**SAS Notes**

Contents

[BASIC OPERATIONS 4](#_Toc121075955)

[/\* CREATING TABLE WITH FORMAT SPECIFIED CHARACTER VAR\*/ 4](#_Toc121075956)

[/\* CREATING TABLE WITHOUT FORMAT SPECIFIED CHARACTER VAR\*/ 4](#_Toc121075957)

[/\* SAVING TABLE TO LIBRARY\*/ 4](#_Toc121075958)

[/\* creating a library - setting library name\*/ 4](#_Toc121075959)

[/\* IMPORTING EXCEL\*/ 4](#_Toc121075960)

[/\* reading table in SAS and assigning it to another table \*/ 4](#_Toc121075961)

[VARIABLES 5](#_Toc121075962)

[/\*creating variable\*/ 5](#_Toc121075963)

[/\* Renaming a variable \*/ 5](#_Toc121075964)

[/\* removing a variable \*/ 5](#_Toc121075965)

[/\* create, rename and remove a variable all at once and saving the resulting file in our library \*/ 5](#_Toc121075966)

[FILTERING DATA 5](#_Toc121075967)

[CONDITIONAL STATEMENTS in SAS 6](#_Toc121075968)

[/\* if then else \*/ 6](#_Toc121075969)

[SAS options 6](#_Toc121075970)

[/\* KEEP - we only want to keep few columns\*/ 6](#_Toc121075971)

[/\* DROP - few columsn we do not want to keep\*/ 6](#_Toc121075972)

[/\* OUTPUT option \*/ 6](#_Toc121075973)

[/\* \_N\_ - represents observation lines - lets say we want to keep lines 10 to 20\*/ 6](#_Toc121075974)

[/\* OBS option - lets only output 15 observations - by default 1st to 15th obs \*/ 6](#_Toc121075975)

[/\* FIRSTOBS option - lets only output 10 observations but it should start from 7th observation \*/ 7](#_Toc121075976)

[/\* PROC PRINT \*/ 7](#_Toc121075977)

[/\* let us limit the number of variables - use VAR statement \*/ 7](#_Toc121075978)

[/\* let us give title to the table \*/ 7](#_Toc121075979)

[/\* let us use by statement to get country wise grouping of printed tables \*/ 7](#_Toc121075980)

[/\* let us use sum statement to get sum of trasaction amount for each sub-table \*/ 7](#_Toc121075981)

[/\* let us use where statement to get sum of trasaction amount for USA \*/ 7](#_Toc121075982)

[/\* adding a footnote \*/ 7](#_Toc121075983)

[/\* PROC SORT \*/ 8](#_Toc121075984)

[/\* if we want to save sorted data in another table \*/ 8](#_Toc121075985)

[/\* if we want to save by country in descending order \*/ 8](#_Toc121075986)

[DEALING WITH DUPLICATES 8](#_Toc121075987)

[/\* NODUPKEY - keep one obs for each country\*/ 8](#_Toc121075988)

[/\* NODUP - remove throughout duplicate rows - duplicate values in all columns \*/ 8](#_Toc121075989)

[/\*by \_all\_\*/ 8](#_Toc121075990)

[/\* DUPOUT - output the duplicate values identified by nodupkey or nodup \*/ 9](#_Toc121075991)

[/\* PROC FREQ \*/ 9](#_Toc121075992)

[/\* 2 dimensional freq report \*/ 9](#_Toc121075993)

[Conditional frequency 9](#_Toc121075994)

[/\* PROC TRANSPOSE \*/ 9](#_Toc121075995)

[dropping unwanted columns 10](#_Toc121075996)

[prefix 10](#_Toc121075997)

[suffix 10](#_Toc121075998)

[transposing 2 variables together 11](#_Toc121075999)

[/\* PROC CONTENTS \*/ 11](#_Toc121076000)

[VARNUM 11](#_Toc121076001)

[DEALING WITH BLANKS 11](#_Toc121076002)

[/\* compressing \*/ 12](#_Toc121076003)

[/\* compress can also be used to remove specific characters from string \*/ 12](#_Toc121076004)

[modifiers – ignoring case (caps or lowcase) 12](#_Toc121076005)

[modifiers – removing digits 12](#_Toc121076006)

[CONCATENATION 12](#_Toc121076007)

[Using pipe (||) function 12](#_Toc121076008)

[/\* now using CAT function \*/ 13](#_Toc121076009)

[/\* to separate using single space - use CATX \*/ 13](#_Toc121076010)

[/\* CATS and CATT are other functions \*/ 13](#_Toc121076011)

[EXTRACTION from string 13](#_Toc121076012)

[/\* let us use SUBSTR \*/ 13](#_Toc121076013)

[/\* CHANGE CASE FUNCTIONS \*/ 14](#_Toc121076014)

[/\* STRING MANIPULATION \*/ 14](#_Toc121076015)

[/\* you can correct e.g. prefixes using TRANWRD and conditional statements like IF\*/ 14](#_Toc121076016)

[TRANSLATE 15](#_Toc121076017)

[/\* finding a word or character within a string using INDEX \*/ 15](#_Toc121076018)

[/\* CONDITIONAL FINDING OF CHAR OR WORD IN A STRING \*/ 15](#_Toc121076019)

[/\* PRXMATCH - search for a pattern in a string \*/ 15](#_Toc121076020)

[SEPARATING/EXTRACTING USING PRXMATCH 16](#_Toc121076021)

[/\* data type conversion \*/ 16](#_Toc121076022)

[DATES 16](#_Toc121076023)

