University of Mumbai

Practical Journal of
Research in Computing
&
Data Science

M.Sc. (Information Technology)
Part-I

Submitted by

KARKHANIS YASH SANDESH

Seat No: 3254075

DEPARTMENT OF INFORMATION TECHNOLOGY
PILLAI HOC COLLEGE OF ARTS, SCIENCE &COMMERCE, RASAYANI
(Affiliated to Mumbai University)
RASAYANI, 410207
MAHARASHTRA

2022-2023

Mahatma Education Society's

Pillai Hoc College of Arts, Science & Commerce, Rasayani

(Affiliated to Mumbai University)

RASAYANI – MAHARASHTRA - 410207

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the experiment work entered in this journal is as per the syllabus in **M.Sc. (Information Technology) Part-I, Semester-I**; class prescribed by University of Mumbai for the subject **Research in Computing** was done in computer lab of Mahatma Education Society's Pillai HOC College of Arts, Science & Commerce, Rasayani by **YASH KARKHANIS** during Academic year 2022-2023.

Exam Seat No: 3254075

Subject In-Charge **Coordinator**

External Examiner **Principal**

Date: **College Seal**

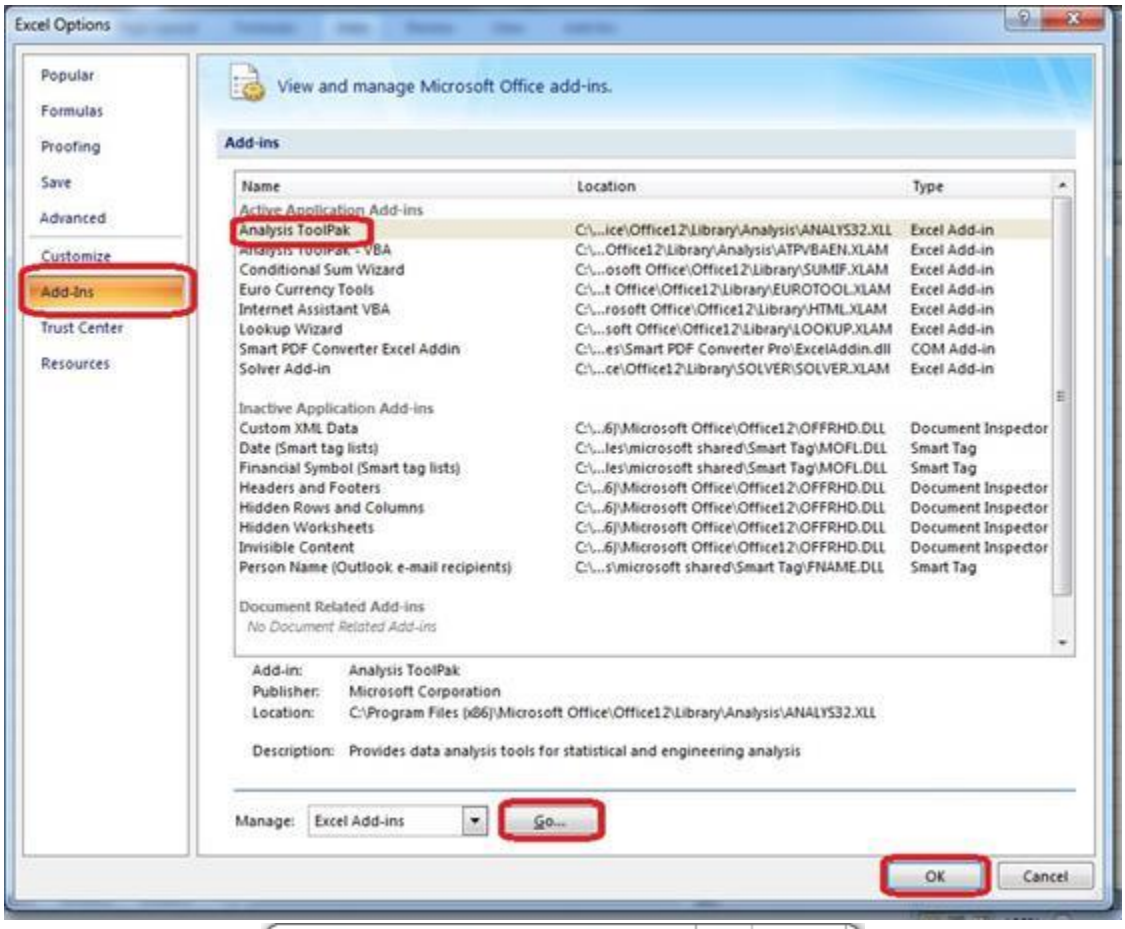
RESEARCH IN COMPUTING

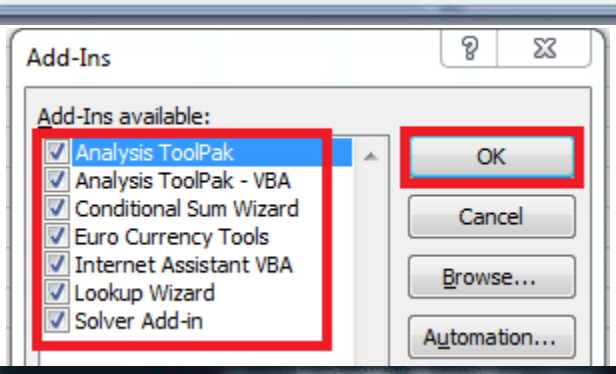
INDEX

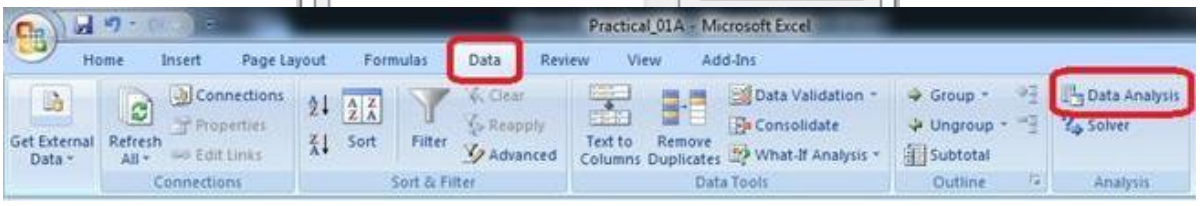
Practical No. 01

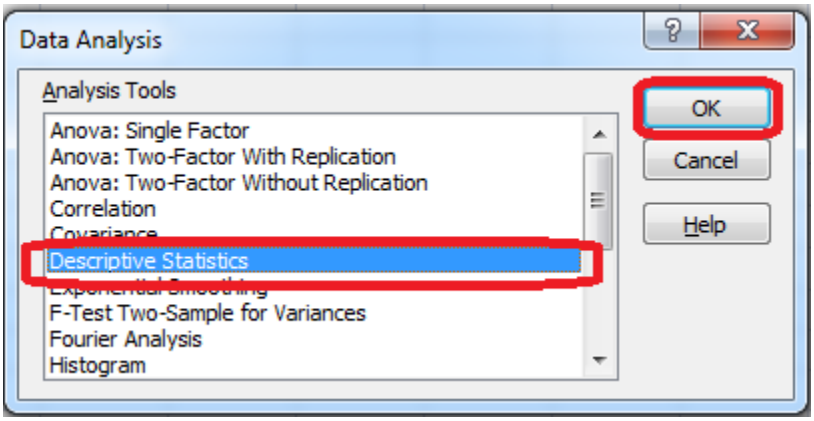
Aim: A) Write a program for obtaining descriptive statistics of data.

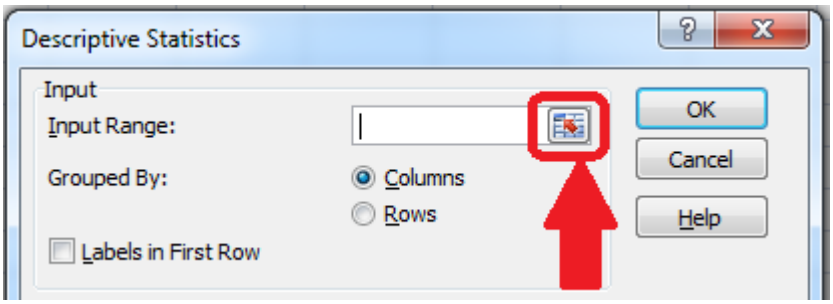
Using Excel
Go to File Menu □ Options □ Add-Ins□ Select Analysis ToolPak □ Press OK





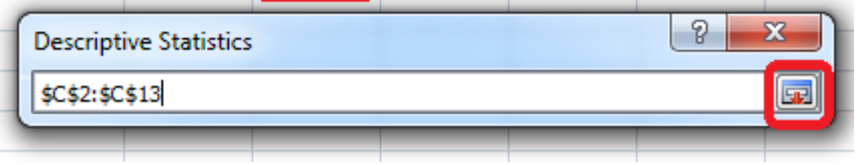


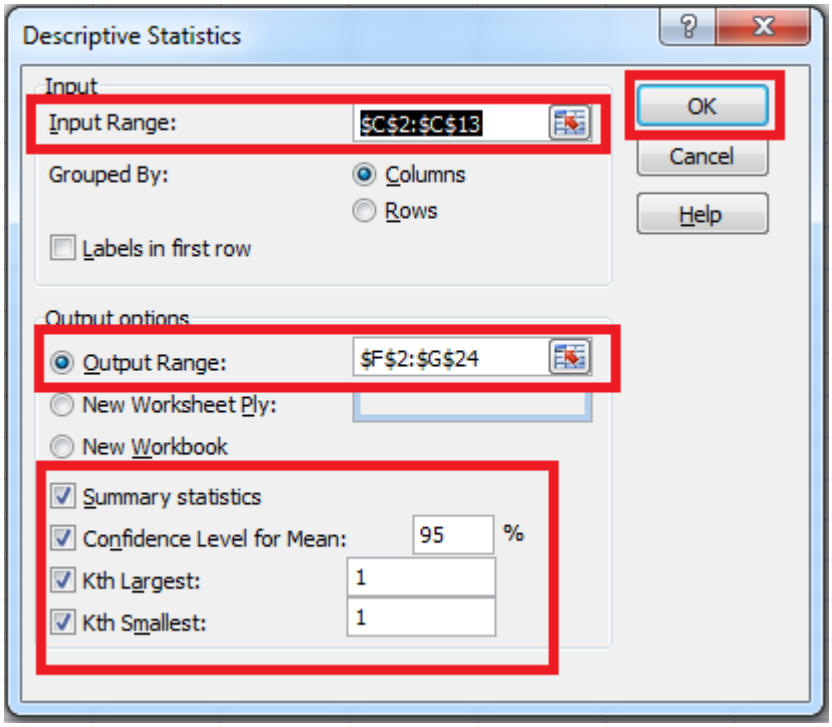




Select the data range from the excel worksheet.

	A	B	C	D	E	F	G
1	Sr. No	Name	Age	Rating			
2	1	AA	25	4.23			
3	2	BB	26	3.24			
4	3	CC	25	3.98			
5	4	DD	23	2.56			
6	5	EE	30	3.2			
7	6	FF	29	4.6			
8	7	GG	23	3.8			
9	8	HH	34	3.78			
10	9	II	40	2.98			
11	10	JJ	30	4.8			
12	11	KK	51	4.1			
13	12	LL	46	3.65			
14							
15							
16							
17							





Output:

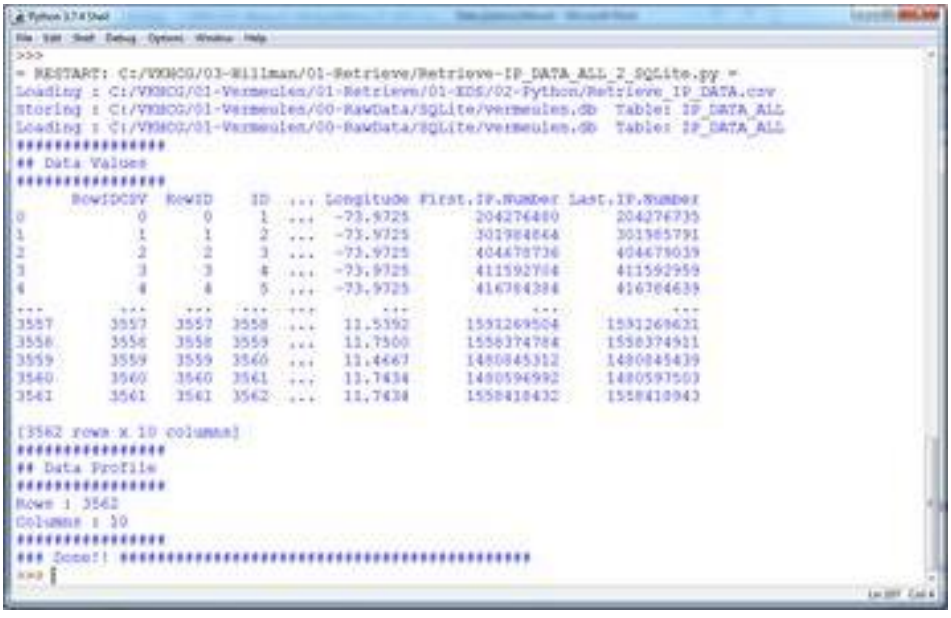
	A	B	C	D	E	F	G
1	Sr. No	Name	Age	Rating			
2	1	AA	25	4.23		*Column1*	
3	2	BB	26	3.24			
4	3	CC	25	3.98		Mean	31.83333
5	4	DD	23	2.56		Standard Error	2.665246
6	5	EE	30	3.2		Median	29.5
7	6	FF	29	4.6		Mode	25
8	7	GG	23	3.8		Standard Deviation	9.232682
9	8	HH	34	3.78		Sample Variance	85.24242
10	9	II	40	2.98		Kurtosis	0.24931
11	10	JJ	30	4.8		Skewness	1.135089
12	11	KK	51	4.1		Range	28
13	12	LL	46	3.65		Minimum	23
14						Maximum	51
15						Sum	382
16						Count	12
17						Largest(1)	51
18						Smallest(1)	23
19						Confidence Level(95.0%)	5.866167

