SAS-kursus

Indhold

SAS-kursus 1

*** Lesson 1: Essentials 4

Summery 4

Understanding SAS Syntax 5

SAS - Program Structure 6

DATA Step 7

Syntax 7

Example 7

Here the string variables have a $ at the end and numeric values are without it 7

PROC Step 7

Syntax 7

Example 7

The OUTPUT Step 8

Syntax 8

Example 8

A complete SAS Program**, eksempel** 8

Program Output 9

SAS - Basic Syntax 10

SAS Statements 10

SAS Variable Names 11

Variables in SAS represent a column in the SAS data set. 11

Example 11

*** Lesson 2: Access data 11

Summery: Understanding SAS Data 11

The CONTENTS PROCEDURE creates a report on the DESCRIPTOR PORTION of the DATA SET, meta-info 12

Accessing Data Through Libraries, fx Excel-data 13

A SAS LIBRARY holds only SAS DATA SETS 13

LIBNAME PG1 BASE 'file-path to downloaded PG1 data'; 13

LIBNAME PG1_EXL XLSX 'file-path to downloaded PG1 data.xlsx'; 14

PROC IMPORT DATAFILE='file-path/storm_damage.csv' DBMS=csv OUT=work.storm_damage_import REPLACE; ... 15
Demofilerne /home/u77510/sasuser.v94/EPG1V2/demos/ ... 16
-kan vist afvikles, fx ep101d01.sas, med nogle fejl ... 16
ODS: The SAS Output Delivery System ... 16
Filerne udskrives til Work temporary mappen, default mappen ... 16
ep101d01.sas er eksempel med access, explore,prepare, analyse data, report ... 16
*laver table med sql ... 18
*Arbejde med excelfiler ... 18
*** Lesson 3: Exploring and Validating Data ... 23
Søge string case-insensitiv med upcase(var) ... 23
Makro-Variable i procedure, med %let Var=.. ... 24
-Var kan vist så bruges overalt i filen ... 24
formatted output ... 25
Sortering ... 28
remove duplicate rows ... 29
Summary of Lesson 3: Exploring and Validating Data ... 31
Exploring Data ... 31
Filtering Rows, med where ... 32
WHERE Operators ... 33
SAS Date Constant ... 33
IN Operator ... 33
Special WHERE Operators ... 33
Filtering Rows with Macro Variables ... 33
Example WHERE Statements with Macro Variables: ... 33
Formatting Columns ... 34
Sorting Data and Removing Duplicates ... 34
*** Lesson 4: Prepare data: datamanipulation ... 36
Filtrering i outputdata med where, medtager kun udvalgte kolonner med keep/drop ... 38
To create storm_cat5 as a permanent table ... 39
Lave nye kolonner ud fra expression ... 42
String-functions ... 43
Common Date Functions for Creating Columns ... 44
Task: indlæs datafil, omform den ... 45
Solution: ... 46

Solution: 46
Solution: 47
If-then-else 47
data tab1 tab2 *der laves to output-tabeller 49
Select-when 49
Understand how to control how data is read and written, merge multiple tables 50
*** Lesson 5: Analyzing and Reporting on Data 51
Der laves frequency-analyse på talværdier i Type-feltet, grupperet efter origin: 52
Label 53
2-way frequency table, pivot-tabel, med felt1*felt2 55
2way fordi man opgører bade for rækkerne, vandret og for hver kolonne, lodret 55
Noprint så udskrives ikke til result-fanen, ... 58

```
proc freq data=pg1.storm_final noprint;
    tables BasinName*StartDate / out=stormcounts;
    format StartDate monname.;
    label BasinName="Basin"
          StartDate="Storm Month";
run;
```

..... 58
her laves til gengæld work.output table, kun midlertidig tabel 58
Summery med output 62
MAP MED DATA 64
*cats(name,"-",maxwindmph,"mph"); string-sammensætning 65
* Creating a Bar Chart with PROC SGPLOT *; 66
* Creating a Line PLOT with PROC SGPLOT *; 67
Practice: Producing a Descriptive Statistic Report 69
Learning More 72
** Summary of Lesson 5: Analyzing and Reporting on Data 72
Enhancing Reports with Titles, Footnotes, and Labels 72
Creating Frequency Reports 73
TABLES statement options: 73
Creating Summary Statistics Reports 74
quiz 75
*** Lesson 6: Export results 77
Export a SAS DATA SET to a variety of file formats outside SAS. 78
export SAS tables and results to Excel, Microsoft Word, and PDF files. 78

Se excel, text fil: right-click storm_final.csv, and select View File as Text. 78
Eksempel på output fil og sheets 80
ODS=output delivery system kan lave csv filer mv 81
proc template list styles; run; viser style-templates i systemet 83
Styling, sheet-name 83
Udskrivning til pp og rtf 85
for pdf: 85
Learning More 88
** Summary of Lesson 6: Exporting Results 89
Exporting Data 89
Exporting Reports 89
Quiz: Lesson 6 91
*** Lesson 7: Using SQL 96
Using Structured Query Language (SQL) in SAS summery 98
Joining Tables Using SQL in SAS 99
Eksempel med title, inner join 100
Quiz: Lesson 7 100

*** Lesson 1: Essentials

Summery

Running a SAS Program

- Programs can be submitted by clicking the **Run** icon or pressing the F3 key.
- To run a subset of a program, highlight the desired statements first. If you are using SAS Studio, click the **Run** icon or press F3. If you are using SAS Enterprise Guide, click the arrow next to **Run** and click **Run Selection** or press F3.
- A program can create a log, results, and output data.
- Submitting a program that has run previously in Enterprise Guide or SAS Studio replaces the log, output data, and results.
- Submitting a program that has run previously in the SAS windowing environment appends the log and results.

Understanding SAS Syntax

- SAS programs consist of DATA and PROC steps, and each step consists of statements.

```
DATA ... ;
      other statements
RUN;
```

-

```
PROC ... ;
      other statements
RUN;
```

- Global statements are outside steps.

```
TITLE ... ;
```

-

```
OPTIONS ... ;
```

-

```
LIBNAME ... ;
```

-
- All statements end with a semicolon.
- Spacing doesn't matter in a SAS program.
- Unquoted values can be lowercase, upper case, or mixed case.
- Consistent program spacing is a good practice to make programs legible.
- Use the following automatic spacing features:

SAS Studio: Click the **Format Code** icon.

Enterprise Guide: Select **Edit** > **Format Code** or press Ctrl+Shift+B.

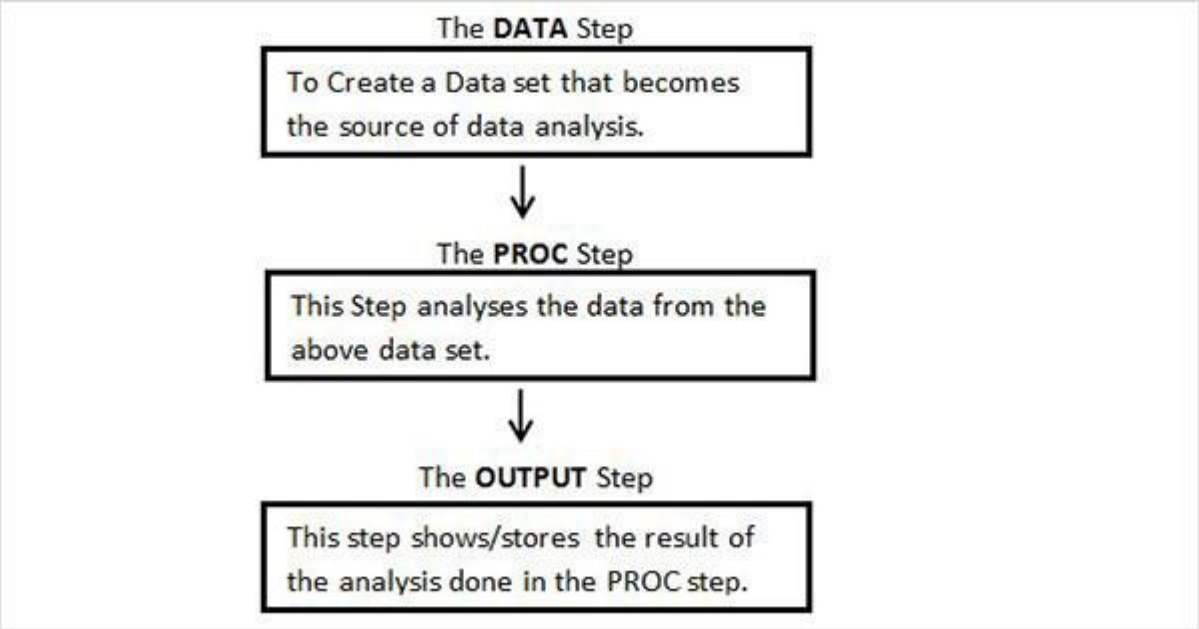
- Comments can be added to prevent text in the program from executing.
- Some common syntax errors are unmatched quotes, missing semicolons, misspelled keywords, and invalid options.
- Syntax errors might result in a warning or error in the log.
- Refer to the log to help diagnose and resolve syntax errors.

SAS- Program Structure

The SAS Programming involves first creating/reading the data sets into the memory and then doing the analysis on this data. We need to understand the flow in which a program is written to achieve this.

SAS Program Structure

The below diagram shows the steps to be written in the given sequence to create a SAS Program.



Every SAS program must have all these steps to complete reading the input data, analysing 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, rows), ligesom R-DataFrame. The syntax for DATA statement is as below.

Syntax

```
DATA data_set_name;          #Name the data set, output
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

PROC Step

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

Syntax

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

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;
```

The OUTPUT Step

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

Syntax

PROC PRINT DATA = data_set;

OPTIONS;

RUN;

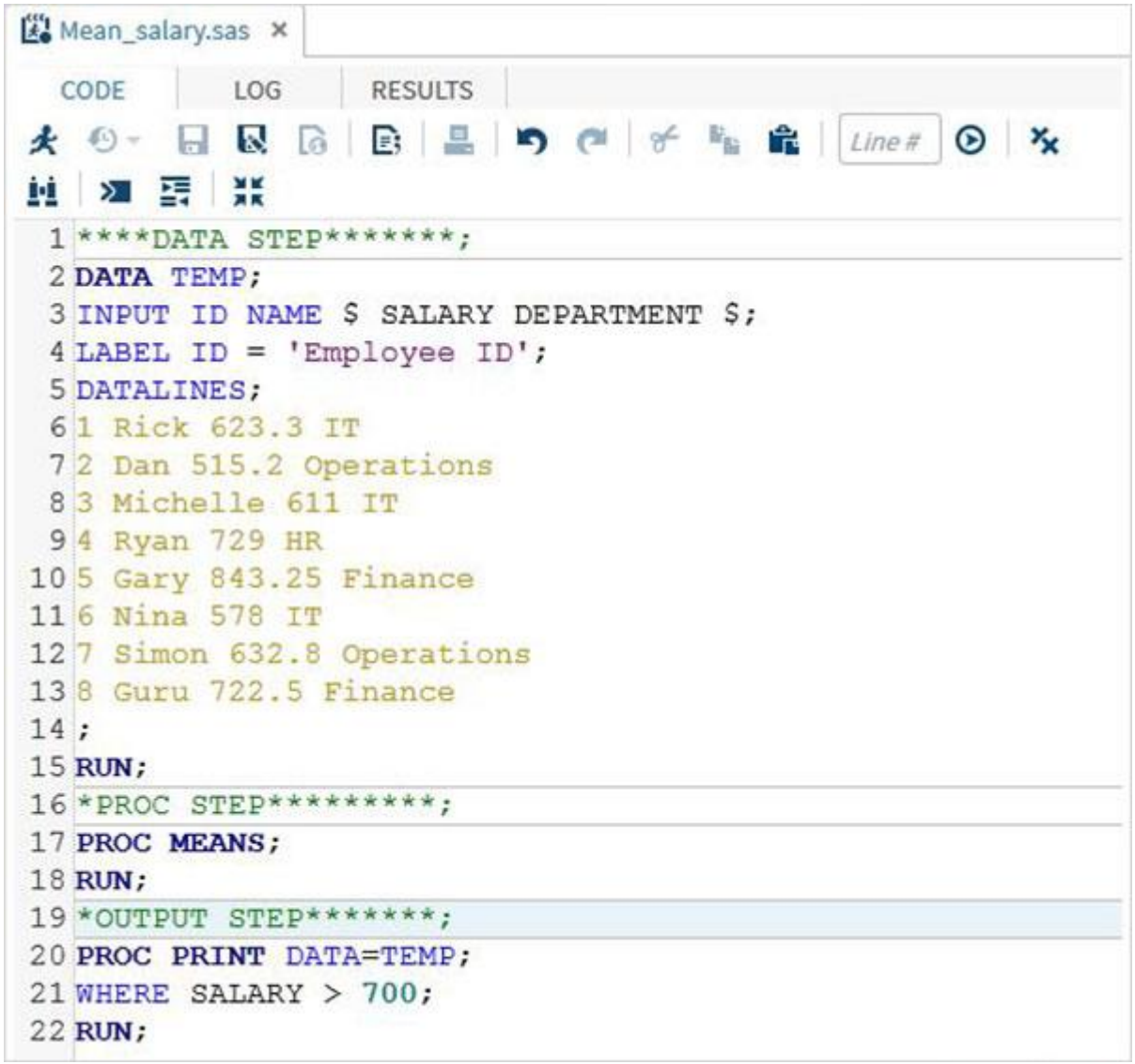
Example

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;
```

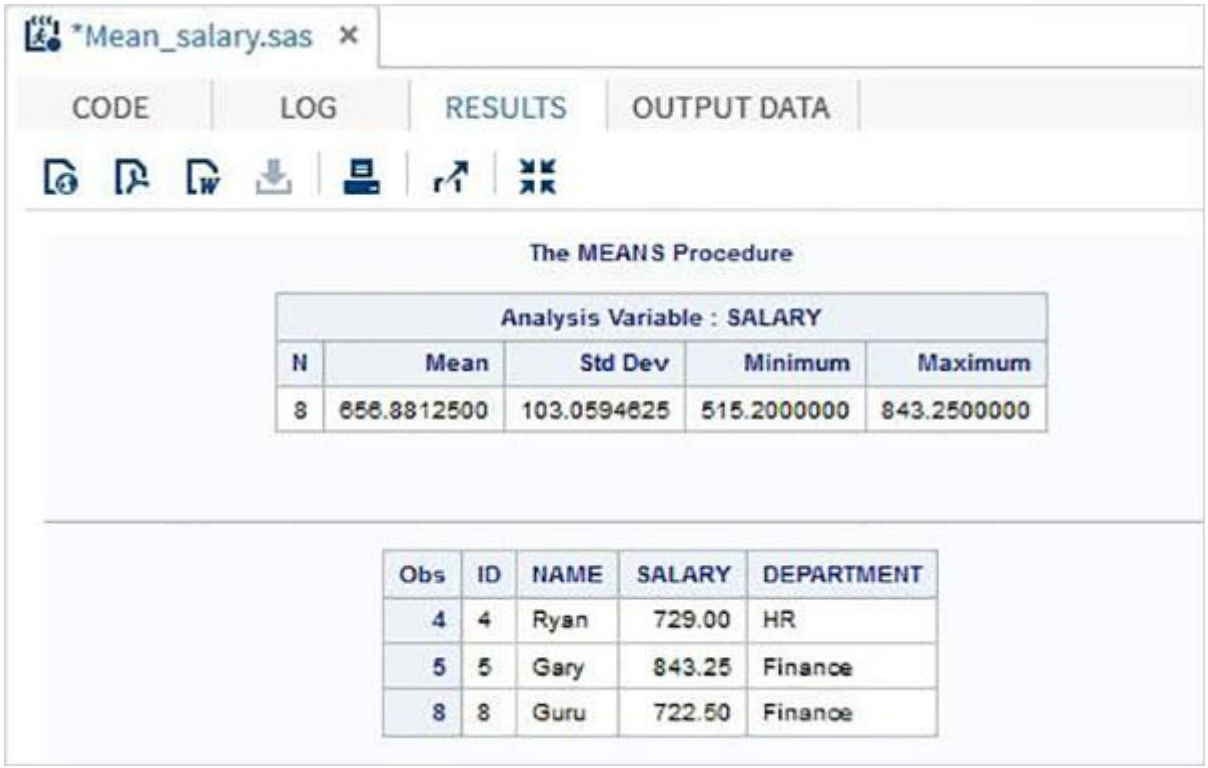
A complete SAS Program, eksempel

Below is the complete code for each of the above steps.



Program Output

The output from above code is seen in the **RESULTS** tab.



SAS - Basic Syntax

Like any other programming language, the SAS language has its own rules of syntax to create the SAS programs.

The three components of any SAS program - Statements, Variables and Data sets follow the below rules on Syntax.

SAS Statements

- Statements can start anywhere and end anywhere. 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.

- It can be 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.

Example

```
# Valid Variable Names
REVENUE_YEAR
MaxVal
_Length

# Invalid variable Names
Miles Per Liter  #contains Space.
RainfFall%      # contains apecial character other than
underscore.
90_high          # Starts with a number.
```

*** Lesson 2: Access data

Summery: Understanding SAS Data

Structured data is stored within tables, tables located within SAS are called SAS DATA SETS and will have the file extension .sas7bdat. These DATA SETS (i.e.

tables) are made up of VARIABLES (i.e. columns) and OBSERVATIONS (i.e. rows). SAS/ACCESS engines can be used to read structured data into SAS. Unstructured data i.e. data stored within raw data files such as .csv, .dat and .txt first needs to be placed into a SAS DATA SET by importing the data before SAS is able to use it. DATA SETS are made up of two components: a DESCRIPTOR PORTION and a DATA PORTION.

- The DESCRIPTOR PORTION of the DATA SET describes the table including the number of VARIABLES and OBSERVATIONS, the date and time the table was created and the VARIABLE attributes such as NAME, TYPE and LENGTH.

The CONTENTS PROCEDURE creates a report on the DESCRIPTOR PORTION of the DATA SET, meta-info

```
PROC CONTENTS data=pg1.storm_summary;

 run;
```

The _ALL_ keyword can be used to display the DESCRIPTOR PORTION for all the DATA SETS in the LIBRARY. It is recommended to use the NODS option alongside the _ALL_ option as reporting on all the table and VARIABLE information in a LIBRARY may take SAS a considerable time.

```
PROC CONTENTS data=pg1._ALL_; * pg1 en mappe, her fås info om filerne

 run;
```

The NODS options can be added to the PROC STATEMENT to only show a list of DATA SETS in the LIBRARY i.e. it will not display the DESCRIPTOR PORTION.

```
PROC CONTENTS data=pg1._ALL_ NODS;

 run;
```

- The DATA PORTION of the DATA SET includes all the OBSERVATIONS. The PRINT PROCEDURE creates a report on the DATA PORTION of the DATA SET. The report contains a print out of the DATA SET i.e. table containing all the OBSERVATIONS and VARIABLES.

