Lesson2

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

Open **p102a04.sas** from the **activities** folder and perform the following task:

1. Write a PROC CONTENTS step to generate a report of the **storm_summary. sas7bdat** table properties. Highlight the step and run only the selected code.
2. How many observations are in the table?

   - **proc contents data="s:/workshop/data/storm_summary.sas7bdat";**

     run;

LIBNAME *libref engine* "*path*";

1. Open a new program. Write a LIBNAME statement to create a library named **PG1** that reads SAS tables in the **data** folder.

   - **libname pg1 base "s:/workshop/data";**

LIBNAME *libref* XLSX "*path/file-name*.xlsx";

OPTIONS VALIDVARNAME=V7;

LIBNAME *libref* CLEAR;

1.Open **p102a07.sas** from the **activities** folder and perform the following tasks:

1. If necessary, update the path of the course files in the LIBNAME statement.
2. Complete the PROC CONTENTS step to read the **parks** table in the **NP** library.

   - ```
     proc contents data=np.parks;
     run;
     ```

2.Write a LIBNAME statement to create a library named **NP** that reads **np_info.xlsx** in the course data as follows

- ```
  libname np xlsx "FILEPATH/np_info.xlsx";
  ```

3.Write an OPTIONS statement to ensure that column names follow SAS naming conventions.
4.Write a PROC CONTENTS step to read the **Parks** table in the **NP** library.
5.Add a LIBNAME statement after PROC CONTENTS to clear the **NP** library.
6.Run the program and examine the log. Which column names have been modified to follow SAS naming conventions?

- ```
  options validvarname=v7;
  ```
- ```
  proc contents data=np.parks;
  run;
  ```
- ```
  libname np clear;
  ```
  ```

```
PROC IMPORT DATAFILE="path/file-name.csv" DBMS=CSV
                OUT=output-table <REPLACE>;
      <GUESSINGROWS=n|MAX;>
RUN;
```

Open **p102a08.sas** from the **activities** folder and perform the following tasks:

1. This program imports a tab-delimited file. Run the program twice and carefully read the log. What is different about the second submission?
2. Fix the program and rerun it to confirm that the import is successful.

   - **proc import datafile="s:/workshop/data/storm_damage.tab"**

     dbms=tab out=storm_damage_tab;

     run;

   - **fix : proc import datafile="s:/workshop/data/storm_damage.tab"**

     dbms=tab out=storm_damage_tab replace;

     run;

```
PROC IMPORT DATAFILE="path/file-name.xlsx" DBMS=XLSX
                OUT=output-table <REPLACE>;
    SHEET=sheet-name;
RUN;
```

Practice:

1. Open **p102p01.sas** from the **practices** folder.
   - Complete the PROC IMPORT step to read **eu_sport_trade.xlsx**.
   - Create a SAS table named **eu_sport_trade** and replace the table if it exists.
2. Modify the PROC CONTENTS code to display the descriptor portion of the **eu_sport_trade** table

   - **proc import datafile="/folders/myfolders/EPG194/data/eu_sport_trade.xlsx"**

     Dbms=xlsx out=eu_sport_trade replace;

     run;

   - **proc contents data=eu_sport_trade ;**

     run;

1. Create a new program.
    - Write a PROC IMPORT step to read the **np_traffic.csv** file and create the **traffic** SAS table.
    - Add a PROC CONTENTS step to view the descriptor portion of the newly created table.
    - Submit the program.
2. Examine the data interactively. Scroll down to row 37. Notice that the values for **ParkName** and **TrafficCounter** seem to be truncated.

3. Modify the program to resolve this issue. Submit the program and verify that **ParkName** and **TrafficCounter** are no longer truncated

```
proc import datafile="FILEPATH/np_traffic.csv"
     dbms=csv  out=traffic replace;
run;

proc contents data=traffic;
run;
```

```
proc import datafile="FILEPATH/np_traffic.csv"
     dbms=csv
     out=traffic
     replace;
   //guessingrows=max;
     run;

proc contents data=traffic;
   run
```

Lesson3

- Exploring data:
  - PRINT Procedure: By default, PROC PRINT lists all columns and rows in the input table.

```
PROC PRINT DATA=input-table (OBS=n);
Var col-name;
RUN;
```

  - means Procedure: By default, PROC MEANS generates simple summary statistics for each numeric column
    in the input data

```
PROC MEANS DATA=input-table;
      VAR col-name(s);
RUN;
```

  - univariate Procedure: By default, PROC UNIVARIATE generates summary statistics for each numeric column in the input data

```
PROC UNIVARIATE DATA=input-table;
      VAR col-name(s);
RUN;
```

  - freq Procedure: By default, PROC FREQ creates a frequency table for each column in the input table.

```
PROC FREQ DATA=input-table;
      TABLES col-name(s);
RUN;
```

practice:

1.Open **p103p01.sas** from the **practices** folder.

- Complete the PROC PRINT statement to list the first 20 observations in **pg1.np_summary**.
- Add a VAR statement to include only the following variables: **Reg**, **Type**, **ParkName**, **DayVisits**, **TentCampers**, and **RVCampers**.
- Highlight the step and run the selected code.
- Do you observe any possible inconsistencies in the data?

```
➢ proc print data=pg1.np_summary(obs=20);
    var Reg Type ParkName DayVisits TentCampers
RVCampers;
run;
```

2.Copy the PROC PRINT step and paste it at the end of the program.