[/\* current date TODAY(), DAY(), WEEKDAY(), WEEK(), MONTH(), QTR() and YEAR() functions \*/ 16](#_Toc121076024)

[/\* creating date from month day year \*/ 17](#_Toc121076025)

[/\* INFORMAT and FORMAT functions \*/ 17](#_Toc121076026)

[Informat 17](#_Toc121076027)

[FORMAT 17](#_Toc121076028)

[/\* INTCK calculates difference betweek 2 dates, times or datetimes \*/ 18](#_Toc121076029)

[/\* INTNX option - increments a date, time or datetime value by a given interval \*/ 18](#_Toc121076030)

[/\* EXPORTING IN SAS \*/ 18](#_Toc121076031)

[/\* how to xport mult tables in one excel file? \*/ 19](#_Toc121076032)

[COMBINING tables 19](#_Toc121076033)

[using set command 19](#_Toc121076034)

[MERGE tables 20](#_Toc121076035)

[/\* APPEND function – stacking tables \*/ 20](#_Toc121076036)

[PROC SQL 21](#_Toc121076037)

[conditional starements 21](#_Toc121076038)

[/\* summarize tables – use group by here\*/ 21](#_Toc121076039)

[/\* conditional with group by \*/ 21](#_Toc121076040)

[/\* filtering data \*/ 22](#_Toc121076041)

[/\* ORDER BY clause – similar to sort \*/ 22](#_Toc121076042)

[/\* CASE WHEN similar to IF ELSE \*/ 22](#_Toc121076043)

[/\* summarizing with COUNT FREQ N \*/ 23](#_Toc121076044)

[/\* MIN MAX AVG MEAN RANGE STD SUM \*/ 23](#_Toc121076045)

[/\* JOINS \*/ 24](#_Toc121076046)

[/\* cross join or cartesian join \*/ 25](#_Toc121076047)

[output result of sql step in a table 25](#_Toc121076048)

# BASIC OPERATIONS

## /\* CREATING TABLE WITH FORMAT SPECIFIED CHARACTER VAR\*/

/\*$ means character 6 means 6 characters max\*/

data first\_data;

input employee $6. salary;

datalines;

manbir 67

jassbr 89

jasdhi 793

khandl 77

grgill 54

cheema 106

;

run;

## /\* CREATING TABLE WITHOUT FORMAT SPECIFIED CHARACTER VAR\*/

/\*no char format specified\*/

data first\_data;

input employee $ salary;

datalines;

manbir 67

jass 89

jas 793

khandal 77

gill 54

cheema 106

;

run;

## /\* SAVING TABLE TO LIBRARY\*/

data SASTUTOR.first\_data; /\*saving table to library\*/

input employee $ salary;

datalines;

manbir 67

jass 89

jas 793

khandal 77

gill 54

cheema 106

;

run;

## /\* creating a library - setting library name\*/

libname sastut "/home/u62306491/sasuser.v94/SAS\_tut\_files";

## /\* IMPORTING EXCEL\*/

proc import datafile="/home/u62306491/sasuser.v94/SAS\_tut\_files/Categories.xlsx" /\* specifying data location \*/

out = sastut.cat /\* specifies name of the SAS table that is to be created in sastut library\*/

dbms = xlsx replace; /\* specifies type of the file imported, replace overwrites existing SAS dataset\*/

sheet = "categories"; /\* specifies sheet on the excel file to be imported\*/

run;

## /\* reading table in SAS and assigning it to another table \*/

data test;

set sastut.cat; /\* set command reads data from specified location\*/

run;

# VARIABLES

/\* working on another dataset\*/

/\* importing\*/

proc import datafile="/home/u62306491/sasuser.v94/SAS\_tut\_files/Order\_Details.xlsx"

out = sastut.OrderDetails (drop = price) /\*drops price column\*/

dbms=xlsx replace;

sheet="OrderDetails";

run;

## /\*creating variable\*/

data test;

set sastut.Orderdetails;

price = quantity\*5; /\*multiplies quantity column with 5 (each row)\*/

run; /\* please note that the resulting dataset is stored as test in work library\*/

## /\* Renaming a variable \*/

data test;

set sastut.orderdetails;

rename quantity = qty;

run;

## /\* removing a variable \*/

data test;

set sastut.orderdetails;

drop orderdetailid;

run;

## /\* create, rename and remove a variable all at once and saving the resulting file in our library \*/

data sastut.test;

set sastut.orderdetails;

price = quantity\*5;

rename quantity = qty;

drop orderdetailid;

run;

# FILTERING DATA

/\* first we import the data \*/

proc import datafile="/home/u62306491/sasuser.v94/SAS\_tut\_files/Customers.xlsx"

out = sastut.customers

dbms=xlsx replace;

sheet="Customers";

run;

/\* filter for USA in country column and amount > 600 in transaction column \*/

data test; /\* create table\*/

set sastut.customers; /\* read data \*/

where country = 'USA' and transaction\_amount > 600;

run;

/\* filter for USA in country column OR amount > 600 in transaction column \*/

data test; /\* create table\*/

set sastut.customers; /\* read data \*/

where country = 'USA' or transaction\_amount > 600;

run;

# CONDITIONAL STATEMENTS in SAS

## /\* if then else \*/

data test;

set sastut.customers;

if transaction\_amount < 500 then category = "Below Average";

else category = "Above Average";

run;

/\* if then, else if the, else \*/

data test;

set sastut.customers;

if transaction\_amount < 500 then category = "Below Average";

else if transaction\_amount >= 500 and transaction\_amount < 699 then category = "Premium";

else category = "Elite";

run;

# SAS options

## /\* KEEP - we only want to keep few columns\*/

/\* first look at the whole table \*/

data keeps;

set sashelp.cars;

run;

