SAS University Edition .ova file, and click **Open**.
4. Click **Next**, and then click **Import** in the Appliance Settings window.

Share your *myfolders* folder with VirtualBox.

1. In VirtualBox, select the SAS University Edition vApp, and then select **Machine > Settings**.
2. In the navigation pane, select **Shared Folders**, and then click the Add Folder icon (+) in the upper right of the Settings window.
3. In the Add Share window, select **Other** as the folder path.
4. In the Select Folder window, open the *SASUniversityEdition* folder, and select the *myfolders* subfolder you created in step 1. Click **Select Folder**.
5. In the Add Share window, confirm that **Read-only** is not selected.
6. Select the **Auto-mount** and **Make Permanent** (if available) options, and click **OK**.
7. Click **OK** again to close the Settings window.

Start SAS University Edition.

1. In VirtualBox, select the SAS University Edition vApp, and then select **Machine > Start.** It might take a few minutes for the virtual machine to start.

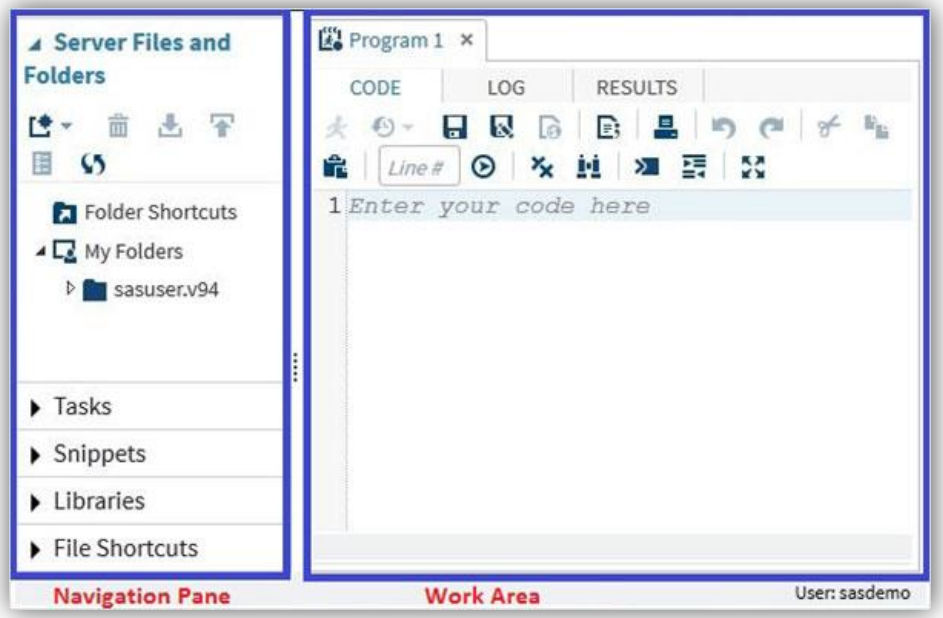
   Note: When the virtual machine is running, the screen with the SAS logo is replaced with a black console screen (called the Welcome window). You can minimize this window, but **do not close the Welcome window until you are ready to end your SAS session**.
2. In a web browser on your local computer, enter **http://localhost:10080**.
3. From the SAS University Edition: Information Center, click **Start SAS Studio**.

1.

