MODUL 7
SAS® DATA PREPARATION APPLICATION

THEME DESCRIPTION

Students are able to carry out Data Preparation which is an important process for analytical and predictive modeling including the process of cleaning, transforming, and managing data using SAS Data Preparation application features that do not require coding-less programming skills.

WEEKLY LEARNING OUTCOMES (SUB-LESSONS)

CLO-2-Sub-CLO-7, Able to properly implement the data curation process, part of the role of data science including the component aspects of the computing environment.

Through the following learning steps:

1. Referencing external files
2. Creating indicators for the first and last observation in a by group
3. Transposing the rows and columns in a table
4. Statistical and mathematical data transformations

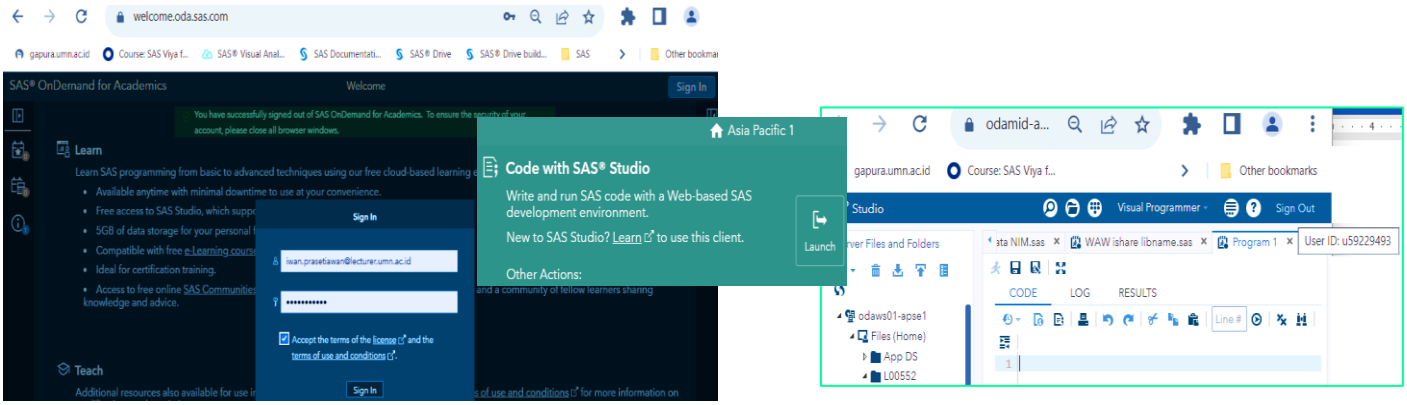
PRACTICUM SUPPORT

a. Windows Operating System
b. (any) Browser Application

PRACTICUM STEPS

Login to SAS Studio

Start, https://welcome.oda.sas.com



1. Referencing external files

- While a SAS library is a pointer to a data storage location that has data stored in tables, SAS programmers also use statements and **data step options to work directly with raw external files**.
- SAS provides programmers with the flexibility of **referencing files** either **directly or indirectly**.

Directly referencing external files

- Data step code has two statements for with external files.
- The infile statement uses the external file for reading or for input, and the file statement is used to write information into an external file for storage.
- Both the infile and file statements refer directly to the external file using the directory location and name of the file.

- For example, here is some pseudo-SAS code for using an external file named **rawfile.txt** as input in a data step:

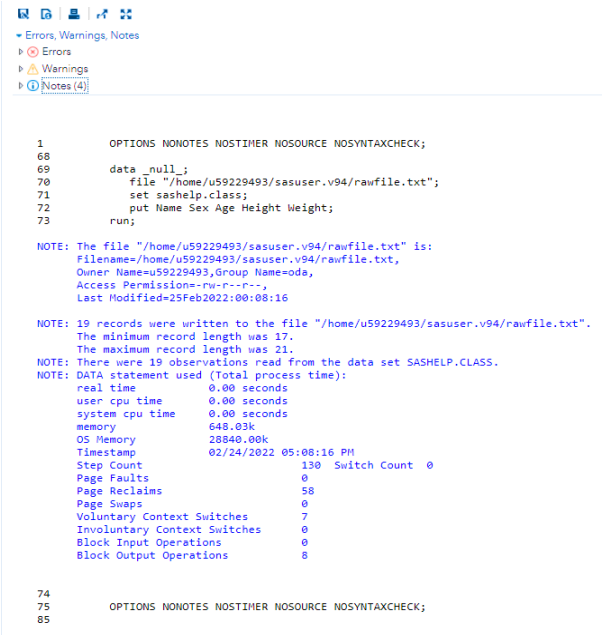
a. SASHELP library has a class table consisting of 19 students inside with the attributes of Name, Sex, Age, Height and Weight of the students.
Step to seeing the table: Library – SASHELP – Class then clicked twice to get inside the table.

b. Use below code to get copy the class table from SASHELP library into your CAS home directory

```
/* copy text file from SASHELP Library for Class table into your home CAS library*/
data _null_;
   file "/home/u59229493/sasuser.v94/rawfile.txt";
   set sashelp.class;
   put Name Sex Age Height Weight;
run;
```

Your home UID

c. Submit this code by selecting the running man icon and check it out what the log says:



d. Not only will you see this information in the LOG, but also the raw.txt file will be seen. Select Server Files and Folders in the left of SAS Studio and expand My Folders and then sasuser.v94.