/\* just select 3 columns out of the whole table \*/

data keeps;

set sashelp.cars (keep=model type msrp);

run;

/\* also works when SAS options are given in data statement line \*/

data keeps(keep=model type msrp);

set sashelp.cars;

run;

## /\* DROP - few columsn we do not want to keep\*/

data drops;

set sashelp.cars (drop = msrp);

run;

## /\* OUTPUT option \*/

/\* OUTPUT - create 2 tables, one with msrp <19000 and that too with only 3 columns\*/

/\* second with msrp >= 19000, but should not keep the 3 variables \*/

/\* we will use keep/drop options in data statement \*/

data price(keep=model type msrp) features(drop=model type msrp);

set sashelp.cars;

if msrp<19000 then output price;

else output features;

run;

## /\* \_N\_ - represents observation lines - lets say we want to keep lines 10 to 20\*/

data linenumber;

set work.price;

if \_N\_>= 10 and \_N\_ < 20;

run;

## /\* OBS option - lets only output 15 observations - by default 1st to 15th obs \*/

data observation;

set work.price(obs = 15);

run;

## /\* FIRSTOBS option - lets only output 10 observations but it should start from 7th observation \*/

data observation;

set work.price(obs = 16 firstobs=7); /\*we use 16 observations because if we count from 7th obs in this 16 obs table, we will get 10 obs which we wanted \*/

run;

# /\* PROC PRINT \*/

proc print data = sastut.customers;

run; /\* this prints the whole table \*/

/\* let us print only first 10 obs \*/

proc print data = sastut.customers(obs=10);

run;

## /\* let us limit the number of variables - use VAR statement \*/

proc print data = sastut.customers(obs=10);

var contactname country transaction\_amount;

run;

## /\* let us give title to the table \*/

proc print data = sastut.customers(obs=10);

var contactname country transaction\_amount;

title "Customers Table";

run;

## /\* let us use by statement to get country wise grouping of printed tables \*/

proc print data = sastut.customers(obs=10);

var contactname country transaction\_amount;

title "Customers Table";

by notsorted country;

run;

## /\* let us use sum statement to get sum of trasaction amount for each sub-table \*/

proc print data = sastut.customers(obs=10);

var contactname country transaction\_amount;

title "Customers Table";

by notsorted country;

sum transaction\_amount;

run;

## /\* let us use where statement to get sum of trasaction amount for USA \*/

proc print data = sastut.customers;

var contactname country transaction\_amount;

title "Customers Table";

by notsorted country; /\* this statement can be deleted here \*/

sum transaction\_amount;

where country = 'USA';

run;

## /\* adding a footnote \*/

proc print data = sastut.customers;

var contactname country transaction\_amount;

title "Customers Table";

by notsorted country; /\* this statement can be deleted here \*/

sum transaction\_amount;

where country = 'USA';

footnote 'footnote text here';

run;

# /\* PROC SORT \*/

/\* let us sort on country \*/

proc sort data = sastut.customers;

by country;

run;

/\* let us sort on country and transaction amount \*/

proc sort data = sastut.customers;

by country transaction\_amount;

run;

## /\* if we want to save sorted data in another table \*/

proc sort data = sastut.customers out = sorted;

by country transaction\_amount;

run;

## /\* if we want to save by country in descending order \*/

proc sort data = sastut.customers out = sorted;

by descending country transaction\_amount;

run;

/\* if we want to save by transaction amount in descending order \*/

proc sort data = sastut.customers out = sorted;

by country descending transaction\_amount;

run;

# DEALING WITH DUPLICATES

## /\* NODUPKEY - keep one obs for each country\*/

proc sort data = sastut.customers nodupkey out = sorted;

by country; /\* duplicate countries will be removed \*/

run;

## /\* NODUP - remove throughout duplicate rows - duplicate values in all columns \*/

proc sort data = sastut.customers nodup out = sorted;

by country; /\* by variable is required for sort command \*/

run;

## /\*by \_all\_\*/

/\* SAS only compares consecutive rows and will not compare e.g. 1st and 3rd rows \*/

/\* use by \_all\_ to workaround that \*/

proc sort data = sastut.customers nodup out = sorted;

by \_all\_; /\* by variable is required for sort command \*/

run;

## /\* DUPOUT - output the duplicate values identified by nodupkey or nodup \*/

proc sort data = sastut.customers nodupkey out = sorted dupout = duplicated;

by country; /\* by variable is required for sort command \*/

run;

# /\* PROC FREQ \*/

/\* first we import the data \*/

proc import datafile="/home/u62306491/sasuser.v94/SAS\_tut\_files/customer\_details.xlsx"

out = sastut.cust

dbms=xlsx replace;

sheet="data";

run;

/\* the procedure \*/

proc freq data = sastut.cust;

tables city;

run;

/\* lets say we only want frequency \*/

proc freq data = sastut.cust;

tables city / nopercent nocum;

run;

## /\* 2 dimensional freq report \*/

/\* e.g. a given city occurs how many times in a country \*/

proc freq data = sastut.cust;

tables city\*country;

run;

/\* lets say we only want frequency \*/

proc freq data = sastut.cust;

tables city\*country / nopercent norow nocol;

run;

## Conditional frequency

/\* check occurences of a city within the country, only when transaction amount > 500 \*/

/\* use where clause \*/

proc freq data = sastut.cust;

tables city\*country / nopercent norow nocol;

where transaction\_amount > 500;

run;