```
PROC PRINT data=pg1.storm_summary;
```

```
run;
```

A VARIABLE must be defined in SAS to form part of a SAS table, required attributes include the VARIABLE's NAME, TYPE and LENGTH. Other optional attributes can also be saved such as FORMATS and LABELS. Note: SAS DATES are stored as numeric values so that calculations using dates can be computed. SAS converts the date to the number of days since 1st January 1960 (01JAN1960 = 0) to calculate the SAS DATE value.

Accessing Data Through Libraries, fx Excel-data

A SAS LIBRARY holds only SAS DATA SETS

A SAS LIBRARY holds only SAS DATA SETS this makes the data available to use within SAS. A LIBRARY is an organisational tool that provides shortcut access to data stored on PHYSICAL FILE.
Note: Data not stored as SAS DATA SETS will need to be converted before it can be placed into a SAS LIBRARY (SAS/ACCESS ENGINES can be used to perform this task)
.

There are two types of LIBRARY; TEMPORARY and PERMANENT

The WORK LIBRARY is a TEMPORARY LIBRARY, meaning when you close your SAS SESSION all of its contents are deleted i.e. the WORK LIBRARY acts as a SCRATCH SPACE. The WORK LIBRARY is also the default LIBRARY therefore, if you do not specify a LIBRARY your DATA SET will automatically placed inside the WORK LIBRARY. PERMANENT LIBRARIES such as SASHELP (contains sample DATA SETS) are connections to a PHYSICAL FILE location such as a folder on a drive. You can reference the data in your SAS PROGRAMS by referring to the LIBRARY LIBREF e.g. pg1.class_birthdate; A LIBNAME STATEMENT can be used to create a user-defined PERMANENT LIBRARY. It is a GLOBAL STATEMENT. For example:

LIBNAME PG1 BASE 'file-path to downloaded PG1 data';

```
Libref=PGI, navnet bruges i koden, BASE er typen af filer, der findes fx
også XLSX type beregnet til excel-filer
```

The LIBNAME STATEMENT contains 4 components: The LIBNAME STATEMENT will need to be run before a SAS PROGRAM referencing it is

processed – the LIBRARY will then remain only for the duration of your SAS SESSION. When you close your SAS SESSION the link between SAS and PHYSICAL FILE location will be lost meaning you will need to re-run the LIBNAME STATEMENT when you open up your SAS SESSION. To manually disconnect the library, use the LIBNAME LIBREF CLEAR STATEMENT. For example:

```
LIBNAME PG1 CLEAR;
```

Different SAS/ACCESS ENGINES can be used to read data stored in different formats into SAS, for example the XLSX ENGINE can be used to read in excel files. The LIBNAME STATEMENT is used to create the user-defined PERMANENT LIBRARY containing each excel worksheet as a separate DATA SET. For example:

```
LIBNAME PG1_EXL XLSX 'file-path to downloaded PG1 data.xlsx';
```

The file path must include the excel file name and file extension. It is best practice to dissociate the excel file from your SAS SESSION once you have finished using the file as the workbook will be locked until LIBRARY has been cleared.

```
LIBNAME PG1_EXL CLEAR;
```

Importing Data into SAS

The IMPORT PROCEDURE converts an external data file into a SAS DATA SET. For example, the PROC IMPORT below is used to read in a .CSV file to produce a new DATA SET in SAS called ~~class_birthdate~~, vel `storm_damage_import, der gemmes I temp- mappen WORK` . The REPLACE option ensures when you re-run the PROCEDURE the new DATA SET pg1.class_birthdate in overwritten with the updates.
Note: Following DATAFILE=, it does not matter which order the options are placed in the PROC STATEMENT.

```
PROC IMPORT DATAFILE='file-path/storm_damage.csv’ DBMS=csv
OUT=work.storm_damage_import REPLACE;
 run;
```

Other common DBMS options include: xls, xlsx, tab and dlm. If the DBMS=dlm option has been specified you will also need to include the DELIMITER STATEMENT in your PROC STEP stating the DELIMITER in quotation marks. For example:

```
PROC IMPORT DATAFILE='file-path/storm_damage.dat’ DBMS=dlm
OUT=work.storm_damage_import REPLACE;

 DELIMITER=‘*’

run;
```

When using PROC IMPORT to a create a DATA SET SAS scans the first 20 rows to determine the variable attributes. The GUESSINGROWS= STATEMENT can be added to the STEP to specify how many rows of the file SAS should be scanned to determine the data TYPE and LENGTH for the VARIABLES. The MAX option can be used instead of 2147483647 rows.

```
GUESSINGROWS=MAX;
```

Demofilerne /home/u77510/sasuser.v94/EPG1V2/demos/

-kan vist afvikles, fx ep101d01.sas, med nogle fejl

ODS: The SAS Output Delivery System

Filerne udskrives til Work temporary mappen, default mappen

ep101d01.sas er eksempel med access, explore,prepare, analyse data, report

```
************************************************;
*   SAS Programming Process                    *;
************************************************;
*   This program is an example of code that you  *;
*   learn in the class to analyze international  *;
*   storm data. The program follows the SAS      *;
*   programming process:                         *;
*    1) Access data                          *;
*    2) Explore data                         *;
*    3) Prepare data                         *;
*    4) Analyze and report on data           *;
*    5) Export results                       *;
************************************************;

***************;
* Section 1:  *;
* Access Data *;
***************;

options validvarname=v7;
ods graphics on;  *for at tegne histogram, eller map ?  senere

*Path is assigned in the cre8data.sas program;
*%let path=s:/workshop;

libname pg1 base "&path";

/*læser fra excelark, output til storm_damage, med replace option,
 Kan ikke finde path/storm.xlsx   */
proc import datafile="&path/storm.xlsx"
                    dbms=xlsx out=storm_damage replace;
                    sheet="Storm_Damage";
```

```
run;

****************;
* Section 2:   *;
* Explore Data *;
****************;

title "Explore Basin and Status Codes";
proc freq data=pg1.storm_summary;  * i /pg1
                tables basin type;  *der laves to tabeller med gruppering over basin og type kolonne værdier,
se tabel nedenfor
run;

title "Summary Statistics for Maximum Wind(MPH) and Minimum Pressure";
proc means data=pg1.storm_summary;
                var MaxWindMPH MinPressure;
run;

title "First 5 Rows from Imported Storm Damage";
proc print data=storm_damage(obs=5);  * i /Work =default mappen
run;

********************;
* Section 3:     *;
* Prepare Data    *;
********************;
/*der laves ny outputfil=storm_summary2, med de to inputfiler i set*/
data storm_summary2;
set   pg1.storm_summary   *ref til inputdatasæt
      pg1.storm_2017(drop=location rename=(Year=Season));
                length OceanCode $ 7 BasinName $ 14;
                drop oceancode;
                Basin=upcase(basin);
                OceanCode=substr(basin,2,1);
                key=cats(season,name);
                StormLength=enddate-startdate;

                if oceancode="A" then Ocean="Atlanic";
                else if oceancode="P" then Ocean="Pacific";
                else if oceancode="I" then Ocean="Indian";

                if Basin="NA" then BasinName="North Atlantic";
                else if Basin="SA" then BasinName="South Atlantic";
                else if Basin="WP" then BasinName="West Pacific";
                else if Basin="EP" then BasinName="East Pacific";
                else if Basin="SP" then BasinName="South Pacific";
                else if Basin="NI" then BasinName="North Indian";
                else if Basin="SI" then BasinName="South Indian";
run;
```

```
data storm_damage2;
                set storm_damage;
                Name=upcase(scan(Event,-1));
                Season=Year(date);
                key=cats(season,name);
                drop Event Date;
                format Cost dollar16.;
run;
```

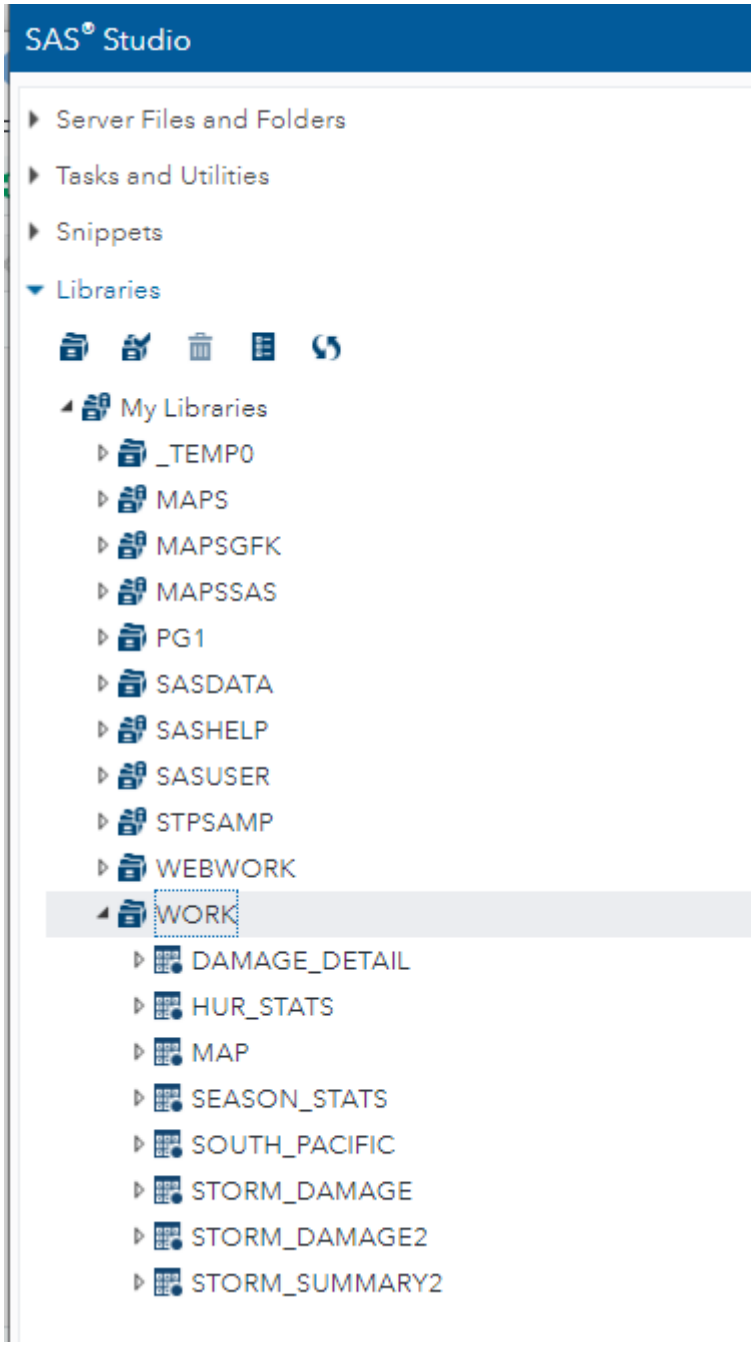
*laver table med sql

```
proc sql;
create table damage_detail as
select d.name, d.season, basinname, maxwindmph, minpressure, stormlength, cost
   from storm_damage2 as D, storm_summary2 as S
   where d.key=s.key order by cost desc;
quit;

*******************************;
* Section 4:              *;
* Analyze and Report on Data  *;
* Export Results          *;
*******************************;
```

*Arbejde med excelfiler

```
%let Year=2016;
%let basin=North Atlantic;
%let outpath=%sysfunc(tranwrd(&path,data,output));
ods noproctitle;
*goptions dev=png;
ods excel file="&outpath/storm_report&year..xlsx"
        options(sheet_interval="proc"
        sheet_name="&Year Storms by Basin"
        embedded_titles="yes");

title1 "Number of Storms by Type and Basin";
title2 "&year Season";
proc freq data=storm_summary2 order=freq;  *findes i /Work sorters efter freq-værdier, der alves tabel og
plot:
   tables basinname / nopercent nocum plots=freqplot;
                tables basinname*type / norow nocol crosslist ;
                where season=&year;
run;

ods excel options(sheet_name="&year Wind Statistics");
title1 "Wind Statistics by Storm";
title2 "Year &year";
proc means data=pg1.storm_detail mean min max maxdec=0 nonobs;
                class name;
                var wind;
```

```
                where season=&year;
                output out=hur_stats mean=AvgWind min=MinWind max=MaxWind;
run;

data map;
                set storm_summary2;
                length maplabel $ 20;
                where season=&year and basinname="&basin";
                if maxwindmph<100 then MapLabel=" ";
                else maplabel=cats(name,"-",maxwindmph,"mph");
                keep lat lon maplabel maxwindmph;
run;

title1 "Tropical Storms in &year Season";
title2 "&basin Basin";
footnote1 "Storms with MaxWind>100mph are labeled";

ods excel options(sheet_name="&year &Basin Basin");
proc sgmap plotdata=map;
   *openstreetmap;
   esrimap url='https://services.arcgisonline.com/arcgis/rest/services/World_Physical_Map';
   bubble x=lon y=lat size=maxwindmph /
                                   datalabel=maplabel datalabelattrs=(color=red size=8);
run;
ods excel close;
ods proctitle;
title;footnote;
```


SAS® Studio
Server Files and Folders
Tasks and Utilities
Snippets
Libraries
My Libraries
_TEMP0
MAPS
MAPSGFK
MAPSSAS
PG1
SASDATA
SASHELP
SASUSER
STPSAMP
WEBWORK
WORK
DAMAGE_DETAIL
HUR_STATS
MAP
SEASON_STATS
SOUTH_PACIFIC
STORM_DAMAGE
STORM_DAMAGE2
STORM_SUMMARY2

Explore Basin and Status Codes

The FREQ Procedure

Basin	Frequency	Percent	Cumulative Frequency	Cumulative Percent
EP	671	21.52	671	21.52
NA	472	15.14	1143	36.66
NI	84	2.69	1227	39.35
SI	588	18.86	1815	58.21
SP	359	11.51	2174	69.72
WP	928	29.76	3102	99.49
na	16	0.51	3118	100.00

Type	Frequency	Percent	Cumulative Frequency	Cumulative Percent
DS	293	9.40	293	9.40
ET	761	24.41	1054	33.80
NR	702	22.51	1756	56.32
SS	5	0.16	1761	56.48
TS	1357	43.52	3118	100.00

Summary Statistics for Maximum Wind(MPH) and Minimum Pressure

The MEANS Procedure

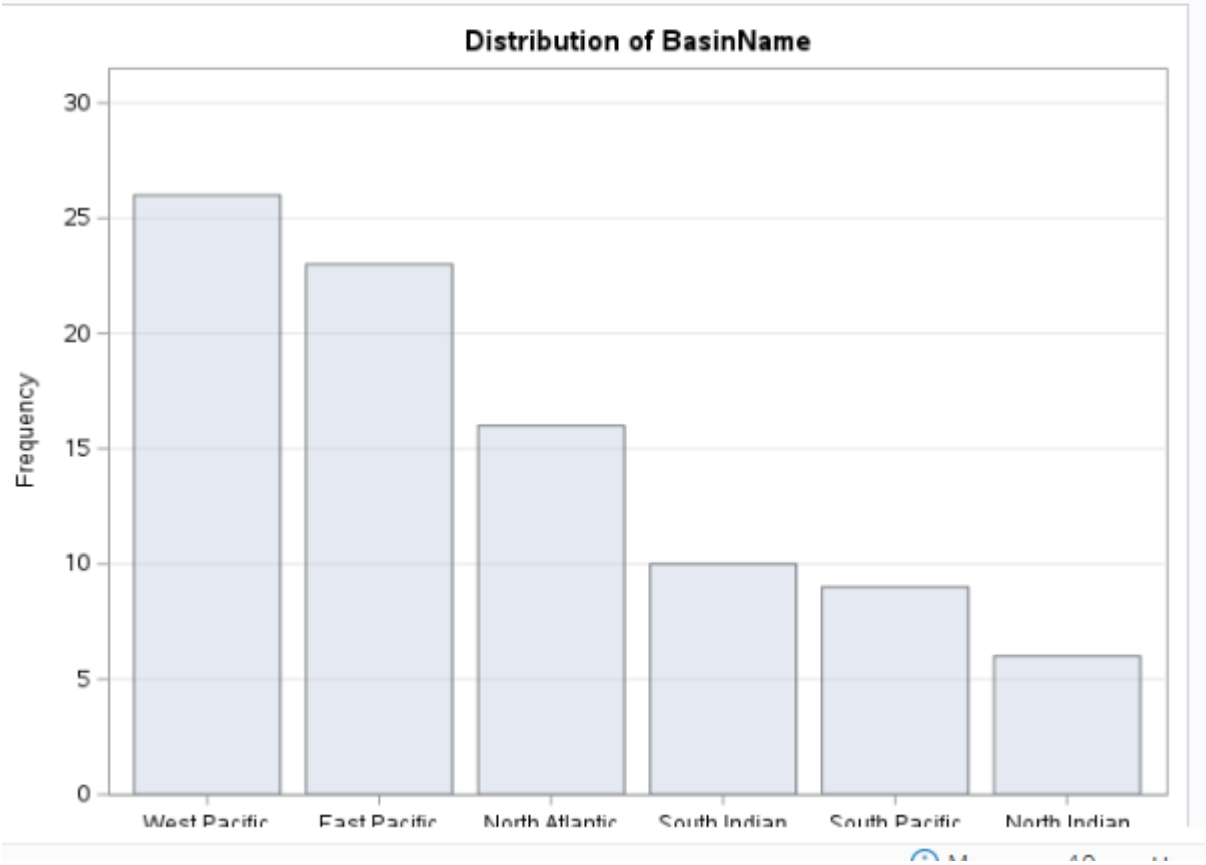
Variable	N	Mean	Std Dev	Minimum	Maximum
MaxWindMPH	3095	79.3179321	31.6853937	6.0000000	213.0000000
MinPressure	2922	961.8545517	288.6582966	-9999.00	1012.00

First 5 Rows from Imported Storm Damage

Obs	Event	Date	Summary	Cost
1	Hurricane Katrina	25AUG2005	Category 3 hurricane initially impacts the U.S. as a Category 1 near Miami, FL, then as a strong Category 3 along the eastern LA-western MS coastlines, resulting in severe storm surge damage (maximum surge probably exceeded 30 feet) along the LA-MS-AL coasts, wind damage, and the failure of parts of the levee system in New Orleans. Inland effects included high winds and some flooding in the states of AL, MS, FL, TN, KY, IN, OH, and GA.	1,613E11 kr.
2	Hurricane Harvey	25AUG2017	Category 4 hurricane made landfall near Rockport, Texas causing widespread damage. Harvey's devastation was most pronounced due to the large region of extreme rainfall producing historic flooding across Houston and surrounding areas. More than 30 inches of rainfall fell on 6.9 million people, while 1.25 million experienced over 45 inches and 11,000 had over 50 inches, based on 7-day rainfall totals ending August 31. This historic U.S. rainfall caused massive flooding that displaced over 30,000 people and damaged or destroyed over 200,000 homes and businesses.	1,25E11 kr.
3	Hurricane Maria	19SEP2017	Category 4 hurricane made landfall in southeast Puerto Rico after striking the U.S. Virgin Island of St. Croix. Maria's high winds caused widespread devastation to Puerto Rico's transportation, agriculture, communication and energy infrastructure. Extreme rainfall up to 37 inches caused widespread flooding and mudslides across the island. The interruption to commerce and standard living conditions will be sustained for a long period, as much of Puerto Rico's infrastructure is rebuilt. Maria tied Hurricane Wilma (2005) for the most rapid intensification, strengthening from tropical depression to a category 5 storm in 54 hours. Maria's	90000000000 kr.

Number of Storms by Type and Basin
2016 Season

BasinName	Frequency
West Pacific	26
East Pacific	23
North Atlantic	16
South Indian	10
South Pacific	9
North Indian	6

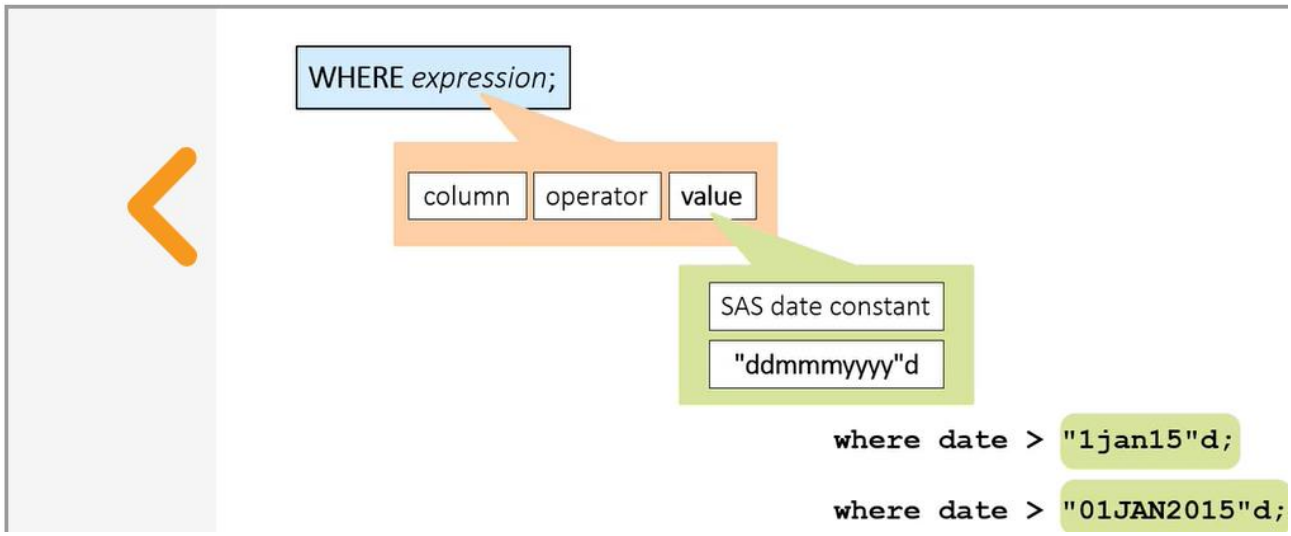


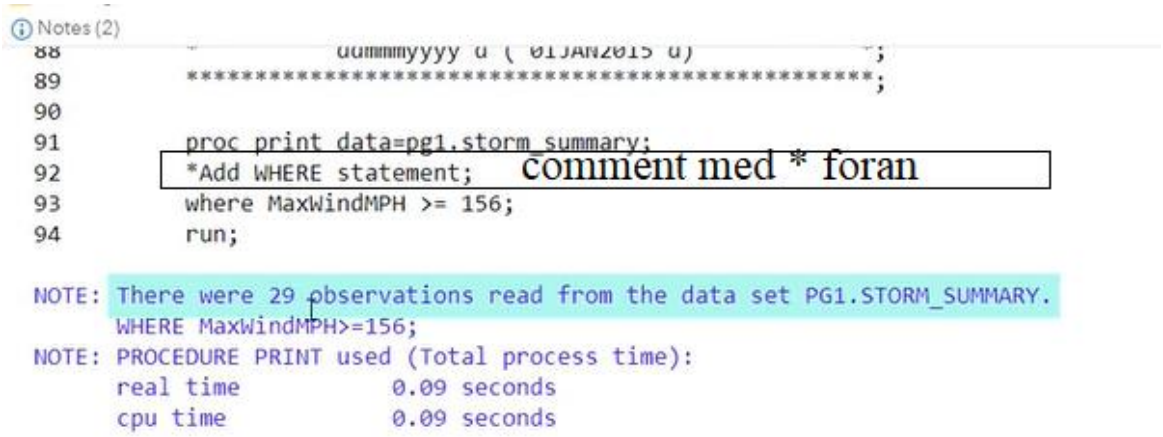
..

Map til sidst

*** Lesson 3: Exploring and Validating Data

Lesson 3: Exploring and Validating Data





Søge string case-insensitiv med upcase(var)