- Replace PRINT with MEANS and remove the OBS= data set option.
- Modify the VAR statement to calculate summary statistics for **DayVisits**, **TentCampers**, and **RVCampers**.
- Highlight the step and run the selected code.

```
➢ proc means data=pg1.np_summary;
  DayVisits TentCampers RVCampers;
   run;
```

4.Copy the PROC MEANS step and paste it at the end of the program.

- Replace MEANS with UNIVARIATE.
- Highlight the step and run the selected code.

```
➢ proc univariate data=pg1.np_summary;
 var DayVisits TentCampers RVCampers;
 run;
```

5.Are there negative values for any of the columns? no

6.Copy the PROC UNIVARIATE step and paste it at the end of the program.

- Replace UNIVARIATE with FREQ.
- Replace the VAR statement with a TABLES statement to produce frequency tables for **Reg** and **Type**.
- Highlight the step and run the selected code.

```
➢ proc freq data=pg1.np_summary;
  tables Reg Type;
   run;
```

- Filtering Rows with the WHERE Statement

```
PROC procedure-name . . . ;
     WHERE expression;
RUN;
```

= or **EQ** | < or **LT** | ^= or ~= or **NE**

> or **GT** | <= or **LE** | >= or **GE**

- Date: **"ddmmmyyyy"d**

- using in operator:

```
WHERE col-name IN (value-1,…,value-n);
WHERE col-name NOT IN (value-1,…,value-n);
```

```
WHERE col-name IS MISSING;
WHERE col-name IS NOT MISSING;
```

```
WHERE col-name BETWEEN value-1 AND value-2;
```

```
WHERE col-name LIKE "value";
```

1.Open **p103a02.sas** from the **activities** folder and perform the following tasks:

1. Add a new WHERE statement to print storms that begin with Z. How many storms are included in the results?

   - **proc print data=pg1.storm_summary(obs=50);**

     where name like "Z%";

     run;

- **Marco variable :**

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

&*macro-var*

1.Open **p103a03.sas** from the **activities** folder and perform the following tasks:

1. Change the value in the %LET statement from **NA** to **SP**.

   - **%let BasinCode=SP;**

     proc means data=pg1.storm_summary;

     where Basin="&BasinCode";

     run;

practice :

1.Open **p103p04.sas** from the **practices** folder.

Add a WHERE statement to print only the rows
where **ParkName** includes *Preserve*. **ParkName** contains character values, and these values are case sensitive.

```
proc print data=pg1.np_summary;
var Type ParkName;
where ParkName like '%Preserve%';
 run;
```

2.Create a new program.

- Write a PROC PRINT step to read the **pg1.eu_occ** table.
- Use a WHERE statement to list rows where **Hotel**, **ShortStay**, and **Camp** are missing.
- Submit the program.

```
proc print data=pg1.eu_occ;
where Hotel is missing and ShortStay is missing and
      Camp is missing;
   run;
```

2. Modify the WHERE statement to list rows with **Hotel** values greater than 40,000,000. Submit the program.

```
proc print data=pg1.eu_occ;
where Hotel > 40000000;
run;
```

3.Create a new program.

- Write a PROC FREQ step to analyze rows from **pg1.np_species**.
- Include only rows where **Species_ID** starts with *YOSE* (Yosemite National Park) and **Category** Equals *Mammal*.
- Generate frequency tables for **Abundance** and **Conservation_Status**.

```
proc freq data=pg1.np_species;
tables Abundance Conservation_Status;
where Species_ID like "YOSE%" and
      Category="Mammal";
 Run;
```

4.Write a PROC PRINT step to list the same subset of rows from **pg1.np_species**.

- Include **Species_ID**, **Category**, **Scientific_Name**, and **Common_Names** in the report..

```
proc print data=pg1.np_species;
var Species_ID Category Scientific_Name Common_Names;
 where Species_ID like "YOSE%" and
  Category="Mammal";
  run;
```

5.Create a macro variable named ParkCode to store *YOSE*, and another macro variable named SpeciesCat to store *Mammal*. Modify the code to reference the macro variables. Submit the program and confirm that the same results are generated.

Solution:

```
%let ParkCode=YOSE;
%let SpeciesCat=Mammal;

proc freq data=pg1.np_species;
 tables Abundance Conservation_Status;
  where Species_ID like "&ParkCode%" and
    Category="&SpeciesCat";
   run;

 proc print data=pg1.np_species;
var Species_ID Category Scientific_Name Common_Names;
where Species_ID like "&ParkCode%" and
Category="&SpeciesCat";
     Run;
```

- **formatting columns**

```
PROC PRINT DATA=input-table;
     FORMAT col-name(s)  format;
RUN;
```

<$>*format-name*<*w*>.<*d*>

Open **p103a05.sas** from the **activities** folder and perform the following tasks:

1. Highlight the PROC PRINT step and run the selected code. Notice how the values of **Lat**, **Lon**, **StartDate**, and **EndDate** are displayed in the report.
2. Change the width of the DATE format to 7 and run the PROC PRINT step. How does the display of **StartDate** and **EndDate** change? DATE7. displays a 2-digit year
3. Change the width of the DATE format to 11 and run the PROC PRINT step. How does the display of **StartDate** and **EndDate** change? DATE11. displays a 4-digit year and adds dashes
4. Highlight the PROC FREQ step and run the selected code. Notice that the report includes the number of storms for each **StartDate**.
5. Add a FORMAT statement to apply the MONNAME. format to **StartDate** and run the PROC FREQ step. How many rows are in the report?

```
proc print data=pg1.storm_summary(obs=20);

 format Lat Lon 4. StartDate EndDate date9.;

 run;

proc freq data=pg1.storm_summary order=freq;

 tables StartDate;

format StartDate MONNAME.;

 run;
```

- **sorting data:**

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

Open **p103a06.sas** from the **activities** folder and perform the following tasks:

1. Modify the OUT= option in the PROC SORT statement to create a temporary table named **storm_sort**.
2. Complete the WHERE and BY statements to answer the following question: Which storm in the North Atlantic basin (*NA* or *na*) had the strongest **MaxWindMPH**?