# /\* PROC TRANSPOSE \*/

proc transpose data = sastut.cust out = transposed;

by city; /\*what we dont want to transpose\*/

id country; /\*needs to be transposed from obs to column\*/

var transaction\_amount; /\*what we want to populate the table\*/

run;

/\* this willl give error as data needs to be sorted based on the "by" and "id" variables \*/

proc sort data = sastut.cust out = sorted\_cust;

by city country;

run;

/\* now do the transpose \*/

proc transpose data = sorted\_cust out = transposed;

by city; /\*what we dont want to transpose\*/

id country; /\*needs to be transposed from obs to column\*/

var transaction\_amount; /\*what we want to populate the table\*/

run;

/\* we see some errors because more than one instance of city and country combination exists \*/

/\* let us fix that first \*/

proc sort data=sorted\_cust nodupkey out = sorted\_cust\_nodup;

by city country;

run;

/\* now lets see \*/

proc transpose data = sorted\_cust\_nodup out = transposed;

by city; /\*what we dont want to transpose\*/

id country; /\*needs to be transposed from obs to column\*/

var transaction\_amount; /\*what we want to populate the table\*/

run;

## dropping unwanted columns

/\* getting rid of the 2 unwanted labels \*/

proc transpose data = sorted\_cust\_nodup out = transposed(drop=\_:); /\*drop columns starting with '\_' and put ':' after\*/

by city; /\*what we dont want to transpose\*/

id country; /\*needs to be transposed from obs to column\*/

var transaction\_amount; /\*what we want to populate the table\*/

run;

## prefix

/\* giving a prefix to the transposed variable \*/

/\* use prefix function - dont use "" with it \*/

proc transpose data = sorted\_cust\_nodup prefix = trans\_ out = transposed(drop=\_:); /\*drop columns starting with '\_' and put ':' after\*/

by city; /\*what we dont want to transpose\*/

id country; /\*needs to be transposed from obs to column\*/

var transaction\_amount; /\*what we want to populate the table\*/

run;

## suffix

/\* using suffix as well \*/

proc transpose data = sorted\_cust\_nodup prefix = trans\_ suffix = \_done out = transposed(drop=\_:); /\*drop columns starting with '\_' and put ':' after\*/

by city; /\*what we dont want to transpose\*/

id country; /\*needs to be transposed from obs to column\*/

var transaction\_amount; /\*what we want to populate the table\*/

run;

/\* another example \*/

/\* first let us sort the data and remove duplicate data\*/

proc sort data=sastut.customers out=sorted\_cust2;

by city contactname country;

run;

/\* now let us transpose \*/

proc transpose data=sorted\_cust2 out=transposed2(drop=\_:);

by city contactname;

id country;

var transaction\_amount;

run;

## transposing 2 variables together

/\* what if we want to transpose city and country value together \*/

/\* change by and id statements accordingly \*/

proc transpose data=sorted\_cust2 out=transposed2(drop=\_:);

by contactname;

id country city;

var transaction\_amount;

run;

/\* we get error so changing the order of by statement variables \*/

proc sort data=sastut.customers out=sorted\_cust2;

by contactname city country;

run;

/\* next try \*/

proc transpose data=sorted\_cust2 out=transposed2(drop=\_:);

by contactname;

id country city;

var transaction\_amount;

run;

# /\* PROC CONTENTS \*/

/\* rows columns and variable type information \*/

proc contents data=sashelp.cars;

run;

## VARNUM

/\* the names of variables listed in the output table is in alphabetical order and not in actual order \*/

/\* for actual order, use VARNUM option \*/

proc contents data=sashelp.cars varnum;

run;

# DEALING WITH BLANKS

/\* TYPES OF BLANKS \*/

/\* trailing blanks - blanks at the end of the string \*/

/\* leading blanks - before the string \*/

/\* blanks in between - as the name suggests \*/

/\* how to remove the blanks/spaces \*/

/\* TRIM TRIMN STRIP COMPRESS COMPBL \*/

/\* HERE WE WILL USE COMPRESS AND COMPBL - REST ARE EASY TO USE \*/

/\* COMPRESS - removes all blank spaces from data \*/

/\* COMPBL - compresses more than one consecutive blank spaces into 1 blank space \*/

/\* creating sample data \*/

data sample;

NAME = "Jaskaran Singh Dhiman";

len = length(NAME);

run;

/\* putting some space \*/

data sample;

NAME = "Jaskaran Singh Dhiman";

len = length(NAME);

run;

## /\* compressing \*/

data sample;

NAME = "Jaskaran Singh Dhiman";

len = length(NAME);

compress = compress(NAME);

len\_c = length(compress);

compbl = compbl(NAME);

len\_cbl = length(compbl);

run;

## /\* compress can also be used to remove specific characters from string \*/

data sample2;

values = '165236837jhHkshdvsvdfk<>:"';

new\_values = compress(values, '<>:"'); /\* removing spcl char \*/

run;

## modifiers – ignoring case (caps or lowcase)

/\* removing a letter - h \*/

data sample2;

values = '165236837jhHkshdvsvdfk<>:"';

new\_values = compress(values, 'h'); /\* removing h \*/

run;

/\* we see that capital H is not removed - if we want all h to be removed irrespective of the case - we use modifier \*/

/\* e.g. 'i' modifier ignores case \*/

data sample2;

values = '165236837jhHkshdvsvdfk<>:"';

new\_values = compress(values, 'h', 'i'); /\* removing h and H \*/

run;

## modifiers – removing digits

/\* we want to remove all number values - use digits modifier - d \*/

data sample2;

values = '165236837jhHkshdvsvdfk<>:"';

new\_values = compress(values, '', 'd'); /\* removing all digits - keep character as blank - '' \*/

run;