```
proc print data=pg1.np_traffic;
    var ParkName Location Count;
    where Count ne 0 and upcase(Location) like '%MAIN
ENTRANCE%';
run;
```

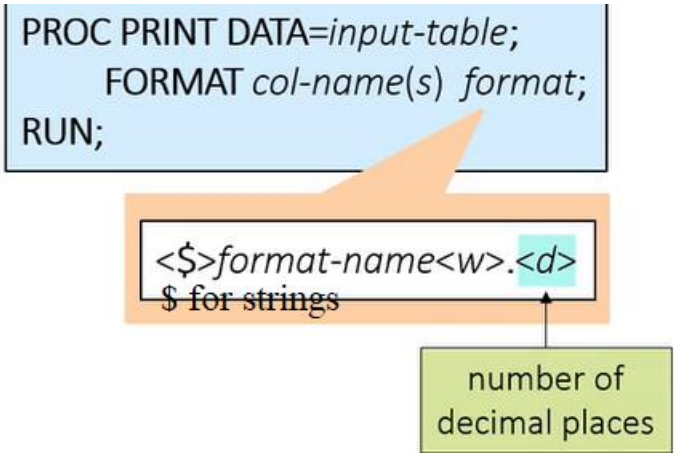
Makro-Variable i procedure, med %let Var=..

-Var kan vist så bruges overalt i filen


```
%let Year=2016;
%let basin=North Atlantic;  *case insensitiv
ods noproctitle;
goptions dev=png;
ods excel file="&path/output/storm_report&year..xlsx"
          options(sheet_interval="proc"
          sheet_name="&Year Storms by Basin"
          embedded_titles="yes");

title1 "Number of Storms by Type and Basin";
title2 "&year Season";
proc freq data=storm_summary2 order=freq;
    tables basinname / nopercent nocum plots=freqplot;
      tables basinname*type / norow nocol crosslist ;
      where season=&year;  * var name caseinsentiv
run;

ods excel options(sheet_name="&year Wind Statistics");
title1 "Wind Statistics by Storm";
title2 "Year &year";
proc means data=pg1.storm_detail mean min max maxdec=0
nonobs;
      class name;
      var wind;
      where season=&year;
      output out=hur_stats mean=AvgWind min=MinWind
max=MaxWind;
run;

data map;
```

```
      set storm_summary2;
      length maplabel $ 20;
      where season=&year and basinname="&basin";
      if maxwindmph<100 then MapLabel=" ";
      else maplabel=cats(name,"-",maxwindmph,"mph");
      keep lat lon maplabel maxwindmph;
run;
```

formatted output




```
proc print data=pg1.class birthdate;
    format Height Weight 3. Birthdate date9.;
run;
```

9 chars i date

3 decimaler i tal

Name	Sex	Age	Height	Weight	Birthdate
Alfred	M	14	69	112.5	16370
Alice	F	13	56.5	84	16756
Barbara	F	13	65.3	98	16451
Carol	F	14	62.8	102.5	16256

Name	Sex	Age	Height	Weight	Birthdate
Alfred	M	14	69	113	26OCT2004
Alice	F	13	57	84	16NOV2005
Barbara	F	13	65	98	15JAN2005
Carol	F	14	63	103	04JUL2004

Common Formats for Numeric Values

Format Name	Example Value	Format Applied	Formatted value
w.d	12345.67	5.	12346
w.d	12345.67	8.1	12345.7
COMMA*w.d*	12345.67	COMMA8.1	12,345.7
DOLLAR*w.d*	12345.67	DOLLAR10.2	$12,345.67
DOLLAR*w.d*	12345.67	DOLLAR10.	$12,346
YEN*w.d*	12345.67	YEN7.	¥12,346
EUROX*w.d*	12345.67	EUROX10.2	€12.345,67

Zw.d format indføre foranstillede 0'er

Look up the **Zw.d** format.

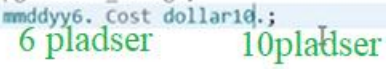
The format displays standard numeric data with leading 0s.

Format Name	Example Value	Format Applied	Formatted value
Zw.d	1350	Z8.	00001350

You can find this information by typing **Zw.d** in the search box and selecting the first link to **Zw.d Format**.

Value	Format applied	Formatted value
21199	DATE7.	15JAN18
21199	DATE9.	15JAN2018
21199	MMDDYY10.	01/15/2018
21199	DDMMYY8.	15/01/18
21199	MONYY7.	JAN2018
21199	MONNAME.	January
21199	WEEKDATE.	Monday, January 15, 2018

```
*    FORMAT col-name(s) format;                          *;
*                                                        *;
*    <$>format-name<w>.<d>                               *;
*                                                        *;
*    Common formats:                                     *;
*       dollar10.2 -> $12,345.67                         *;
*       dollar10.  -> $12,346                            *;
*       comma8.1   -> 9,876.5                            *;
*       date7.     -> 01JAN17                            *;
*       date9.     -> 01JAN2017                          *;
*       mmddyy10.  -> 12/31/2017                         *;
*       ddmmyy8.   -> 31/12/17                           *;
*********************************************************;

*Write a PROC PRINT step and FORMAT statement;
proc print data=pg1.storm_damage;
    format Date mmddyy6. Cost dollar10.;
run;
```



Highlight the code i data eller proc og run selection

Open **p103a05.sas** from the **activities** folder and perform the following tasks:
3 for lection 3

1. Highlight the P ROC 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 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 freq data=pg1.storm_summary order=freq;
    tables StartDate;
    format startdate monname.;
run;
```

The new report has 12 rows. Formats are an easy way to group data in procedures!

Sortering

```
proc sort data=pg1.class_test2 out=test_sort;
    by Subject descending TestScore;
run;
```

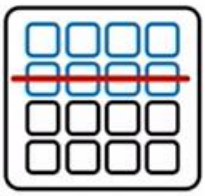
ascending order by Subject, then within Subject by descending TestScore

Name	Subject	TestScore
Judy	Math	97
Barbara	Math	96
Louise	Math	92
James	Math	90
Joyce	Math	88
William	Math	87
Henry	Math	85
Jane	Math	84

```
proc sort data=pg1.storm_summary out=storm_sort;
     where Basin in("NA" "na");
     by descending MaxWindMPH;
 run;
```

remove duplicate rows

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




```
1  ************************************************************;
2  *  Identifying and Removing Duplicate Values               *;
3  ************************************************************;
4
5  *Step 1;
6  proc sort data=pg1.storm_detail out=storm_clean nodupkey dupout=storm_dups;
7      by _all_;
8  run;
9
10 *Step 2;
11 proc sort data=pg1.storm_detail out=min_pressure;
12     where Pressure is not missing and Name is not missing;
13     by descending Season Basin Name Pressure;
14 run;
15
16 *Step 3;
17 proc sort data=min_pressure nodupkey;
18     by descending Season Basin Name;
19 run;
```

p103d05.sas

```
************************************************************;
*  Identifying and Removing Duplicate Values               *;
************************************************************;
*  Syntax and Example                                      *;
*                                                          *;
*    Remove duplicate rows:                                *;
*    PROC SORT DATA=input-table <OUT=output-table>         *;
*      NODUPKEY <DUPOUT=output-table>;                     *;
*      BY _ALL_;                                           *;
*    RUN;                                                  *;
*                                                          *;
*    Remove duplicate key values:                          *;
```

```
*   PROC SORT DATA=input-table <OUT=output-table>       *;
*      NODUPKEY <DUPOUT=output-table>;                  *;
*      BY <DESCENDING> col-name (s);                    *;
*   RUN;
*Step 1;
proc sort data=pg1.storm_detail out=storm_clean;
                by _all_;
run;

*Step 2;
proc sort data=pg1.storm_detail out=min_pressure;
                where Pressure is not missing and Name is not missing;
                by descending Season Basin Name Pressure;
run;

*Step 3;
proc sort data=min_pressure nodupkey;
                by  descending Season Basin Name Pressure;
run;
```

Total rows: 1462 Total columns: 13 Rows 1-100

	Season	Basin	Sub_basin	Name	ISO_time	Type	Latitude	Longitude	Wind	Pressure	Hem_NS	Hem_EW	I
1	2016	EP	MM	AGATHA	03JUL2016:06:00:00.00	TS	16.8	-122.1	45	1002	N	W	
2	2016	EP	MM	BLAS	06JUL2016:00:00:00.00	TS	14.3	-121.2	120	947	N	W	
3	2016	EP	MM	CELIA	11JUL2016:18:00:00.00	TS	15.1	-125.8	85	972	N	W	
4	2016	EP	MM	DARBY	16JUL2016:18:00:00.00	TS	17.9	-124.4	105	958	N	W	
5	2016	EP	MM	ESTELLE	17JUL2016:18:00:00.00	TS	16.5	-112.2	60	990	N	W	
6	2016	EP	MM	FRANK	27JUL2016:00:00:00.00	TS	21.6	-118.2	75	979	N	W	
7	2016	EP	MM	GEORGETTE	25JUL2016:06:00:00.00	TS	16.6	-126.4	115	952	N	W	
8	2016	EP	MM	HOWARD	02AUG2016:06:00:00.0(	TS	18	-127	50	998	N	W	
9	2016	EP	MM	IVETTE	05AUG2016:18:00:00.0(	TS	15.3	-131.2	50	1000	N	W	
10	2016	EP	MM	JAVIER	08AUG2016:18:00:00.0(	TS	22.2	-109.3	55	997	N	W	
11	2016	EP	MM	KAY	21AUG2016:18:00:00.0(	TS	21.3	-115	45	1000	N	W	
12	2016	EP	MM	LESTER	31AUG2016:06:00:00.0(	TS	17.7	-137	125	944	N	W	
13	2016	EP	MM	MADELINE	27AUG2016:18:00:00.0(	TS	15.5	-139	50	1000	N	W	
14	2016	EP	MM	NEWTON	06SEP2016:06:00:00.00	TS	22.1	-109.5	80	977	N	W	
15	2016	EP	MM	ONE	06JUN2016:18:00:00.00	TS	13.9	-97.1	30	1006	N	W	
16	2016	EP	MM	ORLENE	12SEP2016:18:00:00.00	TS	17.7	-119.2	95	967	N	W	
17	2016	EP	MM	OTTO	25NOV2016:06:00:00.0(	TS	10.6	-86.2	60	994	N	W	

1.
   - Write a PROC SORT step to sort **pg1.eu_occ** and create an output table named **countrylist**.
   - Remove duplicate key values.
   - Sort by **Geo** and then **Country**.
   - Submit the program and view the output data.

```
proc sort data=pg1.eu_occ out=countryList
          nodupkey;
    by Geo Country;
run;
```

The value of **Geo** is *AT* and the value of **Country** is *Austria*. Six columns are included.

2. To read only **Geo** and **Country** from the **pg1.eu_occ** table, you can use the KEEP= data set option. Add the KEEP= option immediately after the input table and list **Geo** and **Country**. Submit the program and verify that only one row per country is included. Det sparer data-transport

data-set(**KEEP=***varlist*)

```
proc sort data=pg1.eu_occ(keep=geo country) out=countryList
          nodupkey;
    by Geo Country;
run;
```

Summary of Lesson 3: Exploring and Validating Data

Exploring Data

- PROC PRINT lists all columns and rows in the input table by default. The OBS= data set option limits the number of rows listed. The VAR statement limits and orders the columns listed.

PROC PRINT DATA=*input-table*(OBS=*n*);
    VAR *col-name(s)*;
RUN;

-
- PROC MEANS generates simple summary statistics for each numeric column in the input data by default. The VAR statement limits the variables to analyze.

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

-
- PROC UNIVARIATE also generates summary statistics for each numeric column in the data by default, but includes more detailed statistics related to distribution and extreme values. The VAR statement limits the variables to analyze.

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

-
- PROC FREQ creates a frequency table for each variable in the input table by default. You can limit the variables analyzed by using the TABLES statement.

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

-

Filtering Rows, med where

- The WHERE statement is used to filter rows. If the expression is true, rows are read. If the expression is false, they are not.
- Character values are case sensitive and must be in quotation marks.
- Numeric values are not in quotation marks and must only include digits, decimal points, and negative signs.
- Compound conditions can be created with AND or OR.
- The logic of an operator can be reversed with the NOT keyword.
- When an expression includes a fixed date value, use the SAS date constant syntax: “ddmmmyyyy”d, where *dd* represents a 1- or 2-digit day, *mmm* represents a 3-letter month in any case, and *yyyy* represents a 2- or 4-digit year.

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

-

WHERE Operators

= or EQ
^= or ~= or NE
> or GT
< or LT
>= or GE
<= or LE

SAS Date Constant

"ddMONyyyy"d

IN Operator

WHERE *col-name* **IN(***value-1<...,value-n>***);**
WHERE *col-name* **NOT IN (***value-1<…,value-n>***);**

Special WHERE Operators

WHERE *col-name* **IS MISSING;**
WHERE *col-name* **IS NOT MISSING;**
WHERE *col-name* **IS NULL;**
WHERE *col-name* **BETWEEN** *value-1* **AND** *value-2***;**
WHERE *col-name* **LIKE "***value***";**
WHERE *col-name* **=* "***value***";**

Filtering Rows with Macro Variables

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

Example WHERE Statements with Macro Variables:

WHERE *numvar=&macrovar***;**
WHERE *charvar="&macrovar"***;**
WHERE *datevar="&macrovar"*d

- A macro variable stores a text string that can be substituted into a SAS program.
- The %LET statement defines the macro variable name and assigns a value.

- Macro variable names must follow SAS naming rules.
- Macro variables can be referenced in a program by preceding the macro variable name with an &.
- If a macro variable reference is used inside quotation marks, double quotation marks must be used.

Formatting Columns

- Formats are used to change the way values are displayed in data and reports.
- Formats do not change the underlying data values.
- Formats can be applied in a procedure using the FORMAT statement.
- Visit SAS Language Elements documentation to access a list of available SAS formats.

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

-

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

Sorting Data and Removing Duplicates

- PROC SORT sorts the rows in a table on one or more character or numeric columns.
- The OUT= option specifies an output table. Without this option, PROC SORT changes the order of rows in the input table.
- The BY statement specifies one or more columns in the input table whose values are used to sort the rows. By default, SAS sorts in ascending order.

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

-
- The NODUPKEY option keeps only the first row for each unique value of the column(s) listed in the BY statement.

- The NODUPKEY option together with the BY *_ALL_* statement removes adjacent rows that are entirely duplicated.

- The DUPOUT= option creates an output table containing duplicates removed.

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

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

Based on the following program and data, how many rows will be included in the **payment** table?

```
proc sort data=payment dupout=dups nodupkey;
    by ID;
run;
```

ID	Amount
A	$997.54
A	$833.88
B	$879.05
C	$894.77
C	$894.77
C	$998.26

a. 1
b. 3
c. 5
d. 6

Your answer: c
Correct answer: b

The NODUPKEY option keeps the first row for each unique value of **ID**, which includes A, B and C.

```
proc sort data=orion.staff;
          out=work.staff;
    by descending Salary Manager_ID;
run;
```

a. The sorted table overwrites the input table.

b. The rows are sorted by **Salary** in descending order, and then by **Manager_ID** in descending order.

c. A semicolon should not appear after the input data set name.

d. The sorted table contains only the columns specified in the BY statement.

Your answer: b
Correct answer: c

This PROC SORT step has a syntax error: a semicolon in the middle of the PROC SORT statement. If you correct this syntax error, this step sorts **orion.staff** by **Salary** in descending order and by **Manager_ID** in ascending order. The step then creates the temporary data set **staff** that contains the sorted rows and all columns.

*** Lesson 4: Prepare data: datamanipulation

you learn how to do some common data manipulations, such as filtering rows and columns, computing new columns, and performing conditional processing.