    - **proc sort data=pg1.storm_summary out=storm_sort;**

      where Basin in("NA" "na");

      by descending MaxWindMPH;

      run;

- **Identifying and Removing Duplicate Rows:**

```
PROC SORT DATA=input-table <OUT=output-table>
            NODUPRECS <DUPOUT=output-table>;
      BY _ALL_;
RUN;
```

- **Identifying and Removing Duplicate Key Values:**

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

Practice:

1.Open **p103p08.sas** from the **practices** folder.

- Modify the PROC SORT step to read **pg1.np_summary** and create a temporary, sorted table named **np_sort**.
- Add a BY statement to order the data by **Reg** and descending **DayVisits** values.
- Add a WHERE statement to select **Type** equal to *NP*.
- Submit the program and view the output data

```
proc sort data=pg1.np_summary out=np_sort;
 by Reg descending DayVisits;
 where Type="NP";
 run;
```

2.Create a new program.

- Write a PROC SORT step that creates two tables (**park_clean** and **park_dups**), and removes the duplicate rows.
- Submit the program and view the output data

```
proc sort data=pg1.np_largeparks
     out=park_clean
     dupout=park_dups
     noduprecs;
     by _all_;
     run;
```

Lesson4

- **DATA Step:** filter rows and columns, compute new columns

```
DATA output-table;
      SET input-table;
RUN;
```

- **Compilation:** Check syntax for errors. Identify column attributes. Establish new table metadata**.**
- **Execution:** Read and write data, Perform data manipulations, calculations.

1.Open **p104a01.sas** from the **activities** folder and perform the following tasks:

1. Complete the DATA step to create a temporary table named **storm_new** and read **pg1.storm_summary**. Run the program and read the log.
   - **data storm_new;**

     set pg1.storm_summary;

     run;
2. Define a library named **out** pointing to the **output** folder in the main course files folder.
   - **libname out "s:/workshop/output";**

     data out.storm_new;

     set pg1.storm_summary;

     run;

Change the program to save a permanent version of **storm_new** in the **out** library. Run the modified program.

- **Filtering rows in data step:**

```
DATA output-table;
      SET input-table;
      WHERE expression;
RUN;
```

```
DROP col-name <col-name>;
```

```
KEEP col-name <col-name>;
```

```
DATA output-table;
      SET input-table;
      FORMAT col-name format;
RUN;
```

1. Write a DATA step that reads the **pg1.storm_summary** table and creates an output table named **Storm_cat5**. **Note**: If you are using SAS Studio, try creating **storm_cat5** as a permanent table in the **EPG194/output** folder.

2. Include only Category 5 storms (**MaxWindMPH** greater than or equal to 156) with **StartDate** on or after 01JAN2000.

3. Add a statement to include the following columns in the output data: **Season**, **Basin**, **Name**, **Type**, and **MaxWindMPH**.

```
data storm_cat5;
set pg1.storm_summary;
where StartDate>="01jan2000"d and MaxWindMPH>=156;
keep Season Basin Name Type MaxWindMPH;
run;
```

practice:

1.Open **p104p01.sas** from the **practices** folder.

- Modify the code to create a temporary table named **eu_occ2016** and read **pg1.eu_occ.**
- Complete the WHERE statement to select only the stays that were reported in 2016. Notice that **YearMon** is a character column and the first four positions represent the year.
- Complete the FORMAT statement to apply the COMMA17. format to the **Hotel**, **ShortStay**, and **Camp** columns.
- Complete the DROP statement to exclude **Geo** from the output table.
- Submit the program and view the output data.

```
data eu_occ2016;
set pg1.eu_occ;
where YearMon like "2016%";
format Hotel ShortStay Camp comma17.;
drop geo;
run;
```

2.Create a new program.

- Write a DATA step to read the **pg1.np_species** table and create a new table named **fox**.
  Note: If you are using SAS Studio, try creating **fox** as a permanent table in the **EPG194/output** folder.

- Include only the rows where **Category** is *Mammal* and **Common_Names** includes *Fox* in any case.

- Exclude the **Category**, **Record_Status**, **Occurrence**, and **Nativeness** columns.

```
data fox;
set pg1.np_species;
 where Category='Mammal' and upcase(Common_Names)
like'%FOX%';
drop Category Record_Status Occurrence Nativeness;
run;
```

- Notice that *Fox Squirrels* are included in the output table. Add a condition in the WHERE statement to exclude rows that include *Squirrel*. Submit the program and verify the results.

```
data fox;
 set pg1.np_species;
    where Category='Mammal' and upcase(Common_Names)
like '%FOX%' and upcase(Common_Names) not like
'%SQUIRREL%';
 drop Category Record_Status Occurrence Nativeness;
 run;
```

- Sort the **fox** table by **Common_Names**

```
proc sort data=fox;
 by Common_Names;
 run
```

- **Create New Columns:**

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

1.Open **p104a04.sas** from the **activities** folder and perform the following tasks:

1. Add an assignment statement to create **StormLength** that represents the number of days between **StartDate** and **EndDate**

```
data storm_length;
set pg1.storm_summary;
drop Hem_EW Hem_NS lat lon;
StormLength = EndDate-StartDate;
run;
```

- **Functions:sum,mean,median,range,min,max,n,nmiss**

```
function(argument1, argument2, ...);
```

```
DATA output-table;
      SET input-table;
      new-column=function(arguments);
RUN;
```

2.Open **p104a05.sas** from the **activities** folder and perform the following tasks:

1. Open the **pg1.storm_range** table and examine the columns. Notice that each storm has four wind speed measurements.
2. Create a new column named **WindAvg** that is the mean of **Wind1**, **Wind2**, **Wind3**, and **Wind4**.
3. Create a new column **WindRange** that is the range of **Wind1**, **Wind2**, **Wind3**, and **Wind4**.

```
data storm_windavg;
set pg1.storm_range;
WindAvg=mean(wind1, wind2, wind3, wind4);
WindRange=range(of wind1-wind4);
run;
```

- **Character Functions:**

UPCASE(char), lOWCASE(char) , PROPCASE(char, <*delimiters*>) , CATS(char1,char2,..) , SUBSTR (*char*, *position*, <*length*>)

3.Open **p104a06.sas** from the **activities** folder and perform the following tasks:

1. Add a WHERE statement that uses the SUBSTR function to include rows where the second letter of **Basin** is *P* (Pacific ocean storms).
2. Run the program and view the log and data. How many storms were in the Pacific basin?

```
data pacific;
set pg1.storm_summary;
drop type Hem_EW Hem_NS MinPressure Lat Lon;
where substr(Basin,2,1)="P";
run;
```

- **Date Functions:**

MONTH (*SAS-date*), YEAR (*SAS-date*), DAY (*SAS-date*)
WEEKDAY (*SAS-date*), QTR (*SAS-date*), TODAY(),
MDY (*month*, *day*, *year*), YRDIF (*startdate*, *enddate*, 'AGE').

Practice:

1. Open **p104p04.sas** from the **practices** folder.

- Create a new column named **SqMiles** by multiplying **Acres** by .0015625.
- Create a new column named **Camping** as the sum of **OtherCamping**, **TentCampers**, **RVCampers**, and **BackcountryCampers**.
- Format **SqMiles** and **Camping** to include commas and zero decimal places.
- Modify the KEEP statement to include the new columns.

```
data np_summary_update;
 set pg1.np_summary;
 keep Reg ParkName DayVisits OtherLodging Acres SqMiles Camping;
SqMiles=Acres*.0015625;
Camping=sum(OtherCamping,TentCampers,
RVCampers,BackcountryCampers);
format SqMiles comma6. Camping comma10.;
 run;
```

2. Write a DATA step to create a temporary table named **eu_occ_total** that is based on the **pg1.eu_occ** table.

- Create the following new columns:
  - **Year**: the four-digit year extracted from **YearMon**
  - **Month**: the two-digit month extracted from **YearMon**
  - **ReportDate**: the first day of the reporting month
    Note: Use the MDY function and the new **Year** and **Month** columns
  - **Total**: the total nights spent at any establishment
- Format **Hotel**, **ShortStay**, **Camp**, and **Total** with commas. Format **ReportDate** to display the values in the form JAN2018.
- Keep **Country**, **Hotel**, **ShortStay**, **Camp**, **ReportDate**, and **Total** in the new table.
- Submit the program and view the output data

```
data eu_occ_total;
  set pg1.eu_occ;
  Year=substr(YearMon,1,4);
  Month=substr(YearMon,6,2);
  ReportDate=MDY(Month,1,Year);
  Total=sum(Hotel,ShortStay,Camp);
  format Hotel ShortStay Camp Total comma17.
    ReportDate monyy7.;
keep Country Hotel ShortStay Camp ReportDate Total;
   run;
```

- **Conditional Processing with IF-THEN:**

IF *expression* THEN *statement*;

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

1.Open **p104a07.sas** from the **activities** folder and perform the following tasks:

1. Add the **ELSE** keyword to test conditions sequentially until a true condition is met.
2. Change the final IF-THEN statement to an ELSE statement. Run the code.

```
data storm_cat;
 set pg1.storm_summary;
 keep Name Basin MinPressure StartDate PressureGroup;
*add ELSE keyword and remove final condition;
 if MinPressure=. then PressureGroup=.;
else if MinPressure<=920 then PressureGroup=1;
else PressureGroup=0;
run;
```

LENGTH *char-column* $ *length*;

1.Open **p104a08.sas** from the **activities** folder and perform the following tasks:

1. Run the program and examine the results. Why is **Ocean** truncated?
   What value is assigned when Basin='*na*'?
2. Modify the program to add a LENGTH statement to declare the name, type, and length of **Ocean** before the column is created.
3. Add an assignment statement after the KEEP statement to convert **Basin** to uppercase. Run the program.
4. Move the LENGTH statement to the end of the DATA step. Run the program. Does it matter where the LENGTH statement is in the DATA step?

```
data storm_summary2;
set pg1.storm_summary;
length Ocean $ 8;
keep Basin Season Name MaxWindMPH Ocean;
Basin=upcase(Basin);
OceanCode=substr(Basin,2,1);
if OceanCode="I" then Ocean="Indian";
 else if OceanCode="A" then Ocean="Atlantic";
 else Ocean="Pacific";
 run;
```

DATA *table1 table2...*

OUTPUT *table*;

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

Practice:

1. Open **p104p07.sas** from the **practices** folder. Submit the program and view the generated output.

2. In the DATA step, use IF-THEN/ELSE statements to create a new column, **ParkType**, that is based on the value of **Type**.