Indirectly referencing external files

- SAS also provides a filename statement that similarly to the libname statement.
- While the libname statement creates a pointer or libref to a location that stores data in tables, the filename statement assigns a pointer or fileref to an external file.
- This indirect reference is convenient to make one reference to a file within a program, which may be used multiple times and therefore can easily be updated in the future by changing one line instead of multiple direct references.

d. Here is what a filename statement looks like in code:

```
filename myfile "/home/u59229493/sasuser.v94/rawfile.txt";
```

e. Now, a programmer can use this fileref in combination with the direct infile and file statements. Type the following code in a new SAS Studio program section:

```
filename myfile "/home/u59229493/sasuser.v94/rawfile.txt";
data work.myraw;
   infile myfile;
   input Name $ Sex $ Age Height Weight;
run;
```

You could get your results and **print into pdf file as figure output A by format file name as 'IS-429 Lab Week#7 A results"**.

2. Creating indicators for the first and last observation in a by group

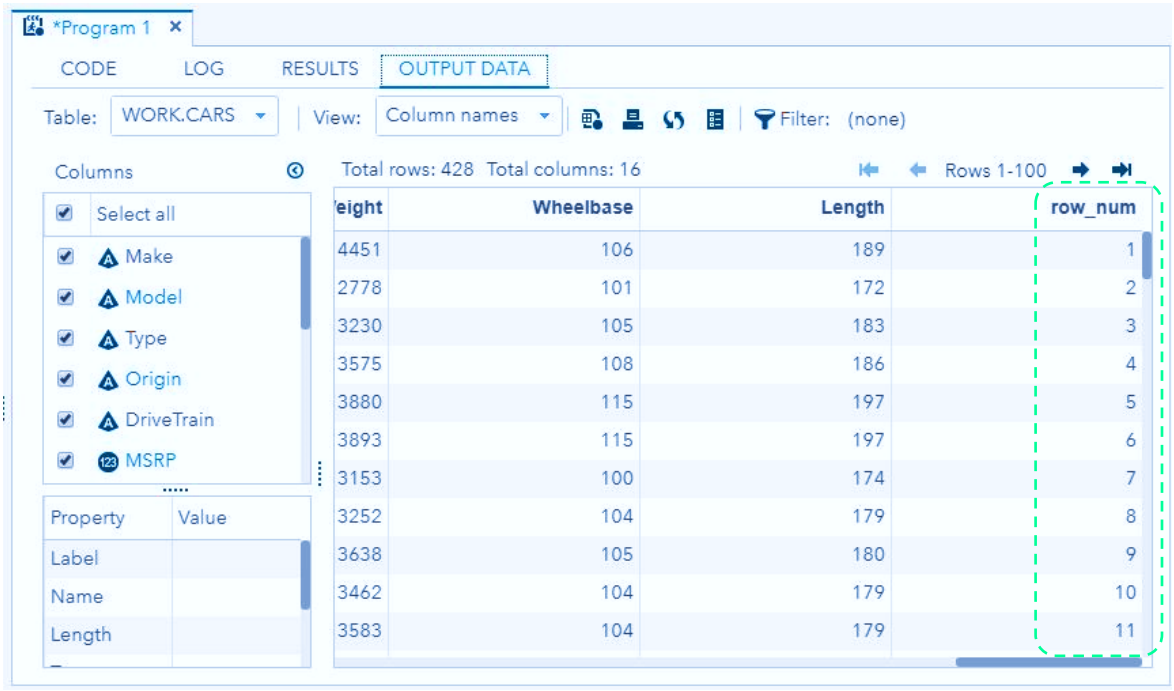
- It is very convenient to be able to easily the first and last observation with a group of observations.
- For example, if you need to identify a sequence of events that some entity performs, but the data you receive from various sources doesn't happen to provide you with the in this type of format,

a. It can be difficult and time-consuming to rearrange this data into the proper sequence. One simple way to use this automatic variable is to add a row number variable to a table that doesn't already have one.

b. Type the following code in a program section in SAS Studio and submit it:

```
data work.cars;
   set sashelp.cars;
   row_num=_N_;
run;
```

c. In the OUTPUT DATA table preview window, scroll all the way to the right within the window and you will see the new column named run_num. It contains the value that corresponds to the row number within the new work.cars table (1 through 428):



- Likewise, the SAS data step identifies the first observation and last observation of a by group by creating temporary variables for each variable listed in the by statement.
- These temporary variables are named **FIRST.variablename** and **LAST.variablename**. Similar to _N_, these temporary variables can be used to control program logic within the execution of the data step.
- The values of **FIRST.variablename** and **LAST.variablename** indicates **whether** an observation with that **by group is one of the following positions**:
  - The first one in a by group
  - The last one in a by group
  - Neither the first nor the last one in a by group
  - Both first and last, as is the case when there is only one observation in a by group

  As a result users can take actions conditionally, based on whether they are processing the first or the last observation in a by group.