Learning Objectives:

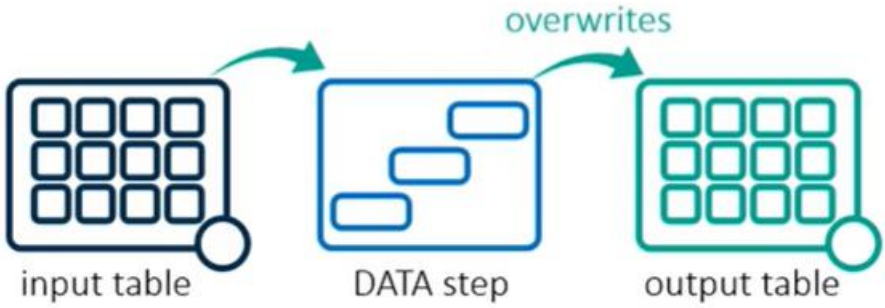
1. Create a new SAS table using the DATA STEP.
2. Create new variables using hardcoded values, calculations and SAS FUNCTIONS.
3. Perform a certain action(s) depending whether a condition is true.

It's rare that a data source is perfect as-is and needs no preparation. After you learn about your data, you'll probably want to make some adjustments based on what you find and what you need. This is where the DATA step, a powerful tool for manipulating data, really shines.

Most analysts agree that manipulating and validating data takes much more time than reporting or analysis. For this reason, you'll appreciate that the DATA step is a robust yet simple programming tool that can do everything from simple querying to providing structure to messy web logs.

In Preparing Data you learn how to do some common data manipulations, such as filtering rows and columns, computing new columns, and performing conditional processing.

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



Execution

1) Read a row from the input table.
2) Sequentially process statements.
3) At RUN, write the row to the output table.
4) Loop back to the top and read the next row from the input table.

```
data myclass;
     set sashelp.class;
     ...other statements...
run;
```

Which DATA step creates a permanent output data set?

- a.
```
data storm_new;
      set pg1.storm_summary;
run;
```
- b.
```
data out.storm_new;
      set pg1.storm_summary;
run;
```
- c.
```
data work.storm_new;
      set pg1.storm_summary;
run;
```

Incorrect.
The correct answer is **b**. If no library or the WORK library is specified, the table is temporary. When you specify a library that points to a permanent location, the table that is created there is permanent.

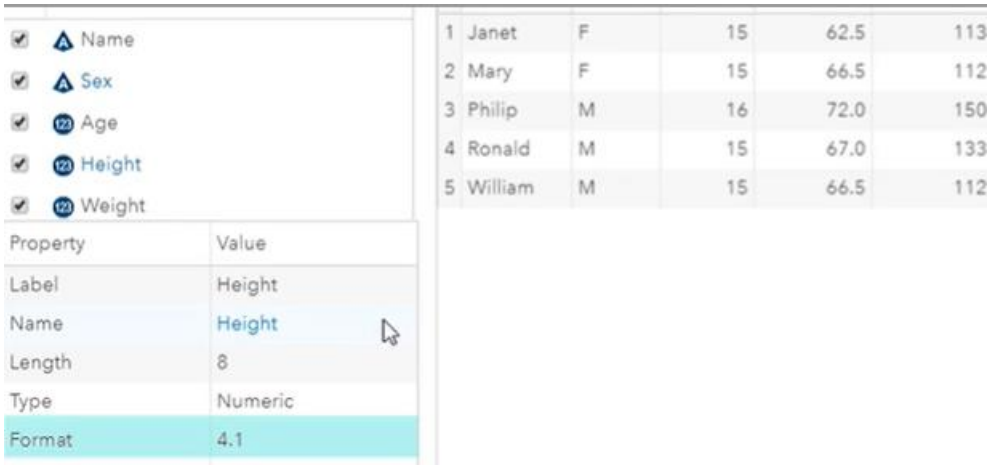
Which data sources can be used in the SET statement? Select all that apply.

- [x] *a.* SAS tables
- [x] *b.* Excel spreadsheets
- [ ] *c.* DBMS tables
- *d.* comma-delimited files

Incorrect.
The correct answer is **a, b** and **c**. Any data source that can be read via a library can be used as the input table in the SET statement.

Filtrering i outputdata med where, medtager kun udvalgte kolonner med keep/drop

```
6  data myclass;
7      set sashelp.class;
8      where Age >=15;
9      *keep Name Age Height;
10     drop Sex Weight;
11 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 **EPG1V2/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**.

4. How many Category 5 storms have there been since January 1, 2000?

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

There have been 18 Category 5 storms since January 1, 2000.

To create storm_cat5 as a permanent table

in the **EPG1V2/output** folder, you would simply submit a LIBNAME statement to define a library and then reference the library on the DATA statement as in:

```
libname out "path-to-EPG1V2/output";
data out.storm_cat5;
. . .
run;
```

1. 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 **EPG1V2/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.
    - Run the program.

```
* if you are creating a permanent table, you must submit
a LIBNAME statement for out-mappen and then reference
out.fox;
* libname out "path-to-EPG1V2/output";

Set=dataset angiver inputfilen, når data output-fil skal
laves
*laver ny datasæt fox, læse inputfilen i set,
pg1.np_species
data fox;
    set pg1.np_species;
    where Category='Mammal' and upcase(Common_Names) like
'%FOX%';
    drop Category Record_Status Occurrence Nativeness;
run;
```

2. 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;
```

3. Sort the **fox** table by **Common_Names**.

```
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;

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

4. What is the value of **Common_Names** in row one?

Arctic Fox

Lave nye kolonner ud fra expression

```
data cars_new;
    set sashelp.cars;
    where Origin ne "USA";
    Profit = MSRP-Invoice;
    Source = "Non-US Cars";
    format Profit dollar10.;
    keep Make Model MSRP Invoice Profit Source;
run;
```

Make	Model	MSRP	Invoice	Profi
Acura	MDX	$36,945	$33,337	$3,6
Acura	RSX Type S 2dr	$23,820	$21,761	$2,0
Acura	TSX 4dr	$26,990	$24,647	$2,3
Acura	TL 4dr	$33,195	$30,299	$2,8
Acura	3.5 RL 4dr	$43,755	$39,014	$4,7
Acura	3.5 RL w/Navi...	$46,100	$41,100	$5,0
Acura	NSX coupe 2d...	$89,765	$79,978	$9,7

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

2. Run the program. In 1980, how long did the storm named *Agatha* last?

```
data storm_length;
    set pg1.storm_summary;
    drop Hem_EW Hem_NS lat lon;
    StormLength = EndDate-StartDate+1;
run;
```

The storm named **Agatha** lasted 7 days in 1980.

1. Create a new column named **WindAvg** that is the mean of **wind1**, **wind2**, **wind3**, and **wind4**.

2. Create a new column **WindRange** that is the range of **wind1**, **wind2**, **wind3**, **wind4**.

3. Run the program and view the data. What are the **WindAvg** and **WindRange** values in row 1?

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

Note: you can use the shorthand notation **OF *col1 - coln*** to list a range of columns when the columns have the same name and end in a consecutive number.

The **WindAvg** is *92.5* and the **WindRange** is *15*.

String-functions

Character Function	What it Does
UPCASE (*char*) **LOWCASE**(*char*)	Changes letters in a character string to uppercase or lowercase
PROPCASE (*char*, <*delimiters*>)	Changes the first letter of each word to uppercase and other letters to lowercase
CATS (*char1*, *char2*, ...)	Concatenates character strings and removes leading and trailing blanks from each argument
SUBSTR (*char*, *position*, <*length*>)	Returns a substring from a character string

```
23 data storm_new;
24     set pg1.storm_summary;
25     drop Type Hem_EW Hem_NS MinPressure Lat Lon;
26     *Add assignment statements;
27     Basin=upcase(basin);
28     Name=propcase(Name);
29     Hemisphere=cats(Hem_NS, Hem_EW);
30     Ocean=substr(Basin, 2, 1);
31 run;
```

	Season	Name	Basin	MaxWindMPH	StartDate	EndDate	Hemisphere	Ocean
1	1980		NA	35	17JUL1980	18NOV1980	NW	A
2	1980		SP	.	27MAR1980	30MAR1980	SE	P
3	1980	Agatha	EP	115	09JUN1980	15JUN1980	NW	P
4	1980	Albine	SI	.	27NOV1979	06DEC1979	SE	I
5	1980	Alex	WP	40	09OCT1980	14OCT1980	NE	P
6	1980	Allen	NA	190	31JUL1980	11AUG1980	NW	A

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;
```

A note in the log indicates that **work.pacific** has 1958 observations, so there were 1958 storms in the Pacific basin.

Common Date Functions for Creating Columns

SAS date functions are incredibly helpful for creating and manipulating SAS dates.

These functions **extract** information from the SAS date column or value provided in the argument:

Date Function	What it Does
MONTH (*SAS-date*)	Returns a number from 1 through 12 that represents the month
YEAR (*SAS-date*)	Returns the four-digit year
DAY (*SAS-date*)	Returns a number from 1 through 31 that represents the day of the month
WEEKDAY (*SAS-date*)	Returns a number from 1 through 7 that represents the day of the week (Sunday=1)

QTR (*SAS-date*)	Returns a number from 1 through 4 that represents the quarter

These functions enable you to **create** SAS date values from the arguments.

Date Function	What it Does
TODAY ()	Returns the current date as a numeric SAS date value (no argument is required because the function reads the system clock)
MDY (*month*, *day*, *year*)	Returns a SAS date value from numeric month, day, and year values
YRDIF (*startdate*, *enddate*, 'AGE')	Calculates a precise age between two dates

```
17 data storm_new;
18     set pg1.storm_damage;
19     drop Summary;
20     *Add assignment and FORMAT statements;
21     YearsPassed=yrdif(Date, today(), "age");
22     Anniversary=mdy(month(Date), day(Date), year(today()));
23     format YearsPassed 4.1 Date Anniversary mmddyy10.;
24 run;
```

Jubilæum hvornår I today()'s year, 2020, er der gået helt antal år

	Event	Date	Cost	YearsPassed	Anniversary
1	Hurricane Katrina	08/25/2005	161300000000	14.5	08/25/2020
2	Hurricane Harvey	08/25/2017	125000000000	2.5	08/25/2020
3	Hurricane Maria	09/19/2017	90000000000	2.4	09/19/2020
4	Hurricane Sandy	10/30/2012	70900000000	7.3	10/30/2020
5	Hurricane Irma	09/06/2017	50000000000	2.5	09/06/2020
6	Hurricane Andrew	08/23/1992	48300000000	27.5	08/23/2020
7	Hurricane Ike	09/12/2008	35100000000	11.5	09/12/2020
8	Hurricane Ivan	09/12/2004	27300000000	15.5	09/12/2020

Task: indlæs datafil, omform den

- 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(7 chars).
    - Keep **Country**, **Hotel**, **ShortStay**, **Camp**, **ReportDate**, and **Total** in the new table.
    - Submit the program and view the output data.

Solution:

```
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;
```

2. What is the value of **ReportDate** in row one?

Solution:

SEP2017

1. Create a new program.
    - Create a new temporary table named **np_summary2** that is based on **pg1.np_summary**.
    - Use the SCAN function to create a new column named **ParkType** that is the last word in the **ParkName** column. Use a negative number for the second argument to count words from right to left in the character string.
    - Keep **Reg**, **Type**, **ParkName**, and **ParkType** in the output table.
    - Submit the program and view the output data.

Solution:

```
data np_summary2;
    set pg1.np_summary;
    ParkType=scan(parkname,-1);
*finder words, bagfra

    keep Reg Type ParkName ParkType;
run;
```

Cape Krusenstern National Monument -> ParkType = Monument

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;
```

2. What is the value of **ParkType** in row four?

Solution:

Preserve

```
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;
```

The FREQ procedure results show that there are **189** storms in **PressureGroup 1**

Ny variabel `PressureGroup` bliver oprettet som ny kolonne i tabellen

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

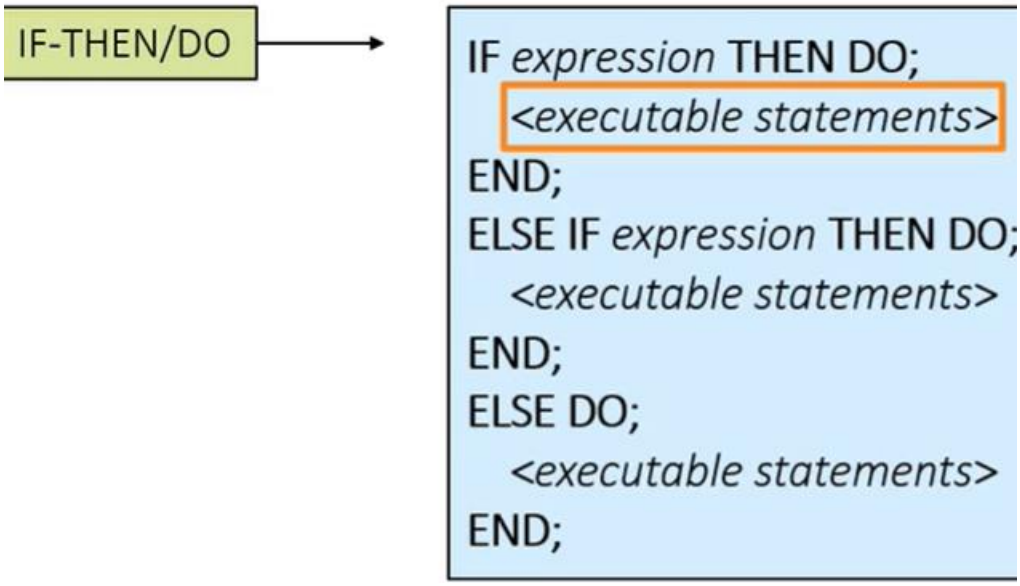
nødvendigt, ellers bestemmer første forekomst, Basic formattet, som så bliver len=5

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	Luxury
Audi	A4 1.8T 4dr	$25,940	Basic

```
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;
```

4. Move the LENGTH statement to the end of the DATA step, just before the RUN statement. Run the program. Does it matter where the LENGTH statement is in the DATA step?

Yes, a warning in the log indicates that the length of character variable **Ocean** has already been set.



data tab1 tab2 *der laves to output-tabeller

```
data under40 over40; to output tabeller
    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;
```

udskrives til , skal være til sidst

Select-when

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.

```
1. data parks monuments;
```

```
2.     set pg1.np_summary;
3.     where type in ('NM', 'NP');
4.     Campers=sum(OtherCamping, TentCampers, RVCampers,
5.                 BackcountryCampers);
6.     format Campers comma17.;
7.     length ParkType $ 8;
8.     select (type);
9.         when ('NP') do;
10.             ParkType='Park';
11.             output parks;
12.            end;
13.            otherwise do;
14.             ParkType='Monument';
15.             output monuments;
16.            end;
17.     end;
18.     keep Reg ParkName DayVisits OtherLodging Campers
   ParkType;
19. run;
```

20. Submit the program and verify that **work.parks** contains 51 rows and **work.monuments** contains 63 rows.

Understand how to control how data is read and written, merge multiple tables

What If You Want to...

...understand how the DATA step processes behind the scenes to control how data is read and written, merge multiple tables with the DATA step, simplify repetitive code with DO loops or arrays?

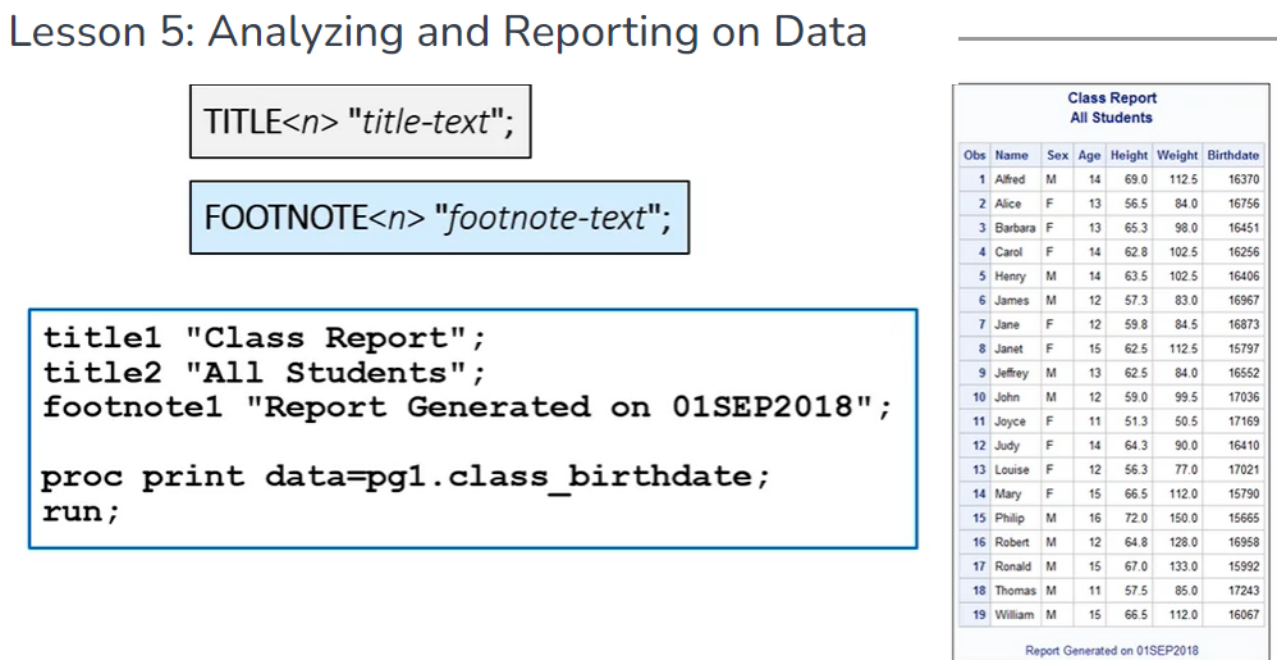
- Take the SAS Programming 2: Data Manipulation Techniques and SAS Programming 3: Advanced Techniques and Efficiencies courses.

...manipulate data with PROC SQL?

...learn more about PROC DS2 to manipulate data?

https://vle.sas.com/mod/scorm/player.php?a=10986&currentorg=ORG-EPG1V2&scoid=25239

*** Lesson 5: Analyzing and Reporting on Data



Makro-varible i titles

TITLE<*n*> "*title-text*";

```
%let age=13;

title1 "Class Report";
title2 "Age=&age";
footnote1 "School Use Only";

proc print data=pg1.class_birthdate;
    where age=&age;
run;

title;
footnote;
```

Class Report
Age=13

Obs	Name	Sex	Age	Height	Weight	Birthdate
2	Alice	F	13	56.5	84	16756
3	Barbara	F	13	65.3	98	16451
9	Jeffrey	M	13	62.5	84	16552

School Use Only

```
proc means data=sashelp.cars;
    where type="Sedan";
    var MSRP MPG_Highway;
    label MSRP="Manufacturer Suggested Retail Price"
          MPG_Highway="Highway Miles per Gallon";
run;
```

Der laves frequency-analyse på talværdier i Type-feltet, grupperet efter origin:

Segmenting Reports

```
proc sort data=sashelp.cars
          out=cars_sort;
    by Origin;
run;

proc freq data=cars_sort;
    by Origin;
    tables Type;
run;
```

Type	Frequency
Hybrid	3
SUV	25
Sedan	94
Sports	17
Truck	8
Wagon	11

Type	Frequency
SUV	10
Sedan	78
Sports	23
Wagon	12

Type	Frequency
SUV	25
Sedan	90
Sports	9
Truck	16
Wagon	7

Label