Type	ParkType
NM	Monument
NP	Park
NPRE, PRE, or PRESERVE	Preserve
NS	Seashore
RVR or RIVERWAYS	River

3. Modify the PROC FREQ step to generate a frequency report for **ParkType**. Submit the program.

```
➢ data park_type;
 set pg1.np_summary;
 length ParkType $ 8;
  if Type='NM' then ParkType='Monument';
  else if Type='NP' then ParkType='Park';
  else if Type in ('NPRE', 'PRE', 'PRESERVE') then
      ParkType='Preserve';
  else if Type in ('RVR', 'RIVERWAYS') then
ParkType='River';
  else if Type='NS' then ParkType='Seashore';
   run;

➢ proc freq data=park_type;
  tables Type;
  run;
```

2.Write a DATA step to create two temporary tables, named **parks** and **monuments**, that are based on the **pg1.np_summary** table. Read only national parks or monuments from the input table. (**Type** is either *NP* or *NM*.)

```
data parks monuments;
 set pg1.np_summary;
 where type in ('NM', 'NP');
 run;
```

1. Create a new column named **Campers** that is the sum of all columns that contain counts of campers. Format the column to include commas.

```
data parks monuments;
 set pg1.np_summary;
 where type in ('NM', 'NP');
 Campers=sum(OtherCamping, TentCampers, RVCampers,
           BackcountryCampers);
format Campers comma17.;
run;
```

2. When **Type** is *NP*, create a new column named **ParkType** that is equal to **Park**, and write the row to the **parks** table. When **Type** is *NM*, assign **ParkType** as **Monument** and write the row to the **monuments** table.

```
data parks monuments;
set pg1.np_summary;
where type in ('NM', 'NP');
Campers=sum(OtherCamping, TentCampers, RVCampers,
         BackcountryCampers);
format Campers comma17.;
length ParkType $ 8;
if type='NP' then do;
  ParkType='Park';
  output parks;
end;
else do;
  ParkType='Monument';
  output monuments;
end;
Keep Reg ParkName DayVisits OtherLodging Campers ParkType ;
run;
```

3. Keep **Reg**, **ParkName**, **DayVisits**, **OtherLodging**, **Campers**, and **ParkType** in both output tables. Submit the program and view the output data

Lesson5

- **Using Titles and Footnotes.**

TITLE<*n*> "*title-text*";	FOOTNOTE<*n*> "*footnote-text*";
clears titles and footnotes	turns off procedure titles
TITLE; FOOTNOTE;	ODS NOPROCTITLE;

1.Open **p105a01.sas** from the **activities** folder and perform the following tasks:

1. In the program, notice that there is a TITLE statement followed by two procedures. Run the program. Where does the title appear in the output?
2. Add a TITLE2 statement above PROC MEANS to print a second line: **Summary Statistics for MaxWind and MinPressure**
3. Add another TITLE2 statement above PROC FREQ with this title: **Frequency Report for Basin**
4. Run the program. Which titles appear above each report?
   - The title appears above both reports.

```
title "Storm Analysis";
title2 "Summary Statistics for MaxWind and MinPressure";
proc means data=pg1.storm_final;
   var MaxWindMPH MinPressure;
run;
title2 "Frequency Report for Basin";
proc freq data=pg1.storm_final;
   tables BasinName;
run;
```

- **The first title appears above both reports. The second title is replaced for the PROC FREQ output.**

- **Applying Temporary Labels to Columns**

LABEL *col-name*="*label-text*";

1.Open p105a03.sas from the activities folder and perform the following tasks:

1. Modify the LABEL statement in the DATA step to label the Invoice column as Invoice Price.

```
data cars_update;
set sashelp.cars;
keep Make Model MSRP Invoice AvgMPG;
AvgMPG=mean(MPG_Highway, MPG_City);
label MSRP="Manufacturer Suggested Retail Price"
      AvgMPG="Average Miles per Gallon"
      Invoice="Invoice Price";
run;
```

2. Run the program. Why do the labels appear in the PROC MEANS report but not in the PROC PRINT report? Fix the program and run it again.

```
proc means data=cars_update min mean max;
 var MSRP Invoice;
 run;
proc print data=cars_update label;
var Make Model MSRP Invoice AvgMPG;
 run;
```

   - **Most procedures automatically display permanent labels but you must use the LABEL option in PROC PRINT**

- **Creating One-Way Frequency Reports and Graphs**

```
PROC FREQ DATA=input-table < options >;
      TABLES  col-name(s) < / options >;
RUN;
```

1.Open **p105a04.sas** from the **activities** folder and perform the following tasks:

1. Create a temporary output table named **storm_count** by completing the OUT= option in the TABLES statement.
2. Add the NOPRINT option in the PROC FREQ statement to suppress the printed report.
3. Run the program. Which statistics are included in the output table?
   Which month has the highest number of storms?
   - **September (With ORDER=FREQ, the highest count is listed first.)**

```
title "Frequency Report for Basin and Storm Month";
proc freq data=pg1.storm_final order=freq noprint;
tables StartDate BasinName / out=storm_count;
format StartDate monname.;
run;
```

- **Creating two-Way Frequency Reports and Graphs**

```
PROC FREQ DATA=input-table < options >;
      TABLES  col-name*col-name < / options >;
RUN;
```

Practice:

1. Write a PROC FREQ step to analyze rows from **pg1.np_species**.

    - Use the TABLES statement to generate a frequency table for **Category**.
    - Use the NOCUM option to suppress the cumulative columns.
    - Use the ORDER=FREQ option in the PROC FREQ statement to sort the results by descending frequency.
    - Use **Categories of Reported Species** as the report title.
    - Submit the program and review the results.

```
title1 "Categories of Reported Species";
proc freq data=pg1.np_species order=freq;
tables Category / nocum;
run;
```

2. Modify the PROC FREQ step to make the following changes:

    - Include only the rows where **Species_ID** starts with *EVER* and **Category** is not *Vascular Plant*.
      Note: *EVER* represents Everglades National Park.
    - Turn on ODS Graphics before the PROC FREQ step and turn off the procedure title.
    - Add the PLOTS=FREQPLOT option to display frequency plots.
    - Add **in the Everglades** as a second title.
    - Submit the program and review the results

```
ods graphics on;
ods noproctitle;
title1 "Categories of Reported Species";
title2 "in the Everglades";
proc freq data=pg1.np_species order=freq;
tables Category / nocum plots=freqplot;
where Species_ID like "EVER%" and
Category ne "Vascular Plant";
run;
title;
```

1. Create a new program. Write a PROC FREQ step to analyze rows from **pg1.np_codelookup**.

    - Generate a two-way frequency table for **Type** by **Region**.
    - Exclude any park type that contains the word *Other*.
    - The levels with the most rows should come first in the order.
    - Suppress the display of column percentages.
    - Use **Park Types by Region** as the report title.
    - Submit the program and review the results.

```
title1 'Park Types by Region';
proc freq data=pg1.np_codelookup order=freq;
tables Type*Region / nocol;
where Type not like '%Other%';
 run;
```

2. What are the top three park types based on total frequency count?

    Note: Statistics labels appear in the main table in Enterprise Guide if SAS Report is the output format.

3. Modify the PROC FREQ step to make the following changes:

    - Limit the park types to the three that were determined in the previous step.
    - In addition to suppressing the display of column percentages, use the CROSSLIST option to display the table.
    - Add a frequency plot that groups the bars by the row variable, displays row percentages, and has a horizontal orientation.
      Note: Use SAS documentation to learn how the GROUPBY=, SCALE=, and ORIENT= options can be used to control the appearance of the plot.
    - Use **Selected Park Types by Region** as the report title.
    - Submit the program and review the results

```
title1 'Selected Park Types by Region';
ods graphics on;
proc freq data=pg1.np_codelookup order=freq;
tables Type*Region /  nocol crosslist
     plots=freqplot(groupby=row scale=grouppercent
orient=horizontal);
    where Type in ('National Historic Site', 'National
Monument', 'National Park');
    run;
    title;
```

- **Creating a Summary Statistics Report**

```
PROC MEANS DATA=input-table <stat-list> <options>;
      VAR col-name(s);
      CLASS col-name(s);
      WAYS n;
RUN;
```

Ways: specify ways to combine values of columns in the CLASS statement

Class: group the statistics based on distinct values of columns in the CLASS statement

Open **p105a06.sas** from the **activities** folder and perform the following tasks:

1. Add options to specify the following statistics in the PROC MEANS output: N (count), MEAN, and MIN. Round each statistic to the nearest integer.
2. Add a CLASS statement to group the data by **Season** and **Ocean** so statistics are calculated for each unique combination of season and ocean. Run the program.

```
proc means data=pg1.storm_final maxdec=0 n mean min;
var MinPressure;
where Season >=2010;
 class Season Ocean;
run;
```

3.Modify the program to add the WAYS statement so that separate reports are created for **Season** and **Ocean** statistics. Run the program

```
ways 1;
```

- **Creating an Output Summary Table**

```
OUTPUT OUT=output-table <statistic=col-name>;
```

1.Open **p105a06.sas** from the **activities** folder and perform the following tasks:

1. Run the PROC MEANS step and compare the report and the **wind_stats** table. Are the same statistics in the report and table? What do the first five rows in the table represent?
2. Uncomment the WAYS statement. Delete the statistics listed in the PROC MEANS statement and add the NOPRINT option. Run the program. Notice that a report is not generated and the first five rows from the previous table are excluded.

```
proc means data=pg1.storm_final noprint;
     var MaxWindMPH;
     class BasinName;
     ways 1;
     output out=wind_stats;
run;
```

```
output out=wind_stats mean=AvgWind max=MaxWind;
```

practice:

1.Create a new program. Write a PROC MEANS step to analyze rows from **pg1.np_westweather** with the following specifications:

- Generate the mean, minimum, and maximum statistics for the **Precip**, **Snow**, **TempMin**, and **TempMax** columns.
- Use the MAXDEC= option to display the values with a maximum of two decimal positions.
- Use the CLASS statement to group the data by **Year** and **Name**.
- Use **Weather Statistics by Year and Park** as the report title.
- Submit the program and review the results

```
title1 'Weather Statistics by Year and Park';
proc means data=pg1.np_westweather mean min max maxdec=2;
    var Precip Snow TempMin TempMax;
    class Year Name;
run;
```

2.Create a new program. Write a PROC MEANS step to analyze rows from **pg1.np_westweather** with the following specifications:

- Exclude rows where values for **Precip** are equal to *0*.
- Analyze precipitation amounts grouped by **Name** and **Year**.
- Create only an output table, named **rainstats**, with columns for the N and SUM statistics.
- Name the columns **RainDays** and **TotalRain**, respectively.
- Keep only those rows that are the combination of **Year** and **Name**.
- Submit the program and view the output data.

```
proc means data=pg1.np_westweather noprint;
where Precip ne 0;
var Precip;
class Name Year;
 ways 2;
 output out=rainstats n=RainDays sum=TotalRain;
 run;
```

2. Write a PROC PRINT step to print the **rainstats** table.

- Suppress the printing of observation numbers, and display column labels.
- Display the columns in the following order: **Name**, **Year**, **RainDays**, and **TotalRain**.
- Label **Name** as **Park Name**, **RainDays** as **Number of Days Raining**, and **TotalRain** as **Total Rain Amount (inches)**.
- Use **Rain Statistics by Year and Park** as the report title.

```
title1 'Rain Statistics by Year and Park';
 proc print data=rainstats label noobs;
   var Name Year RainDays TotalRain;
   label Name='Park Name'
     RainDays='Number of Days Raining'
     TotalRain='Total Rain Amount (inches)';
   run;
   title;
```

lesson6

- **Exporting Data Using Code**

```
PROC EXPORT DATA=input-table OUTFILE="output-file"
            <DBMS=identifier> <REPLACE>;
RUN;
```

1.Open **p106a02.sas** from the **activities** folder and perform the following tasks:

1. Complete the PROC EXPORT step to read the **pg1.storm_final** SAS table and create a comma-delimited file named **storm_final.csv**. Use **&outpath** to provide the path to the file.

2. Run the program and check the log to see if the records were written to the **storm_final.csv** file. If possible, open and view the text file

```
 proc export data=pg1.storm_final
outfile="&outpath/storm_final.csv"
dbms=csv replace;
run;
```

- **Exporting Data to excel**

```
libname myxl xlsx "&outpath/cars.xlsx";

data myxl.asiacars;
    set sashelp.cars;
    where origin='Asia';
run;
libname myxl clear;
```

2.Open **p106a03.sas** from the **activities** folder and perform the following tasks:

1. Complete the LIBNAME statement using the XLSX engine to create an Excel workbook named **storm.xlsx**.

2. Modify the DATA step to write the **storm_final** table to the **storm.xlsx** file. Use **&outpath** to provide the path to the file.

3. At the end of the DATA step, write a statement to clear the library

```
libname xl_lib xlsx "&outpath/storm.xlsx";
 data xl_lib.storm_final;
set pg1.storm_final;
drop Lat Lon Basin OceanCode;
run;

libname xl_lib clear;
```

- **exporting reports:**
- **Using the SAS Output Delivery System**

```
ODS <destination> <destination-specifications>;
/* SAS code that produces output */
ODS <destination> CLOSE;
```

- **Exporting results to a CSV File**

```
ODS CSVALL FILE="filename.csv";
/* SAS code that produces output */
ODS CSVALL CLOSE;
```

- **Exporting Results to Excel**

```
ODS EXCEL FILE="filename.xlsx" STYLE=style
          OPTIONS(SHEET_NAME='label');
 /* SAS code that produces output */
ODS EXCEL CLOSE;
```

1.Open **p106a04.sas** from the **activities** folder and perform the following tasks:

1. Add ODS statements to create an Excel file named **pressure.xlsx**. Use **&outpath** to provide the path to the file. Be sure to close the ODS location at the end of the program.

2. Add the STYLE=ANALYSIS option in the first ODS EXCEL statement. Run the program again and open the Excel file

```
➢ ods excel file="&outpath/pressure.xlsx"
   style=analysis;
title "Minimum Pressure Statistics by Basin";
```

.....

```
➢ ods excel close;
```

- **Exporting Output to PowerPoint and Microsoft Word**

```
ODS POWERPOINT FILE="filename.pptx" STYLE=style;
/* SAS code that produces output */
ODS POWERPOINT CLOSE;
```

```
ODS RTF FILE="filename.rtf" STARTPAGE=NO;
/* SAS code that produces output */
ODS RTF CLOSE;
```

1.Open **p106a05.sas** from the **activities** folder and perform the following tasks:

1. Modify the ODS statements to change the output destination to RTF. Change the style to **sapphire**.

2. Add the STARTPAGE=NO option in the first ODS RTF statement to eliminate the page break

```
➢ ods rtf file="&outpath/pressure.rtf"
   style=sapphire
  startpage=no;
title "Minimum Pressure Statistics by Basin";
ods noproctitle;
...
➢ ods rtf close;
```

- **export results to pdf**

ODS PDF FILE="*filename.pdf***" STYLE=***style*
STARTPAGE=NO PDFTOC=*n***;**
ODS PROCLABEL "*label***";**
/* SAS code that produces output */
ODS PDF CLOSE;

Practice:

1. Open **p106p01.sas** from the **practices** folder. Before the PROC MEANS step, add an ODS EXCEL statement to do the following:

   - Write the output file to **"&outpath/StormStats.xlsx"**.
   - Set the style for the Excel file to **snow**.
   - Set the sheet name for the first tab to **South Pacific Summary**.

```
➢ ods excel file="&outpath/StormStats.xlsx"
 style=snow
 options(sheet_name='South Pacific Summary');
proc means data=pg1.storm_detail maxdec=0 median max;
 class Season;
……..
 run;
```

2. Turn off the procedure titles and report titles at the start of the program. Turn the procedure titles on at the end of the program.

```
ods excel file="&outpath/StormStats.xlsx"
 style=snow
 options(sheet_name='South Pacific Summary');
  ➢ ods noproctitle;
proc means data=pg1.storm_detail maxdec=0 median max;
 class Season
  ➢ ods proctitle;
```