B) Import data from different data sources (from Excel, csv, mysql, sql server, oracle to R/Python/Excel)

SQLite:

```
########################################## ######################
# -*- coding: utf-8 -*-
######################################################################
import sqlite3 as sq import pandas as pd
######################################################################
Base='C:/VKHCG'
sDatabaseName=Base + '/01-Vermeulen/00-RawData/SQLite/vermeulen.db' conn =
sq.connect(sDatabaseName)
######################################################################
sFileName='C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
print('Loading :',sFileName)
IP_DATA_ALL_FIX=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL_FIX.index.names = ['RowIDCSV']
sTable='IP_DATA_ALL'
print('Storing :',sDatabaseName,' Table:',sTable) IP_DATA_ALL_FIX.to_sql(sTable, conn,
if_exists="replace") print('Loading :',sDatabaseName,' Table:',sTable)
TestData=pd.read_sql_query("select * from IP_DATA_ALL;", conn)
print('#################')
print('## Data Values') print('#################')
print(TestData) print('#################')
print('## Data Profile') print('#################')
print('Rows :',TestData.shape[0]) print('Columns :',TestData.shape[1])
print('#################')
print('### Done!! ############################################')
```

Output:



Microsoft Excel

```
##################Retrieve-Country-Currency.py
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
importos
import pandas as pd
#####################################################################
Base='C:/VKHCG'
#####################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python' #if not os.path.exists(sFileDir):
#os.makedirs(sFileDir)
#####################################################################
CurrencyRawData = pd.read_excel('C:/VKHCG/01-Vermeulen/00-
RawData/Country_Currency.xlsx') sColumns = ['Country or territory', 'Currency', 'ISO-4217']
CurrencyData = CurrencyRawData[sColumns] CurrencyData.rename(columns={'Country or
territory': 'Country', 'ISO-4217': 'CurrencyCode'}, inplace=True)
CurrencyData.dropna(subset=['Currency'],inplace=True) CurrencyData['Country'] =
CurrencyData['Country'].map(lambda x: x.strip()) CurrencyData['Currency'] =
CurrencyData['Currency'].map(lambda x: x.strip())
CurrencyData['CurrencyCode'] = CurrencyData['CurrencyCode'].map(lambda x: x.strip())
print(CurrencyData)
print('~~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~ ')
#####################################################################
sFileName=sFileDir + '/Retrieve-Country-Currency.csv'
CurrencyData.to_csv(sFileName, index = False)
#####################################################################
```

Output:

Practical No. 02

Aim: A) Design a survey form for a given case study, collect the primary data and analyze it

Case 1:
A researcher wants to conduct a Survey in colleges on Use of ICT in higher education from Mumbai, Thane and Navi Mumbai. The survey focuses on access to and use of ICT in teaching and learning, as well as on attitudes towards the use of ICT in teaching and learning.
Design questionnaire addressed to teachers seeks information about the target class, his experience using ICT for teaching, access to ICT infrastructure, support available, ICT based activities and material used, obstacles to the use of ICT in teaching, learning activities with the target class, your skills and attitudes to ICT, and some personal background information.
Arrange question in following groups:

1. Information about the target class you teach
2. Experience with ICT for teaching
3. ICT access for teaching
4. Support to teachers for ICT use
5. ICT based activities and material used for teaching
6. Obstacles to using ICT in teaching and learning
7. Learning activities with the target class
8. Teacher skills
9. Teacher opinions and attitudes
10. Personal background information

Case 2:
A research agency wants to study the perception about App based taxi service in Mumbai, Thane and Navi Mumbai. The survey focuses on customers attitude towards app base taxi service as well as on attitudes towards regular taxi cab.
Design questionnaire seeks information about the target taxi service, his experience using taxi services, access, support available, obstacles and some personal background information, with the following objectives:

1. To find out the customer satisfaction towards the App based-taxi services.
2. To find the level of convenience and comfort with App based -taxi services.
3. To know their opinion about the tariff system and promptness of service.
4. To ascertain the customer view towards the driver behaviour and courtesy.
5. To provide inputs to enhance the services to delight the customers.
6. To examine relationship between service quality factors and taxi passenger satisfaction.
7. To suggest better regulations for transportation authorities regarding customer protection and effective monitoring of taxi services.

Case 3:

A popular electronic store want to conduct a survey to develop awareness of branded laptop baseline estimates and determine popularity of different company's laptop. It suggests steps to be initiated or strengthened in the field of demand in a region. The key indicators are among the general population, demand branded laptop and the problem users.

The objectives of this particular study are:-

1. To know the preferences of different types of branded laptops by students and professionals.
2. To study which factor influence for choosing different types of branded laptops.
3. To know about the level of satisfaction towards different types of branded laptops.
4. To identify the perception of consumers towards the laptop positioning strategy.
5. To know the consumer preference towards laptop in the present era

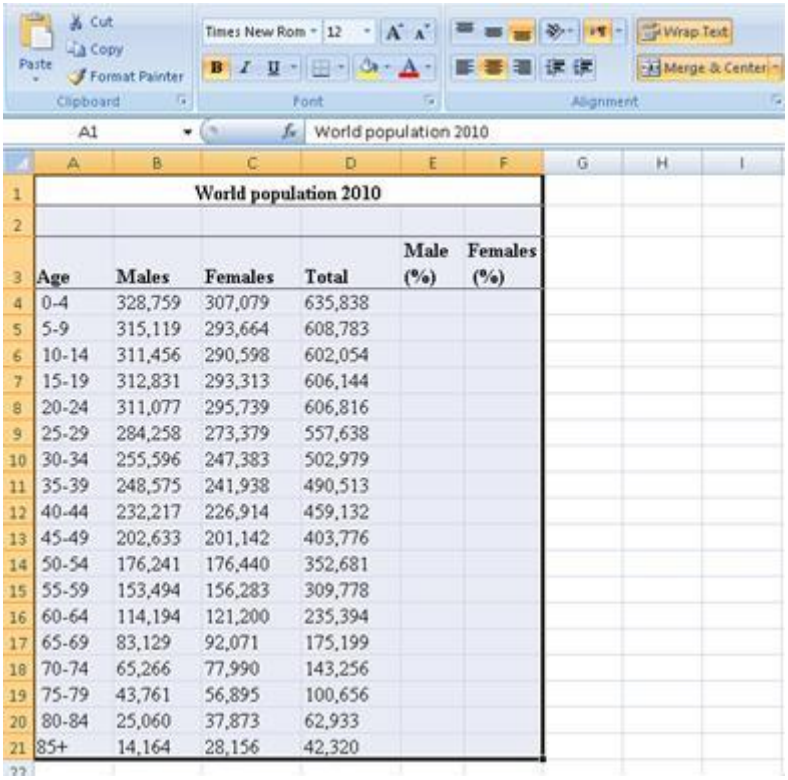
Use the collected data for analysis.

B) Perform analysis of given secondary data.

Steps in Secondary Data Analysis

1. Determine your research question – Knowing exactly what you are looking for.
2. Locating data– Knowing what is out there and whether you can gain access to it. A quick Internet search, possibly with the help of a librarian, will reveal a wealth of options.
3. Evaluating relevance of the data – Considering things like the data's original purpose, when it was collected, population, sampling strategy/sample, data collection protocols, operationalization of concepts, questions asked, and form/shape of the data.
4. Assessing credibility of the data – Establishing the credentials of the original researchers, searching for full explication of methods including any problems encountered, determining how consistent the data is with data from other sources, and discovering whether the data has been used in any credible published research.
5. Analysis – This will generally involve a range of statistical processes.

Example: Analyze the given Population Census Data for Planning and Decision Making by using the size and composition of populations.

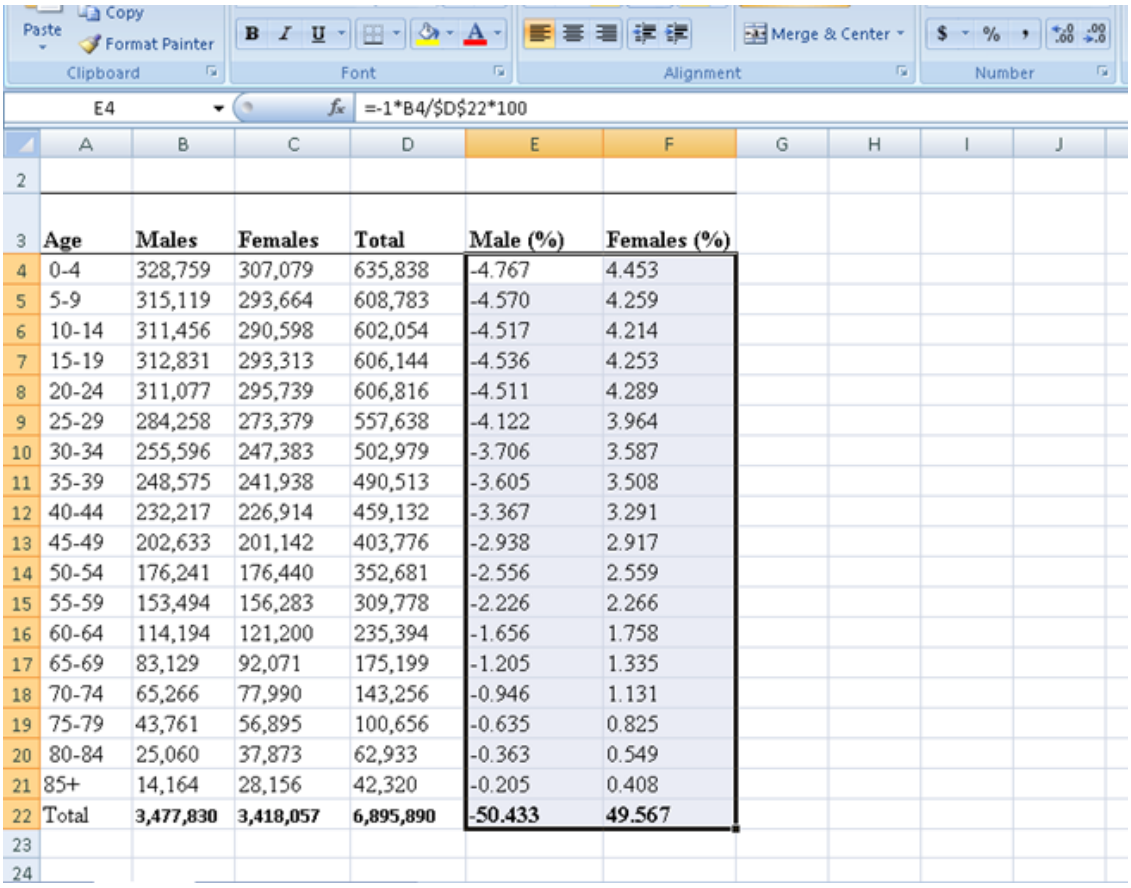


	A	B	C	D	E	F	G	H	I
1	World population 2010								
2									
3	Age	Males	Females	Total	Male (%)	Females (%)			
4	0-4	328,759	307,079	635,838					
5	5-9	315,119	293,664	608,783					
6	10-14	311,456	290,598	602,054					
7	15-19	312,831	293,313	606,144					
8	20-24	311,077	295,739	606,816					
9	25-29	284,258	273,379	557,638					
10	30-34	255,596	247,383	502,979					
11	35-39	248,575	241,938	490,513					
12	40-44	232,217	226,914	459,132					
13	45-49	202,633	201,142	403,776					
14	50-54	176,241	176,440	352,681					
15	55-59	153,494	156,283	309,778					
16	60-64	114,194	121,200	235,394					
17	65-69	83,129	92,071	175,199					
18	70-74	65,266	77,990	143,256					
19	75-79	43,761	56,895	100,656					
20	80-84	25,060	37,873	62,933					
21	85+	14,164	28,156	42,320					

Put the cursor in cell B22 and click on the AutoSum and then click Enter. This will calculate the total population. Then copy the formula in cell D22 across the row 22.