```
*    LABEL col-name="label-text"                                *;
*          col-name="label-text";                               *;
*                                                               *;
*    Grouped Reports (sort first):                              *;
*    PROC procedure-name;                                       *;
*        BY col-name;                                           *;
*    RUN;                                                       *;
*****************************************************************;
proc sort data=pg1.storm_final out=storm_sort;
    by BasinName descending MaxWindMPH;
    where MaxWindMPH > 156;
run;

title "Category 5 Storms";
proc print data=storm_sort label;
    var Season Name MaxWindMPH MinPressure StartDate StormLength;
    by BasinName;
    label MaxWindMPH="Max Wind (MPH)"
        MinPressure="Min Pressure"
        StartDate="Start Date"
        StormLength="Length of Storm (days)";
run;

title;
```

```
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;
```

```
*    ODS GRAPHICS ON;
*    PROC FREQ DATA=input-table <proc-options>;
*        TABLES col-name(s) / options;
*    RUN;
*
*    PROC FREQ statement options:
*        ORDER=FREQ|FORMATTED|DATA
*        NLEVELS
*    TABLES statement options:
*        NOCUM
*        NOPERCENT
*        PLOTS=FREQPLOT
*           (must turn on ODS Graphics)
*        OUT=output-table
**************************************************

proc freq data=pg1.storm_final;
    tables BasinName Season;
run;
```

frekvenstabel for hver felt nævnt

```
ods graphics on;
ods noproctitle;normalt står proc-title proc freq som title, det stoppes her
title "Frequency Report for Basin and Storm Month";
proc freq data=pg1.storm_final order=freq nlevels;
    tables BasinName StartDate / nocum plots=freqplot(orient=horizontal scale=percent);
    format StartDate monname.;
    label BasinName="Basin"
run;
title;
```

```
title "Frequency Report for Basin and Storm Month";

proc freq data=pg1.storm_final order=freq noprint;
    tables StartDate / out=storm_count;
    format StartDate monname.;
run;
```

2-way frequency table, pivot-tabel, med felt1*felt2

2way fordi man opgører bade for rækkerne, vandret og for hver kolonne, lodret

**********;

```
*  Creating Two-Way Frequency Reports                *;
************************************************************;
*  Syntax and Example                              *;
*                                    *;
*   PROC FREQ DATA=input-table;                    *;
*      TABLES col-name*col-name </ options>;       *;
*   RUN;                                  *;
*                                    *;
*   PROC FREQ statement options:                   *;
*      NOPRINT                                *;
*   TABLES statement options:                      *;
*      NOROW, NOCOL, NOPERCENT                     *;
*      CROSSLIST, LIST                          *;
*      OUT=output-table                           *;
************************************************************;

title "Blood Pressure by Cholesterol Status";
proc freq data=sashelp.heart;
                tables BP_Status*Chol_Status; *pivottabel med BP_Status rækker og Chol_Status som
kolonner
run;
title;

************************************************************;
*  Demo (Highlight the PROC FREQ step and run            *;
*       the selected code after each step.)              *;
*   1) Highlight the PROC FREQ step, run the selected    *;
*      code, and examine the default results.            *;
*   2) Add the NOPERCENT, NOROW, and NOCOL options in    *;
*      the TABLES statement.                             *;
*   3) Delete the options in the TABLES statement and    *;
*      add the CROSSLIST option.                         *;
*   4) Change the CROSSLIST option to the LIST option in *;
*      the TABLES statement.                             *;
*   5) Delete the previous options and add               *;
*      OUT=STORMCOUNTS. Add NOPRINT to the PROC FREQ     *;
*      statement to suppress the report.                 *;
************************************************************;


proc freq data=pg1.storm_final;
                tables BasinName*StartDate;
                format StartDate monname.;
                label BasinName="Basin"
                                StartDate="Storm Month";
run;
```

Blood Pressure by Cholesterol Status

The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

Table of BP_Status by Chol_Status				
	Chol_Status(Cholesterol Status)			
BP_Status(Blood Pressure Status)	Borderline	Desirable	High	Total
High	798 15.78 36.21 42.88	456 9.02 20.69 32.46	950 18.79 43.10 53.04	2204 43.58
Normal	793 15.68 38.09 42.61	634 12.54 30.45 45.12	655 12.95 31.46 36.57	2082 41.17
Optimal	270 5.34 35.02 14.51	315 6.23 40.86 22.42	186 3.68 24.12 10.39	771 15.25
Total	1861 36.80	1405 27.78	1791 35.42	5057 100.00

Frequency Missing = 152

=2204/5057

1861/5057

demo

Frequency
Percent
Row Pct
Col Pct

Table of BasinName by StartDate													
	StartDate(Storm Month)												
BasinName(Basin)	November	December	January	February	March	April	May	June	July	August	September	October	Total
East Pacific	15 0.49 2.22 7.94	3 0.10	2 0.06	0 0.00 0.00 0.00	1 0.03 0.15 0.55	0 0.00 0.00 0.00	29 0.94 4.30 24.79	76 2.46 11.26 40.86	156 5.05 23.11 45.09	163 5.27 24.15 33.61	142 4.59 21.04 29.22	88 2.85 13.04 26.99	675 21.83
North Atlantic	26 0.84 5.44 13.76	5 0.16 1.05 2.78	1 0.03 0.21 0.41	0 0.00 0.00 0.00	0 0.00 0.00 0.00	2 0.06 0.42 1.60	10 0.32 2.09 8.55	30 0.97 6.28 16.13	45 1.46 9.41 13.01	132 4.27 27.62 27.22	152 4.92 31.80 31.28	75 2.43 15.69 23.01	478 15.46
North Indian	19 0.61 31.67 10.05	7 0.23 11.67 3.89	1 0.03 1.67 0.41	0 0.00 0.00 0.00	0 0.00 0.00 0.00	4 0.13 6.67 3.20	8 0.26 13.33 6.84	5 0.16 8.33 2.69	1 0.03 1.67 0.29	0 0.00 0.00 0.00	2 0.06 3.33 0.41	13 0.42 21.67 3.99	60 1.94
South Indian	41 1.33 6.90 21.69	89 2.88 14.98 49.44	139 4.50 23.40 56.50	126 4.08 21.21 56.25	93 3.01 15.66 51.10	66 2.13 11.11 52.80	14 0.45 2.36 11.97	4 0.13 0.67 2.15	3 0.10 0.51 0.87	1 0.03 0.17 0.21	4 0.13 0.67 0.82	14 0.45 2.36 4.29	594 19.21
South Pacific	14 0.45 3.90 7.41	37 1.20 10.31 20.56	88 2.85 24.51 35.77	93 3.01 25.91 41.52	74 2.39 20.61 40.66	32 1.03 8.91 25.60	13 0.42 3.62 11.11	4 0.13 1.11 2.15	0 0.00 0.00 0.00	0 0.00 0.00 0.00	0 0.00 0.00 0.00	4 0.13 1.11 1.23	359 11.61
West Pacific	74 2.39 7.99 39.15	39 1.26 4.21 21.67	15 0.49 1.62 6.10	5 0.16 0.54 2.23	14 0.45 1.51 7.69	21 0.68 2.27 16.80	43 1.39 4.64 36.75	67 2.17 7.24 36.02	141 4.56 15.23 40.75	189 6.11 20.41 38.97	186 6.02 20.09 38.27	132 4.27 14.25 40.49	926 29.95
Total	189 6.11	180 5.82	246 7.96	224 7.24	182 5.89	125 4.04	117 3.78	186 6.02	346 11.19	485 15.69	486 15.72	326 10.54	3092 100.00

udgør 21.8% af alle

=15/675

=15/189

=675/3092

nov udgør 6.11% af alle måneder

Noprint så udskrives ikke til result-fanen,

```
proc freq data=pg1.storm_final noprint;
    tables BasinName*StartDate / out=stormcounts;
    format StartDate monname.;
    label BasinName="Basin"
          StartDate="Storm Month";
run;
```

her laves til gengæld work.output table, kun midlertidig tabel

2-way stat, pivot-tabel

```
proc freq data=pg1.storm_final;
    tables BasinName*StartDate;
    format StartDate monname.;
    label BasinName="Basin"
          StartDate="Storm Month";
run;
```

basinName=række,
StartDate=kolonner

The FREQ Procedure

Frequency
Percent
Row Pct
Col Pct

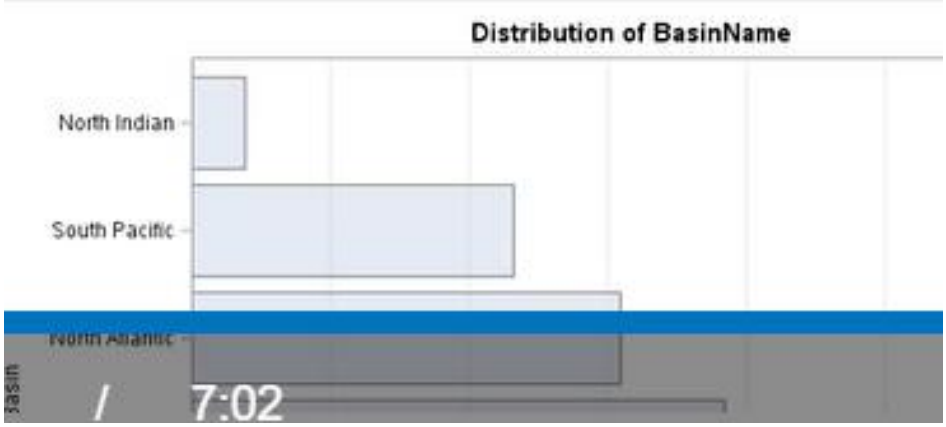
Table of BasinName by StartDate

BasinName(Basin)	StartDate(Storm Month) November	December	January	February	March	April	May	June	July	August	September	October	Total
East Pacific	15 0.49 2.22 7.94	3 0.10 0.44 1.67	2 0.06 0.30 0.81	0 0.00 0.00 0.00	1 0.03 0.15 0.55	0 0.00 0.00 0.00	29 0.94 4.30 24.79	76 2.46 11.26 40.86	156 5.05 23.11 45.09	163 5.27 24.15 33.61	142 4.59 21.04 29.22	88 2.85 13.04 26.99	675 21.83
North Atlantic	26 0.84 5.44 13.76	6 0.19 1.05 2.78	1 0.03 0.21 0.41	0 0.00 0.00 0.00	0 0.00 0.00 0.00	2 0.06 0.42 1.60	10 0.32 2.09 8.55	30 0.97 6.28 16.13	45 1.46 9.41 13.01	132 4.27 27.62 27.22	152 4.92 31.80 31.28	75 2.43 15.69 23.01	478 15.46
North Indian	19 0.61 31.67 10.05	7 0.23 11.67 3.89	1 0.03 1.67 0.41	0 0.00 0.00 0.00	0 0.00 0.00 0.00	4 0.13 6.67 3.20	8 0.26 13.33 6.84	5 0.16 8.33 2.69	1 0.03 1.67 0.29	0 0.00 0.00 0.00	2 0.06 3.33 0.41	13 0.42 21.67 3.99	60 1.94
South Indian	41 1.33 6.90 21.69	89 2.88 14.99 49.44	139 4.50 23.40 56.50	126 4.08 21.21 56.25	93 3.01 15.66 51.10	66 2.13 11.11 52.80	14 0.45 2.36 11.97	4 0.13 0.67 2.15	3 0.10 0.51 0.87	1 0.03 0.17 0.21	4 0.13 0.67 0.82	14 0.45 2.36 4.29	594 19.21
South Pacific	14 0.45 3.90 7.41	37 1.20 10.31 20.56	88 2.85 24.51 35.77	93 3.01 25.91 41.52	74 2.39 20.61 40.66	32 1.03 8.91 25.60	13 0.42 3.62 11.11	4 0.13 1.11 2.15	0 0.00 0.00 0.00	0 0.00 0.00 0.00	0 0.00 0.00 0.00	4 0.13 1.11 1.23	359 11.61
West Pacific	74 2.39 7.99 39.15	39 1.26 4.21 21.67	15 0.49 1.62 6.10	5 0.16 0.54 2.23	14 0.45 1.51 7.69	21 0.68 2.27 16.80	43 1.39 4.64 36.75	67 2.17 7.24 36.02	141 4.56 15.23 40.75	189 6.11 20.41 38.97	186 6.02 20.09 38.27	132 4.27 14.25 40.49	926 29.95
								186	346	485	488	326	3092

Frequency Report for Basin and Storm Month

Number of Variable Levels		
Variable	Label	Levels
BasinName	Basin	6
StartDate	Storm Month	12

Basin		
BasinName	Frequency	Percent
West Pacific	928	29.95
East Pacific	675	21.83
South Indian	594	19.21
North Atlantic	478	15.46
South Pacific	359	11.61
North Indian	60	1.94



The FREQ Procedure

BasinName	Frequency	Percent	Cumulative Frequency	Cumulative Percent
East Pacific	675	21.83	675	21.83
North Atlantic	478	15.46	1153	37.29
North Indian	60	1.94	1213	39.23
South Indian	594	19.21	1807	58.44
South Pacific	359	11.61	2166	70.05
West Pacific	926	29.95	3092	100.00

Season	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1980	79	2.55	79	2.55
1981	87	2.81	166	5.37
1982	79	2.55	245	7.92
1983	76	2.46	321	10.38
1984	90	2.91	411	13.29
1985	92	2.98	503	16.27
1986	80	2.59	583	18.86
1987	70	2.26	653	21.12
1988	72	2.33	725	23.45
1989	90	2.91	815	26.36
1990	87	2.81	902	29.17
1991	69	2.23	971	31.40
1992	93	3.01	1064	34.41
1993	79	2.55	1143	36.97
1994	89	2.88	1232	39.84
1995	73	2.36	1305	42.21

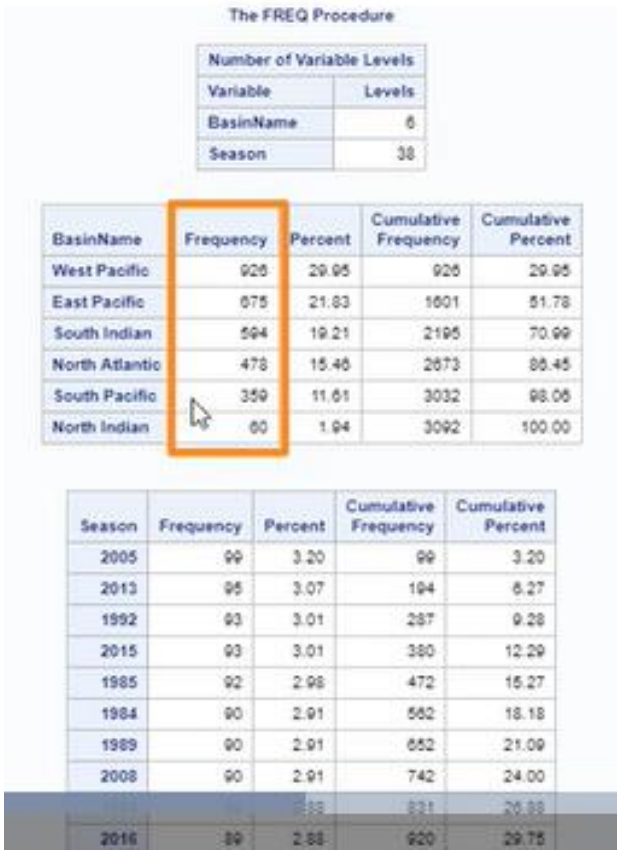
```
22 proc freq data=pg1.storm_final order=freq nlevels;
23     tables BasinName Season;
24 run;
25
```

order by freq, aftagende; levels er antal unikke værdier

```
proc freq data=pg1.storm_final order=freq nlevels;
    tables BasinName StartDate / nocum;
    format StartDate monname.;
run;
```

så sker frekvensbehandlingen på formatted data,

StartDate	Frequency	Percent
September	486	15.72
August	485	15.69
July	346	11.19
October	326	10.54
January	246	7.96
February	224	7.24
November	189	6.11
June	186	6.02
March	182	5.89
April	125	4.04
May	117	3.78



The FREQ Procedure

Number of Variable Levels	
Variable	Levels
BasinName	6
Season	38

BasinName	Frequency	Percent	Cumulative Frequency	Cumulative Percent
West Pacific	926	29.95	926	29.95
East Pacific	675	21.83	1601	51.78
South Indian	594	19.21	2195	70.99
North Atlantic	478	15.46	2673	86.45
South Pacific	359	11.61	3032	98.06
North Indian	60	1.94	3092	100.00

Season	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2005	99	3.20	99	3.20
2013	95	3.07	194	6.27
1992	93	3.01	287	9.28
2015	93	3.01	380	12.29
1985	92	2.98	472	15.27
1984	90	2.91	562	18.18
1989	90	2.91	652	21.09
2008	90	2.91	742	24.00
2016	89	2.88	920	29.75

1. Add a CLASS(for classification) 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.
   - Which ocean had the lowest mean for minimum pressure?
   - Which season had the lowest mean for minimum pressure?