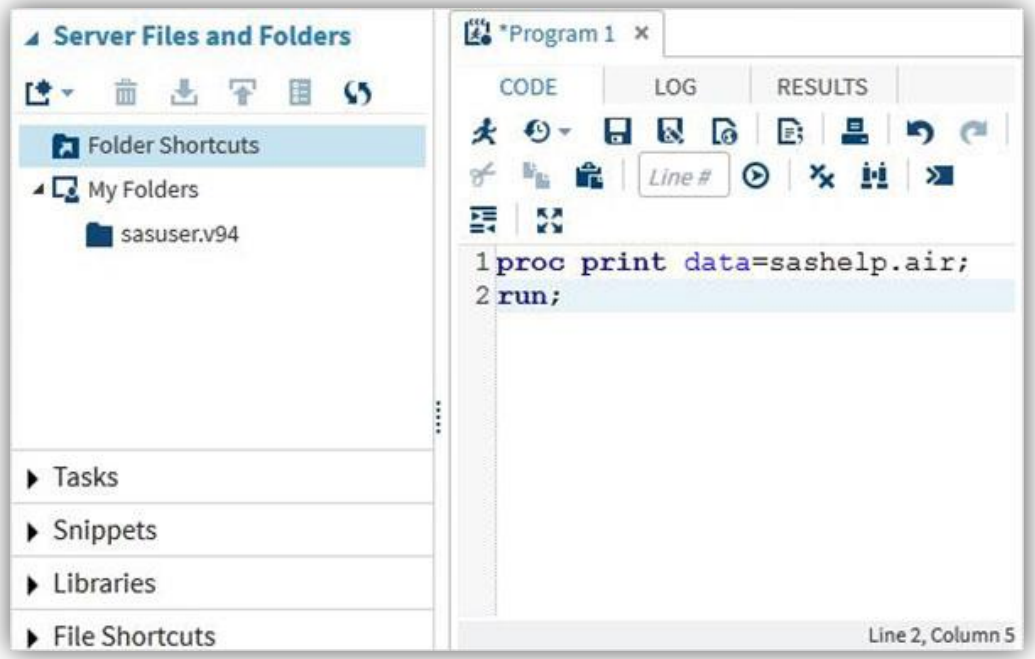
SAS environment

SAS University Edition

SAS Main Window

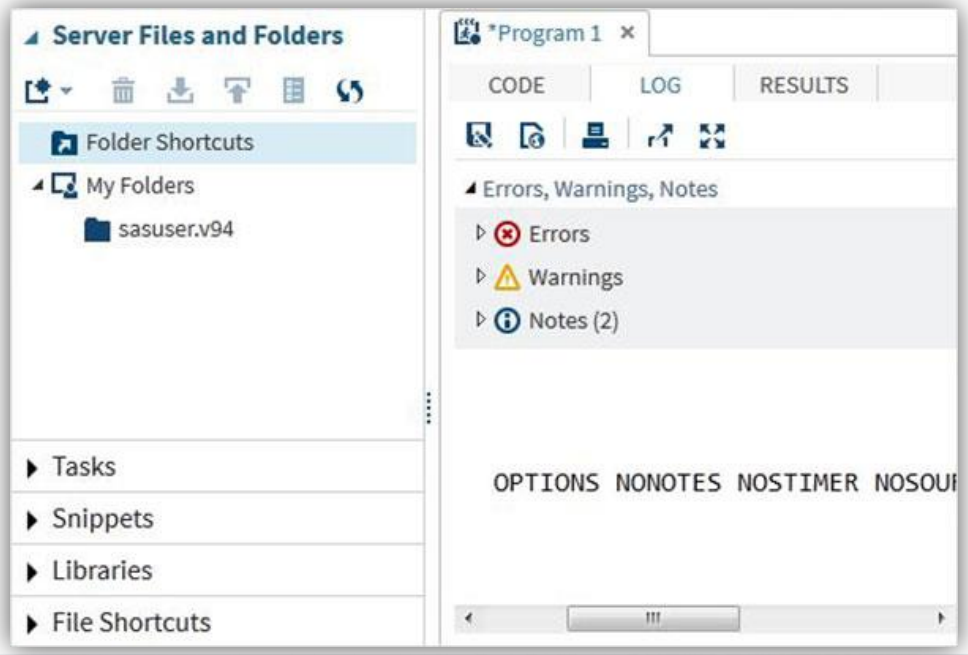


Program Execution



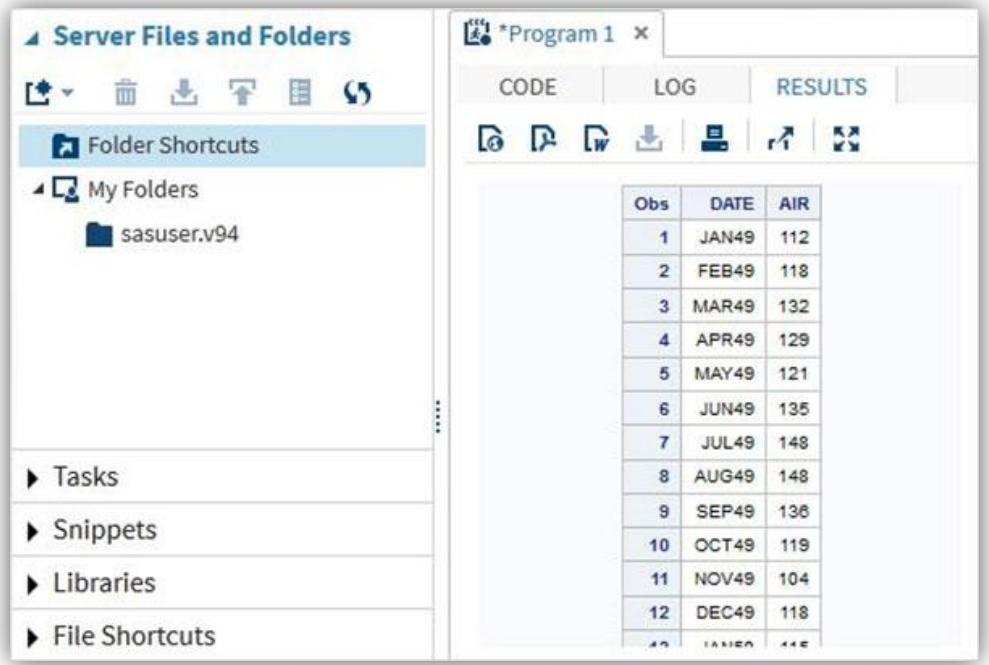
- The execution of code is done by pressing the run icon, which is the first icon from left or the F3 button.