- The value of FIRST.variablename = 1 for the first row associated with the BY variable; otherwise, it has a value of ø.
- Likewise, the value of LAST.variablename = 1 for the last row associated with the by variable; otherwise it has a value ofø.

d. Type the following code in a program section in SAS Studio and submit it:

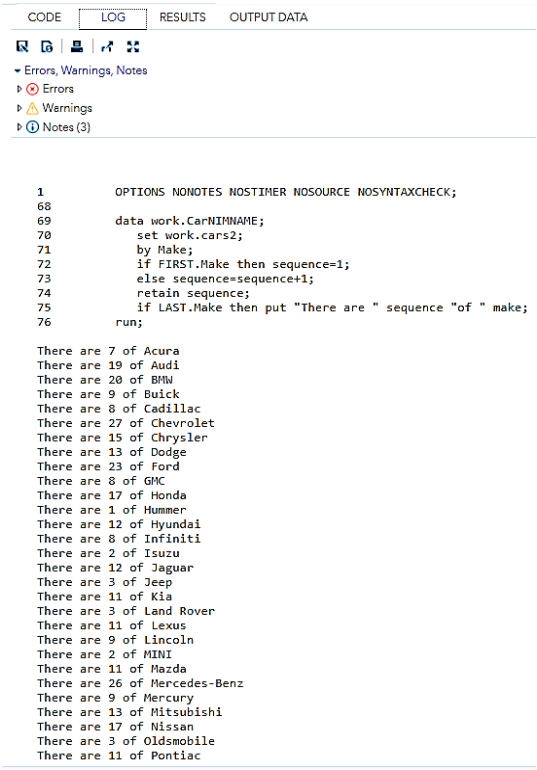
```
proc sort data=sashelp.cars out=work.cars2;
   by Make Type;
quit;

data work.cars3;
   set work.cars2;
   by Make Type;
   if FIRST.Make and FIRST.Type then output;
   if NOT FIRST.Make and FIRST.Type then output;
run;
```

- *You will have the result by no sequence number occurs.*
- *This code will result in a dataset, work.cars3, which is a single type of car from each make (or manufacture) instead of multiple types from the same make.*
- ***The first if statement** outputs the row corresponding to the car associated **with a specific make** and the **first type from that make**, while **the second if is responsible** for **outputting the first type** of each **other type associated** with that particular make*

e. To further illustrate the usefulness of these temporary variables, let's add a fake sequence number associated with each make from work.cars2:

```
data work.cars4;
   set work.cars2;
   by Make;
   if FIRST.Make then sequence=1;
   else sequence=sequence+1;
   retain sequence;
   if LAST.Make then put "There are " sequence "of " make;
run;
```

- Notice **the use of the retain statement** in order **to make sure your sequence variable is assigned the proper value for each iteration** through the dataset.
- This code also introduced **the put statement,** which **is a simple way to write messages or variable values out in the LOG**. In this case, the put statement writes out to the **LOG how make types of each make happen** to exist in the original **work.cars2** dataset.
- A SAS programmer can make use of the put statement as a simple way to help them debug the data step code while in the process of writing it:



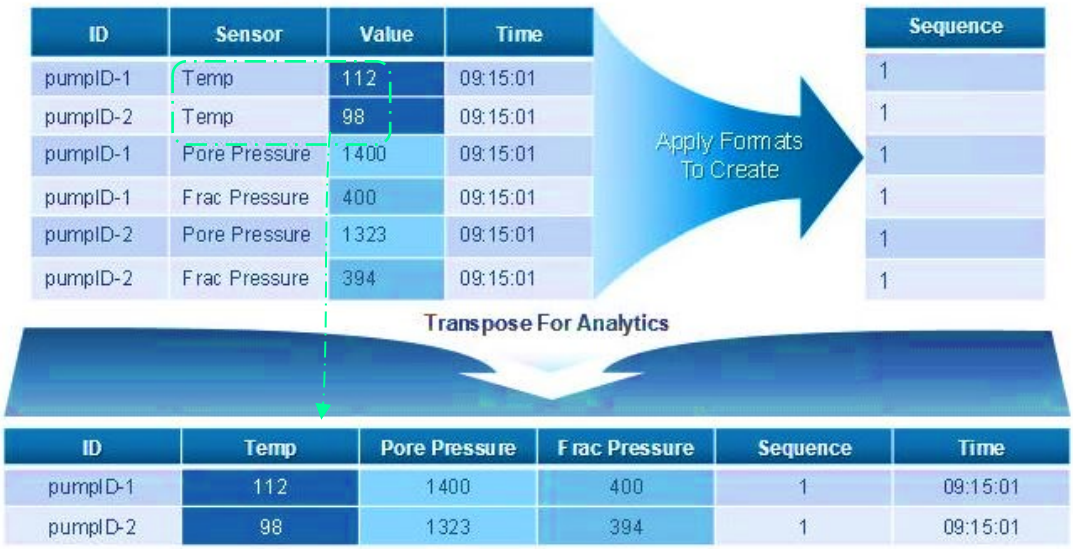
f. Now go to the OUTPUT DATA tab and once again scroll the table preview all the way to the right so that you can see the sequence variable added. See that it has been added correctly by restarting at 1 for each new make the program encounters in reading the work.cars2 dataset.