```
proc means data=pg1.storm_final maxdec=0 n mean min;
    var MinPressure;
    where Season >=2010;
    class Season Ocean;
    ways 1;
run;
```

The **Pacific** ocean had the lowest mean for minimum pressure.
The season that had the lowest mean for minimum pressure was **2015**.

Summery med output

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

```
proc means data=sashelp.heart noprint;
    var Weight;
    class Chol_Status;
    ways 1;
    output out=heart_stats mean=AvgWeight;
run;
```

Chol_Status	_TYPE_	_FREQ_	AvgWeight
Borderline	1	1861	154.31827957
Desirable	1	1405	148.43121882
High	1	1791	155.4082774

```
*  Activity 5.06                                *;
*    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.        *;
*    3) Add the following options in the OUTPUT statement *;
*       and run the program again. How many rows are in   *;
*       the output table?                                 *;
*        output out=wind_stats mean=AvgWind max=MaxWind; *;
************************************************************;


proc means data=pg1.storm_final mean median max;
                    var MaxWindMPH;
                    class BasinName;
                    *ways 1;
                    output out=wind_stats;
run;
```

The MEANS Procedure

Analysis Variable : MaxWindMPH				
BasinName	N Obs	Mean	Median	Maximum
East Pacific	675	82.7629630	75.0000000	213.0000000
North Atlantic	478	82.8953975	75.0000000	190.0000000
North Indian	60	63.6000000	52.0000000	146.0000000
South Indian	594	77.3593750	69.0000000	155.0000000
South Pacific	359	78.0421348	69.0000000	173.0000000
West Pacific	926	79.6511879	81.0000000	144.0000000

```
proc means data=pg1.storm_final

    var MaxWindMPH;
    class BasinName;
    ways 1;
    *output out=wind_stats;
    output out=wind_stats mean=AvgWind max=MaxWind;
run;
```

The MEANS Procedure

BasinName	N Obs	Variable	Variance	Maximum
East Pacific	675	Season	130.9924739	2017.00
		MaxWindMPH	1334.47	213.0000000
		MaxWindKM	3456.24	342.7894200
		MinPressure	616.5301272	1010.00
		StartDate	17486311.21	21119.00
		EndDate	17486655.80	21120.00
		StormLength	34.2596681	112.0000000
		Lat	10.0515450	34.4000000
		Lon	287.8129722	-86.2000000
North Atlantic	478	Season	110.1357377	2017.00

1. Uncomment the WAYS statement. Delete the statistics listed in the PROC MEANS statement and add the NOPRINT option. Run the program. Notice a report is not generated and the first 5 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;
```

MAP MED DATA

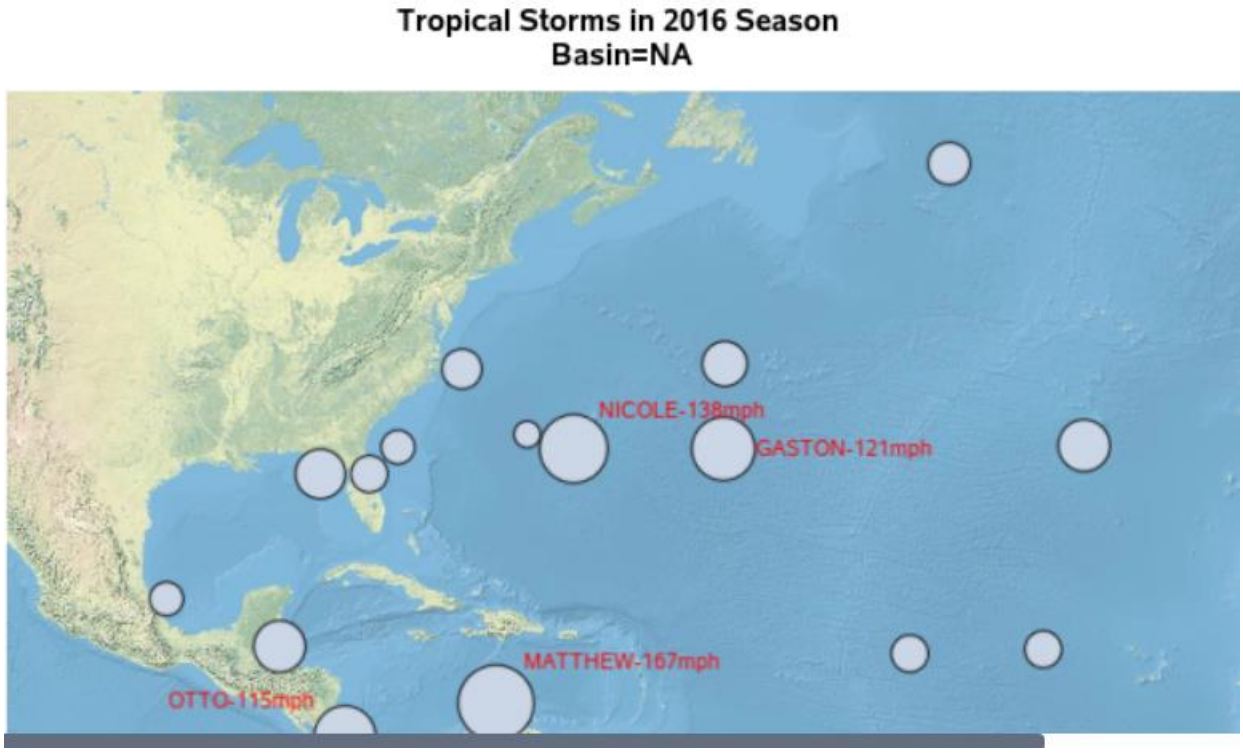
Activity 5.07 p105a07.sas

```
**********************************************;
*  Creating a Map with PROC SGMAP            *;
```

```
*   Requires SAS 9.4M5 or later                *;
*************************************************;
%let Year=2016;
%let basin=NA;

*Preparing the data for map labels;   output herfra map er input til selve plottet
data map;
                set pg1.storm_final;
                length maplabel $ 20;
                where season=&year and basin="&basin";
                if maxwindmph<100 then MapLabel=" ";
                else maplabel=cats(name,"-",maxwindmph,"mph");   *string-sammensætning
                keep lat lon maplabel maxwindmph;
run;
```

*cats(name,"-",maxwindmph,"mph");   string-sammensætning

```
*Creating the map;
title1 "Tropical Storms in &year Season";
title2 "Basin=&basin";
footnote1 "Storms with MaxWind>100mph are labeled";

proc sgmap plotdata=map;
   *openstreetmap;
   esrimap url='https://services.arcgisonline.com/arcgis/rest/services/World_Physical_Map';
        bubble x=lon y=lat size=maxwindmph / datalabel=maplabel datalabelattrs=(color=red size=8);
run;
title;footnote;
```

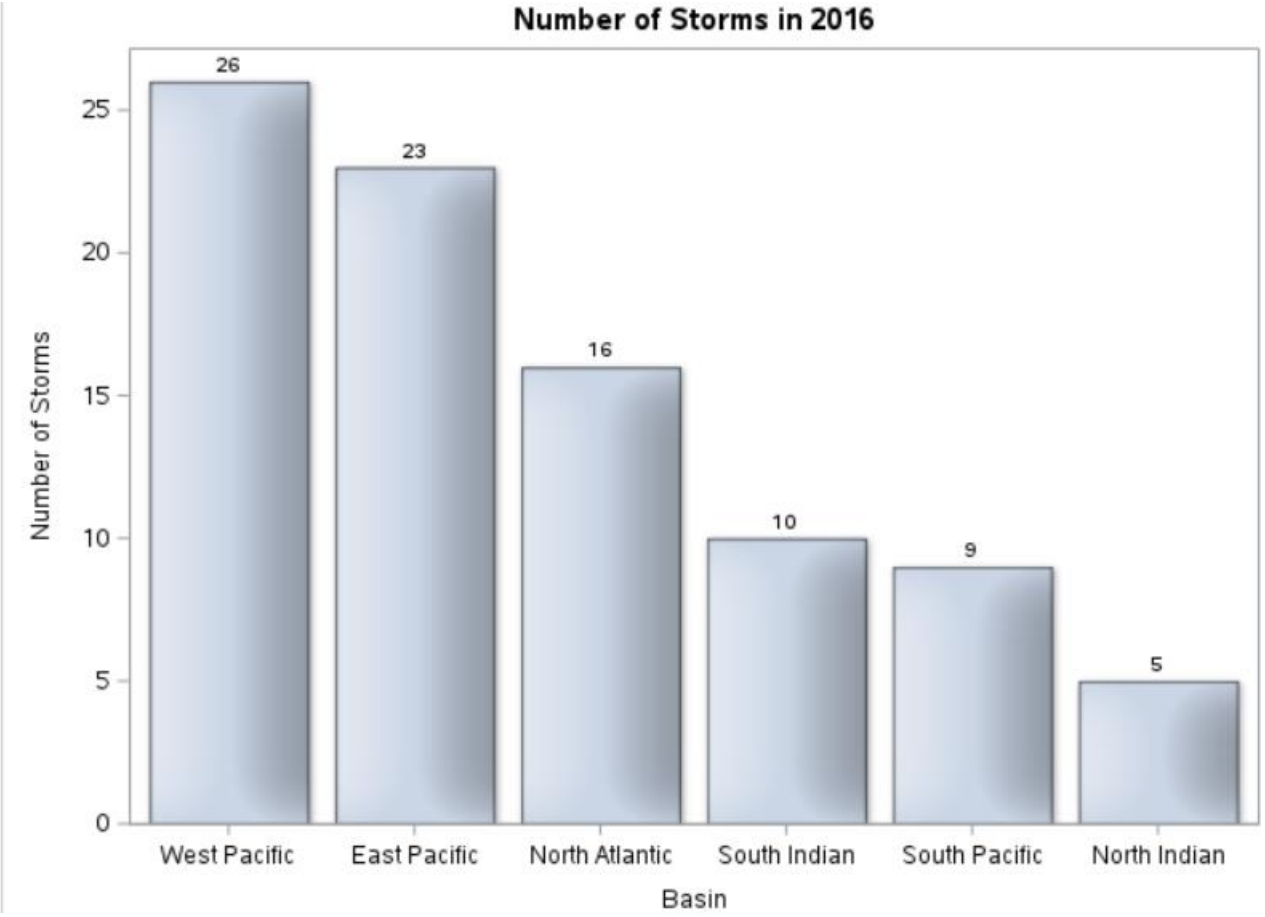


**********;

* Creating a Bar Chart with PROC SGPLOT *;

**********;

```
title "Number of Storms in &year";
proc sgplot data=pg1.storm_final;
                where season=&year;
                vbar BasinName / datalabel dataskin=matte categoryorder=respdesc;
                xaxis label="Basin";
                yaxis label="Number of Storms";
run;
```

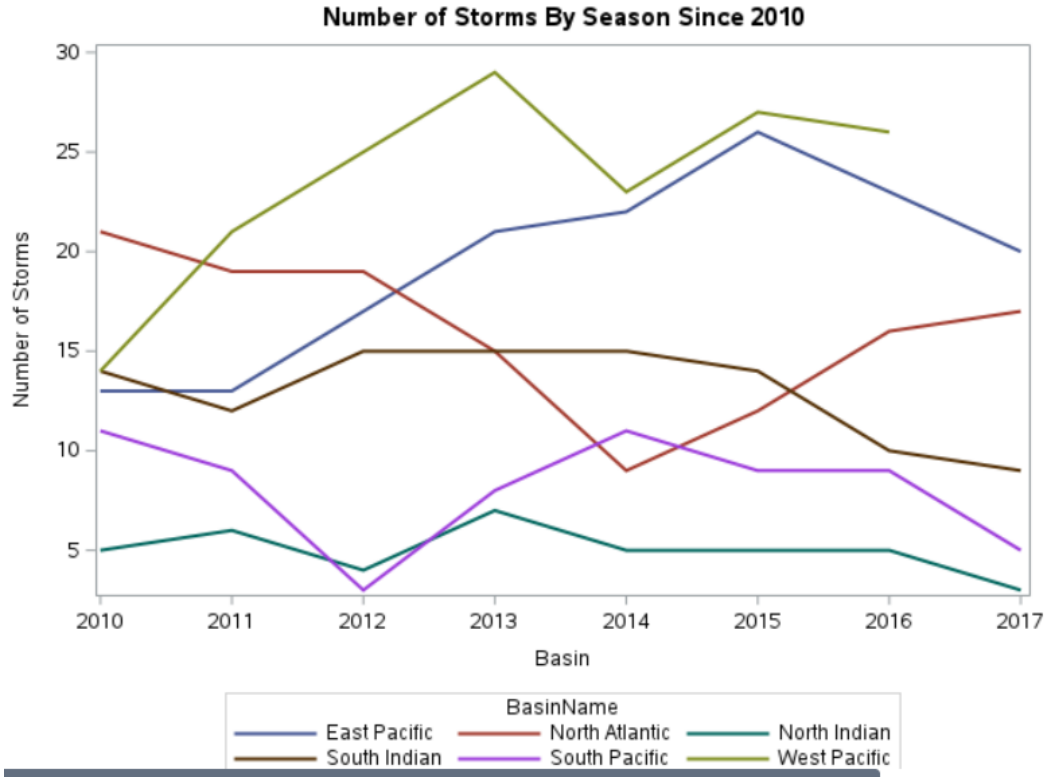


**********;

* Creating a Line PLOT with PROC SGPLOT *;

**********;

```
title "Number of Storms By Season Since 2010";
proc sgplot data=pg1.storm_final;
                where Season>=2010;
                vline Season / group=BasinName lineattrs=(thickness=2);
                yaxis label="Number of Storms";
                xaxis label="Basin";
run;
```




```
proc format;
   value count 25-high="lightsalmon";  *25 og opad
   value maxwind 90-high="lightblue";
run;

title "Storm Summary since 2000";
footnote1 "Storm Counts 25+ Highlighted";
footnote2 "Max Wind 90+ Highlighted";

proc tabulate data=pg1.storm_final format=comma5.;
                  where Season>=2000;
                  var MaxWindMPH;
                  class BasinName;
                  class Season;
                  table Season={label=""} all={label="Total"}*{style={background=white}},
                                    BasinName={LABEL="Basin"}*(MaxWindMPH={label=" "}*N={label="Number
of Storms"}*{style={background=count.}}
                                    MaxWindMPH={label=" "}*Mean={label="Average Max
Wind"}*{style={background=maxwind.}})
```

```
                              ALL={label="Total"  style={vjust=b}}*(MaxWindMPH={label="
"}*N={label="Number of Storms"}
                              MaxWindMPH={label=" "}*Mean={label="Average Max
Wind"})/style_precedence=row;
run;
title;
footnote;
```

Storm Summary since 2000

	Basin											
	East Pacific		North Atlantic		North Indian		South Indian		South Pacific		West Pacific	
	Number of Storms	Average Max Wind	Number of Storms	Average Max Wind	Number of Storms	Average Max Wind	Number of Storms	Average Max Wind	Number of Storms	Average Max Wind	Number of Storms	Avera Max Win
2000	19	71	14	84	.	.	19	83	10	73	23	
2001	15	79	15	87	.	.	13	81	8	62	26	
2002	15	99	12	75	.	.	14	93	5	65	24	
2003	17	66	16	82	.	.	18	82	10	99	21	
2004	12	84	15	95	.	.	15	90	5	78	29	
2005	17	71	30	85	2	52	16	74	11	94	23	
2006	24	78	9	82	2	89	12	85	8	90	22	
2007	15	59	17	75	3	110	15	92	8	71	24	
2008	19	66	17	88	7	51	20	75	5	91	22	
2009	23	76	11	68	4	55	17	70	9	63	22	

..

Practice: Producing a Descriptive Statistic Report

The **pg1.np_westweathe**r table contains weather-related information for four national parks: Death Valley National Park, Grand Canyon National Park, Yellowstone National Park, and Zion National Park. Use the MEANS procedure to analyze the data in this table.

Reminder: If you restarted your SAS session,you must recreate the **PG1** library so you can access your practice files. In SAS Studio, open and submit the **libname.sas** program in the **EPG1V2** folder. In Enterprise Guide, run the **Autoexec** process flow.

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.

Solution:

```
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;
```

CODE LOG RESULTS

Table of Contents

Weather Statistics by Year and Park

The MEANS Procedure

Year	NAME	N Obs	Variable	Mean	Minimum	Maximum
2015	DEATH VALLEY, CA US	365	PRECIP	0.01	0.00	0.55
			SNOW	0.00	0.00	0.00
			TEMPMIN	64.44	26.00	97.00
			TEMPMAX	93.29	53.00	125.00
	GRAND CANYON VISITOR CENTER, AZ US	365	PRECIP	0.07	0.00	2.20
			SNOW	0.13	0.00	5.70
			TEMPMIN	40.99	8.00	69.00
			TEMPMAX	61.60	17.00	93.00
	YELLOWSTONE NATIONAL PARK EAST ENTRANCE, WY US	363	PRECIP	0.06	0.00	0.85

-

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;
```

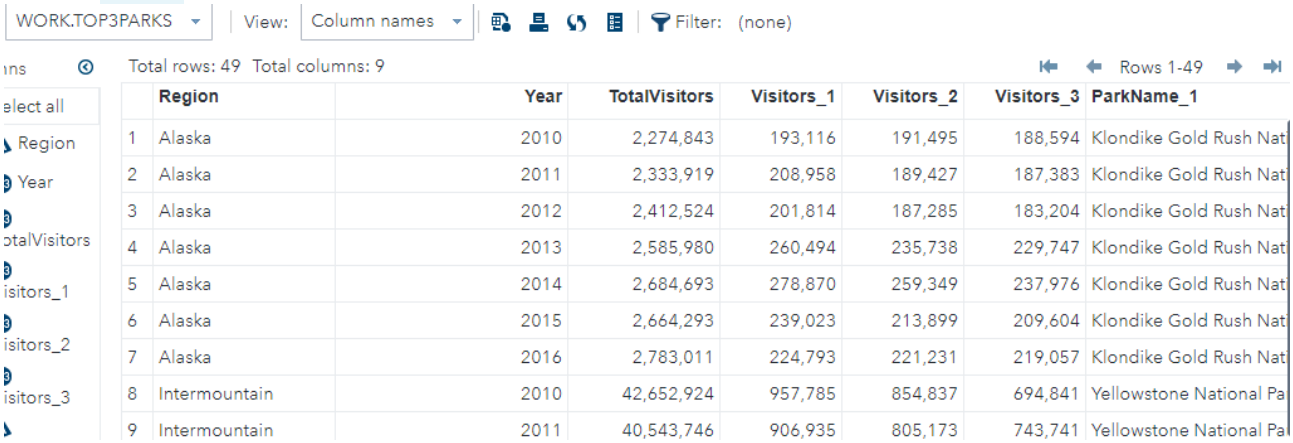
	Year	_TYPE_	_FREQ_	RainDays
	2015	3	15	15
	2016	3	16	16
	2017	3	11	11
	2015	3	97	97
	2016	3	84	82
	2017	3	65	65
/Y L	nameX 15	3	155	150
/Y L	16	3	154	149
/Y L	17	3	172	143
	2015	3	77	77
	2016	3	68	68
	2017	3	56	56

1. Create a new program. Write a PROC MEANS step to analyze rows from **pg1.np_multiyr** and create a table named **top3parks** with the following attributes:
    - Suppress the display of the PROC MEANS report.
    - Analyze **Visitors** grouped by **Region** and **Year**.
    - Drop the **_FREQ_** and **_TYPE_** columns from **top3parks** and keep only the rows that are a result of a combination of **Region** and **Year**.
    - Create a column for **TotalVisitors** in the output table.
    - Use the IDGROUP option on the OUTPUT statement to add additional columns with the top three maximum values of **Visitors** for each **Region** and **Year**. Columns named **Visitors_1**, **Visitors_2**, and **Visitors_3** should include the top 3 visitor counts. Columns named **ParkName_1**, **ParkName_2**, and **ParkName_3** should include the corresponding park name.
      Note: Use SAS Help to learn about the IDGROUP option in the OUTPUT statement.
    - Submit the program and view the output data.

Solution:

```
proc means data=pg1.np_multiyr noprint;
    var Visitors;
    class Region Year;
    ways 2;
    output out=top3parks(drop=_freq_ _type_)
           sum=TotalVisitors /*sum total visitors*/
```

```
            idgroup(max(Visitors) /*find the max of
visitors*, laver 3 grupper ud fra max /
            out[3] /*top 3*/
            (Visitors ParkName)=); /*output columns for top
3 parks*/
run;
```



WORK.TOP3PARKS | View: Column names | Filter: (none)

Total rows: 49 Total columns: 9 | Rows 1-49

	Region	Year	TotalVisitors	Visitors_1	Visitors_2	Visitors_3	ParkName_1
1	Alaska	2010	2,274,843	193,116	191,495	188,594	Klondike Gold Rush Nat
2	Alaska	2011	2,333,919	208,958	189,427	187,383	Klondike Gold Rush Nat
3	Alaska	2012	2,412,524	201,814	187,285	183,204	Klondike Gold Rush Nat
4	Alaska	2013	2,585,980	260,494	235,738	229,747	Klondike Gold Rush Nat
5	Alaska	2014	2,684,693	278,870	259,349	237,976	Klondike Gold Rush Nat
6	Alaska	2015	2,664,293	239,023	213,899	209,604	Klondike Gold Rush Nat
7	Alaska	2016	2,783,011	224,793	221,231	219,057	Klondike Gold Rush Nat
8	Intermountain	2010	42,652,924	957,785	854,837	694,841	Yellowstone National Pa
9	Intermountain	2011	40,543,746	906,935	805,173	743,741	Yellowstone National Pa

Learning More

What If You Want to...

...create high-quality, customized graphs?

...learn about basic statistics procedures to analyze your data?

...generate detail and summary tabular reports?

** Summary of Lesson 5: Analyzing and Reporting on Data

Enhancing Reports with Titles, Footnotes, and Labels

- TITLE is a global statement that establishes a permanent title for all reports created in your SAS session.
- You can have up to 10 titles, in which case you would just use a number 1-10 after the keyword TITLE to indicate the line number. TITLE and TITLE1 are equivalent.
- Titles can be replaced with an additional TITLE statement with the same number. TITLE; clears all titles.

- You can also add footnotes to any report with the FOOTNOTE statement. The same rules for titles apply for footnotes.
- Labels can be used to provide more descriptive column headers. A label can include any text up to 256 characters.
- All procedures automatically display labels with the exception of PROC PRINT. You must add the LABEL option in the PROC PRINT statement.