Program Log



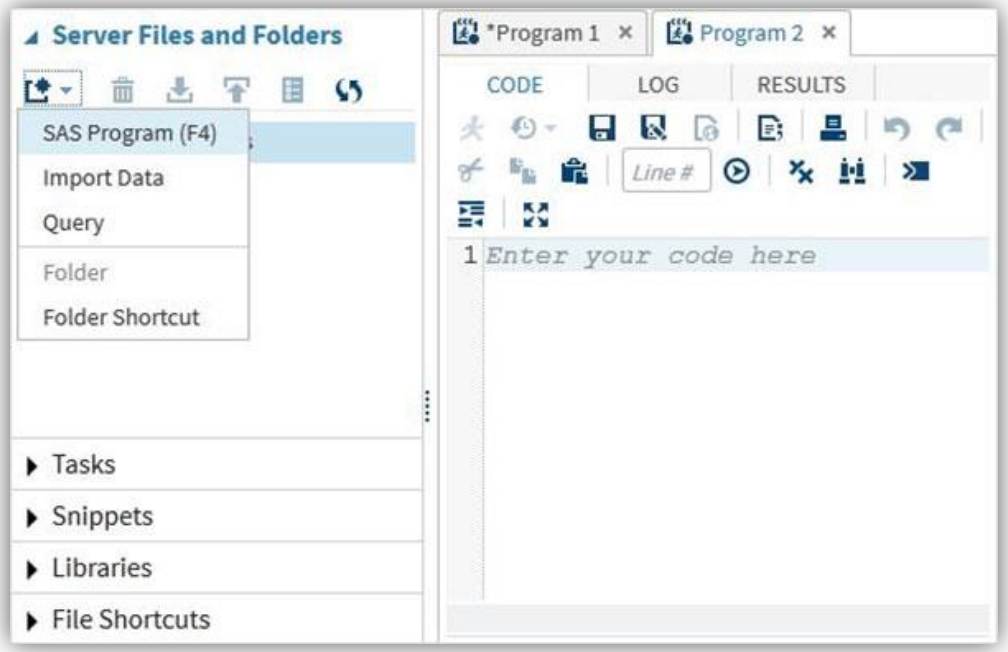
- The log of the executed code is available under the **Log** tab.
- It describes the errors, warnings or notes about the program's execution.
- This is the window where you get all the clues to troubleshoot your code.

Program Result



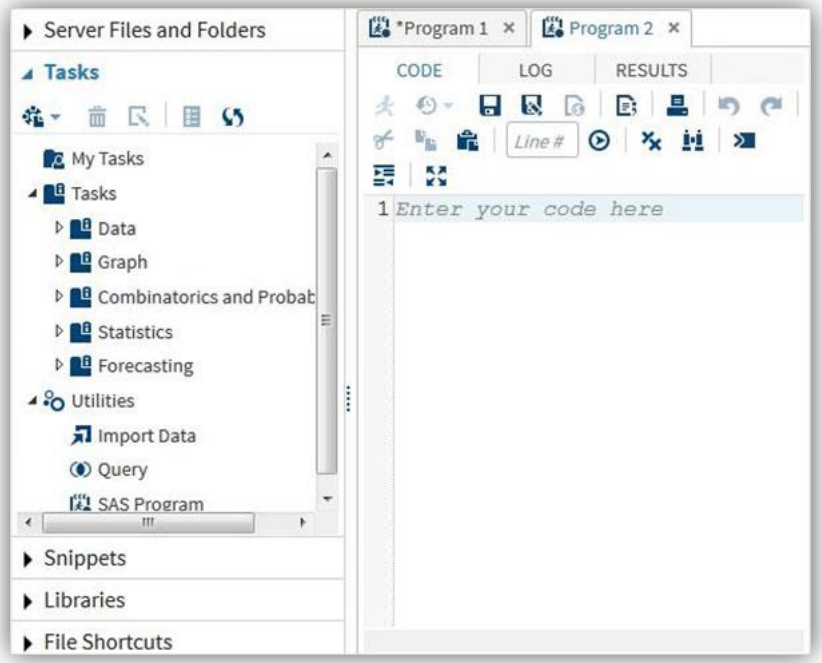
- The result of the code execution is seen in the RESULTS tab.
- By default they are formatted as html tables.