3. Immediately before the PROC PRINT step, add an ODS EXCEL statement to change the sheet name to **Detail**.

   - **ods excel options(sheet_name='Detail');**

```
proc print data=pg1.storm_detail noobs;
where Basin='SP' and Season in (2014,2015,2016);
by Season;
run;
ods proctitle;
```

4. At the end of the program, add an ODS EXCEL statement to close the Excel destination.

   - **ods excel close;**

1. Open **p106p02.sas** from the **practices** folder. Modify the program with the following specifications:

   - Add an ODS statement to write the output file to **&outpath/ParkReport.rtf**.
   - Set the style for the output file to **Journal** and remove page breaks between procedure results.
   - Turn off the procedure titles at the start of the program. Turn the procedure titles on at the end of the program.
   - At the end of the program, close the ODS destination.

```
ods rtf file="&outpath/ParkReport.rtf" style=Journal
startpage=no;
ods noproctitle;
..
..
ods proctitle;
ods rtf close;
```

2. Modify your SAS program so that both tables are created using the **Journal** style, but the **SASDOCPRINTER** style is used to create the graph.

   Note: An ODS destination statement enables you to specify a style without requiring you to redefine the output file location.

```
ods rtf style=SASDocPrinter;
title2 'Day Vists vs. Camping';
….
ods proctitle;
ods rtf close;
```

3. Add an OPTIONS statement with the NODATE option at the beginning of the program to suppress the date and time in the RTF file. Restore the option for future submissions by adding an OPTIONS statement with the DATE option at the end of the program.

   - **options nodate;**

lesson7

proc sql syntax:

PROC SQL;

SELECT clause
FROM clause
<WHERE clause>
<ORDER BY clause>;

PROC SQL;
SELECT *col-name*, *col-name*
FROM *input-table*;
QUIT;

QUIT;

expression AS *col-name*

1.Open **p107a01.sas** from the **activities** folder.

1. What are the similarities and differences in the syntax of the two steps?
2. What are the similarities and differences in the results.

```
proc print data=pg1.class_birthdate;
    var Name Age Height Birthdate;
    format Birthdate date9.;
run;
```

```
proc sql;
select Name, Age, Height*2.54 as HeightCM format=5.1,
       Birthdate format=date9.
    from pg1.class_birthdate;
quit;
```

1. Both the PRINT procedure and the SQL procedure provide the input table name, column list, and format.
2. PROC PRINT uses multiple statements to specify input data and report contents. PROC SQL uses one statement.
3. PROC PRINT separates columns with spaces. PROC SQL separates columns with commas.
4. PROC PRINT ends with a RUN statement. PROC SQL ends with a QUIT statement.
5. PROC SQL allows computed columns in the SELECT clause

1.All rows and selected columns are included in both reports.
2.PROC PRINT adds the OBS column by default.
.

2. Open **p107a02.sas** from the **activities** folder and perform the following tasks:

1. Complete the SQL query to display **Event** and **Cost** from **pg1.storm_damage**. Format the values of **Cost**.
2. Add a new column named **Season** that extracts the year from **Date**.
3. Add a WHERE clause to return rows where **Cost** is greater than 25 billion.
4. Add an ORDER BY clause to arrange rows by descending **Cost**.

```
proc sql;
select Event,
 Cost format=dollar16.,
 year(Date) as Season
from pg1.storm_damage
where Cost>25000000000
order by Cost desc;
  quit;
```

2. Open **p107a03.sas** from the **activities** folder and perform the following tasks:

1. Modify the query to create a temporary table named **top_damage**.
2. Add an additional query in the same PROC SQL step to generate a report listing all columns for the first 10 storms in the **top_damage** table.
3. Add a TITLE statement before the second query to display the following text: **Top 10 Storms by Damage Cost**.

```
      proc sql;
create table top_damage as
select Event,
       Date format=monyy7.,
       Cost format=dollar16.
       from pg1.storm_damage
       order by Cost desc;
title "Top 10 Storms by Damage Cost";
    select *
        from top_damage(obs=10);
quit;
```

- **creates table in sql** CREATE TABLE *table-name* AS
- **deleting table in sql** DROP TABLE *table-name*;
- **Creating Inner Joins in SQL**

FROM *table1* INNER JOIN *table2*
ON *table1.column* = *table2.column*

- **Combining Tables with SQL**

FROM *table1* AS *alias1* INNER JOIN *table2* AS *alias2*

1.Open **p107a04.sas** from the **activities** folder and perform the following tasks:

1. Define aliases for **storm_summary** and **storm_basincodes** in the FROM clause.

2. Use one table alias to qualify **Basin** in the SELECT clause.

3. Complete the ON expression to match rows when **Basin** is equal in the two tables. Use the table aliases to qualify **Basin** in the expression. Run the code.

```
      proc sql;
select Season, Name, s.Basin, BasinName, MaxWindMPH
    from pg1.storm_summary as s
        inner join pg1.storm_basincodes as b
        on s.basin=b.basin
    order by Season desc, Name;
quit;
```

The lowercase Basin values are not in the results because they are not matching values. To fix this, you can use the UPCASE function in the ON expression to change the values to uppercase.

```
proc sql;
select Season, Name, s.Basin, BasinName, MaxWindMPH
    from pg1.storm_summary as s
        inner join pg1.storm_basincodes as b
        on upcase(s.basin)=b.basin
    order by Season desc, Name;
quit;
```