```
TITLE<n> "title-text";
```

-

```
FOOTNOTE<n> "footnote-text";
```

-

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

-
- To create a grouped report, first use PROC SORT to arrange the data by the grouping variable, and then use the BY statement in the reporting procedure.

```
PROC procedure-name;
    BY col-name;
RUN;
```

Creating Frequency Reports

- PROC FREQ creates a frequency table for each variable in the input table by default. You can limit the variables analyzed by using the TABLES statement.

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

-
- PROC FREQ statement options:

  ORDER=FREQ|FORMATTED|DATA
  NLEVELS

TABLES statement options:

NOCUM
NOPERCENT
PLOT=FREQPLOT (must turn on ODS graphics)
OUT=*output-table*

- One or more TABLES statements can be used to define frequency tables and options.
- ODS graphics enable graph options to be used in the TABLES statement.
- WHERE, FORMAT, and LABEL statements can be used in PROC FREQ to customize the report.
- When you place an asterisk between two columns in the TABLES statement, PROC FREQ produces a two-way frequency or crosstabulation report.

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

-

Creating Summary Statistics Reports

- Options in the PROC MEANS statement control the statistics included in the report.
- The CLASS statement specifies variables to group the data, classification, before calculating statistics.
- The WAYS statement specifies the number of ways to make unique combinations of class variables.

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

-
- The OUTPUT statement creates an output SAS table with summary statistics. Options in the OUTPUT statement determine the contents of the table.

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

OUTPUT OUT=*output-table <statistic(col-name)=col-name> < / option(s)*;
RUN;

quiz

https://vle.sas.com/pluginfile.php/1262586/mod_scorm/content/286/05/epg1v205_4_c_quiz.htm

When you run this program, which title or titles appear in the final PROC PRINT results?

```
title1 'The First Line';
title2 'The Second Line';
proc print data=sales;
run;
title2 'The Next Line';
proc print data=sales;
run;
title 'The Top Line';
proc print data=sales;
run;
```

a. The Top Line

TITLE is the same as TITLE1. The TITLE statement for the last PROC PRINT step cancels all the higher TITLE*n* statements.

```
%let Year=2018;
```

d. `footnote "&year Sales";`

```
data baseball2;
    set sashelp.baseball;
    BatAvg=CrHits/CrAtBat;
    label BatAvg="Batting Average";
run;

proc print data=baseball2;
    var Name Team BatAvg;
run;

proc means data=baseball2;
    var BatAvg;
    class Team;
run;
```

a. The column **BatAvg** will have a permanent label in the **sashelp.baseball** data set.
b. The label for **BatAvg** will appear in the PROC PRINT report.
c. The label for **BatAvg** will appear in the PROC MEANS report.

d. The label for **BatAvg** will appear in both reports.

Your answer: a
Correct answer: c

The label will appear in the PROC MEANS report. The label will not appear in the PROC PRINT report because the LABEL option is missing in the PROC PRINT statement. The output table **work.baseball2** (not the input table **sashelp.baseball**) contains a permanent label for **BatAvg**.

Which statement is true regarding a BY statement in a reporting procedure such as PROC PRINT?

a. The BY statement is responsible for sorting the table.
b. Only one column can be specified in the BY statement.
c. The BY statement groups the report by the specified columns.
d. The BY statement must be the first statement after the PROC statement.

Your answer: x
Correct answer: c

The BY statement in a reporting procedure is responsible for grouping the report by the specified columns. One or multiple columns can be in the BY statement. The BY statement can be placed in any order within a PROC step. The BY statement in PROC SORT is responsible for sorting the table.

```
proc freq data=sashelp.cars;
    where Cylinders in (4,6) and Type in ('Sedan','SUV');
    tables Type*Cylinders / nocol norow crosslist;
run;
```

Which statement is false concerning the FREQ procedure?

a. The NOPROCTITLE option can be placed in the PROC FREQ statement to remove the procedure title **The FREQ Procedure**.

c.

Table of Type by Cylinders			
Type	**Cylinders**	**Frequency**	**Percent**
SUV	**4**	7	2.77
	6	30	11.86
	Total	37	14.62

NOCOL removes the Column Percent, NOROW removes the Row Percent, and CROSSLIST displays statistic values in columns instead of stacked in a cell.

Which statement is true concerning the MEANS procedure?

b. The WAYS statement specifies the number of ways to make unique combinations of class columns.

Which statement from PROC MEANS contains valid syntax for creating a summary output table?

d. output out=work.summary mean(Weight)=TotW;

The OUTPUT statement writes statistics to an output table. The OUT= option names the output table. *Statistic*(*input-variable*)=*output-variable* can be specified in the OUTPUT statement.

*** Lesson 6: Export results

-

Summary

SAS procedures empower us to analyse data and generate reports whilst the OUTPUT DELIVERY SYSTEM enables us to create attractive reports in a format the user can consume.

In Exporting Results you learn to export SAS tables and results to Excel, Microsoft Word, and PDF files.

Learning Objectives:

Export a SAS DATA SET to a variety of file formats outside SAS.

1. Use the OUTPUT DELIVERY SYSTEM to export reports from SAS.
2. Know additional options can be added to ODS STATEMENTS to change the look of the report exported.

Now that you've analyzed your data and created accurate, meaningful reports, you're ready to share your insights with others. But suppose not everyone who needs to view your results uses SAS.

Fortunately, SAS enables you to export data and reports in formats that are easy to view outside of SAS. In Exporting Results you learn to

export SAS tables and results to Excel, Microsoft Word, and PDF files.

- If you do **not** have write access to a folder on the server, copy and paste the following %LET statement that creates the **outpath** macro variable and stores the path to a location where you can create temporary files. Note that you will be able to run code to create output but in some cases you will not be able to view the output.

```
%let outpath = %sysfunc(pathname(work));
```

Se excel, text fil: right-click storm_final.csv, and select View File as Text.

- SAS Studio: In your **output** folder, right-click **storm_final.csv**, and select **View File as Text**.
- SAS Enterprise Guide: You can only view this file if you set **outpath** to a folder where you can write files. In the Servers pane, navigate to the folder location you specified for **outpath**. Right-click **storm_final.csv** and select **Open**.

libref= myxl er reference til filen cars.xlsx

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

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

libname myxl clear;
```

```
************************************************************;
*  Exporting Data to an Excel Workbook                     *;
************************************************************;
*  Syntax and Example                                      *;
*                                                          *;
*    LIBNAME libref XLSX "path/file.xlsx";                 *;
*      <use libref for output table(s)>                    *;
*    LIBNAME libref CLEAR;                                 *;
************************************************************;
libname myxl xlsx "&outpath/cars-1109.xlsx";

data myxl.asiacars;
   set sashelp.cars;
   where origin='Asia';
run;
*mon ikke der opretttes sheet asiacars i cars-1109.xlsx   ?!

libname myxl clear;
```

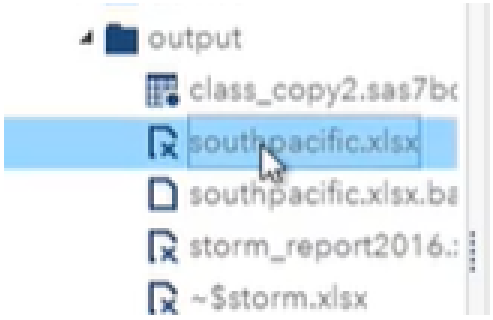
Der oprettes excelfil cars-1109.xlsx

Eksempel på output fil og sheets

```
*  Exporting Data to an Excel Workbook          *;
*************************************************;
*  Syntax and Example                           *;
*                                               *;
*    LIBNAME libref XLSX "path/file.xlsx";      *;
*     <use libref for output table(s)>          *;
*    LIBNAME libref CLEAR;                      *;
*************************************************;

libname xlout xlsx "&outpath/southpacific.xlsx";
data xlout.South_Pacific;
    set pg1.storm_final;
    where Basin="SP";
run;

proc means data=pg1.storm_final noprint maxdec=1;
    where Basin="SP";
    var MaxWindKM;
    class Season;
    ways 1;
    output out=xlout.Season_Stats n=Count mean=AvgMaxWindKM max=StrongestWindKM;
run;

libname xlout clear;
```

*xlout er ref til filen southpacifi. I første udskrift angives output til sheet=South_Pacific i southpacific.xlsx filen, med data xlout= South_Pacific. I nr 2. Nederst oprettes sheet Season_Stats i denne xlsx-fil.

Libename laver reference til work-book, excelark, laver en i mappe beregnet til filer af xlsx



to tabs/faner i excelarket

ODS=output delivery system kan lave csv filer mv

```
ods csvall file="&outpath/cars.csv";
proc print data=sashelp.cars noobs;
     var Make Model Type MSRP MPG_City MPG_Highway;
     format MSRP dollar8.;
run;
ods csvall close;
```

```
"Make","Model","Type","MSRP","MPG_City","MPG_Highway"
"Acura","MDX","SUV","$36,945",17,23
"Acura","RSX Type S 2dr","Sedan","$23,820",24,31
"Acura","TSX 4dr","Sedan","$26,990",22,29
"Acura","TL 4dr","Sedan","$33,195",20,28
"Acura","3.5 RL 4dr","Sedan","$43,755",18,24
"Acura","3.5 RL w/Navigation 4dr","Sedan","$46,100",18,24
```

MSRP formateres som $værdi

```
************************************************************;
*  Exporting Results to Excel                              *;
************************************************************;
*  Syntax and Example                                      *;
*                                                          *;
*   ODS EXCEL FILE="filename.xlsx" <STYLE=style>           *;
*           <OPTIONS (SHEET_NAME='label')>;                *;
*      /* SAS code that produces output */                 *;
*   ODS EXCEL OPTIONS (SHEET_NAME='label');                *;
*      /* SAS code that produces output */                 *;
*   ODS EXCEL CLOSE;                                       *;
************************************************************;

************************************************************;
*  Demo                                                    *;
*  1) Add an ODS statement to create an Excel file named   *;
*     wind.xlsx in the output folder of the course files.  *;
*     Close the excel desination at the end of the         *;
*     program. Highlight the demo program and run the      *;
*     selected code.                                       *;
*  2) Open the Excel file.                                 *;
*     * SAS Studio: Navigate to the output folder in the   *;
*       Files and Folders section of the navigation        *;
*       pane. Select wind.xlsx and click Download .        *;
*     * Enterprise Guide: Click Results and select the     *;
*       Excel file. Right-click and select Open.           *;
*  3) Examine the Excel workbook. Notice the light blue    *;
```

```
*    background in the results generated by the default  *;
*    style. Also notice the default spreadsheet names.   *;
*    Close the Excel file.                               *;
*  4) Examine the available style options.               *;
*    * SAS Studio: Submit the following program:         *;
*      proc template;                                    *;
*        list styles;                                    *;
*      run;                                              *;
*    * Enterprise Guide: Select Tools -> Style Manager.  *;
*  5) Change the style by adding the STYLE=SASDOCPRINTER  *;
*    option in the first ODS statement.                  *;
*  6) Use the SHEET_NAME= option on the first ODS EXCEL   *;
*    statement to name the first worksheet Wind Stats    *;
*    Add another ODS EXCEL statement with the SHEET_NAME=*;
*    option before the second TITLE statement and SGPLOT *;
*    step. Name the second worksheet Wind Distribution *;
*    Highlight the demo program and run the selected     *;
*    code. Open the Excel file to view the results.      *;
************************************************************;

*Add ODS statement;

title "Wind Statistics by Basin";
ods noproctitle;
proc means data=pg1.storm_final min mean median max maxdec=0;   *pg1.storm_final er inputfilen
   class BasinName;
   var MaxWindMPH;
run;

title "Distribution of Maximum Wind";
proc sgplot data=pg1.storm_final;
   histogram MaxWindMPH;
   density MaxWindMPH;
run;
title;
ods proctitle;
*Add ODS statement;   med ods statement, der laves excelark med to faner, én til hver proc
```

```
15 ods excel file="&outpath/wind.xlsx";
16 title "Wind Statistics by Basin";
17 ods noproctitle;
18 proc means data=pg1.storm_final min mean median max maxdec=0;
19     class BasinName;
20     var MaxWindMPH;
21 run;
22
23 title "Distribution of Maximum Wind";
24 proc sgplot data=pg1.storm_final;
25     histogram MaxWindMPH;
26     density MaxWindMPH;
27 run;
28 title;
29 ods proctitle;

31 ods excel close;
```
S:/Workshop/demos/p106d02.sas

proc template list styles; run; viser style-templates i systemet

```
proc template;
    list styles;
run;
```

Styling, sheet-name

```
*Add ODS statement;
ods excel file="&outpath/wind.xlsx" style=sasdocprinter
          options(sheet_name='Wind Stats');
title "Wind Statistics by Basin";
ods noproctitle;
proc means data=pg1.storm_final min mean median max maxdec=0;
    class BasinName;
    var MaxWindMPH;
run;

ods excel options(sheet_name='Wind Distribution');
title "Distribution of Maximum Wind";
proc sgplot data=pg1.storm_final;
    histogram MaxWindMPH;
    density MaxWindMPH;
run;
title;
ods proctitle;
*Add ODS statement;
ods excel close;
```

output er givet ved ODS-stmt.

med option:

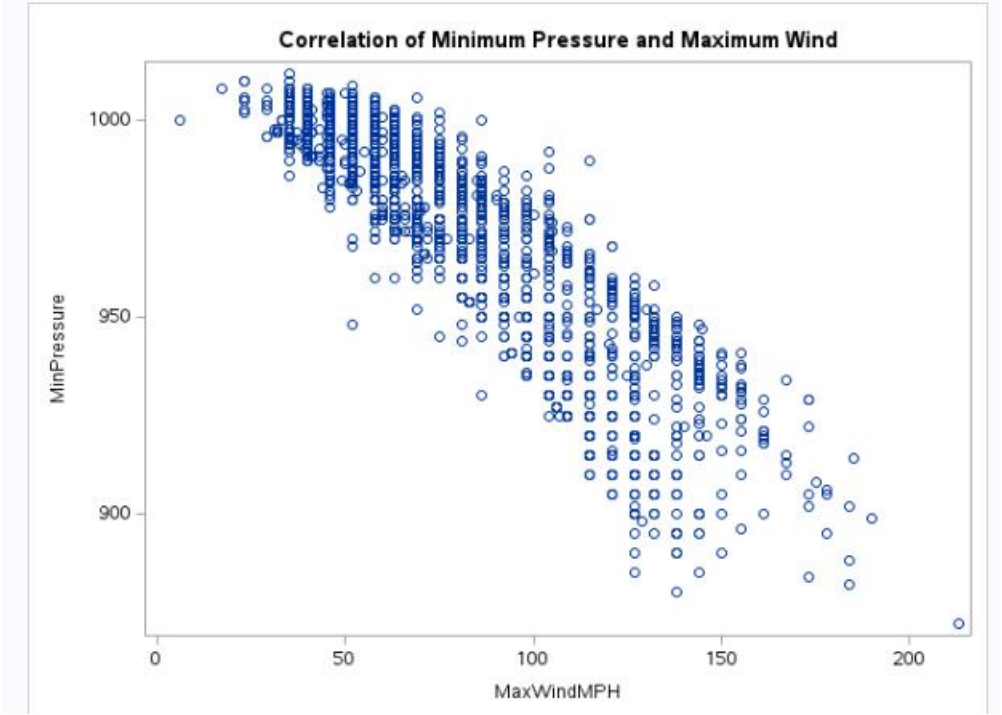
```
ods excel file="&outpath/pressure.xlsx" style=analysis;
```

```
title "Minimum Pressure Statistics by Basin";
ods noproctitle;
proc means data=pg1.storm_final mean median min maxdec=0;
    class BasinName;
    var MinPressure;
run;

title "Correlation of Minimum Pressure and Maximum Wind";
proc sgscatter data=pg1.storm_final;
      plot minpressure*maxwindmph;
run;
title;

ods proctitle;
ods excel close;
```

Result:

Minimum Pressure Statistics by Basin

Analysis Variable : MinPressure				
BasinName	N Obs	Mean	Median	Minimum
East Pacific	675	979	987	872
North Atlantic	478	980	988	882
North Indian	60	983	989	920
South Indian	594	965	974	895
South Pacific	359	967	975	884
West Pacific	926	962	965	880



Udskrivning til pp og rtf

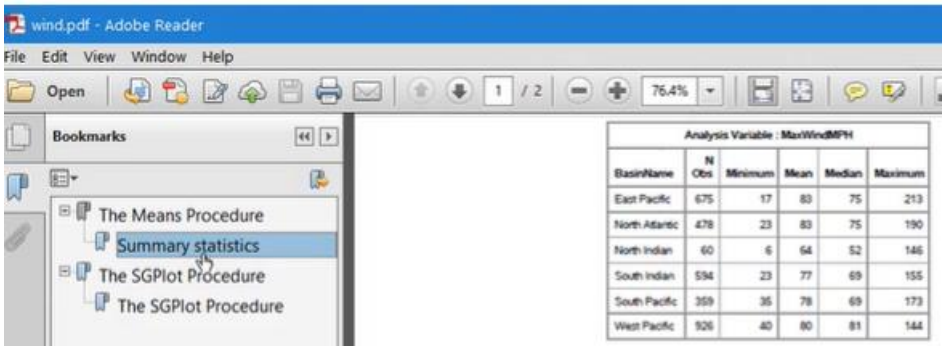
```
ODS POWERPOINT FILE="filename.pptx" STYLE=style;
/* SAS code that produces output */
ODS POWERPOINT CLOSE;
```

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


```
ods rtf file="&outpath/pressure.rtf" style=sapphire
        startpage=no;

title "Minimum Pressure Statistics by Basin";
ods noproctitle;
...

ods rtf close;
```

for pdf:

```
ods pdf file="&outpath/wind.pdf";
ods noproctitle;

title "Wind Statistics by Basin";
proc means data=pg1.storm_final min mean median max maxdec=0;
    class BasinName;
    var MaxWindMPH;
run;

title "Distribution of Maximum Wind";
proc sgplot data=pg1.storm_final;
    histogram MaxWindMPH;
    density MaxWindMPH;
run;
title;

ods pdf close;
```



1. Open **p106p03.sas** from the **practices** folder. Run the program and examine the output. The program produces a table and a map for North Atlantic region storms in the 2016 season. **Note**: You must have SAS 9.4M5 to run this code.

2. Modify the program to produce a PDF file named **StormSummary.pdf** in the output folder in the course files. Set the output style to **Journal**. Specify **not** to generate bookmarks in the file.

3. Use SAS Help to find a SAS system option that changes the page layout to landscape.

4. Use SAS Help to learn about the ODS LAYOUT GRIDDED statement as a way that you can control the layout of multiple result objects. Force the results to be arranged in one row and two columns.

5. Reset the system option at the end of the program so that future results have a portrait layout.

6. Submit the program. Open the **StormSummary.pdf** file to confirm the results.
   - SAS Studio: In the Files and Folders panel, navigate to **EPG1V2 > output**. Select **StormSummary.pdf** and click **Download**.
   - Enterprise Guide: Click the **Results - PDF** tab and click **Download**.

