SAS CODE
INTRODUCTION AND SIMPLE PROCEDURES

1 **Set Library**

```
libname MYLIB "C:/FOLDER/";
```

Library name = MYLIB

Input Data

```
data MYLIB.FILE;
infile 'C:/Folder/FILE.EXT';
input $1-10 Var1 $11-20 Var2 21-30 Var3 31-33 Var4;
run;
```

ex. 'G:\Files\Airlines\flights.txt'

Input Data (2)

```
data MYLIB.FILE
infile 'C:/folder/file.ext'
input Var1 $ 1-15 Var2 $16-30;
format Var5 mmddyy10.;
label Var5='Date';
run;
```

```
input @1 Var1 $15. @30 Var3 3.;
```

2 **Import Excel**

```
PROC IMPORT OUT= MYLIB.FILE
DATAFILE= "G:\Files\...\File.xls"
DBMS=EXCEL REPLACE;    SHEET="Sheet1$";
GETNAMES=YES;
MIXED=YES;
SCANTEXT=YES;
USEDATE=YES;
run;
```

Name of Output SAS file
Name of Input Excel file
Sheet(tab) name
Use the first row to get variable name (Generally YES)
(Generally YES) since most data is not clean

SAS CODE

PROC MEANS

Simple Means

```
proc means
data=MYLIB.FILE
MISSING
mean;
class Var1;
var Var3;
output out=MYLIB.FILE2 mean=;
run;
```

Input dataset
consider missing values as valid values
mean function
for each variable
by this variable
Output dataset

Sums by grouping

```
proc means data=MYLIB.FILE sum;
by Var1 Var2;
output out=MYLIB.FILE2 sum=;
run;
```

Input dataset, sum function
data must be sorted by Var1 var2
Output dataset

Mean Functions:

```
mean, std , var
min, max, range
sum
median or P50, Q1 or P25, Q3 or P75
QRANGE
P1, P5, P10, P90, P95, P99
(other functions available)
```

arithmetic mean (average), standard deviation, variance
minimum, maximum, range
sum
median, and quartiles
Quantile Range
percentiles

PROC TRANSPOSE

reshaping data
(wide to long)

```
proc transpose data=MYLIB.FILE1 out=MYLIB.FILE2
(drop = _Name_ );
By Var1;
ID Var2;
Var Var3;
run;
```

Input and output datasets
drop _name_ variable created in the process
New row headings
New column headings
data to be transposed

SAS CODE

DATA MANIPULATION

Functions

	eq	=	equal to
	ne	^=, ~=	not equal to
	gt	>	greater than
	lt	<	less than
	ge	>=	greater than or equal to
	le	<=	less than or equal to
where Var1 in ('CAT' 'DOG');	in( )		equal to one of a list
where Var1 like 'H_RS%';	_		one character
	%		any number of characters
where Var3 is missing <null>;			missing values

Formats

w.d		number width and number of decimals
$w		character width
Commaw.d		Commas, width and number of decimals
Dollarw.d		Dollar sign, width and number of decimals
mmddyy6.		101002
mmddyy8.		10/10/02
mmddyy10.		10/10/2002
Date9.		10OCT2002
Worddate.		October 10, 2002
Weekdate.		Sunday, October 10, 2002

Dates Options

Today()		todays date
MDY(Var8[month],Var9[day],var10[year])		uses numeric values and creates SAS date value
Year(SAS date)	QTR(..), month(..), weekday(..)	gives year from SAS Date
	ddMMMyyyy'd	Date constant

Formatting Options

State names	Var2 = stnamel(Var1); put state=;		
	MonthVar=month(SasDateVar); YearVar=Year(SasDateVar);		
(Zip Codes)	ZipCode = put(Zip,z5.);	Leading Zeroes	Formats numbers to include leading zeroes ie. Zip 6 will appear as Zipcode 00006
upcase(Var1)	NAME = upcase (name);		puts character variable in caps
propcase(Var1)	Name = propcase(name);		puts character variable in Proper Format (first letter Upper Case)

	SAS CODE	
Sort and Delete	proc sort data=MYLIB.FILE; by Var1 Var2 Var3; data Mylib.File2; set Mylib.File; by; if first.Var3 then delete;	
Lag	LagVar = lag(Var1);	
Erasing first row after lag	proc sort data=my.lib; by factory; if first.factory then delete; run;	
Finding text within data	if find(Var1,"Apple") gt 0;	If Apple is within Var1, find = 1; find is gt 0, so observation is kept
Replacing text within variable	tranwrd(Var1,"word","replacement");	finds "word" in Var1, replaces it with "replacement"

SAS CODE

DATA STEP

Data Step

```
data MYLIB.FILEOUT (drop<keep>= Var8 Var9);
set MYLIB.FILEIN;
length Var1 $11;
drop Var3;
Var6 = Var5 - Var3;
<Var6=sum(Var5,Var3);>
if upcase(Var2)='MUTT' then Var 11= Var4 / Var3;
else then Var 11 = Var4;
if ... then do;  ...; end;
else do; ...; end;
run;
```

drop/keeps only selected variables

define length of a variable
removes variable from new data set
Adds. Missing values lead to missing values
sum, mean, stand. Deviation — ignores missing values
if ... then delete; — if command, upcase converts to uppercase
else comand