To calculate the percent of males in cell E4, enter the formula =-1*100*B4/$D$22 . And copy the formula in cell E4 down to cell E21.

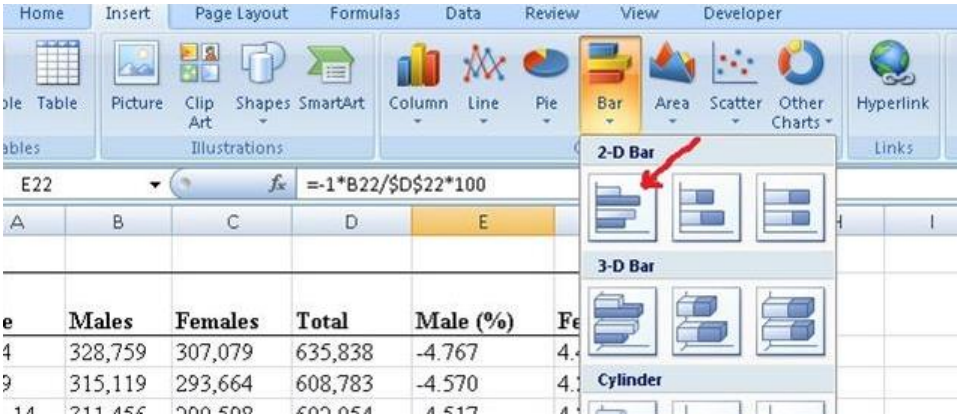
To calculate the percent of females in cell F4, enter the formula =100*C4/$D$22. Copy the formula in cell F4 down to cell F21.



Age	Males	Females	Total	Male (%)	Females (%)
0-4	328,759	307,079	635,838	-4.767	4.453
5-9	315,119	293,664	608,783	-4.570	4.259
10-14	311,456	290,598	602,054	-4.517	4.214
15-19	312,831	293,313	606,144	-4.536	4.253
20-24	311,077	295,739	606,816	-4.511	4.289
25-29	284,258	273,379	557,638	-4.122	3.964
30-34	255,596	247,383	502,979	-3.706	3.587
35-39	248,575	241,938	490,513	-3.605	3.508
40-44	232,217	226,914	459,132	-3.367	3.291
45-49	202,633	201,142	403,776	-2.938	2.917
50-54	176,241	176,440	352,681	-2.556	2.559
55-59	153,494	156,283	309,778	-2.226	2.266
60-64	114,194	121,200	235,394	-1.656	1.758
65-69	83,129	92,071	175,199	-1.205	1.335
70-74	65,266	77,990	143,256	-0.946	1.131
75-79	43,761	56,895	100,656	-0.635	0.825
80-84	25,060	37,873	62,933	-0.363	0.549
85+	14,164	28,156	42,320	-0.205	0.408
Total	**3,477,830**	**3,418,057**	**6,895,890**	**-50.433**	**49.567**

To build the population pyramid, we need to choose a horizontal bar chart with two series of data (% male and % female) and the age labels in column A as the Category X-axis labels. Highlight the range A3:A21, hold down the CTRL key and highlight the range E3:F21

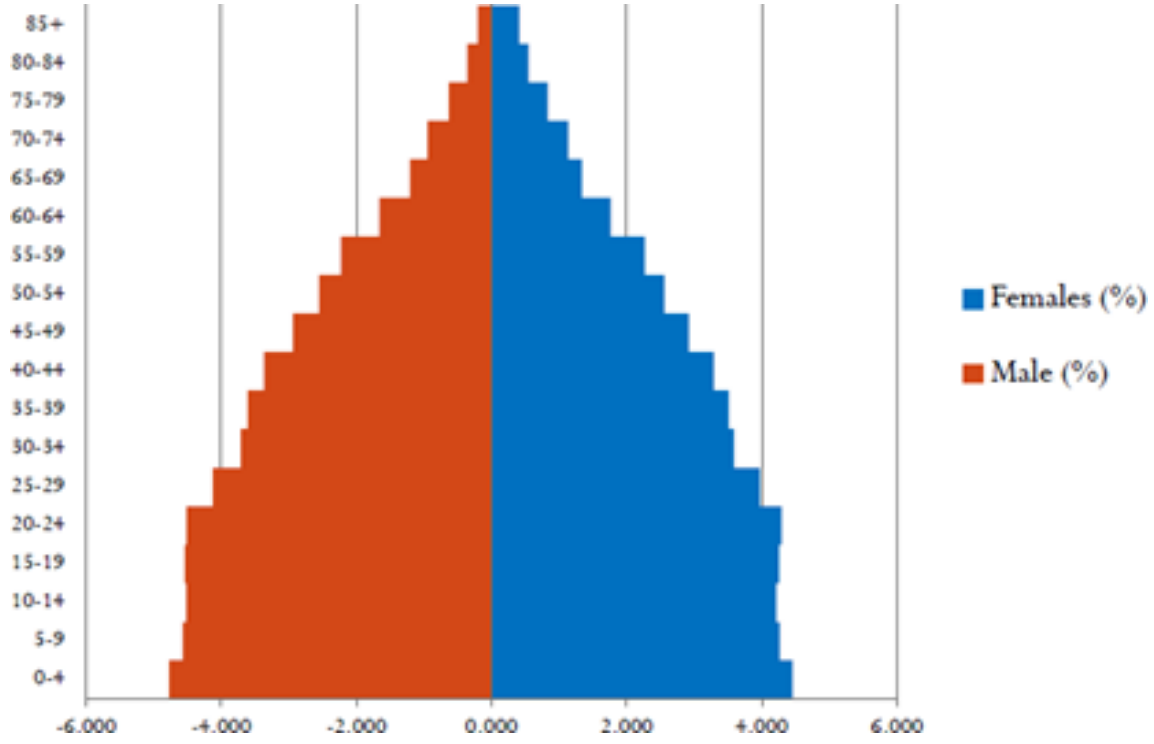
Under inset tab, under horizontal bar charts select clustered bar chart



Put the tip of your mouse arrow on the Y-axis (vertical axis) so it says “Category Axis”, right click and chose Format Axis

Choose Axis options tab and set the major and minor tick mark type to None, Axis labels to Low, and click OK.

Click on any of the bars in your pyramid, click right and select “format data series”. Set the Overlap to 100 and Gap Width to 0. Click OK.



Practical No. 03

Aim: A) Perform testing of hypothesis using one sample t-test.

One sample t-test : The One Sample t Test determines whether the sample mean is statistically different from a known or hypothesised population mean. The One Sample t Test is a parametric test.

Program Code:

```
fromscipy.stats import ttest_1samp
importnumpy as np
ages = np.genfromtxt('ages.csv')
print(ages)
ages_mean = np.mean(ages)
print(ages_mean)
tset, pval = ttest_1samp(ages, 30)
print('p-values - ',pval)
if pval< 0.05: # alpha value is 0.05
print(" we are rejecting null hypothesis")
else:
print("we are accepting null hypothesis")
```

Output:

```
In [4]: runfile('K:/Research In Computing/Practical Material/Programs/
Practical_05/Prac_3A.py', wdir='K:/Research In Computing/Practical Material/
Programs/Practical_05')
[20. 30. 25. 13. 16. 17. 34. 35. 38. 42. 43. 45. 48. 49. 50. 51. 54. 55.
 56. 59. 61. 62. 18. 22. 29. 30. 31. 39. 52. 53. 67. 36. 47. 54. 40. 40.
 35. 22. 59. 58. 30. 43. 22. 45. 21. 59. 51. 47. 25. 58. 50. 23. 24. 45.
 37. 59. 28. 28. 48. 42. 54. 36. 36. 24. 26. 24. 50. 48. 34. 44. 56. 55.
 35. 33. 39. 53. 34. 28. 56. 24. 21. 29. 28. 58. 35. 57. 26. 25. 59. 56.
 22. 57. 48. 33. 23. 26. 57. 32. 53. 31. 35. 44. 54. 25. 31. 58. 26. 32.
 26. 50. 41. 49. 26. 33. 34. 24. 43. 42. 51. 36. 38. 38. 40. 38. 56. 39.
 23. 33. 53. 30. 38.]
39.47328244274809
p-values -  5.362905195437013e-14
 we are rejecting null hypothesis
```

B) Write a program for t-test comparing two means for independent samples.

The t distribution provides a good way to perform one sample tests on the mean when the population variance is not known provided the population is normal or the sample is sufficiently large so that the Central Limit Theorem applies.

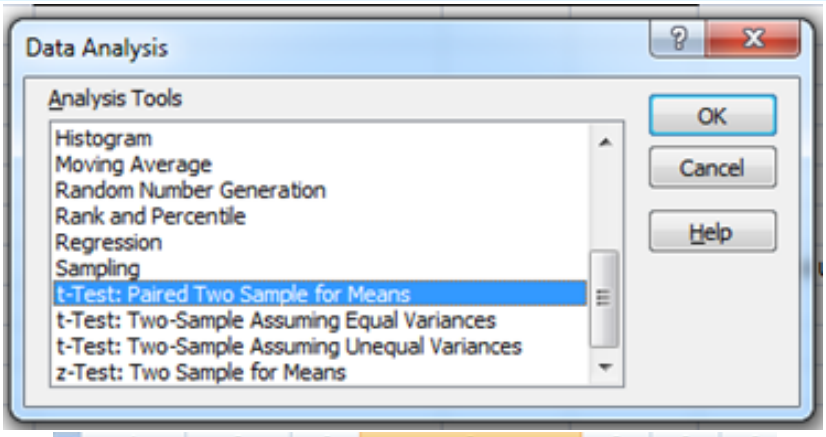
Two Sample t Test
Example: A college Princiapal informed classroom teachers that some of their students showedunusual potential for intellectual gains. One months later the students identified to teachers ashaving potentional for unusual intellectual gains showed significiantly greater gains performanceon a test said to measure IQ than did students who were not so identified. Below are the data forthe students:

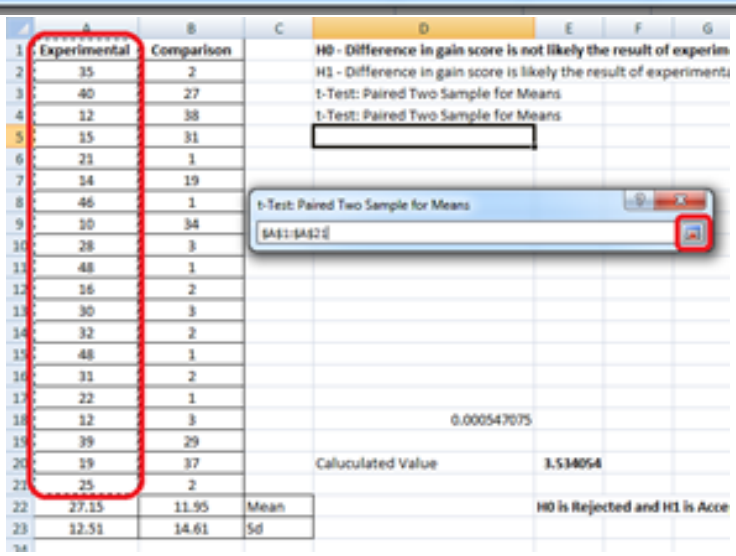
Experimental	Comparison	
35	2	
40	27	
12	38	
15	31	
21	1	
14	19	
46	1	
10	34	
28	3	
48	1	
16	2	
30	3	
32	2	
48	1	
31	2	
22	1	
12	3	
39	29	
19	37	
25	2	
27.15	11.95	Mean
12.51	14.61	Sd

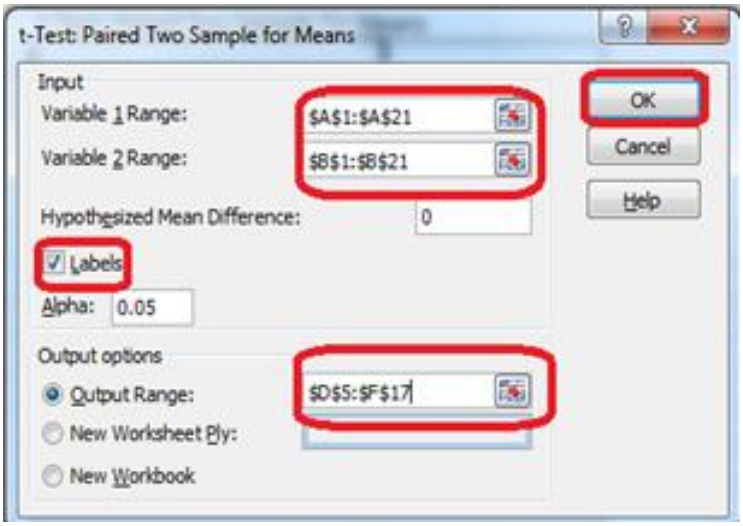
Experimental Data
To calculate Standard Mean go to cell A22 and type =SUM(A2:A21)/20
To calculate Standard Deviation go to cell A23 and type =STDEV(A2:A21)