     Note: SAS Studio generates a warning in the log because the wrapper code is creating an RTF file behind the scenes. ODS LAYOUT is not supported in RTF. The warning can be ignored because it doesn't impact the PDF results.

```
ods pdf file="&outpath/StormSummary.PDF" style=Journal
nobookmarkgen;
```

```
title1 "2016 Northern Atlantic Storms";
*tegner kort i pdf-filen
proc sgmap plotdata=pg1.storm_final;
    *openstreetmap;
    esrimap
url='http://services.arcgisonline.com/arcgis/rest/services/Wo
rld_Physical_Map';
    bubble x=lon y=lat size=maxwindmph / datalabel=name
datalabelattrs=(color=red size=8);
    where Basin='NA' and Season=2016;
    keylegend 'wind';
run;

proc print data=pg1.storm_final noobs;
    var name StartDate MaxWindMPH StormLength;
    where Basin="NA" and Season=2016;
    format StartDate monyy7.;
run;

ods pdf close;
```

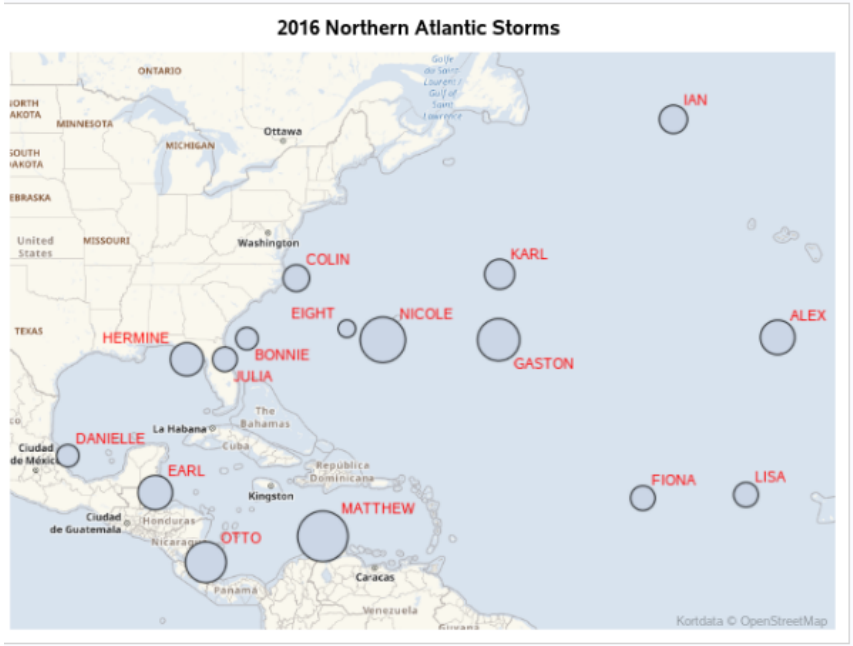
Use SAS Help to learn about the ODS LAYOUT GRIDDED statement as a way that you can control the layout of multiple result objects. Force the results to be arranged in one row and two columns.

Solution:

```
options orientation=landscape;
ods pdf file="&outpath/StormSummary.PDF" style=Journal
nobookmarkgen;

title1 "2016 Northern Atlantic Storms";

ods layout gridded columns=2 rows=1;
ods region;

proc sgmap plotdata=pg1.storm_final;
    *openstreetmap;
    esrimap
url='http://services.arcgisonline.com/arcgis/rest/services/World_P
hysical_Map';
    bubble x=lon y=lat size=maxwindmph / datalabel=name
datalabelattrs=(color=red size=8);
```

```
    where Basin='NA' and Season=2016;
    keylegend 'wind';
run;

ods region;
proc print data=pg1.storm_final noobs;
    var name StartDate MaxWindMPH StormLength;
    where Basin="NA" and Season=2016;
    format StartDate monyy7.;
run;

ods layout end;
ods pdf close;
```

2016 Northern Atlantic Storms



Name	StartDate	MaxWindMPH	StormLength
ALEX	JAN2016	86	10
BONNIE	MAY2016	46	13
COLIN	JUN2016	58	3
DANIELLE	JUN2016	46	3
EARL	AUG2016	86	4
FIONA	AUG2016	52	8
GASTON	AUG2016	121	13
EIGHT	AUG2016	35	5
HERMINE	AUG2016	81	11
IAN	SEP2016	63	5
KARL	SEP2016	69	14
JULIA	SEP2016	52	8
LISA	SEP2016	52	7
MATTHEW	SEP2016	167	12
NICOLE	OCT2016	138	15
OTTO	NOV2016	115	8

Ser I log at der ikke kan oprettes forbindelser til esri,

Openstreetmap kan bruges

Learning More

What If You Want to...

...look up additional options for exporting data or results?

...learn to create and customize reports with ODS?

** Summary of Lesson 6: Exporting Results

Exporting Data

- PROC EXPORT can export a SAS table to a variety of file formats outside SAS.

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

-
- The XLSX engine requires a license for SAS/ACCESS to PC Files.
- The XLSX engine can read and write data in Excel files.
- To write data to a new or existing Excel workbook, use the LIBNAME statement to assign a libref pointing to the Excel file. Use the libref when naming output tables. The table name is the worksheet label in the Excel file.

```
OPTIONS VALIDVARNAME=V7;
```

-

```
LIBNAME libref XLSX "path/file.xlsx";
```

-
- Use libref som shortname til sheet for output table(s).

```
LIBNAME libref CLEAR;
```

Exporting Reports

- The SAS Output Delivery System (ODS) is programmable and flexible, making it simple to automate the entire process of exporting reports to other formats.

```
ODS <destination> <destination-specifications>;
```

```
/* SAS code that produces output */

ODS <destination> CLOSE;
```

-
- Selected destinations:
    - CSVALL
    - PowerPoint
    - RTF
    - PDF
- Exporting results to Excel, se ”Eksempel på output til fil.. ” § ovenfor

```
ODS EXCEL FILE="filename.xlsx" STYLE=style
                OPTIONS(SHEET_NAME='label');

/* SAS code that produces output */

ODS EXCEL OPTIONS(SHEET_NAME='label');

/* SAS code that produces output */

ODS EXCEL CLOSE;
```

-
- The ODS EXCEL destination creates an .XLSX file.
- By default, each procedure output is written to a separate worksheet with a default worksheet name. The default style is also applied.
- Use the STYLE= option on the ODS EXCEL statement to apply a different style.
- Use the OPTIONS(SHEET_NAME=’label’) on the ODS EXCEL statement to provide a custom label for each worksheet.
- Exporting results to PDF:

```
ODS PDF FILE="filename.pdf" STYLE=style
            STARTPAGE=NO PDFTOC=1;
ODS PROCLABEL"label";

/* SAS code that produces output */

ODS PDF CLOSE;
```

-
- The ODS PDF destination creates a .PDF file.
- The PDFTOC=n option controls the level of the expansion of the table of contents in PDF documents.
- The ODS PROCLABEL statement enables you to change a procedure label.

Quiz

Quiz: Lesson 6

Your score: 60%

Your score indicates that you would benefit from reviewing topics in this lesson. Check the feedback below and when you are ready, take the quiz again.

1. Which statement is false concerning the options for the PROC EXPORT statement?

   a. The DATA= option identifies the input SAS table.

   b. The REPLACE option specifies to overwrite an existing file.

   c. The DBMS= option specifies the database identifier for the type of file being created.

   d. The OUT= option specifies the path and file name of the external data file being created.

Your answer: d
Correct answer: d

2. The OUTFILE= (not OUT=) option specifies the path and filename of the external data file being created.

3.

2. Which PROC EXPORT step contains valid syntax?

a.
```
proc export outfile="c:\temp\cars.txt" tab
             data=sashelp.cars replace; run;
```

b.
```
proc export data=sashelp.cars dbms=csv
             outfile="c:\temp\cars.csv"; run;
```

c.
```
proc export data=sashelp.class; dbms=csv;
             outfile="c:\temp\cars.csv"; run;
```

d.
```
proc export dbms=tab data=sashelp.cars replace=yes
             outfile="c:\temp\cars.txt"; run;
```

Your answer: a
Correct answer: b

3. DATA=, DBMS=, and OUTFILE= are valid PROC EXPORT options. For answer a, DBMS= is missing in front of TAB. For answer c, there shouldn't be semicolons after each option. For answer d, =YES is not valid after REPLACE.
4.
5.

---

3. What does the following program create?

```
4. libname sales xlsx 'c:\mydata\midyear.xlsx';
5.
6. data sales.q1_2018;
7.     set sasdata.qtr1_2018;
8. run;
9. data sales.q2_2018;
10.     set sasdata.qtr2_2018;
   run;
```

a. two SAS tables: **sales.q1_2018** and **sales.q2_2018**

b. two Excel workbooks: **sales.q1_2018** and **sales.q2_2018**

c. two worksheets in the Excel workbook: **midyear: q1_2018** and **q2_2018**

d. two worksheets in the Excel workbook: **sales: q1_2018** and **q2_2018**

Your answer: c
Correct answer: c

The LIBNAME statement specifies the Excel workbook **midyear**. The DATA statements create the worksheets **q1_2018** and **q2_2018** within the workbook **midyear**. The library reference of **sales** is what links the LIBNAME and DATA statements.

---

4. Which statement disassociates the **sales** libref?

```
5. libname sales xlsx 'c:\mydata\midyear.xlsx';
```

a. `libname sales end;`

b. `libname sales clear;`

c. `libname sales close;`

d. `libname sales disassociate;`

Your answer: b
Correct answer: b

The CLEAR option in the LIBNAME statement disassociates one or more currently assigned librefs.

5. What type of output file does this program create?

```
6. libname mylib xlsx "s:/workshop/output/test.xlsx";
7.
8. data class_list;
9.     set sashelp.class;
   run;
```

a. SAS table

b. delimited file

c. Microsoft Excel XLS file

d. Microsoft Excel XLSX file

Your answer: d
Correct answer: a

The DATA statement references **class_list**. A libref is not specified, so the **work** library is assumed. **Work.class_list** is a temporary SAS table.

6. Which of these programs creates a Microsoft Excel file?

   a.
   ```
   ods excel file="s:/workshop/output/class.xlsx";
   proc print data=sashelp.class;
   run;
   ods excel close;
   ```

   b.
   ```
   libname mylib xlsx "s:/workshop/output/class.xlsx";
   data mylib.class_list;
       set sashelp.class;
   run;
   ```

   c. both

   d. neither

7.
Your answer: d
Correct answer: c

Both ODS EXCEL and the LIBNAME statement with the XLSX engine create Excel workbooks.

8.

7. Which of the following is not a valid ODS statement?

   a.
   ```
   ods csvall file='c:\temp\myfile.csv';
   ```

   b.
   ```
   ods pdf file='c:\temp\myfile.pdf';
   ```

   c.
   ```
   ods powerpoint file='c:\temp\myfile.ppt';
   ```

   d.
   ```
   ods word file='c:\temp\myfile.doc';
   ```

8.

Your answer: d
Correct answer: d

9. WORD is not a valid destination for the ODS statement. The RTF destination creates a file that can be opened by word processors: **ods rtf file='c:\temp\myfile.rtf';**
10.

11.

---

8. Which statement needs to be added to the end of this program?

```
9. ods pdf file='c:\temp\myfile.pdf';
10.
11. proc print data=sashelp.class;
   run;
```

a. `ods clear;`

b. `ods close;`

c. `ods pdf clear;`

d. `ods pdf close;`

Your answer: c
Correct answer: d

The CLOSE argument closes the destination and the file that is associated with it.

---

9. Which statement is false concerning the options for the ODS statement?

a. The STYLE= option names the desired font.

b. The FILE= option specifies the output file to create.

c. The STARTPAGE= option controls the behavior of page breaks.

d. The PDFTOC= option controls the level of the expansion of the table of contents in PDF documents.

10.

Your answer: a
Correct answer: a

11. The STYLE= option names the style to use in the output file. The style controls visual aspects such as colors and fonts.
12.

10. Which statement contains valid syntax for specifying a worksheet name?

a. `ods excel sheet_name='Males';`

b. `ods excel (sheet_name='Males');`

c. `ods excel option(sheet_name='Males');`

d. `ods excel options(sheet_name='Males');`

11.

Your answer: d
Correct answer: d

12. SHEET_NAME= is a sub-option that goes in a set of parentheses for the OPTIONS option.
13.

Close

*** Lesson 7: Using SQL

```
proc sql;
select * from pg1.storm_basincodes as sb;


quit;


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

```
NOTE: PROCEDURE SQL used (Total process time):
      real time           1.48 seconds
      user cpu time       1.48 seconds
      system cpu time     0.01 seconds
      memory              6175.43k
      OS Memory           28204.00k
      Timestamp           18/11/2023 07.11.06 e.m.
      Step Count                        66  Switch Count  8
      Page Faults                       0
      Page Reclaims                     786
      Page Swaps                        0
      Voluntary Context Switches        47
      Involuntary Context Switches      3
      Block Input Operations            0
      Block Output Operations           960
```

Season	Name	Basin	Type	MaxWindMPH	MinPressure	StartDate	EndDate	Hem_NS	Hem_EW	Lat	Lon
1980		na	TS	35	.	17JUL1980	18NOV1980	N	W	25.7	-91.2

proc sql;

select * from pg1.storm_basincodes as sb;

quit;

Basin	BasinName
NA	North Atlantic
WP	West Pacific
EP	East Pacific
SP	South Pacific
NI	North Indian
SI	South Indian

The DATA step and SQL each have their own strengths, and it is helpful to know how to use both for processing data. There are some situations where one might be easier or more efficient than the other. This following table highlights some of the strengths of each.

DATA Step	PROC SQL
provides more control of reading, writing, and manipulating data	ANSI-standard language used by most databases
can create multiple tables in one step	code can be more streamlined
includes looping and array processing	can manipulate, summarize, and sort data in one step

Using Structured Query Language (SQL) in SAS summery

- PROC SQL creates a report by default.
- The SELECT statement describes the query. List columns to include in the results after SELECT, separated by commas.
- The FROM clause lists the input table(s).

- The ORDER BY clause arranges rows based on the columns listed. The default order is ascending. Use DESC after a column name to reverse the sort sequence.
- PROC SQL ends with a QUIT statement.

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

-
- Subsetting data:

```
WHERE expression
```

-
- Sorting data:

```
ORDER BY col-name <DESC>
```

-
- Creating output data:

```
CREATE TABLE table-name AS
```

-
- Deleting data:

```
DROP TABLE table-name;
```

Joining Tables Using SQL in SAS

- An SQL inner join combines matching rows between two tables.
- The two tables to be joined are listed in the FROM clause.

```
FROM from table1 INNER JOIN table2
ON table1.column = table2.column
```

-
- Assign an alias (or nickname) to a table in the FROM clause by adding the keyword AS and the alias of your choice. Then you can use the alias in place of the full table name to qualify columns in the other clauses of a query.

FROM *table1* **AS** *alias1, table2* **AS** *alias2*

-

Eksempel med title, inner join

```
title "Storme i bassiner";
title2 "Storme i bassiner-undertitel";
proc sql;
select Season, Name, ss.Basin, sb.BasinName, MaxWindMPH
   from pg1.storm_summary as ss
   inner join pg1.storm_basincodes as sb
                                 on  upcase(ss.basin)=sb.basin
   order by Season desc, Name;
quit;
```

Quiz: Lesson 7

Your score: 90%

Congratulations! Your score indicates that you have mastered the topics in this lesson. You can review the feedback and when you're finished, exit the lesson.

1. What is the correct order of the following four clauses?

a.
```
from ...
     select ...
      where ...
       order by ...
```

b.
```
order by ...
     from ...
      select ...
       where ...
```

c.
```
select ...
     where ...
```

```
order by ...
 from ...
```

d.
```
select ...
  from ...
 where ...
  order by ...
```

2.

Your answer: d
Correct answer: d

First is SELECT, second is FROM, third is WHERE, and fourth is ORDER BY.

3.

---

2. Which of the following is false regarding the SQL procedure?

   a. Column names are separated with commas.

   b. The procedure ends with a QUIT statement.

   c. Formats can be specified in the FROM clause.

   d. The SELECT and FROM clauses are required in the SELECT statement.

3.

Your answer: c
Correct answer: c

Formats are specified in the SELECT clause after the column name.

4.

---

3. Which syntax is valid for creating a computed column in the SELECT clause?

   a. `Ratio = Height/Weight`

   b. `Ratio as Height/Weight`

   c. `Height/Weight = Ratio`

   d. `Height/Weight as Ratio`

4.

Your answer: b
Correct answer: d

Computed columns are created by specifying the expression, the keyword AS, and the column name, in that order.

5.

4. The SELECT statement creates a report. Which clause can be added before the SELECT clause to create a table?

a. `create work.new =`

b. `create work.new table`

c. `create table work.new as`

d. `create table=work.new as`

5.

Your answer: c
Correct answer: c

To create a table, add the CREATE TABLE *NEW-TABLE-NAME* AS clause before the SELECT clause.

5. Which SELECT statement produces the given output?

Name	Height
Thomas	57.5
Joyce	51.3

a.
```
select Name Height
      from sashelp.class
      where age=12
      order by Height;
```

b.
```
select Name, Height
      from sashelp.class
      where age=12
      order by Height desc;
```

c.
```
select Name Height
      from sashelp.class
      where age=12
      order by desc Height;
```

d.
```
select Name, Height
      from sashelp.class
```

```
where age=12
order by desc Height;
```

7.

Your answer: b
Correct answer: b

In the SELECT clause, column names are separated with commas. In the ORDER BY clause, DESC goes after the column name. ASC is the default sort order.

8.

---

6. Which SQL statement can delete tables?

   a. DROP

   b. VOID

   c. DELETE

   d. SELECT

7.

Your answer: a
Correct answer: a

The DROP TABLE statement deletes tables.

8.

---

7. If an inner join is performed on the following tables based on the **ID** and **IDNO** columns, how many rows will be in the PROC SQL report?

Name	ID
Jack	111
Mary	333
Jane	555

IDNO	Salary
111	75000
222	83000

333	82000

8.
9.

a. one

b. two

c. three

d. four

10.
Your answer: b
Correct answer: b

An inner join gives matches only. Jack (111) and Mary (222) are the matches in this example.

11.

8. Which statement has the correct syntax for performing an inner join?

a.
```
select ID, Name, Salary
       from one join two
       on ID=IDNO;
```

b.
```
select ID, Name, Salary
       from one join two
       where ID=IDNO;
```

c.
```
select ID, Name, Salary
       from one inner join two
       on ID=IDNO;
```

d.
```
select ID, Name, Salary
       from one inner join two
       where ID=IDNO;
```

9.
Your answer: c
Correct answer: c

To perform an inner join, specify INNER JOIN between two table names and specify the matching condition in an ON clause (not a WHERE clause).

10.

9. Which ON clause has valid qualifying syntax?

a.
```
from empsau inner join phonec
      on e.empid=p.empid;
```

b.
```
from empsau inner join phonec
      on left.empid=right.empid;
```

c.
```
from empsau inner join phonec
      on first.empid=second.empid;
```

d.
```
from empsau inner join phonec
      on empsau.empid=phonec.empid;
```

10.
Your answer: d
Correct answer: d

To qualify a column, put the table name and a period before the column name. Qualifying is needed when a column is in multiple tables. The name **empsau.empid** refers to the **empid** column in the **empsau** table.

10. Which FROM clause properly creates aliases?

a.
```
from empsau=e inner join phonec=p
```

b.
```
from empsau(e) inner join phonec(p)
```

c.
```
from empsau as e inner join phonec as p
```

d.
```
from empsau of e inner join phonec of p
```

11.
Your answer: c
Correct answer: c

To create an alias in the FROM clause, put the word AS and the alias after the table name. The word AS is optional. The alias can be used when qualifying a column.