Server Files and Folders



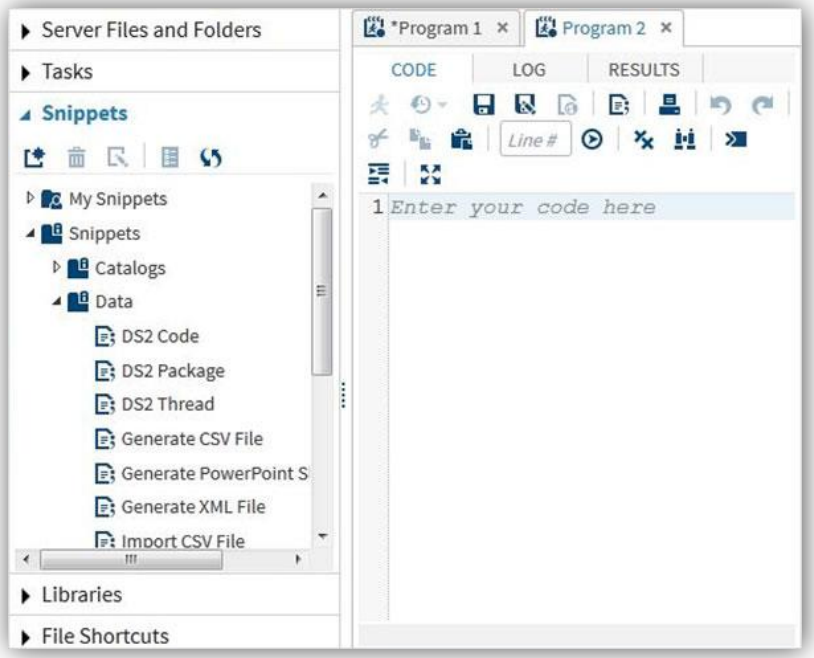
- Under this tab we can create additional programs, import data to be analyzed and query the existing data.
- It can also be used to create folder shortcuts.

Tasks



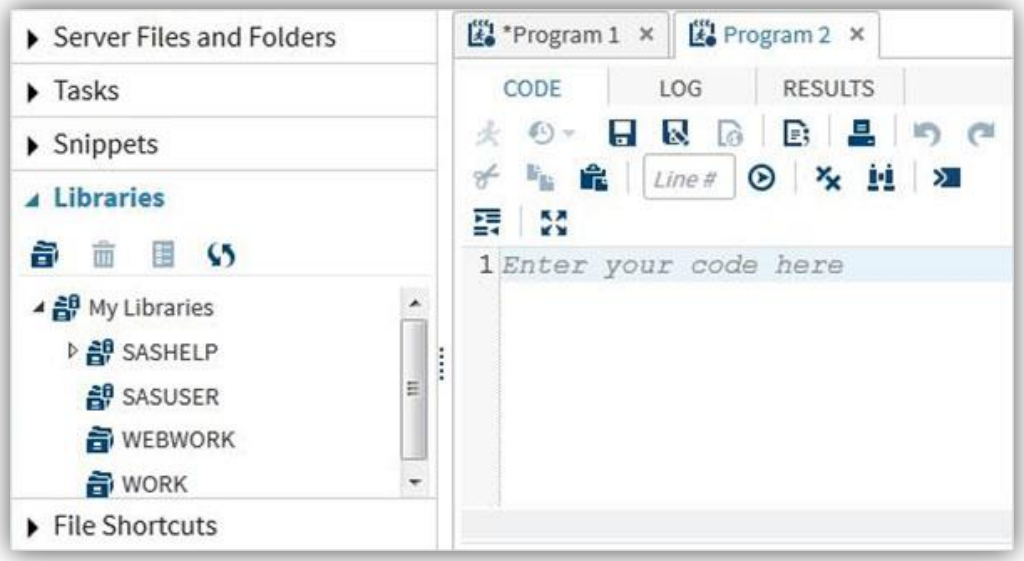
- The Tasks tab provides features to use in-built SAS programs by supplying only the input variables.
- For example under the statistics folder you can find a SAS program to do linear regression by only supplying the SAS data set name and variable names.