Merging Sets

```
data MYLIB.NEWFILE;
set MYLIB.FILE1 MYLIB.FILE2;
<set  ML.F1 ML.FL2 (rename= (V2=Var2));>
by Var1;
run;
```

merges two datasets
corrects variable name of dataset to be merged
merges in order, but sets are to be pre-ordered independently

MACROS

Simple Macro

```
%macro Name(data,x,y,z);
data &data; set  &x;
if Var1=&y then Var2=&z;
run; quit;
% mend

%Name(mylib.file3, mylib.file2, 5 , 7)
```

Do Loops

```
data Hours;
do x = 1 to 24 by  1;
Hour = x||":00";
end;
drop x; run;
```

creates dataset Hours (or any other) using do loops

dataset with 24 obs
creates Hour = 3:00 from x = 3

REGRESSIONS

SAS CODE

Regular Regression

```
proc reg data=mylib.file;
model VarY = VarX1 VarX2 VarX3 ;
Title "Regressing X on Y";
run;
```

GLM Regression

```
proc glm data=mylib.file;
class VarX1;                                  fixed-effect variable
model VarY = VarX1 varX2 varX3  / solution;
run;
```

GLM Regression
(Fixed Effects)

```
proc glm data=mylib.file;
absorb varX2;                                 class variable with too many unique obs or estimates not needed
class varX1;                                  fixed-effect variable
model varY = varX1 varX2 varX3  varX4 / solution;
run;
```

SAS CODE

SQL

Simplest Sql

```
proc sql;                                   begins sql statements
create table mylib.file2 as                 name new file
select var1, var3                           choose variables separated by commas
from mylib.file2;                           original dataset
quit;                                       ends sql statements
```

Complex Sql

```
proc sql;
create table mylib.file2 as                 name new file
select unique var1, sum ( var3) as Total_var   unique restricts results   new variable that will be calculated from var3
from mylib.file2                            original dataset
where var1 = "January"                      condition restricting input dataset
group by var1, var2                         grouping order
having Total_var  = min (Total_var)         condition restricting output dataset
order by var1, var2 desc;                   desc = descending order   order condition
quit;
```

Inbedded SQL

```
proc sql;
create table mylib.file2 as
select unique *, sum(var3) as M_Tot from
(select *, sum(var3) as Y_Tot from file1 group by var1)
where var1 in ( 2008 , 2009 ) and
var2 not  in (select unique var2 from file 0)
group by var1, var2;
quit;
```

Cartesian join

```
proc sql;
create table mylib.cartesian as             name of new file
select *                                    select all (*)
from mylib.file1, mylib.file2;              original files
quit;                                       ** if file1 is X obs long, and file2 is Y obs long
                                            then cartesian is XY obs long.
```

Left join

```
proc sql;
create table mylib.join as                  name of new file
select * from
(select  *, trim(var1)||trim(var2) as var3
from mylib.file1 ) as a
left join
(select * from mylib.file2) as b
on a.var3 =  b.var3 and a.var4 = b.var4;
quit;
```

Calculating Straight line (Crow-fly) distances in SAS

You need to have two SAS datasets with longitude and latitude (Origins and Destinations), and each one has to have the variables named differently.

Example. The Origins dataset will have the following variables:

Facility1
Latitude1
Longitude1

And the Destinations will have:

Facility2
Latitude2
Longitude2

You then need to create a dataset that has every origin matched with every destination. To create this dataset you create a Cartesian join using proc sql. This resulting dataset can get very big, its size will be the number of observations of the dataset multiplied by each other.

Example. If the Origins dataset has 2 locations {A and B} and Destinations has 3 locations {1 2 3}, you would end up with a dataset with 6 observations (3 * 2). The dataset would have the following pairs of locations:

{A1 A2 A3 B1 B2 B3}

```
proc sql;
create table Cartesian as
select *
from Origins, Destinations;
quit;
```

Once the Cartesian dataset is created we do the calculation with the formula below:

```
data Calculation;
set Cartesian;
Distance =
3963 * ARCOS ((SIN (atan (1) / 45 * Latitude1) * SIN (atan (1) / 45 *
Latitude2) + COS (atan (1)/45* Latitude1) * COS (atan (1)/45 *
Latitude2) * COS (atan (1)/45 * ABS (Longitude2 - Longitude1)));
run;
```

Same two steps in a single proc sql statement.

```
proc sql;
create table Calculation as
select *, 3963 * ARCOS (SIN (atan (1) / 45 * Latitude2) * SIN (atan (1)
/ 45 * Latitude1) + COS (atan (1)/45* Latitude2) * COS (atan (1)/45 *
Latitude1) * COS (atan (1)/45 * ABS (Longitude2 - Longitude1))) as
Distance
from (select * from Origins, Destinations);
quit;
```