g. **Print it out your output result as shown on Output figure B with name file "IS-429 Lab Week#7B results"**

➲ **Save your code to "IS-429 Lab Week#7B NIM Name.sas"**

3. Transposing the rows and columns in a table

- **Transposing variables (columns) into observations (rows)** is a task when preparing data for analytics.
- In data step or other languages, it can take quite a lot of code as well as processing time to get the results
- An example of data transposition to prepare analytics can be illustrated as follows:



- Above example provided the actually combines the power of using formats along with transposition in order to get the data in the proper format for analytics to be applied.
- In this case, a custom format based on the time stamp associated with the various pumps would be created in order to align the data up in the sequence of events that are being captured at different times throughout the day.
- Notice how the resulting table at the bottom lines up the entity, in this case pumps, in such a way that all the data or attributes associated with individual pumps are in one row per pump per sequence number.
- Furthermore, PROC TRANSPOSE has been re-engineered over the years to take advantage of the hardware and memory of the environment it is running on, whether it is a Symmetrical Multiprocessing (SMP) based server or hardware, such as the Massively Parallel Processing (MPP) based database or Hadoop data storage platform.
- PROC TRANSPOSE restructures the of an input data table into an output data table by transposing selected columns (or variables) into rows (or observations). PROC TRANSPOSE can perform both simple transpositions of columns into rows or complex transpositions that involve transposing with by groups and/or renaming transposed columns.

a. Type the following code in a program section of SAS Studio and submit it:

```
/* Week#7C                                    */
proc transpose data=sashelp.failure
               out=work.failure_transposed(drop=_NAME_);
               id cause;
               by process day;
quit;
```

What the above codes do:

SAS Table Properties

General | Columns | Extended Attributes | Column Extended Attributes

Name:	FAILURE
Description:	MOS Capacitor Failure
Type:	Table
Location:	SASHELP.FAILURE
Rows:	70
Columns:	4
Date created:	Oct 25, 2018, 9:21:29 AM
Date modified:	Oct 25, 2018, 9:21:29 AM

General | Columns | Extended Attributes | Column Extended Attributes

Column Name	Type	Length	Format	Informat	Label
CAUSE	Char	14			Cause of Failure
PROCESS	Char	9			
COUNT	Numeric	8			
DAY	Numeric	8	DOWNAME.		

CODE | LOG | RESULTS

```
NOTE: There were 70 observations read from the data set SASHELP.FAILURE.
NOTE: The data set WORK.NIMNAME_TRANSPOSED has 10 observations and 9 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
      real time           0.00 seconds
      user cpu time       0.00 seconds
      system cpu time     0.01 seconds
      memory              2700.21k
      OS Memory           27048.00k
      Timestamp           02/25/2022 04:11:05 PM
      Step Count                        30  Switch Count  4
      Page Faults                       0
      Page Reclaims                     233
      Page Swaps                        0
      Voluntary Context Switches        18
      Involuntary Context Switches      0
      Block Input Operations            0
      Block Output Operations           528
```

Transposed

CODE | LOG | RESULTS | OUTPUT DATA

Table: WORK.NIMNAME_TRANSPOSED | View: Column names | Filter: (none)

Columns

- Select all
- Process
- Day
- Contamination
- Corrosion
- Doping
- Metallization
- Miscellaneous
- Oxide Defect
- Silicon Defect