Snippets



- The snippets tab provides features to write SAS Macro and generate files from the existing data set.

Program Libraries

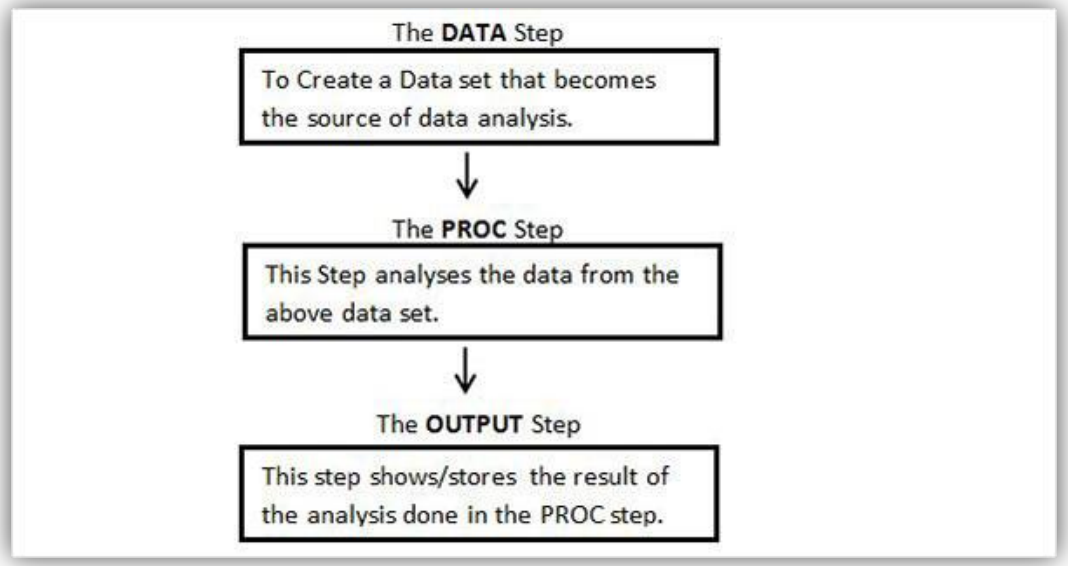


- SAS stores the datasets in SAS libraries.
- The temporary library is available only for a single session and it is named as WORK.
- But the permanent libraries are available always.

2.

SAS Program Structure

SAS Program Structure



Every SAS program must have all these steps to complete reading the input data, analyzing the data and giving the output of the analysis.

Also the **RUN** statement at the end of each step is required to complete the execution of that step.

DATA STEP

- This step involves loading the required data set into SAS memory and identifying the variables (also called columns) of the data set.
- It also captures the records (also called observations or subjects).

Syntax

```
DATA data_set_name;              #Name the data set.
INPUT var1,var2,var3;            #Define the variables in this data set.
NEW_VAR;                         #Create new variables.
LABEL;                           #Assign labels to variables.
DATALINES;                       #Enter the data.
RUN;
```

Example

The below example shows a simple case of naming the data set, defining the variables, creating new variables and entering the data. Here the string variables have a $ at the end and numeric values are without it.

```
DATA TEMP;
INPUT ID $ NAME $ SALARY DEPARTMENT $;
comm = SALARY*0.25;
LABEL ID = 'Employee ID' comm = 'COMMISION';
DATALINES;
1 Rick 623.3 IT
2 Dan 515.2 Operations
3 Michelle 611 IT
4 Ryan 729 HR
5 Gary 843.25 Finance
6 Nina 578 IT
7 Simon 632.8 Operations
8 Guru 722.5 Finance
;
RUN;
```

PROC STEP

» This step involves invoking a SAS built-in procedure to analyse the data .

Example

The below example shows using the **MEANS** procedure to print the mean values of the numeric variables in the data set.

```
PROC MEANS;
RUN;
```

Syntax

```
PROC procedure_name options; #The name of the proc.
RUN;
```

OUTPUT STEP

» The data from the data sets can be displayed with conditional output statements.