Comparison Data
To calculate Standard Mean go to cell B22 and type =SUM(B2:B21)/20
To calculate Standard Deviation go to cell B23 and type =STDEV(B2:B21) To find T-Test Statistics go to data □Data Analysis


Practical_01A - Microsoft Excel
Home
Insert
Page Layout
Formulas
Data
Review
View
Add-Ins
Get External Data
Refresh All
Connections
Properties
Edit Links
Sort
Filter
Clear
Reapply
Advanced
Text to Columns
Remove Duplicates
Data Validation
Consolidate
What-If Analysis
Group
Ungroup
Subtotal
Data Analysis
Solver
Connections
Sort & Filter
Data Tools
Outline
Analysis


Data Analysis
Analysis Tools
Histogram
Moving Average
Random Number Generation
Rank and Percentile
Regression
Sampling
t-Test: Paired Two Sample for Means
t-Test: Two-Sample Assuming Equal Variances
t-Test: Two-Sample Assuming Unequal Variances
z-Test: Two Sample for Means
OK
Cancel
Help


Experimental
Comparison
H0 - Difference in gain score is not likely the result of experim
H1 - Difference in gain score is likely the result of experiment
t-Test: Paired Two Sample for Means
t-Test: Paired Two Sample for Means
0.000547075
Caluculated Value
3.534054
H0 is Rejected and H1 is Acce
Mean
Sd


t-Test: Paired Two Sample for Means
Input
Variable 1 Range:
$A$1:$A$21
Variable 2 Range:
$B$1:$B$21
Hypothesized Mean Difference:
0
Labels
Alpha:
0.05
Output options
Output Range:
$D$5:$F$17
New Worksheet Ply:
New Workbook
OK
Cancel
Help

To caluculate the T-Test square value go to cell E20 and type
=(A22-B22)/SQRT((A23*A23)/COUNT(A2:A21)+(B23*B23)/COUNT(A2:A21))

Now go to cell E20 and type
=IF(E20<E12,"H0 is Accepted", "H0 is Rejected and H1 is Accepted")

Our calculated value is larger than the tabled value at alpha = .01, so we reject the null hypothesisand accept the alternative hypothesis, namely, that the difference in gain scores is likely the resultof the experimental treatment and not the result of chance variation.

Output:

	A	B	C	D	E	F	G	H	I	J	K
1	**Experimental**	**Comparison**		**H0 - Difference in gain score is not likely the result of experimental treatment.**							
2	35	2		H1 - Difference in gain score is likely the result of experimental treatment and not the result of change variation.							
3	40	27		t-Test: Paired Two Sample for Means							
4	12	38		t-Test: Paired Two Sample for Means							
5	15	31		t-Test: Paired Two Sample for Means							
6	21	1									
7	14	19			*Experimental*	*Comparison*					
8	46	1		Mean	27.15	11.95					
9	10	34		Variance	156.45	213.5236842					
10	28	3		Observations	20	20					
11	48	1		Pearson Correlation	-0.395904927						
12	16	2		Hypothesized Mean Difference	0						
13	30	3		df	19						
14	32	2		t Stat	2.996289153						
15	48	1		P(T<=t) one-tail	0.003711226						
16	31	2		t Critical one-tail	1.729132792						
17	22	1		P(T<=t) two-tail	0.007422452						
18	12	3		t Critical two-tail	2.09302405						
19	39	29									
20	19	37		Caluculated Value	**3.534053898**						
21	25	2									
22	27.15	11.95	Mean		**H0 is Rejected and H1 is Accepted**						
23	12.51	14.61	Sd								

C) Perform testing of hypothesis using paired t-test.

The paired sample t-test is also called dependent sample t-test. It's an univariate test that tests for a significant difference between 2 related variables. An example of this is if you where to collect the blood pressure for an individual before and after some treatment, condition, or time point. The data set contains blood pressure readings before and after an intervention. These are variables "bp_before" and "bp_after".

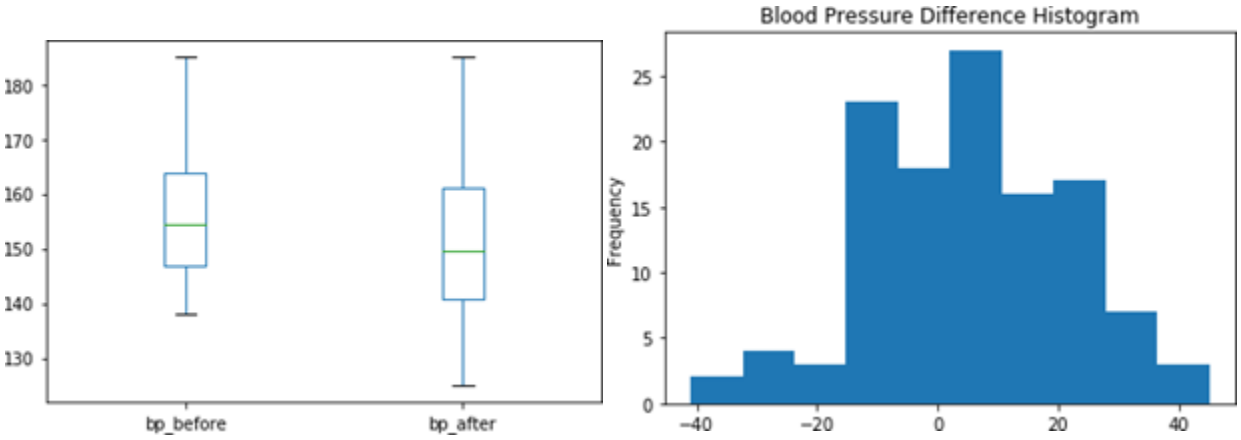
The hypothesis being test is:

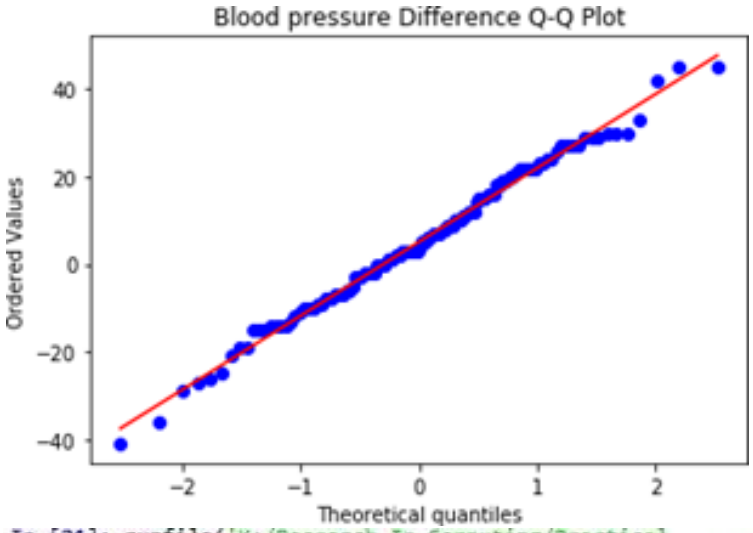
- H0 - The mean difference between sample 1 and sample 2 is equal to 0.
- H0 - The mean difference between sample 1 and sample 2 is not equal to 0

Program Code:

```
from scipy import stats
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("blood_pressure.csv")
print(df[['bp_before','bp_after']].describe())
df[['bp_before', 'bp_after']].plot(kind='box')
plt.savefig('boxplot_outliers.png')
df['bp_difference'] = df['bp_before'] - df['bp_after']
df['bp_difference'].plot(kind='hist', title= 'Blood Pressure Difference Histogram')
plt.savefig('blood pressure difference histogram.png')
stats.probplot(df['bp_difference'], plot= plt)
plt.title('Blood pressure Difference Q-Q Plot')
plt.savefig('blood pressure difference qq plot.png')
stats.shapiro(df['bp_difference'])
stats.ttest_rel(df['bp_before'], df['bp_after'])
```

Output:






```
In [21]: runfile('K:/Research In Computing/Practical
Material/Programs/Practical_03/Practical_03C.py', wdir='K:/
Research In Computing/Practical Material/Programs/
Practical_03')
        bp_before    bp_after
count  120.000000  120.000000
mean   156.450000  151.358333
std     11.389845   14.177622
min    138.000000  125.000000
25%    147.000000  140.750000
50%    154.500000  149.500000
75%    164.000000  161.000000
max    185.000000  185.000000
0.0011297914644840823
Ttest_relResult(statistic=3.3371870510833657,
pvalue=0.0011297914644840823)
reject null hypothesis
```

A paired sample t-test was used to analyze the blood pressure before and after the intervention to test if the intervention had a significant affect on the blood pressure. The blood pressure before the intervention was higher (156.45 ± 11.39 units) compared to the blood pressure post intervention (151.36 ± 14.18 units); there was a statistically significant decrease in blood pressure (t(119)=3.34, p= 0.0011) of 5.09 units.

Practical No. 04

Aim: A) Perform testing of hypothesis using chi-squared goodness- of-fit test.

Problem

Ansystem administrator needs to upgrade the computers for his division. He wants to know what sort of computer system his workers prefer. He gives three choices: Windows, Mac, or Linux. Test the hypothesis or theory that an equal percentage of the population prefers each type of computer system .

System	O	Ei	$\sum \frac{(O_i - E_i)^2}{Ei}$
Windows	20	**33.33 %**	
Mac	60	**33.33 %**	
Linux	20	**33.33 %**	

H0 : The population distribution of the variable is the same as the proposed distribution HA : The distributions are different

To calculate the Chi –Squred value for Windows go to cell D2 and type =((B2- C2)*(B2-C2))/C2

To calculate the Chi –Squred value for Mac go to cell D3 and type =((B3-C3)*(B3- C3))/C3

To calculate the Chi –Squred value for Mac go to cell D3 and type =((B4-C4)*(B4- C4))/C4

Go to Cell D5 for and type=SUM(D2:D4)

To get the table value for Chi-Square for α = 0.05 and dof = 2, go to cell D7 and type =CHIINV(0.05,2)

At cell D8 type =IF(D5>D7, "H0 Accepted","H0 Rejected")

Output:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	System	O	Ei	$\sum \frac{(O_i - E_i)^2}{Ei}$										
2	Windows	20	33.33	5.333333		H0 : The population distribution of the variable is the same as the proposed distribution								
3	Mac	60	33.33	21.33333		H1 - : The distributions are different								
4	Linux	20	33.33	5.333333										
5	Total	100	100	32										
6														
7			Table Value	5.991465										
8			H0 Accepted											

B) Perform testing of hypothesis using chi-squared test of independence.

In a study to understatnd the permormacne of M. Sc. IT Part -1 class, a college selects a random sample of 100 students. Each student was asked his grade obtained in B. Sc. IT. The sample is as given below

Sr. No	Roll No	Student's Name	Gen	Grade
1	1	Gaborone	m	O
2	2	Francistown	m	O
3	5	Niamey	m	O
4	13	Maxixe	m	O
5	16	Tema	m	O
6	17	Kumasi	m	O
7	34	Blida	m	O
8	35	Oran	m	O
9	38	Saefda	m	O
10	42	Constantine	m	O
11	43	Annaba	m	O
12	45	Bejaefa	m	O
13	48	Medea	m	O
14	49	Djelfa	m	O
15	50	Tipaza	m	O
16	51	Bechar	m	O
17	54	Mostaganem	m	O
18	55	Tiaret	m	O
19	56	Bouira	m	O
20	59	Tebessa	m	O
21	61	El Harrach	m	O
22	62	Mila	m	O
23	65	Fouka	m	O
24	66	El Eulma	m	O
25	68	SidiBel Abbes	m	O
26	69	Jijel	m	O
27	70	Guelma	m	O
28	85	Khemis El Khechna	m	O
29	87	Bordj El Kiffan	m	O
30	88	Lakhdaria	m	O
31	6	Maputo	m	D
32	12	Lichinga	m	D
33	15	Ressano Garcia	m	D
34	19	Accra	m	D
35	27	Wa	m	D
36	28	Navrongo	m	D
37	37	Mascara	m	D
38	44	Batna	m	D
39	57	El Biar	m	D
40	60	Boufarik	m	D
41	63	OuedRhiou	m	D
42	64	Souk Ahras	m	D
43	71	Dar El Befda	m	D
44	86	Birtouta	m	D
45	18	Takoradi	m	C
46	22	Cape Coast	m	C
47	29	Kwabeng	m	C
48	30	Algiers	m	C
49	31	Laghouat	m	C
50	39	Relizane	m	C
51	52	Setif	m	C
52	53	Biskra	m	C
53	67	Kolea	m	C
54	100	AefnFakroun	m	C
55	26	Nima	m	B
56	32	TiziOuzou	m	B
57	33	Chlef	m	B