Total rows: 10 Total columns: 9 Rows 1-10

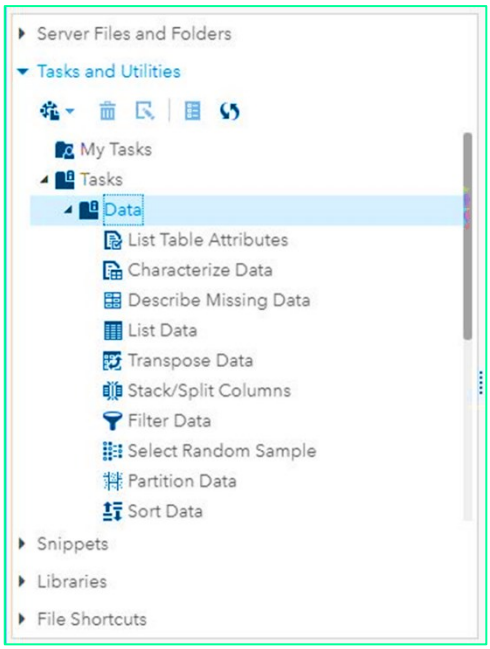
	Process	Day	Contamination	Corrosion	Doping
1	Process A	Monday	15	2	1
2	Process A	Tuesday	16	3	1
3	Process A	Wednesday	20	1	1
4	Process A	Thursday	12	1	1
5	Process A	Friday	23	1	1
6	Process B	Monday	8	2	1
7	Process B	Tuesday	9	0	1
8	Process B	Wednesday	4	1	1
9	Process B	Thursday	2	2	1
10	Process B	Friday	1	3	1

- As shown in the above screenshot, look how **little code is needed to be written using PROC TRANSPOSE** to easily rearrange the data within a table in order to view information much more clearly.

Next, is **another way** to do the same transpose **using the TASK function** available in SAS Studio which **will generate the program code** for you.

SAS Studio Transpose Data task

- SAS Studio also provides a task that generates the necessary code, including PROC TRANSPOSE. In the left of SAS Studio, expand Tasks and Utilities; then expand Tasks and Data so that you can see the list of Data tasks:



b. Now double-click on the Transpose Data task, which will open up a program section on the right-hand side of SAS Studio. In the Data drop-down box, select **SASHELP.FAILURE**.
c. On the left of your workbench select **the "Data" tab**:
   1) **Under ROLES**, assign the **Count column**;
   2) then expand **ADDITIONAL ROLES** and add **Process** and **Day**.
d. Now select the **OPTIONS tab** :
   1) **Uncheck** the **Use prefix** checkbox.
   2) **Check** the **Select a variable that contains the names of the new variables** box and **add Cause.**
e. Return to the DATA tab in the left-hand-side section and scroll all the way to the bottom:
   1) **Check** the Show output data box.
   2) Give your Data Set name **output as "work.NIMNAME_Transposed"**
f. Now **submit the code** by selecting the running man icon. Since we checked **Show output data,** a **PROC PRINT** was added to the generated code and shows this in the **RESULTS tab**.
g. On Output **figure C** the Transpose task example **provides the exact same transposed results** as the **previous example** using **PROC TRANSPOSE code.**
h. **Print it out your output result as shown on Output figure C as name file "IS-429 Lab Week#7C results"**

➲ **Save the code given as "IS-429 Lab Week#7C NIM Name"**

4. **Statistical and mathematical data transformations**

Data scientists and analysts not only want to transpose data within tables, but also tend **to enrich the data with statistics related** to the existing numeric within the table. SAS again makes this enrichment easy with procedures, **for example, PROC MEANS**.

PROC MEANS

- The MEANS procedure provides summarization **tools to descriptive statistics** for variables across all observations and within groups of observations. For example, PROC MEANS does the following:
  - Calculates **descriptive statistics** based **on moments**
  - Estimates quantiles, which includes the **median**
  - Calculates confidence limits for the **mean**
  - Identifies **extreme values**
  - Performs a **t test**
- The following example of PROC MEANS will continue to use the work.failure_transposed dataset that was produced using the prior PROC TRANSPOSE example.

h. Type it on and submitted these below codes and **save it on to IS-429 Lab Week#7D NIM Name**:

```
/* Week#7D yourNIM and Name                                          */
proc transpose data=sashelp.failure
               out=work.nimname_transposed(drop=_NAME_);
               id cause;
               by process day;
quit;

proc means data=work.nimname_transposed n mean max min Q1 median Q3 std;
   class Process;
quit;
```

i. Submitting the above code will produce the output in the RESULTS tab as Output Figure D.

i. **Print it out your output result as shown on Output figure D as name file "IS-429 Lab Week#7D results**"

❖ **Finally**, today's practicum is over, **collect your pdf file and all code programs into "IS-429 Week#7 NIM yourName.zip** and submit immediately today to e-Learning IS-429 Practicum Week#7.