Example

The below example shows using the where clause in the output to produce only few records from the data set.

```
PROC PRINT DATA=TEMP;
WHERE SALARY > 700;
RUN;
```

Syntax

```
PROC PRINT DATA = data_set;
OPTIONS;
RUN;
```

Complete SAS Program

Mean_salary.sas

CODE | LOG | RESULTS

```
1  ****DATA STEP*******;
2  DATA TEMP;
3  INPUT ID NAME $ SALARY DEPARTMENT $;
4  LABEL ID = 'Employee ID';
5  DATALINES;
6  1 Rick 623.3 IT
7  2 Dan 515.2 Operations
8  3 Michelle 611 IT
9  4 Ryan 729 HR
10 5 Gary 843.25 Finance
11 6 Nina 578 IT
12 7 Simon 632.8 Operations
13 8 Guru 722.5 Finance
14 ;
15 RUN;
16 *PROC STEP*********;
17 PROC MEANS;
18 RUN;
19 *OUTPUT STEP*******;
20 PROC PRINT DATA=TEMP;
21 WHERE SALARY > 700;
22 RUN;
```

*Mean_salary.sas

CODE | LOG | RESULTS | OUTPUT DATA

The MEANS Procedure

Analysis Variable : SALARY				
N	Mean	Std Dev	Minimum	Maximum
8	656.8812500	103.0594625	515.2000000	843.2500000

Obs	ID	NAME	SALARY	DEPARTMENT
4	4	Ryan	729.00	HR
5	5	Gary	843.25	Finance
8	8	Guru	722.50	Finance

3.

SAS Programming

SAS Statements

» A semicolon at the end of the last line marks the end of the statement.
»Many SAS statements can be on the same line, with each statement ending with a semicolon.
»Space can be used to separate the components in a SAS program statement.
»SAS keywords are not case sensitive.
»Every SAS program must end with a RUN statement.

SAS Variable Names

Variables in SAS represent a column in the SAS data set. The variable names follow the below rules.
»Maximum 32 characters long.
»It can not include blanks.
»It must start with the letters A through Z (not case sensitive) or an underscore (_).
»Can include numbers but not as the first character.
»Variable names are case insensitive.

```
# Valid Variable Names
REVENUE_YEAR
MaxVal
_Length
```

SAS Data Set

The DATA statement marks the creation of a new SAS data set. The rules for DATA set creation are as below.

» A single word after the DATA statement indicates a temporary data set name.
» The data set name can be prefixed with a library name which makes it a permanent data set.

```
# Temporary data sets.
DATA TempData;
DATA abc;
DATA newdat;

# Permanent data sets.
DATA LIBRARY1.DATA1
DATA MYLIB.newdat;
```

SAS Options

» SAS includes a large suite of system options that will affect your SAS session.
»Specific options are invoked by default when you open SAS.
»The options can vary depending what computing environment you are using (e.g. Windows, Unix).
»The **OPTIONS** procedure lists the current settings of SAS system options in the SAS log.

```
PROC OPTIONS;
RUN;
```

Diagnosing and Correcting Syntax Errors

» Color-Coded Syntax

```
TITLE "REGRESSION";
PROC GLM DATA = newdata;
CLASS female prog ;
MODEL read =  write female math prog /SOLUTION;
FORMAT female female. prog prog.;
RUN;

TITLE "REGRESSION;
PROC GLM DATA = newdata;
CLASS female prog ;
MODEL read =  write female math prog /SOLUTION;
FORMAT female female. prog prog.;
RUN;
```

» Log File

Manipulating DataSets

» **WHERE** statement :

```
PROC PRINT DATA=idre.sales;
WHERE Country='AU';
RUN;
```

» **IN** operator :

```
PROC PRINT DATA=idre.sales;
WHERE Country IN ('AU', 'US');
RUN;
```

One limitation of using a **WHERE** statement is that more than 1 can not be used simultaneously, except in special cases.

»Comparison Operators

```
PROC PRINT DATA=idre.sales;
WHERE Salary<30000;
RUN;

PROC PRINT DATA=idre.sales;
WHERE Salary ge 30000;
RUN;
```

»We also can specify SAS to output only certain ranges of values for numeric variables. In the first example above we ask SAS to output salary salary values that are less than (< <) $30,000. In the second example, we output salary salary values greater than or equal to (**ge**) $30,000.