# CONCATENATION

## Using pipe (||) function

/\* CONCATENATION - COMBINE MULTIPLE STRINGS \*/

/\* MANUAL WAY TO JOIN USING PIPE - | \*/

data sample;

firstname = 'Jaskaran';

middlename = 'Singh';

lastname = 'Dhiman';

fullname = firstname||middlename||lastname; /\* using double pipe - || \*/

run;

/\* here there is no space - let us add space manually \*/

data sample;

firstname = 'Jaskaran';

middlename = 'Singh';

lastname = 'Dhiman';

fullname = firstname||' '||middlename||' '||lastname; /\* using double pipe - || and space \*/

run;

## /\* now using CAT function \*/

data sample;

firstname = 'Jaskaran';

middlename = 'Singh';

lastname = 'Dhiman';

fullname = firstname||' '||middlename||' '||lastname;

fullname2 = cat(firstname, middlename, lastname);

run;

## /\* to separate using single space - use CATX \*/

data sample;

firstname = 'Jaskaran';

middlename = 'Singh';

lastname = 'Dhiman';

fullname = firstname||' '||middlename||' '||lastname;

fullname2 = cat(firstname, middlename, lastname);

fullname3 = catx(' ',firstname, middlename, lastname); /\* catx uses a separator before - this can be any value \*/

run;

## /\* CATS and CATT are other functions \*/

/\* CATS is similar to CAT and CATT is combination of CAT and TRIM function - you can try \*/

# EXTRACTION from string

/\* EXTRACTING a part of the string \*/

/\* e.g. getting last 4 digits of credit card number \*/

/\* in EXCEL we use MID function \*/

/\* in SAS we use SUBSTR \*/

data card\_info;

input card\_num $ 16.;

datalines;

572848221752577

337904780994715

401860789452657

382609120165135

709455401928619

618737423328593

874440208793747

126776030172856

280138656321611

;

run;

## /\* let us use SUBSTR \*/

data sample;

set card\_info;

last\_digits = substr(card\_num, 12, 4); /\* SUBSTR(var\_name, position of first value, no. of values to follow) \*/

run;

/\* you can get any middle characters as well using SUBSTR \*/

/\* we can replace a part of string with any character using SUBSTR - let's see \*/

data sample;

set card\_info;

substr(card\_num, 5, 8) = '\*\*\*\*\*\*\*\*'; /\* from 5th value, we want 8 \* values \*/

run;

# /\* CHANGE CASE FUNCTIONS \*/

/\* capital, lower, sentence, all first letters caps, etc \*/

data sample;

input name $;

cards;

Yusuf

renee

CORRINNE

aLLyX

aHMAD

;

run;

data change\_case;

set sample;

CAPS = upcase(name);

LOW = lowcase(name);

sentence = propcase(name);

run;

# /\* STRING MANIPULATION \*/

/\* TRANWRD - replaces all occurences of a specific word in a string \*/

/\* TRANWRD(var\_name, REPLACE WHAT, REPLACE WITH) \*/

data sample;

name = "Amanpreet Kaur Punni";

changed = tranwrd(name, 'Punni', 'Dhiman');

run;

data sample;

string = "boy boy boy girl girl girl";

changed = tranwrd(string, 'boy', 'girl');

run;

## /\* you can correct e.g. prefixes using TRANWRD and conditional statements like IF\*/

/\* making a dataset \*/

data sample2;

input name $ sex $;

cards;

Mr.Julie F

Ms.Ron M

Mrs.Daniel m

Mr.Allyx f

;

run;

/\* trying TRANWRD \*/

data sample2\_fixed;

set sample2;

if upcase(sex) = 'M' then name2 = tranwrd(name, 'Ms.', 'Mr. ');

else name2 = tranwrd(name, 'Mr.', 'Ms. ');

run;

/\* we see that mrs is not changed \*/

data sample2\_fixed\_again;

set sample2\_fixed;

if upcase(sex) = 'M' then name3 = tranwrd(name2, 'Mrs.', 'Mr. ');

else name3 = name2;

run;

## TRANSLATE

/\* TRANSLATE option - just like TRANWRD replaces a word in a sentence or string, TRANSLATE replaces characters \*/

/\* TRANSLATE(var\_name, REPLACE WITH, REPLACE WHAT) \*/

data sample3;

name = 'Justkaran';

name2 = translate(name, 'as', 'ust'); /\* this will give extra space because we are changing 3 to 2\*/

name3 = compress(name2, ' '); /\* fixing the name \*/

run;

## /\* finding a word or character within a string using INDEX \*/

/\* returns the position of specified string value \*/

/\* INDEX(var\_name, find what) \*/

/\* first we import file \*/

proc import datafile='/home/u62306491/sasuser.v94/SAS\_tut\_files/online\_sales.xlsx'

out = sastut.online

dbms= xlsx replace;

sheet= 'Sheet1';

run;

/\* lets try now \*/

data sample;

set sastut.online;

exists = index(remarks, '.com'); /\* non-0 value under exist column will tell that .com exists there \*/

/\* now let us see only that data where exist>0 \*/

if exists > 0 then output;

run;

## /\* CONDITIONAL FINDING OF CHAR OR WORD IN A STRING \*/

/\* FIND FUNCTION - LIKE index BUT PROVIDE OPTIONAL FEATURES \*/

data sample;

set sastut.online;

exists = find(remarks, '.com', 'i'); /\* i means modifier - ignore cases - caps or lower \*/

if exists > 0 then output;

run;

/\* giving starting position \*/

data sample;

string = 'hanni minni goin gaa dimb babi illin aan dimb';