Sr. No	Roll No	Student's Name	Gen	Grade
62	3	Maun	f	O
63	7	Tete	f	O
64	9	Chimoio	f	O
65	11	Pemba	f	O
66	14	Chibuto	f	O
67	25	Mampong	f	O
68	36	Tlemcen	f	O
69	40	Adrar	f	O
70	41	Tindouf	f	O
71	46	Skikda	f	O
72	47	Ouargla	f	O
73	10	Matola	f	D
74	20	Legon	f	D
75	21	Sunyani	f	D
76	72	Teenas	f	D
77	73	Kouba	f	D
78	75	HussenDey	f	D
79	77	Khenchela	f	D
80	82	HassiBahbah	f	D
81	84	Baraki	f	D
82	91	Boudouaou	f	D
83	95	Tadjenanet	f	D
84	4	Molepolole	f	C
85	8	Quelimane	f	C
86	23	Bolgatanga	f	C
87	58	Mohammadia	f	C
88	83	Merouana	f	C
89	24	Ashaiman	f	B
90	76	N'gaous	f	B
91	90	Bab El Oued	f	B
92	92	BordjMenael	f	B
93	93	Ksar El Boukhari	f	B
94	74	Reghaa	f	A
95	78	Cheria	f	A
96	79	Mouzaa	f	A
97	80	Meskiana	f	A
98	81	Miliana	f	A
99	94	Sig	f	A
100	99	Kadiria	f	A

Null Hypothesis - H0 : The performance of girls students is same as boys students. Alternate Hypothesis - H1 : The performance of boys and girls students are different. Open Excel Workbook

	O	A	B	C	D	Total	$\sum \frac{(O_i - E_i)^2}{Ei}$
Girls	11	7	5	5	11	**39**	6.075
Boys	30	4	3	10	14	**61**	6.075
Total	41	11	8	15	25	**100**	**12.150**
Ei	**20.5**	**5.5**	**4**	**7.5**	**12.5**	**50**	

Prepare a contingency table as shown above. To calculate Girls Students with ‘O’ Grade
Go to Cell N6 and type =COUNTIF($J$2:$K$40,"O")

To calculate Girls Students with ‘A’ Grade
Go to Cell O6 and type =COUNTIF($J$2:$K$40,"A")

To calculate Girls Students with ‘B’ Grade
Go to Cell P6 and type =COUNTIF($J$2:$K$40,"B")

To calculate Girls Students with ‘C’ Grade
Go to Cell Q6 and type =COUNTIF($J$2:$K$40,"C")

To calculate Girls Students with ‘D’ Grade
Go to Cell R6 and type =COUNTIF($J$2:$K$40,"D")

To calculate Boys Students with ‘O’ Grade
Go to Cell N7 and type =COUNTIF($D$2:$E$62,"O")

To calculate Boys Students with ‘A’ Grade
Go to Cell O7 and type =COUNTIF($D$2:$E$62,"A")

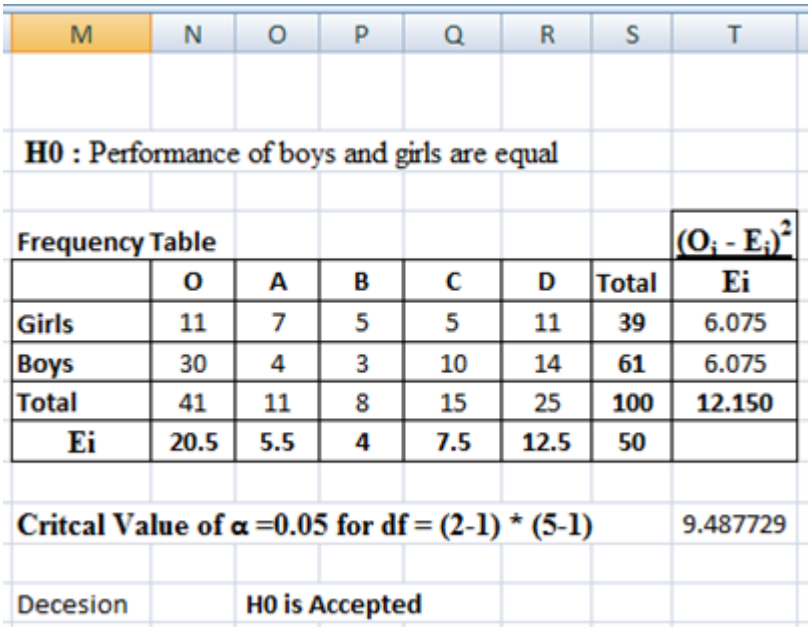
To calculate Boys Students with ‘B’ Grade
Go to Cell P7 and type =COUNTIF($D$2:$E$62,"B")

To calculate Boys Students with ‘C’ Grade
Go to Cell Q7 and type =COUNTIF($D$2:$E$62,"C")

To calculate Boys Students with ‘D’ Grade
Go to Cell R7 and type =COUNTIF($D$2:$E$62,"D")

To calculated the expected value Ei
Go to Cell N9 and type =N8/2 Go to Cell O9 and type =O8/2 Go to Cell P9 and type =P8/2 Go to Cell Q9 and type =Q8/2 Go to Cell R9 and type =R8/2
Go to Cell S6 and calculate total girl students = SUM(N6:R6) Go to Cell S7 and calculate total girl students = SUM(N7:R7)
Now Calculate
Go to cell T6 and type

=SUM((N6-$N$9)^2/$N$9,(O6-$O$9)^2/$O$9,(P6-$P$9)^2/$P$9,(Q6-Q$9)^2/$Q$9, (R6-$R$9)^2/$R$9)
Go to cell T7 and type
=SUM((N7-$N$9)^2/$N$9,(O7-$O$9)^2/$O$9,(P7-$P$9)^2/$P$9,(Q7-Q$9)^2/$Q$9, (R7-$R$9)^2/$R$9)
To get the table value go to cell T11 and type =CHIINV(0.05,4)
Go to cell O13 and type =IF(T8>=T11," H0 is Accepted", "H0 is Rejected")



M	N	O	P	Q	R	S	T
H0 : Performance of boys and girls are equal							
Frequency Table							$(O_i - E_i)^2$
	O	**A**	**B**	**C**	**D**	**Total**	**Ei**
Girls	11	7	5	5	11	**39**	6.075
Boys	30	4	3	10	14	**61**	6.075
Total	41	11	8	15	25	**100**	**12.150**
Ei	**20.5**	**5.5**	**4**	**7.5**	**12.5**	**50**	
Critcal Value of α =0.05 for df = (2-1) * (5-1)							9.487729
Decesion		**H0 is Accepted**					

Practical No. 05

Aim: Perform testing of hypothesis using Z-test.

Use a Z test if:

- Your sample size is greater than 30. Otherwise, use a t test.
- Data points should be independent from each other. In other words, one data point isn't related or doesn't affect another data point.
- Your data should be normally distributed. However, for large sample sizes (over 30) this doesn't always matter.
- Your data should be randomly selected from a population, where each item has an equal chance of being selected.
- Sample sizes should be equal if at all possible.

Ho - Blood pressure has a mean of 156 units

Program Code for one-sample Z test.

```
from statsmodels.stats import weightstats as stests
import pandas as pd
from scipy import stats
df = pd.read_csv("blood_pressure.csv")
df[['bp_before','bp_after']].describe()
print(df)
ztest ,pval = stests.ztest(df['bp_before'], x2=None, value=156)
print(float(pval))
if pval<0.05:
print("reject null hypothesis")
else:
print("accept null hypothesis")
```

Output:

```
In [26]: runfile('K:/Research In Computing/Practical
Material/Programs/Practical_05/Z_Test_One_Sample.py',
wdir='K:/Research In Computing/Practical Material/Programs/
Practical_05')
     patient  gender agegrp  bp_before  bp_after
0          1    Male  30-45        143       153
1          2    Male  30-45        163       170
2          3    Male  30-45        153       168
3          4    Male  30-45        153       142
4          5    Male  30-45        146       141
..       ...     ...    ...        ...       ...
115      116  Female    60+        152       152
116      117  Female    60+        161       152
117      118  Female    60+        165       174
118      119  Female    60+        149       151
119      120  Female    60+        185       163

[120 rows x 5 columns]
0.6651614730255063
accept null hypothesis
```

Two-sample Z test- In two sample z-test , similar to t-test here we are checking two independent data groups and deciding whether sample mean of two group is equal or not.

H0 : mean of two group is 0

H1 : mean of two group is not 0

```
import pandas as pd
from statsmodels.stats import weightstats as stests
df = pd.read_csv("blood_pressure.csv")
df[['bp_before','bp_after']].describe()
print(df)
ztest ,pval = stests.ztest(df['bp_before'], x2=df['bp_after'], value=0,alternative='two- sided')
print(float(pval))
if pval<0.05:
print("reject null hypothesis")
else:
print("accept null hypothesis")
```

Output:

```
In [29]: runfile('K:/Research In Computing/Practical
Material/Programs/Practical_05/Z_Test_Two_Sample.py',
wdir='K:/Research In Computing/Practical Material/Programs/
Practical_05')
     patient  gender agegrp  bp_before  bp_after
0          1    Male  30-45        143       153
1          2    Male  30-45        163       170
2          3    Male  30-45        153       168
3          4    Male  30-45        153       142
4          5    Male  30-45        146       141
..       ...     ...    ...        ...       ...
115      116  Female    60+        152       152
116      117  Female    60+        161       152
117      118  Female    60+        165       174
118      119  Female    60+        149       151
119      120  Female    60+        185       163

[120 rows x 5 columns]
0.002162306611369422
reject null hypothesis
```

Practical No. 06

Aim: Perform testing of hypothesis using One-way ANOVA.

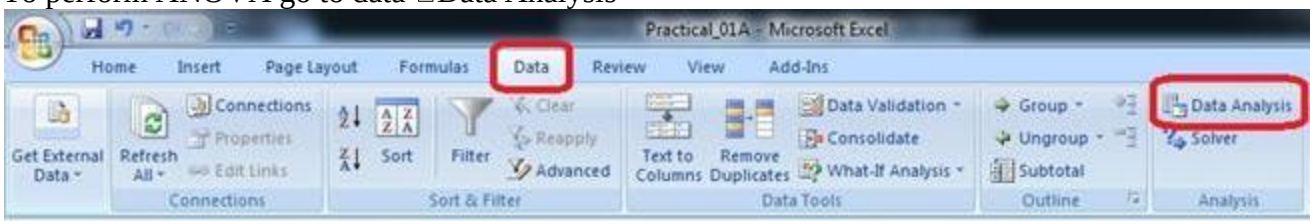
Using Excel

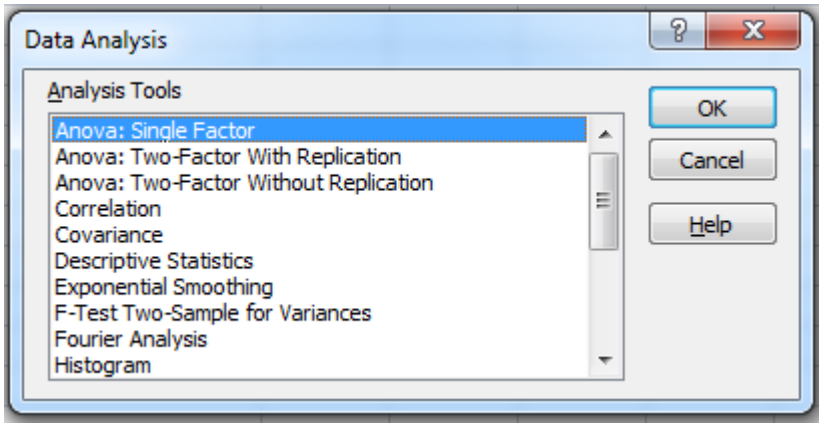
H0 - There are no significant differences between the Subject's mean SAT scores.