Manipulating DataSets

» **Logical Operators** :

Symbol	Mnemonic
^ ~	NOT
&	AND
|	OR

```
PROC PRINT DATA=idre.sales;
WHERE Country='AU' AND Salary<30000;
RUN;

PROC PRINT data=idre.sales;
WHERE Country='AU' & Salary<30000;
RUN;
```

» **WHERE Operators** :

»Between-And
»Contains
»Is Null or Is Missing
»Like
»=*
»Same-And or Also

```
PROC PRINT DATA=idre.shoes_eclipse;
VAR product_name;
WHERE product_name LIKE "Woman's %";
RUN;
```

Obs	Product_Name
9	Woman's Air Amend Mid
10	Woman's Air Converge Triax X
11	Woman's Air Imara
12	Woman's Air Rasp Suede
13	Woman's Air Zoom Drive
14	Woman's Air Zoom Sterling

Manipulating DataSets

» **Arithmetic Operators** :

Symbol	Description
**	Exponentiation
*	Multiplication
/	Division
+	Addition
–	Subtraction

» If you are calculating values using a variable(s) with **missing data**, the resulting value will also **be missing**.

Example : we use parentheses to change the estimation of a compound (more then one operator) expression. We will use a Data step to create two new variables profit1 and profit2.

```
DATA profit;
SET idre.order_fact;
profit1 = total_retail_price - costPrice_per_unit * quantity;
profit2 = (total_retail_price - costPrice_per_unit) * quantity;
RUN;
```

Manipulating DataSets

» **Conditional Processing** :

- Conditional processing in SAS allows the user to manipulate and output portions of data instead of the whole file.

- While both **WHERE** and **IF** can be used with a **DATA** step, only **WHERE** is allowed in a **PROC** step.

» **IF-THEN / IF-THEN ELSE statements**

»An If-then statement is a commonly used assignment statement that is typically carried out within the context of Data Step. It executes a SAS statement that fulfills a certain condition.

```
DATA comp1;
SET idre.sales;
IF Job_Title='Sales Rep. IV' THEN Bonus=1000;
IF Job_Title='Sales Manager' THEN Bonus=1500;
IF Job_Title='Senior Sales Manager' THEN Bonus=2000;
IF Job_Title='Chief Sales Officer' THEN Bonus=2500;
RUN;
```

```
DATA comp2;
SET idre.sales;
IF Job_Title='Sales Rep. IV' THEN Bonus=1000;
ELSE IF Job_Title='Sales Manager' THEN Bonus=1500;
ELSE IF Job_Title='Senior Sales Manager' THEN Bonus=2000;
ELSE IF Job_Title='Chief Sales Officer' THEN Bonus=2500;
RUN;
```

Manipulating DataSets

» **Using Do**:

- Typically with an **IF-THEN** statement only one executable statement is allowed.

```
DATA bonus;
SET idre.sales;
IF Country='US' THEN DO;
Bonus=500;
Freq='Once a Year';
END;
ELSE DO;
Bonus=300;
Freq='Twice a Year';
END;
RUN;
```

» **SAS Functions**

- Arithmetic Functions :

```
DATA budget;
SET idre.oldbudget;
sum1 = yr2003 + yr2004 + yr2005 + yr2006 + yr2007;
sum2 = SUM(yr2003, yr2004, yr2005, yr2006, yr2007);
sum3 = SUM( of yr2003-yr2007);

mean1 = (yr2003 + yr2004 + yr2005 + yr2006 + yr2007)/5;
mean2 = MEAN(yr2003, yr2004, yr2005, yr2006, yr2007);
mean3 = MEAN( of yr2003-yr2007);

RUN;
```

- Date Functions :

```
DATA comp;
SET idre.sales;
Hire_Month=MONTH(Hire_Date);
Birth_Day = WEEKDAY(Birth_date);
Day_Dif = DATDIF(Birth_date,Hire_Date, 'actual');
Month_dif= INTCK('years',Birth_date,Hire_Date);
Bonus_1 = INTNX('month', Hire_Date, 6);
RUN;
```

Manipulating DataSets

» **Sorting** :

- The are many instances when having your data sorted in a particular way will be helpful for visualizing your data.

```
PROC SORT DATA=idre.sales OUT=sales;
BY Salary Country;
RUN;
```

Obs	Employee_ID	Last_Name	Salary	Country
71	120160	Segrave	27115	AU
72	121102	Flammia	27115	US
73	120162	Scordia	27215	AU
74	121029	Mcelwee	27225	US
75	121088	Kernitzki	27240	US