/\* we want to look where dimb occurs \*/

exists = find(string, 'dimb');

run;

/\* we see that dimb occurs first at 22 position \*/

/\* we want to search after 22nd position \*/

data sample;

string = 'hanni minni goin gaa dimb babi illin aan dimb';

exists = find(string, 'dimb', 25); /\* search after 25th position \*/

run;

## /\* PRXMATCH - search for a pattern in a string \*/

/\* getting all entries where remarks start with letter 's' or 'c' \*/

data test;

set sastut.online;

flag = prxmatch("/^s|^c/", remarks); /\* we need to put expression in double quote and in double forward slash // \*/

run;

/\* all flag values where expression is satisfied is 1 \*/

/\* if we want case insesitive \*/

data test;

set sastut.online;

flag = prxmatch("/^s|^c/i", remarks); /\* put an 'i' after closing forward slash / \*/

run;

/\* if we want flag where digit value occurs \*/

data test;

set sastut.online;

flag = prxmatch("/\d/", remarks); /\* use \d - flag variabe here will also give the position\*/

run;

## SEPARATING/EXTRACTING USING PRXMATCH

/\* let us test for SEPARATING mobile numbers \*/

/\* create data first \*/

data mobile;

input comments $70.;

cards;

he called yesterday and his phone number is 5148885643. Thats it

9888885431 is her phone number. Call her and ask for $450

you dont know her number? it is 6543219999. Thanks! now go call her

;

run;

/\* now we extract mobile numbers \*/

data extract;

set mobile;

mobile\_num\_pos = prxmatch("/\d{10}/", comments); /\* 10 digit digit number \*/

mobile\_number = substr(comments, mobile\_num\_pos, 10); /\* from the specified position to 10 characters \*/

run;

# /\* data type conversion \*/

/\* INPUT function - char to num \*/

/\* PUT function - num to char \*/

data sample;

set sastut.online;

acct\_num = put(acctno, best.); /\* converts num to char \*/

acc\_num = input(acct\_num, best.); /\* converts char to num \*/

run;

# DATES

## /\* current date TODAY(), DAY(), WEEKDAY(), WEEK(), MONTH(), QTR() and YEAR() functions \*/

data dates;

format curr\_date date9.; /\* to get in date format - if not used you will get a number \*/

curr\_date = today();

run;

/\* previous day's date \*/

data dates;

format curr\_date date9.; /\* to get in date format - if not used you will get a number \*/

curr\_date = today()-1;

day = day(curr\_date); /\* get day of the month \*/

weekday = weekday(curr\_date); /\* which day of the week 1-7 with 1 as sunday\*/

week = week(curr\_date); /\* which week of the year 1-52 range\*/

month = month(curr\_date); /\* which month of the year \*/

quarter = qtr(curr\_date); /\* quarter of year 1-4 \*/

year = year(curr\_date); /\* 4 digit year value \*/

run;

## /\* creating date from month day year \*/

/\* MDY() function \*/

data sample;

format next\_day date9.;

next\_day = mdy(12, 04, 2022);

run;

## /\* INFORMAT and FORMAT functions \*/

## Informat

data test;

input date;

cards;

01/01/1957

;

run;

/\* we get missing value \*/

/\* SAS get confused as / is a spl char \*/

/\* we use informat to fix this \*/

/\* informat can be a method for SAS to read the data \*/

data test;

input date ddmmyy10.; /\* ddmmyy10. is informat \*/

cards;

01/01/1961

;

run;

/\* SAS will count the number of days after 01/01/1960 and give here \*/

/\* if date occurs before 01/01/1960, SAS will give number of days before this date with a -ve sign \*/

/\* use format to fix the issue \*/

## FORMAT

data test;

input date ddmmyy10.;

format date ddmmyy10.;

cards;

01/01/1961

;

run;

/\* lets try date9. format \*/

data test;

input date ddmmyy10.;

format date date9.;

cards;

01/01/1961

;

run;

/\* note that informat has to match the data input structure, whereas format does not have to match \*/

## /\* INTCK calculates difference betweek 2 dates, times or datetimes \*/

data sample;

format addmission\_date discharge\_date application\_date current\_date date9.;

addmission\_date= '10jan2020'd;

discharge\_date= '12may2020'd;

application\_date= '12jun2020'd;

current\_date= today();

run;

/\* lets do \*/

data test;

set sample;

days = intck('day', addmission\_date, discharge\_date);

months = intck('month', addmission\_date, discharge\_date);

run;

/\* optional statements in intck \*/

/\* discrete option - diff between 31 dec 2019 and 1 jan 2020 will be 1 month - default method \*/

/\* continous option - diff between 31 dec 2019 and 1 jan 2020 will be 0 months \*/

data test;

set sample;

format start\_date end\_date date9.;

start\_date = '31dec2019'd;

end\_date = '01jan2020'd;

months\_cont = intck('month', start\_date, end\_date, 'C');

months\_discrete = intck('month', start\_date, end\_date, 'D');

run;

## /\* INTNX option - increments a date, time or datetime value by a given interval \*/

/\* we want to increase number of days by 20 units from a given date - lets see \*/

data dates;

format first\_visit second\_visit third\_visit fourth\_visit date9.;

first\_visit = '10oct2020'd;

second\_visit = intnx('day', first\_visit, 20);

third\_visit = intnx('day', second\_visit, 20);

fourth\_visit = intnx('day', third\_visit, 20);

run;