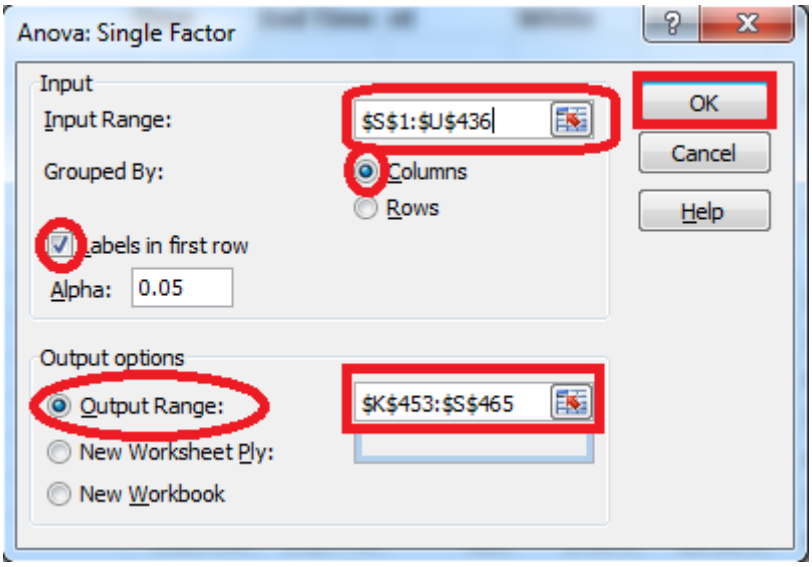
$$\mu 1 = \mu 2 = \mu 3 = \mu 4 = \mu 5$$

H1 - There is a significant difference between the Subject's mean SAT scores.

To perform ANOVA go to data □Data Analysis







Input Range: $S$1:$U$436 (Select columns to be analyzed in group)

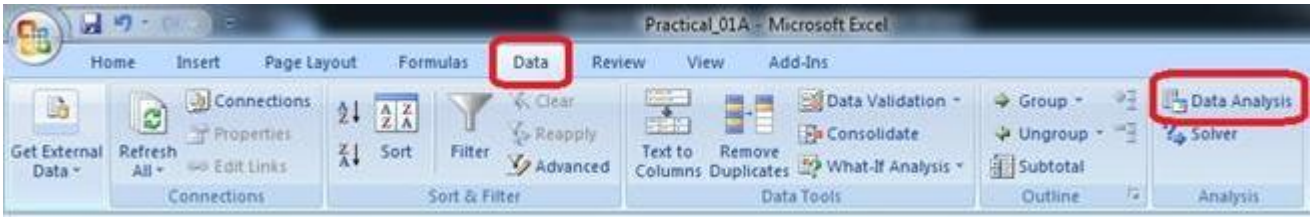
Output Range: $K$453:$S$465 (Can be any Range)

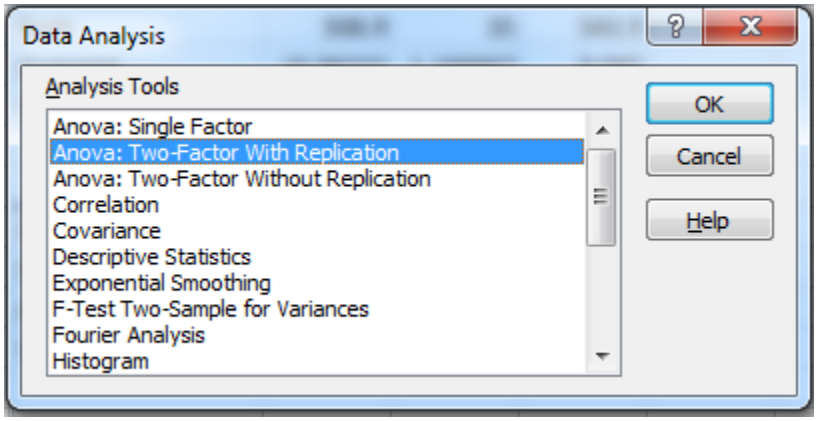
Anova: Single Factor						
SUMMARY						
Groups	*Count*	*Sum*	*Average*	*Variance*		
Average Score (SAT Math)	375	162354	432.944	5177.144		
Average Score (SAT Reading)	375	159189	424.504	3829.267		
Average Score (SAT Writing)	375	156922	418.4587	4166.522		
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	39700.57	2	19850.28	4.520698	0.01108	3.003745
Within Groups	4926677	1122	4390.977			
Total	4966377	1124				

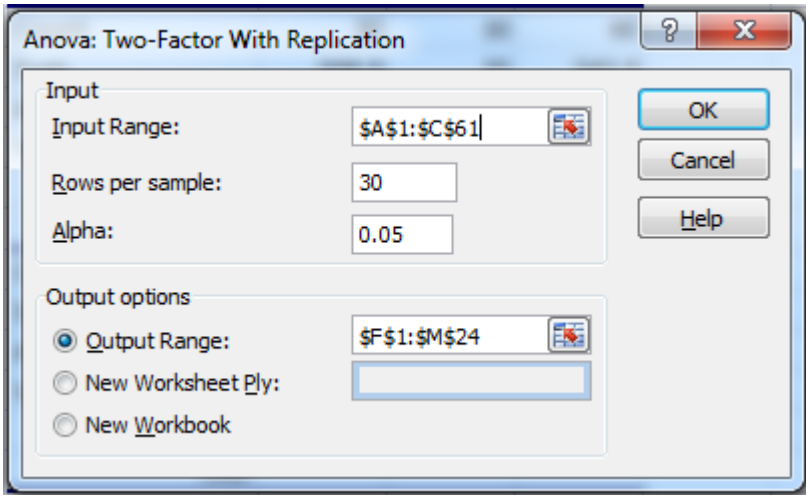
Since theresulting pvalueis less than 0.05. The null hypothesis (H0) is rejected and conclude that there is a significant difference between the SAT scores for each subject.

B) Perform testing of hypothesis using Two-way ANOVA.

Using Excel:
Go to Data tab Data Analysis







Rows Per Sample – 30 (Beacause 30 Patients are given each dose)
Alpha – 0.05
Output Range - $F$1:$M$24

Output:

Anova: Two-Factor With Replication						
SUMMARY	len	dose	Total			
1						
Count	30	30	60			
Sum	508.9	35	543.9			
Average	16.96333	1.166667	9.065			
Variance	68.32723	0.402299	97.22333			
31						
Count	30	30	60			
Sum	619.9	35	654.9			
Average	20.66333	1.166667	10.915			
Variance	43.63344	0.402299	118.2854			
Total						
Count	60	60				
Sum	1128.8	70				
Average	18.81333	1.166667				
Variance	58.51202	0.39548				
ANOVA						
Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Sample	102.675	1	102.675	3.642079	0.058808	3.922879
Columns	9342.145	1	9342.145	331.3838	8.55E-36	3.922879
Interaction	102.675	1	102.675	3.642079	0.058808	3.922879
Within	3270.193	116	28.19132			
Total	12817.69	119				

P-value = 0.0588079 column in the ANOVA Source of Variation table at the bottom of the output. Because the p-values for both medicin dose and interaction are less than our significance level, these factors are statistically significant. On the other hand, the interaction effect is not significant because its p-value (0.0588) is greater than our significance level. Because the interaction effect is not significant, we can focus on only the main effects and not consider the interaction effect of the dose.

C) Perform testing of hypothesis using MANOVA.

Excel:
Go to http://www.real-statistics.com/free-download/

1. Download Real Statistics Resource Pack

Real Statistics Resource Pack: contains a variety of supplemental functions and data analysis tools not provided by Excel. These complement the standard Excel capabilities and make it easier for you to perform the statistical analyses described in the rest of this website.

Free Download
Real Statistics Resource Pack

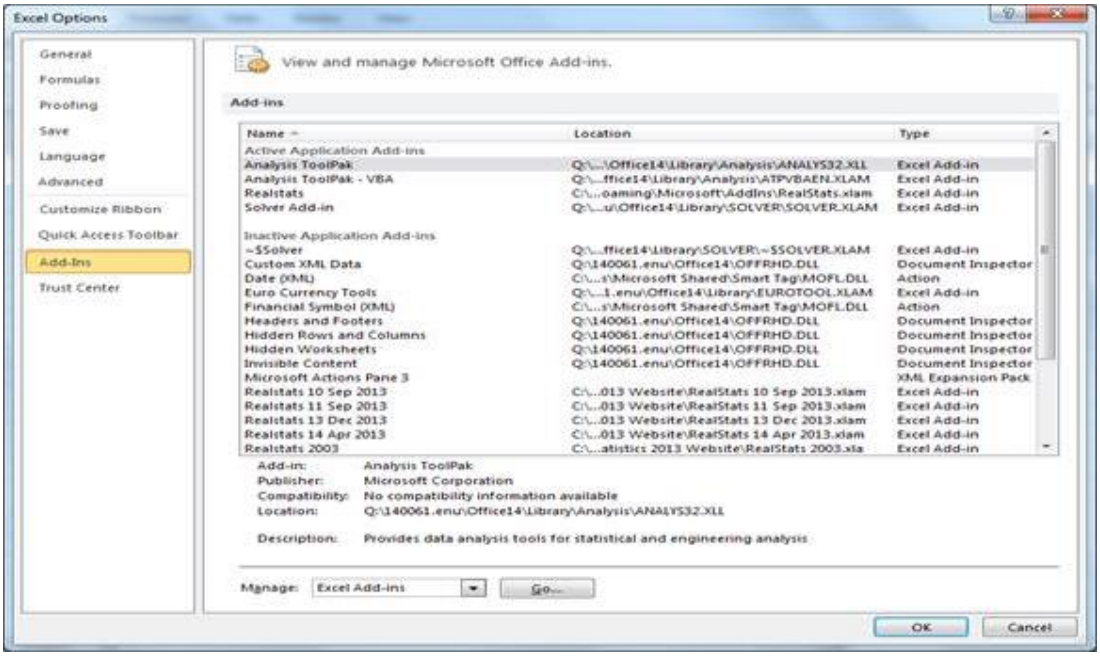
Real Statistics Resource Pack for Excel 2010, 2013, 2016, 2019 or 365 for Windows

If you accept the License Agreement, click here on Real Statistics Resource Pack for Excel 2010/2013/2016/2019/365 to download the latest Excel for Windows version of the

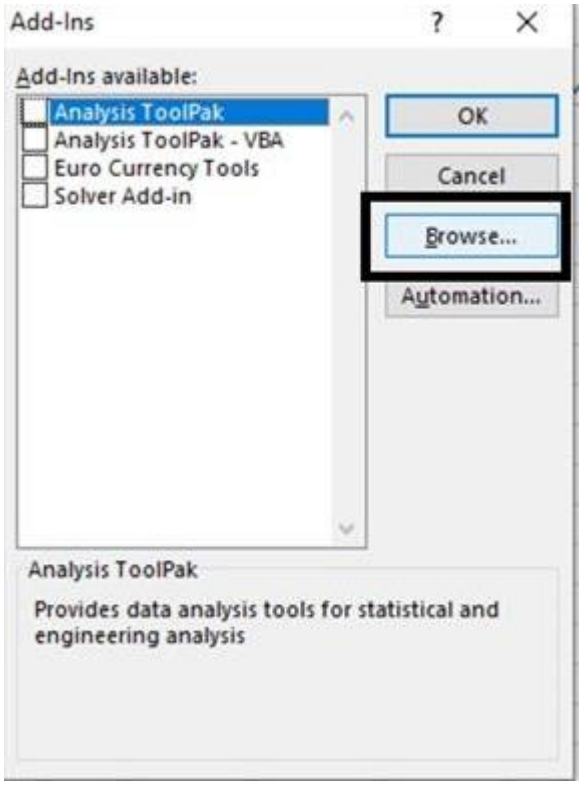
OR

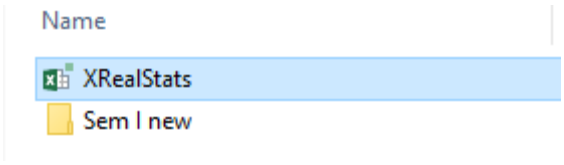
http://www.real-statistics.com/wp-content/uploads/2019/11/XRealStats.xlam

Install Add-in in excel. Select File > Help Options > Add-Ins and click on the Go button at the bottom of the window (see Figure 1).
Add-ins -> Analysis Pack -> Go

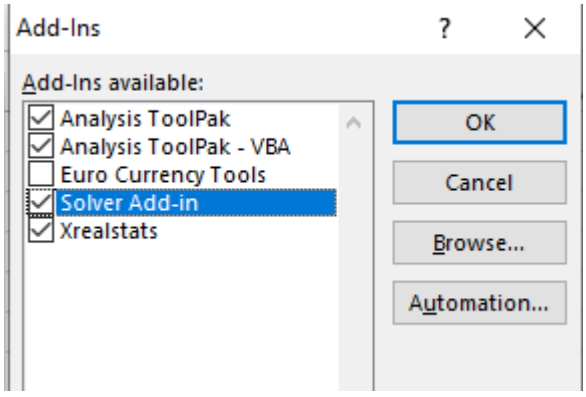


Click on browse and select XrealStats file (previously downloaded).





Select the following Add-Ins. Click OK.