» **Merging** :

- One data management task that requires proper sorting is merging.
- Merging (One-to-One) or (One-to-Many)
- The datasets to be merged must be sorted by the same variable(s).

```
DATA payadd;
MERGE payroll addresses;
BY Employee_ID;
RUN;
```

Manipulating DataSets

» **Appending** :

- Appending or concatenating observations is the process of adding rows or observations to a dataset as opposed to merging which adds variables.

```
DATA mnth7_8_9_2011 ;
SET idre.mnth7_2011 idre.mnth8_2011 idre.mnth9_2011;
RUN;
```

» All you have to do is list the datasets to be appended on the **SET** statement line.
» The ordering and the number of datasets does not matter

Modifying SAS Output

» **Titles and Footnotes** :

- As a researcher it is important to know how to manipulate and change your output to convey important information to your audience.
- One of the procedures we have been using to obtain output from our various Data steps is **PROC PRINT**.

```
TITLE1 'Orion Star Sales Staff';
TITLE2 'Salary Report';
FOOTNOTE1 'Confidential';
PROC PRINT DATA=idre.sales (OBS=5);
VAR Employee_ID
Last_Name Salary;
RUN;
```

Orion Star Sales Staff
Salary Report

Obs	Employee_ID	Last_Name	Salary
1	121070	Holthouse	29385
2	120170	Kingston	28830
3	120171	Moody	26205
4	120135	Platts	32490
5	121027	Rudder	26165

Confidential

Modifying SAS Output

» **Label Options**:

```
PROC PRINT DATA=idre.sales (OBS=5) SPLIT='*';
VAR Employee_ID Last_Name Salary;
LABEL Employee_ID = 'Sales ID'
      Last_Name = 'Last*Name'
      Salary = 'Annual*Salary';
RUN;
```

Orion Star Sales Staff
Salary Report

Obs	Sales ID	Last Name	Annual Salary
1	121070	Holthouse	29385
2	120170	Kingston	28830
3	120171	Moody	26205
4	120135	Platts	32490
5	121027	Rudder	26165

Confidential

» **Formats** :

```
PROC FORMAT;
VALUE $ctryfmt      'AU'='Australia'
                    'US'='United States'
                     other ='Miscoded';

VALUE tiers          0-49999='Tier 1'
                     50000-99999='Tier 2'
                     100000-250000='Tier 3';
RUN;
```

```
PROC PRINT DATA=idre.sales (OBS=5);
VAR Employee_ID Salary Country Birth_Date Hire_Date;
FORMAT Salary tiers. Birth_Date Hire_Date monyy7. Country $ctryfmt.;
RUN;
```

- Formatting values changes the appearance of those values in output but the underlying values does NOT change.

Special Issues

» **Dealing with Duplicates** :

An issue that comes up a lot in data management is how to handle duplicates.

- The **FREQ** procedure :

```
PROC FREQ DATA=idre.nonsales ORDER=FREQ;
TABLES Employee_ID;
RUN;
```

Employee_ID	Frequency	Percent	Cumulative Frequency	Cumulative Percent
120108	2	0.85	2	0.85
120101	1	0.43	3	1.28
120104	1	0.43	4	1.71
120105	1	0.43	5	2.14
120106	1	0.43	6	2.56

- Useful option **NLEVELS**:

```
PROC FREQ DATA=idre.nonsales NLEVELS;
TABLES Employee_ID /NOPRINT;
RUN;
```

Number of Variable Levels			
Variable	Levels	Missing Levels	Nonmissing Levels
Employee_ID	234	1	233

- There are 235 unique employees in the “nonsales” data but only 234 unique levels, meaning that one employee ID# is duplicated.

Special Issues

» **Identifying Outliers** :
Another issue that comes up a lot in dealing with data is outliers.

- The **UNIVARIATE** procedure :

```
PROC UNIVARIATE DATA=idre.price_new;
VAR unit_cost_price;
RUN;
```

Extreme Observations			
Lowest		Highest	
Value	Obs	Value	Obs
1.3	45	89.95	9
1.3	43	120.30	42
2.2	82	126.70	86
2.2	81	159.15	68
2.6	8	231.90	69

- Useful option **NEXTROBS** :

- Indicate the number of outliers to display. You can specify any number between 0 and half of the total observations.

Any questions?

You can find me at:

- zoumpeka@inf.uth.gr