/\* let us use INTNX options \*/

data alignment;

format start endd end\_beginning end\_middle end\_end end\_sameday date9.;

start = '25oct2020'd;

endd = intnx('month', start, 3);

end\_middle = intnx('month', start, 3, 'm');

end\_beginning = intnx('month', start, 3, 'b');

end\_end = intnx('month', start, 3, 'e');

end\_sameday = intnx('month', start, 3, 's');

run;

# /\* EXPORTING IN SAS \*/

proc export data=sastut.customers

outfile = "/home/u62306491/sasuser.v94/SAS\_tut\_files/test\_export.xlsx"

dbms = xlsx replace;

sheet = "you\_can\_give\_sheet\_name";

run;

/\* doing it in CSV \*/

proc export data=sastut.customers

outfile = "/home/u62306491/sasuser.v94/SAS\_tut\_files/test\_export.csv"

dbms = csv replace;

run;

## /\* how to xport mult tables in one excel file? \*/

/\* provide same file address but different sheet names \*/

proc export data=sastut.customers

outfile = "/home/u62306491/sasuser.v94/SAS\_tut\_files/test\_export.xlsx"

dbms = xlsx replace;

sheet='customers';

run;

/\* run the proc steps one by one \*/

proc export data=sastut.online

outfile = "/home/u62306491/sasuser.v94/SAS\_tut\_files/test\_export.xlsx"

dbms = xlsx replace;

sheet='online';

run;

/\* run the proc steps one by one \*/

proc export data=sastut.orderdetails

outfile = "/home/u62306491/sasuser.v94/SAS\_tut\_files/test\_export.xlsx"

dbms = xlsx replace;

sheet='order\_details';

run;

# COMBINING tables

/\* if a column is exactlly the same between 2 datasets \*/

/\* lets create 2 tables to check this \*/

data table1;

set sastut.cust(keep = contactname transaction\_amount obs=5);

run;

data table2;

set sastut.cust(keep = contactname city country obs=5);

run;

## using set command

data combined;

set table1;

set table2;

run;

/\* this will combine the 2 datasets with a common and exactly same column in both \*/

/\* if we have 2 tables with same column name but different values, still this would work but \*/

/\* data from column (with common column name) of 2nd table will be written in resulting table \*/

/\* create another table \*/

data table3;

set sastut.cust (keep = contactname city country obs=10 firstobs=6);

run;

data combined;

set table1;

set table3;

run;

/\* this table is not correct \*/

/\* use only 1 set statement to fix this \*/

data combined;

set table1

table3;

run;

/\* this is not what we want - we want to merge the tables \*/

/\* there should be atleast 2 tables \*/

/\* one of the variables (columns) should be common between the 2 tables \*/

/\* both tables must be sorted by the common column \*/

/\* let us create a table to be used in this excercise \*/

data table4;

set sastut.cust (keep = contactname city country obs=8 firstobs=4);

run;

## MERGE tables

/\* now let us merge \*/

/\* first we sort \*/

proc sort data=table1; by contactname; run;

proc sort data=table4; by contactname; run;

data combined;

merge table1 table4;

by contactname;

run;

/\* let us say we want values only based on table 1 in the contactname variable \*/

/\* this is also called left merge \*/

data combined;

merge table1(in=a) table4(in=b); /\* in command is used to name a table and should be used when merging based on a condition like this \*/

by contactname;

if a;

run;

/\* right merge \*/

data combined;

merge table1(in=a) table4(in=b); /\* in command is used to name a table and should be used when merging based on a condition like this \*/

by contactname;

if b;

run;

/\* now we want only those observations which are common in both tables \*/

/\* inner merge \*/

data combined;

merge table1(in=a) table4(in=b); /\* in command is used to name a table and should be used when merging based on a condition like this \*/

by contactname;

if a and b;

run;

/\* outer merge \*/

data combined;

merge table1(in=a) table4(in=b); /\* in command is used to name a table and should be used when merging based on a condition like this \*/

by contactname;

if a or b;

run;

## /\* APPEND function – stacking tables \*/

/\* let us create tables required first \*/

data table5;

set sastut.cust (keep= contactname transaction\_amount city country obs=5);

run;

data table6;

set sastut.cust (keep= contactname transaction\_amount city country obs=10 firstobs=6);

run;

/\* lets append \*/

proc append base=table5 data=table6; run;

proc print data = table5; run;

/\* keep in mind that here the base dataset gets changed \*/

/\* an option for APPEND procedure is the FORCE statement (e.g. if column names are not same in 2 tables) \*/

/\* let us try without force first \*/

proc append base=table1 data=table3; run;

/\* we get error - use FORCE now \*/

proc append base=table1 data=table3 force; run;

proc print data=table1; run;

/\* keep in mind, only columns from base table will be available in output table \*/

/\* Appending all tables with similar initial name (e.g. starts with CUST) into one table within library \*/

data combined\_all;

set sastut.cust:; /\* put a colon after \*/

run;

# PROC SQL

/\* PROC SQL - structured query language \*/

/\* if we need 2 columns from a dataset \*/

proc sql;

select model, type from sashelp.cars;

quit;

/\* select all columns \*/

proc sql;

select \* from sashelp.cars; /\* all variables \*/

quit;

## conditional starements

/\* conditional statements - WHERE \*/

proc sql;

select model, type, invoice, weight from sashelp.cars

where type = 'SUV' and weight > 5000;

quit;

## /\* summarize tables – use group by here\*/

proc sql;

select type

,count(type) as num\_units

,sum(invoice) as total\_invoice

,sum(weight) as total\_weight

from sashelp.cars

group by type;

quit;

## /\* conditional with group by \*/

proc sql;

select type

,count(type) as num\_units

,sum(invoice) as total\_invoice

,sum(weight) as total\_weight

from sashelp.cars

where weight >= 5000

group by type;

quit;