RESULTS/ OUTPUT

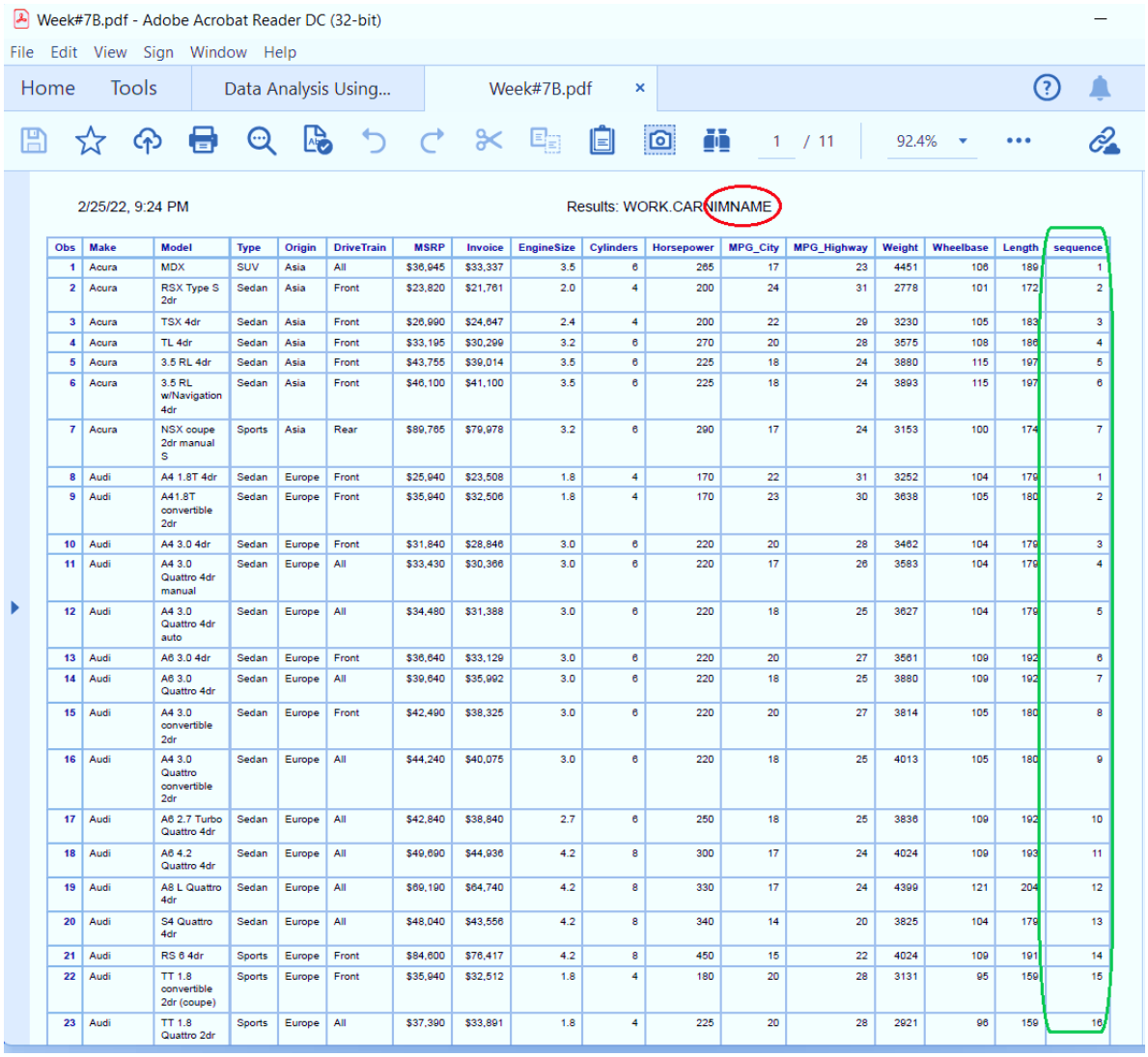
A. Referencing external files: explore and load the CLIENT_INFO table

2/25/22, 12:57 AM Results: WORK.YOURNIMNAME_RAW

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

https://odamid-apse1.oda.sas.com/SASStudio/sasexec/submissions/f741e3ba-672c-4e90-ac1e-6d1805e545b8/results

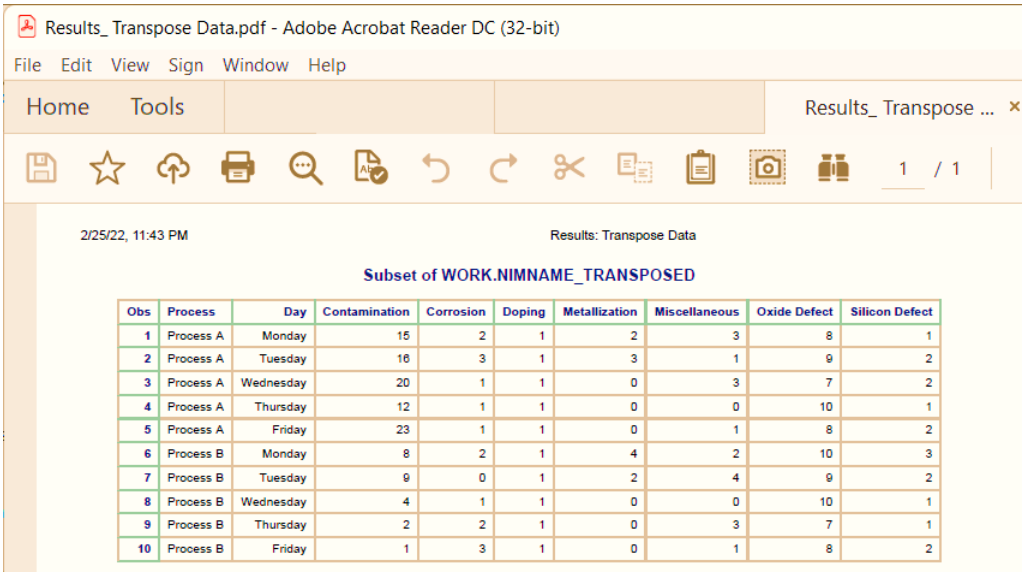
B. Creating indicators for the first and last observation in a by group



2/25/22, 9:24 PM Results: WORK.CARNIMNAME

Obs	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length	sequence
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265	17	23	4451	106	189	1
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2.0	4	200	24	31	2778	101	172	2
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22	29	3230	105	183	3
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20	28	3575	108	186	4
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18	24	3880	115	197	5
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225	18	24	3893	115	197	6
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290	17	24	3153	100	174	7
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	170	22	31	3252	104	179	1
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	$35,940	$32,506	1.8	4	170	23	30	3638	105	180	2
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3.0	6	220	20	28	3462	104	179	3
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430	$30,366	3.0	6	220	17	26	3583	104	179	4
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480	$31,388	3.0	6	220	18	25	3627	104	179	5
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3.0	6	220	20	27	3561	109	192	6
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	$39,640	$35,992	3.0	6	220	18	25	3880	109	192	7
15	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	$42,490	$38,325	3.0	6	220	20	27	3814	105	180	8
16	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	$44,240	$40,075	3.0	6	220	18	25	4013	105	180	9
17	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	$42,840	$38,840	2.7	6	250	18	25	3836	109	192	10
18	Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	$49,690	$44,936	4.2	8	300	17	24	4024	109	193	11
19	Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	330	17	24	4399	121	204	12
20	Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	340	14	20	3825	104	179	13
21	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	450	15	22	4024	109	191	14
22	Audi	TT 1.8 convertible 2dr (coupe)	Sports	Europe	Front	$35,940	$32,512	1.8	4	180	20	28	3131	95	159	15
23	Audi	TT 1.8 Quattro 2dr	Sports	Europe	All	$37,390	$33,891	1.8	4	225	20	28	2921	96	159	16

C. Transposing the rows and columns in a table


Results_ Transpose Data.pdf - Adobe Acrobat Reader DC (32-bit)