## /\* filtering data \*/

proc sql;

select type

,count(type) as num\_units

,sum(invoice) as total\_invoice

,sum(weight) as total\_weight

from sashelp.cars

group by type

where num\_units < 50;

quit;

/\* this will give error as num\_units is not existing in source data (CARS data) \*/

/\* use HAVING instead of WHERE \*/

proc sql;

select type

,count(type) as num\_units

,sum(invoice) as total\_invoice

,sum(weight) as total\_weight

from sashelp.cars

group by type

having num\_units <= 50;

quit;

## /\* ORDER BY clause – similar to sort \*/

proc sql;

select type

,count(type) as num\_units

,sum(invoice) as total\_invoice

,sum(weight) as total\_weight

from sashelp.cars

group by type

having num\_units <= 50

order by num\_units;

quit;

/\* ORDER BY reverse order \*/

proc sql;

select type

,count(type) as num\_units

,sum(invoice) as total\_invoice

,sum(weight) as total\_weight

from sashelp.cars

group by type

having num\_units <= 50

order by num\_units desc;

quit;

## /\* CASE WHEN similar to IF ELSE \*/

proc sql;

select \*

,case when weight>=5000 then 'heavy'

else 'light'

end as weight\_class

from sashelp.cars;

quit;

/\* 3 categories \*/

proc sql;

select \*

,case when weight>=5000 then 'heavy'

when weight < 5000 and weight >=3000 then 'medium'

else 'light'

end as weight\_class

from sashelp.cars;

quit;

/\* multiple columns based on different conditions can be made \*/

proc sql;

select \*

,case when weight>=5000 then 'heavy'

when weight < 5000 and weight >=3000 then 'medium'

else 'light'

end as weight\_class

,case when msrp > 35000 and msrp < 40000 and horsepower < 250 then 'expensive'

else 'ok or luxury'

end as price\_class

from sashelp.cars;

quit;

## /\* summarizing with COUNT FREQ N \*/

proc sql;

select type

,count(\*) as count\_of\_obs /\* \* will count all obs \*/

,freq(type)as frequency /\* non missing value will not be counted \*/

,n(type) as num\_obs

from sashelp.cars

group by type; /\* we can also write group by 1 - 1st variable \*/

quit;

/\* results show there are no missing value \*/

## /\* MIN MAX AVG MEAN RANGE STD SUM \*/

proc sql;

select type

,min(invoice) as minimum\_val

,max(invoice) as maximum\_val

,mean(invoice) as mean\_val

,avg(invoice) as avg\_val

,range(invoice) as range\_val

,std(invoice) as std\_val

,sum(invoice) as sum\_val

from sashelp.cars

group by type; /\* we can also write group by 1 - 1st variable \*/

quit;

/\* by 2 variables group and summarize \*/

proc sql;

select type, drivetrain

,min(invoice) as minimum\_val

,max(invoice) as maximum\_val

,mean(invoice) as mean\_val

,avg(invoice) as avg\_val

,range(invoice) as range\_val

,std(invoice) as std\_val

,sum(invoice) as sum\_val

from sashelp.cars

group by type, drivetrain; /\* we can also write group by 1 - 1st variable \*/

quit;

## /\* JOINS \*/

/\* first we make tables required for this ex \*/

data class1;

set sashelp.class (keep= name sex age obs=8);

run;

data class2;

set sashelp.class (keep=name height weight obs=15 firstobs=6);

run;

/\* left join \*/

proc sql;

select a.name /\* tables are defined as a and b, so selecting the variables within them \*/

,a.sex

,a.age

,b.height

,b.weight

from class1 as a

left join class2 as b

on a.name = b.name;

quit;

/\* alternate \*/

proc sql;

select a.\* /\* since all vars/cols are selected for table a, we can use '\*' \*/

,b.height

,b.weight

from class1 as a

left join class2 as b

on a.name = b.name;

quit;

/\* right join \*/

proc sql;

select a.\* /\* since all vars/cols are selected for table a, we can use '\*' \*/

,b.height

,b.weight

from class1 as a

right join class2 as b

on a.name = b.name;

quit;

/\* inner join \*/

proc sql;

select a.\* /\* since all vars/cols are selected for table a, we can use '\*' \*/

,b.height

,b.weight

from class1 as a

inner join class2 as b

on a.name = b.name;

quit;

/\* full join \*/

proc sql;

select a.\* /\* since all vars/cols are selected for table a, we can use '\*' \*/

,b.height

,b.weight

from class1 as a

full join class2 as b

on a.name = b.name;

quit;

/\* a merge requires a common col name, but join just requires a column with common shared values - name can be diff \*/

/\* merge needs the tables to be sorted on the common variable - not join \*/

## /\* cross join or cartesian join \*/

/\* table 1 has 3 rows, table 2 has 2 rows, crossjoin will have 3 x 2 = 6 rows \*/

/\* let us make tables for this ex \*/

data personal;

input name $ sex $ age;

cards;

John M 23

Jane F 21

James M 22

;

run;

data physical;

input height;

cards;

5.25

5.5

5.8

;

run;

/\* now let's join \*/

proc sql;

select a.\*

,b.height

from personal as a

cross join physical as b; /\* no need to provide 'on' statement in cross join \*/

quit;

## output result of sql step in a table

/\* what of we have to put the result of a sql statement in a table \*/

/\* let us look at example bellow \*/

proc sql;

create table combined as /\* add this statement with a table name – other steps are the same\*/

select a.\*

,b.height

from personal as a

cross join physical as b; /\* no need to provide 'on' statement in cross join \*/

quit;