File Edit View Sign Window Help

Home Tools Results_ Transpose ... ×

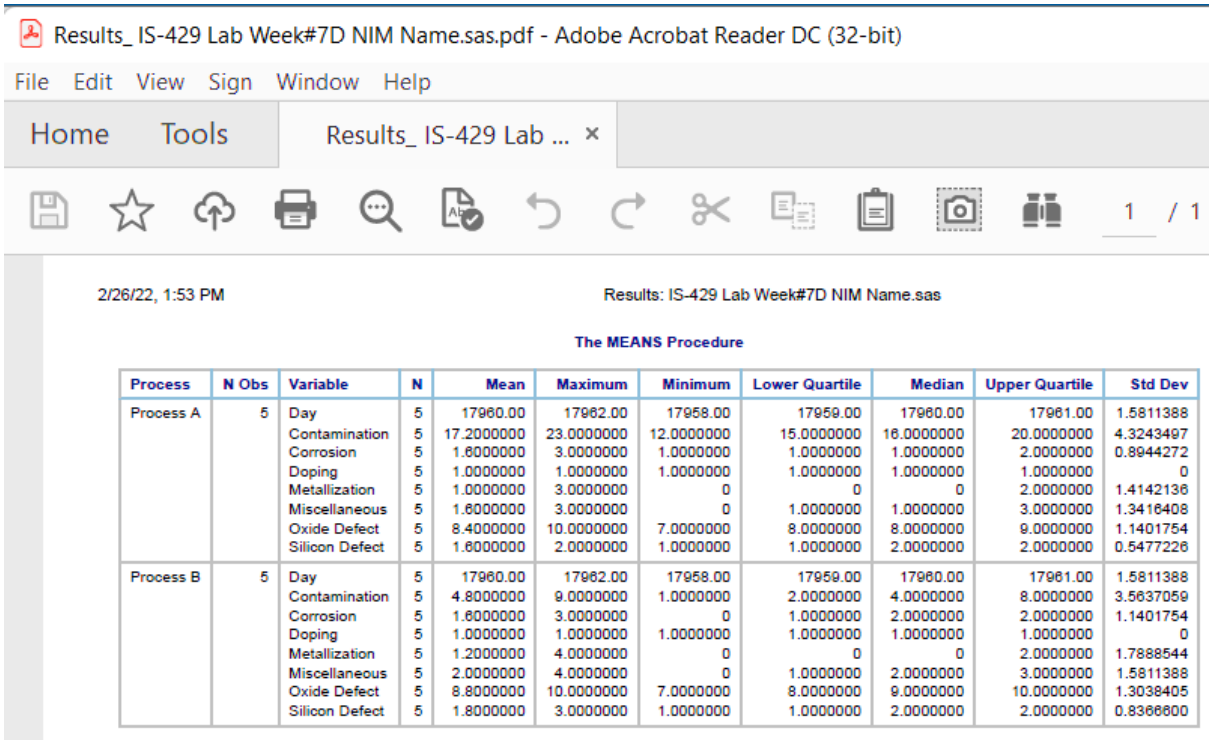
1 / 1

2/25/22, 11:43 PM Results: Transpose Data

Subset of WORK.NIMNAME_TRANSPOSED

Obs	Process	Day	Contamination	Corrosion	Doping	Metallization	Miscellaneous	Oxide Defect	Silicon Defect
1	Process A	Monday	15	2	1	2	3	8	1
2	Process A	Tuesday	16	3	1	3	1	9	2
3	Process A	Wednesday	20	1	1	0	3	7	2
4	Process A	Thursday	12	1	1	0	0	10	1
5	Process A	Friday	23	1	1	0	1	8	2
6	Process B	Monday	8	2	1	4	2	10	3
7	Process B	Tuesday	9	0	1	2	4	9	2
8	Process B	Wednesday	4	1	1	0	0	10	1
9	Process B	Thursday	2	2	1	0	3	7	1
10	Process B	Friday	1	3	1	0	1	8	2

D. Statistical and mathematical data transformations


Results_ IS-429 Lab Week#7D NIM Name.sas.pdf - Adobe Acrobat Reader DC (32-bit)

File Edit View Sign Window Help

Home Tools Results_ IS-429 Lab ... ×

1 / 1

2/26/22, 1:53 PM Results: IS-429 Lab Week#7D NIM Name.sas

The MEANS Procedure

Process	N Obs	Variable	N	Mean	Maximum	Minimum	Lower Quartile	Median	Upper Quartile	Std Dev
Process A	5	Day	5	17960.00	17962.00	17958.00	17959.00	17960.00	17961.00	1.5811388
		Contamination	5	17.2000000	23.0000000	12.0000000	15.0000000	16.0000000	20.0000000	4.3243497
		Corrosion	5	1.6000000	3.0000000	1.0000000	1.0000000	1.0000000	2.0000000	0.8944272
		Doping	5	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	0
		Metallization	5	1.0000000	3.0000000	0	0	0	2.0000000	1.4142136
		Miscellaneous	5	1.6000000	3.0000000	0	1.0000000	1.0000000	3.0000000	1.3416408
		Oxide Defect	5	8.4000000	10.0000000	7.0000000	8.0000000	8.0000000	9.0000000	1.1401754
		Silicon Defect	5	1.6000000	2.0000000	1.0000000	1.0000000	2.0000000	2.0000000	0.5477226
Process B	5	Day	5	17960.00	17962.00	17958.00	17959.00	17960.00	17961.00	1.5811388
		Contamination	5	4.8000000	9.0000000	1.0000000	2.0000000	4.0000000	8.0000000	3.5637059
		Corrosion	5	1.6000000	3.0000000	0	1.0000000	2.0000000	2.0000000	1.1401754
		Doping	5	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000	0
		Metallization	5	1.2000000	4.0000000	0	0	0	2.0000000	1.7888544
		Miscellaneous	5	2.0000000	4.0000000	0	1.0000000	2.0000000	3.0000000	1.5811388
		Oxide Defect	5	8.8000000	10.0000000	7.0000000	8.0000000	9.0000000	10.0000000	1.3038405
		Silicon Defect	5	1.8000000	3.0000000	1.0000000	1.0000000	2.0000000	2.0000000	0.8366600

REFERENCE

1. Anna Yarbrough. 2020. Introduction to Data Curation for SAS® Data Scientists Course Notes. SAS Institute Inc. Cary, NC, USA.
2. SAS Institute Inc. 2020. SAS® Viya® Programming: Getting Started. SAS Institute Inc. Cary, NC, USA.
3. SAS Institute Inc. 2019. Exploring SAS® Viya®: Programming and Data Management. SAS Institute Inc. Cary, NC, USA.
4. SAS® Support | Documentation
5. Other additional references are excerpts from various Online Learning/websites.