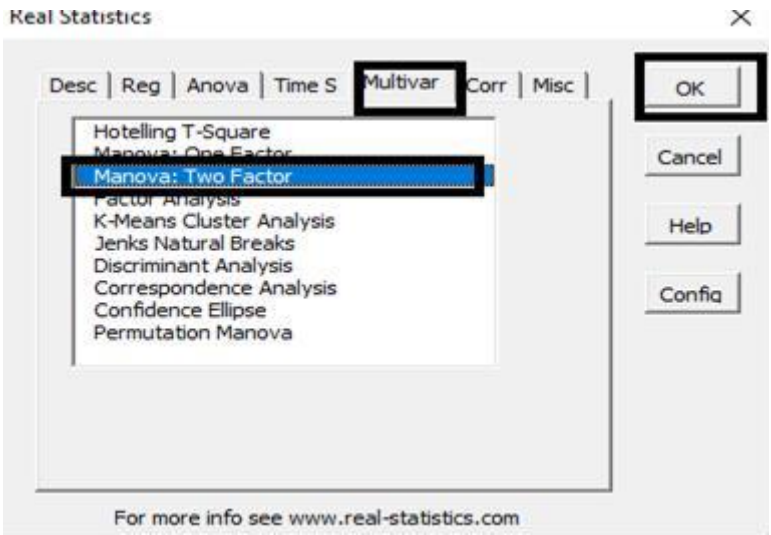


Now create an excel sheet with following data.

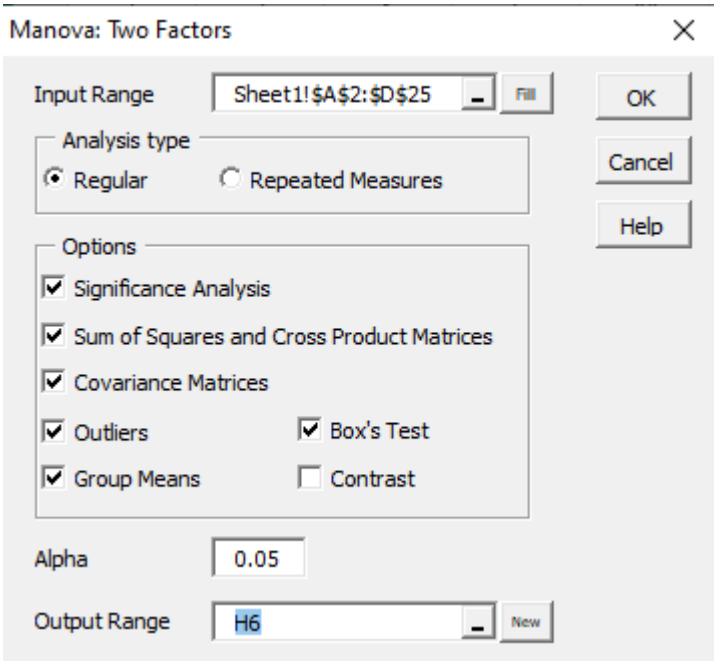
A study was conducted to see the impact of social-economic class (rich, middle, poor) and gender (male, female) on kindness and optimism using on a sample of 24 people based on the data in Figure 1.

	A	B	C	D
3	gender	economic	kindness	optimism
4	male	wealthy	5	3
5	male	wealthy	4	6
6	male	wealthy	3	4
7	male	wealthy	2	4
8	male	middle	4	6
9	male	middle	3	6
10	male	middle	5	4
11	male	middle	5	5
12	male	poor	7	5
13	male	poor	4	3
14	male	poor	3	1
15	male	poor	7	2
16	female	wealthy	2	3
17	female	wealthy	3	5
18	female	wealthy	5	3
19	female	wealthy	4	2
20	female	middle	9	8
21	female	middle	6	5
22	female	middle	7	6
23	female	middle	8	9
24	female	poor	8	9
25	female	poor	9	8
26	female	poor	3	7
27	female	poor	5	7

Press ctrl-m to open Real Statistics menu.



Select the data excluding column names. Select a cell for output.



Output:

Two-Way MANOVA							SSCP Matrices	
fact A	*stat*	*df1*	*df2*	*F*	*p-value*	*part eta-sq*	Tot	
Pillai Trac	0.190764	2	16	1.885866	0.183909	0.190764	104.9565	59.86957
Wilk's Lam	0.809236	2	16	1.885866	0.183909	0.190764	59.86957	110.6087
Hotelling	0.235733	2	16	1.885866	0.183909	0.190764		
Roy's Lg R	0.235733						Row (A)	
							12.5247	15.41502
fact B	*stat*	*df1*	*df2*	*F*	*p-value*	*part eta-sq*	15.41502	18.97233
Pillai Trac	0.340249	4	34	1.742501	0.163458	0.170125		
Wilk's Lam	0.8181	4	32	1.778757	0.157443	0.1819	Column (B)	
Hotelling	0.479878	4	30	1.799541	0.155008	0.193509	31.15295	22.95885
Roy's Lg R	0.448078						22.95885	19.37655

Practical No. 07

Aim: Perform the Random sampling for the given data and analyze it.

Example 1: From a population of 10 women and 10 men as given in the table in Figure 1 on the left below, create a random sample of 6 people for Group 1 and a periodic sample consisting of every 3rd woman for Group 2.

You need to run the sampling data analysis tool twice, once to create Group 1 and again to create Group 2. For Group 1 you select all 20 population cells as the Input Range and Random as the Sampling Method with 6 for the Random Number of Samples. For Group 2 you select the 10 cells in the Women column as Input Range and Periodic with Period 3.

Open existing excel sheet with population data Sample Sheet looks as given below:

	A	B	C	D	E	F	G	H	I	J	K
1	Sr. No	Roll No	Student's Name	Gender	Grade		Sr. No	Roll No	Student's Name	Gender	Grade
2	1	1	Gaborone	m	O		62	3	Maun	f	O
3	2	2	Francistown	m	O		63	7	Tete	f	O
4	3	5	Niamey	m	O		64	9	Chimoio	f	O
5	4	13	Maxixe	m	O		65	11	Pemba	f	O
6	5	16	Tema	m	O		66	14	Chibuto	f	O
7	6	17	Kumasi	m	O		67	25	Mampong	f	O
8	7	34	Blida	m	O		68	36	Tlemcen	f	O
9	8	35	Oran	m	O		69	40	Adrar	f	O
10	9	38	Saefda	m	O		70	41	Tindouf	f	O
11	10	42	Constantine	m	O		71	46	Skikda	f	O
12	11	43	Annaba	m	O		72	47	Ouargla	f	O
13	12	45	Bejaefa	m	O		73	10	Matola	f	D
14	13	48	Medea	m	O		74	20	Legon	f	D
15	14	49	Djelfa	m	O		75	21	Sunyani	f	D
16	15	50	Tipaza	m	O		76	72	Teenas	f	D
17	16	51	Bechar	m	O		77	73	Kouba	f	D
18	17	54	Mostaganem	m	O		78	75	Hussen Dey	f	D
19	18	55	Tiaret	m	O		79	77	Khenchela	f	D
20	19	56	Bouira	m	O		80	82	Hassi Bahbah	f	D
21	20	59	Tebessa	m	O		81	84	Baraki	f	D
22	21	61	El Harrach	m	O		82	91	Boudouaou	f	D
23	22	62	Mila	m	O		83	95	Tadjenanet	f	D
24	23	65	Fouka	m	O		84	4	Molepolole	f	C

Set Cell O1 = Male and Cell O2 = Female

To generate a random sample for male students from given population go to Cell O1 and type =INDEX(E$2:E$62,RANK(B2,B$2:B$62))

Drag the formula to the desired no of cell to select random sample.

Now, to generate a random sample for female students go to cell P1 and type
=INDEX(K$2:K$40,RANK(H2,H$2:H$40))
Drag the formula to the desired no of cell to select random sample.

Output:

O	P
Male	**Female**
A	A
A	A
A	A
B	A
C	B
C	C
D	C
D	C
D	C
D	C
D	D
D	A
D	B
D	B
O	D
O	D
O	D
O	D
O	O
O	O
O	O
O	O
O	A

Practical No. 08

Aim: Write a program for computing different correlation.

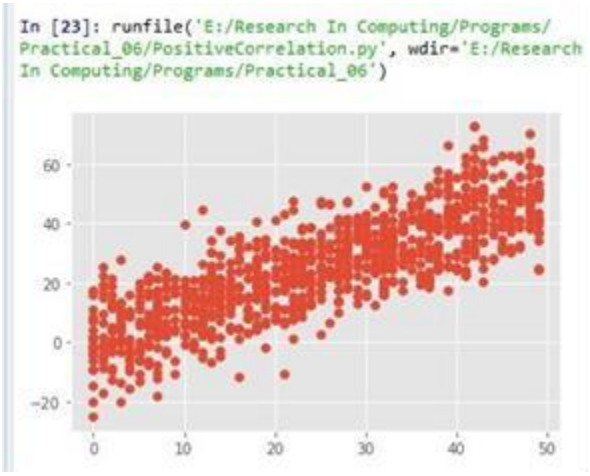
A) Positive Correlation:

Let's take a look at a positive correlation. Numpy implements a corrcoef() function that returns a matrix of correlations of x with x, x with y, y with x and y with y. We're interested in the values of correlation of x with y (so position (1, 0) or (0, 1)).

Code:

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)
# 1000 random integers between 0 and 50
x = np.random.randint(0, 50, 1000)
# Positive Correlation with some noise
y = x + np.random.normal(0, 10, 1000)
np.corrcoef(x, y)
matplotlib.style.use('ggplot')
plt.scatter(x, y)
plt.show()
```

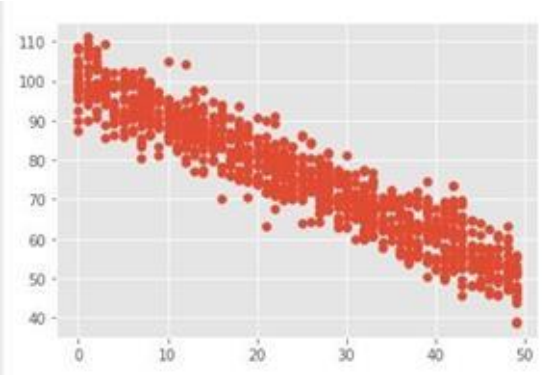
Output:



B) Negative Correlation

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)
#1000 random integers between 0 and 50
x = np.random.randint(0, 50, 1000)
# Negative Correlation with some noise
y = 100 - x + np.random.normal(0, 5, 1000)
np.corrcoef(x, y)
plt.scatter(x, y)
plt.show()
```

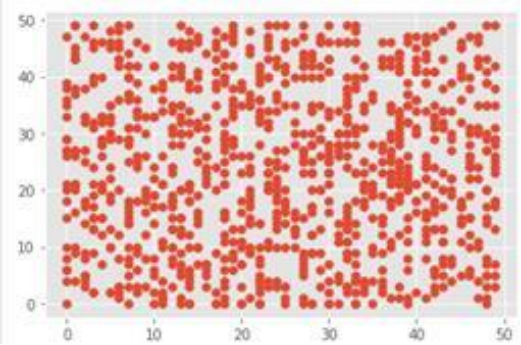
Output:



C) No/Weak Correlation:

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)
x = np.random.randint(0, 50, 1000)
y = np.random.randint(0, 50, 1000)
np.corrcoef(x, y)
plt.scatter(x, y)
plt.show()
```

Output:



University of Mumbai

Practical Journal of
Research in Computing
&
Data Science

M.Sc. (Information Technology)
Part-I

Submitted by

KARKHANIS YASH SANDESH

Seat No: 3254075

DEPARTMENT OF INFORMATION TECHNOLOGY
PILLAI HOC COLLEGE OF ARTS, SCIENCE &COMMERCE, RASAYANI
(Affiliated to Mumbai University)
RASAYANI, 410207
MAHARASHTRA

2022-2023

Mahatma Education Society's

Pillai Hoc College of Arts, Science & Commerce, Rasayani

(Affiliated to Mumbai University)

RASAYANI – MAHARASHTRA - 410207

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the experiment work entered in this journal is as per the syllabus in **M.Sc. (Information Technology) Part-I, Semester-I**; class prescribed by University of Mumbai for the subject **Data Science** was done in computer lab of Mahatma Education Society's Pillai HOC College of Arts, Science & Commerce, Rasayani by **YASH KARKHANIS** during Academic year 2022-2023.

Exam Seat No: 3254075

Subject In-Charge **Coordinator**

External Examiner **Principal**

Date: **College Seal**

DATA SCIENCE

INDEX

Practical No: 01

Aim: Creating Data Model using Cassandra.

Create keyspace keyspace1 with replication = { 'class':'SimpleStratergy','replication_factor': '3' };
Create table dept (dept_id int PRIMARY KEY, dept_name text, dept_loc text);
Create table emp (emp_id int PRIMARY KEY, emp_name text, dept_id int,email text, phone text);
Insert Queries for dept table
Insert into dept (dept_id, dept_name, dept_loc) values (1001,'Accounts', 'Mumbai');
Insert into dept (dept_id, dept_name, dept_loc) values (1002,'Marketing', 'Delhi');
Insert into dept (dept_id, dept_name, dept_loc) values (1003, 'HR','Chennai');
Insert Queries for emp table
Insert into emp (emp_id, emp_name, dept_id, email, phone) values
(1001, 'ABCD',1001, 'abcd@company.com', '1122334455');
Insert into emp (emp_id, emp_name, dept_id, email, phone) values
(1002, 'DEFG',1001, 'defg@company.com', '2233445566');
Insert into emp (emp_id, emp_name, dept_id, email, phone) values
(1003, 'GHIJ',1002, 'ghij@company.com', '3344556677');
Insert into emp (emp_id, emp_name, dept_id, email, phone) values
(1004, 'JKLM',1002, 'jklm@company.com', '4455667788');
Insert into emp (emp_id, emp_name, dept_id, email, phone) values
(1005, 'MNOP',1003, 'mnop@company.com', '5566778899');
Insert into emp (emp_id, emp_name, dept_id, email, phone) values
(1006, 'MNOP',1003, 'mnop@company.com', '5566778844');
Select-all Query for emp table
select * from emp
Select-all Query for dept table
select * from dept
Update query for dept table where dept_id=1003 set dept_name is Human Resource
update dept set dept_name='Human Resource' where dept_id=1003;
Delete query for emp table where emp_id=1006 and Select-all from emp
delete from emp where emp_id=1006;
select * from emp

Output:

emp_id	dept_id	email	emp_name	phone
1006	1003	mnop@company.com	MNOP	5566778844
1004	1002	jklm@company.com	JKLM	4455667788
1005	1003	mnop@company.com	MNOP	5566778899
1001	1001	abcd@company.com	ABCD	1122334455
1003	1002	ghij@company.com	GHIJ	3344556677
1002	1001	defg@company.com	DEFG	2233445566

dept_id	dept_loc	dept_name
1001	Mumbai	Accounts
1003	Chennai	HR
1002	Delhi	Marketing

dept_id	dept_loc	dept_name
1001	Mumbai	Accounts
1003	Chennai	Human Resource
1002	Delhi	Marketing

(3 rows)

emp_id	dept_id	email	emp_name	phone
1004	1002	jklm@company.com	JKLM	4455667788
1005	1003	mnop@company.com	MNOP	5566778899
1001	1001	abcd@company.com	ABCD	1122334455
1003	1002	ghij@company.com	GHIJ	3344556677
1002	1001	defg@company.com	DEFG	2233445566

(5 rows)

Practical No: 02

Aim: Write Python / R Program to convert from the following formats to HORUS format
2A. Text delimited CSV to HORUS format

```
# Utility Start CSV to HORUS ================================
# Standard Tools
# ============================================================
import pandas as pd
# Input Agreement ============================================
sInputFileName = "D:/OneDrive - South Indian Education Society/Online
MSIT Classes/Practicals/Data Science/Practical 2/Country_Code.csv"
InputData = pd.read_csv(sInputFileName, encoding="latin-1")
print("Input Data Values ==================================")
print(InputData)
print("====================================================")
# Processing Rules ===========================================
ProcessData = InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop("ISO-2-CODE", axis=1, inplace=True)
ProcessData.drop("ISO-3-Code", axis=1, inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={"Country": "CountryName"}, inplace=True)
ProcessData.rename(columns={"ISO-M49": "CountryNumber"}, inplace=True)
# Set new Index
ProcessData.set_index("CountryNumber", inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values("CountryName", axis=0, ascending=False,
inplace=True)
print("Process Data Values ================================")
print(ProcessData)
print("====================================================")
# Output Agreement ===========================================
OutputData = ProcessData
sOutputFileName = "D:/OneDrive - South Indian Education Society/Online
MSIT Classes/Practicals/Data Science/Practical 2/HORUS-CSV-
Country.csv"
OutputData.to_csv(sOutputFileName, index=False)
print("CSV to HORUS - Done")
# Utility done ===============================================
```

Output:

```
Input Data Values ==================================
                      Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan         AF         AFG         4
1               Aland Islands         AX         ALA       248
2                     Albania         AL         ALB         8
3                     Algeria         DZ         DZA        12
4              American Samoa         AS         ASM        16
..                        ...        ...         ...       ...
242  Wallis and Futuna Islands         WF         WLF       876
243            Western Sahara         EH         ESH       732
244                     Yemen         YE         YEM       887
245                    Zambia         ZM         ZMB       894
246                  Zimbabwe         ZW         ZWE       716

[247 rows x 4 columns]
====================================================
Process Data Values ================================
                            CountryName
CountryNumber
716                            Zimbabwe
894                              Zambia
887                               Yemen
732                      Western Sahara
876              Wallis and Futuna Islands
...                                  ...
16                       American Samoa
12                              Algeria
8                               Albania
248                       Aland Islands
4                           Afghanistan

[247 rows x 1 columns]
====================================================
CSV to HORUS - Done
```

2B. XML to HORUS Format

```
# Utility Start XML to HORUS ================================
# Standard Tools
import pandas as pd
import xml.etree.ElementTree as ET
def df2xml(data):
header = data.columns
root = ET.Element("root")
for row in range(data.shape[0]):
entry = ET.SubElement(root, "entry")
for index in range(data.shape[1]):
schild = str(header[index])
child = ET.SubElement(entry, schild)
if str(data[schild][row]) != "nan":
child.text = str(data[schild][row])
else:
child.text = "n/a"
entry.append(child)
result = ET.tostring(root)
return result
def xml2df(xml_data):
root = ET.XML(xml_data)
all_records = []
for i, child in enumerate(root):
record = {}
for subchild in child:
record[subchild.tag] = subchild.text
all_records.append(record)
return pd.DataFrame(all_records)
sInputFileName = "D:/VKHCG/05-DS/9999-Data/Country_Code.xml"
InputData = open(sInputFileName).read()
print("=====================================================")
print("Input Data Values ===================================")
print("=====================================================")
# print(InputData)
print("=====================================================")
# =============================================================
# Processing Rules ============================================
# =============================================================
ProcessDataXML = InputData
# XML to Data Frame
ProcessData = xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop("ISO-2-CODE", axis=1, inplace=True)
ProcessData.drop("ISO-3-Code", axis=1, inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={"Country": "CountryName"}, inplace=True)
ProcessData.rename(columns={"ISO-M49": "CountryNumber"}, inplace=True)
# Set new Index
ProcessData.set_index("CountryNumber", inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values("CountryName", axis=0, ascending=False,
```

```
inplace=True)
print("=====================================================")
print("Process Data Values ================================")
print("=====================================================")
print(ProcessData)
print("=====================================================")
OutputData = ProcessData
sOutputFileName = "D:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv"
OutputData.to_csv(sOutputFileName, index=False)
print("=====================================================")
print("XML to HORUS - Done")
print("=====================================================")
# Utility done ================================================
```

Output:

```
=====================================================
Input Data Values ==================================
=====================================================
=====================================================
=====================================================
Process Data Values ================================
=====================================================
                            CountryName
CountryNumber
716                            Zimbabwe
894                              Zambia
887                               Yemen
732                      Western Sahara
876           Wallis and Futuna Islands
...                                 ...
16                       American Samoa
12                              Algeria
8                               Albania
248                        Aland Islands
4                            Afghanistan

[247 rows x 1 columns]
=====================================================
=====================================================
XML to HORUS - Done
=====================================================
```

2C. JSON to HORUS Format

```
# Utility Start JSON to HORUS ================================
# Standard Tools
# ============================================================
import pandas as pd
# Input Agreement ============================================
sInputFileName = "D:/VKHCG/05-DS/9999-Data/Country_Code.json"
InputData = pd.read_json(sInputFileName, orient="index",
encoding="latin-1")
print("Input Data Values ==================================")
print(InputData)
print("====================================================")
# Processing Rules ===========================================
ProcessData = InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop("ISO-2-CODE", axis=1, inplace=True)
ProcessData.drop("ISO-3-Code", axis=1, inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={"Country": "CountryName"}, inplace=True)
ProcessData.rename(columns={"ISO-M49": "CountryNumber"}, inplace=True)
# Set new Index
ProcessData.set_index("CountryNumber", inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values("CountryName", axis=0, ascending=False,
inplace=True)
print("Process Data Values ================================")
print(ProcessData)
print("====================================================")
# Output Agreement ===========================================
OutputData = ProcessData
sOutputFileName = "D:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv"
OutputData.to_csv(sOutputFileName, index=False)
print("JSON to HORUS - Done")
# Utility done ===============================================
```

Output:

```
Input Data Values ===================================
                      Country ISO-2-CODE ISO-3-Code  ISO-M49
0                 Afghanistan         AF         AFG        4
1               Aland Islands         AX         ALA      248
2                     Albania         AL         ALB        8
3                     Algeria         DZ         DZA       12
4              American Samoa         AS         ASM       16
..                        ...        ...         ...      ...
242  Wallis and Futuna Islands         WF         WLF      876
243            Western Sahara         EH         ESH      732
244                     Yemen         YE         YEM      887
245                    Zambia         ZM         ZMB      894
246                  Zimbabwe         ZW         ZWE      716

[247 rows x 4 columns]
=====================================================
Process Data Values ===================================
                            CountryName
CountryNumber
716                            Zimbabwe
894                              Zambia
887                               Yemen
732                      Western Sahara
876              Wallis and Futuna Islands
...                                  ...
16                       American Samoa
12                              Algeria
8                               Albania
248                       Aland Islands
4                           Afghanistan

[247 rows x 1 columns]
=====================================================
JSON to HORUS - Done
```

2D. MySql Database to HORUS Format

```
# Standard Tools
# =============================================================
import pandas as pd
import sqlite3 as sq
# Input Agreement ============================================
sInputFileName = "D:/VKHCG/05-DS/9999-Data/utility.db"
sInputTable = "Country_Code"
conn = sq.connect(sInputFileName)
sSQL = "select * FROM " + sInputTable + ";"
InputData = pd.read_sql_query(sSQL, conn)
print("Input Data Values ===================================")
print(InputData)
print("=====================================================")
# Processing Rules ===========================================
ProcessData = InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop("ISO-2-CODE", axis=1, inplace=True)
ProcessData.drop("ISO-3-Code", axis=1,  inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={"Country": "CountryName"}, inplace=True)
ProcessData.rename(columns={"ISO-M49": "CountryNumber"}, inplace=True)
# Set new Index
ProcessData.set_index("CountryNumber", inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values("CountryName", axis=0, ascending=False,
inplace=True)
print("Process Data Values ================================")
print(ProcessData)
print("=====================================================")
# Output Agreement ===========================================
OutputData = ProcessData
sOutputFileName = "D:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv"
OutputData.to_csv(sOutputFileName, index=False)
print("Database to HORUS - Done")
# Utility done ===============================================
```

Output:

```
Input Data Values ===================================
     index                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0        0                Afghanistan         AF        AFG        4
1        1              Aland Islands         AX        ALA      248
2        2                    Albania         AL        ALB        8
3        3                    Algeria         DZ        DZA       12
4        4             American Samoa         AS        ASM       16
..     ...                        ...        ...        ...      ...
242    242  Wallis and Futuna Islands         WF        WLF      876
243    243             Western Sahara         EH        ESH      732
244    244                      Yemen         YE        YEM      887
245    245                     Zambia         ZM        ZMB      894
246    246                   Zimbabwe         ZW        ZWE      716

[247 rows x 5 columns]
=====================================================
Process Data Values =================================
               index                CountryName
CountryNumber
716              246                   Zimbabwe
894              245                     Zambia
887              244                      Yemen
732              243             Western Sahara
876              242  Wallis and Futuna Islands
...              ...                        ...
16                 4             American Samoa
12                 3                    Algeria
8                  2                    Albania
248                1                Aland Islands
4                  0                Afghanistan

[247 rows x 2 columns]
=====================================================
Database to HORUS - Done
```

Practical No: 03

Aim: Utilities and Auditing
3A. Fixers Utilities

```
#----------------------------- Program to Demonstrate Fixers utilities
- - - - - - - - - - - - -.
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print("\n#1 Removing leading or lagging spaces from a data entry")
baddata = " Data Science with too many spaces is bad!!! "
print(">", baddata, "<")
cleandata = baddata.strip()
print(">", cleandata, "<")
# 2 Removing nonprintable characters from a data entry
print("#2 Removing nonprintable characters from a data entry")
printable = set(string.printable)
baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"
cleandata = "".join(filter(lambda x: x in string.printable, baddata))
print("Bad Data : ", baddata)
print("Clean Data : ", cleandata)
# 3 Reformatting data entry to match specific formatting criteria.
# Convert YYYY/MM/DD to DD Month YYYY
print("# 3 Reformatting data entry to match specific formatting
criteria.")
baddate = dt.date(2019, 10, 31)
baddata = format(baddate, "%Y-%m-%d")
gooddate = dt.datetime.strptime(baddata, "%Y-%m-%d")
gooddata = format(gooddate, "%d %B %Y")
print("Bad Data : ", baddata)
print("Good Data : ", gooddata)
```

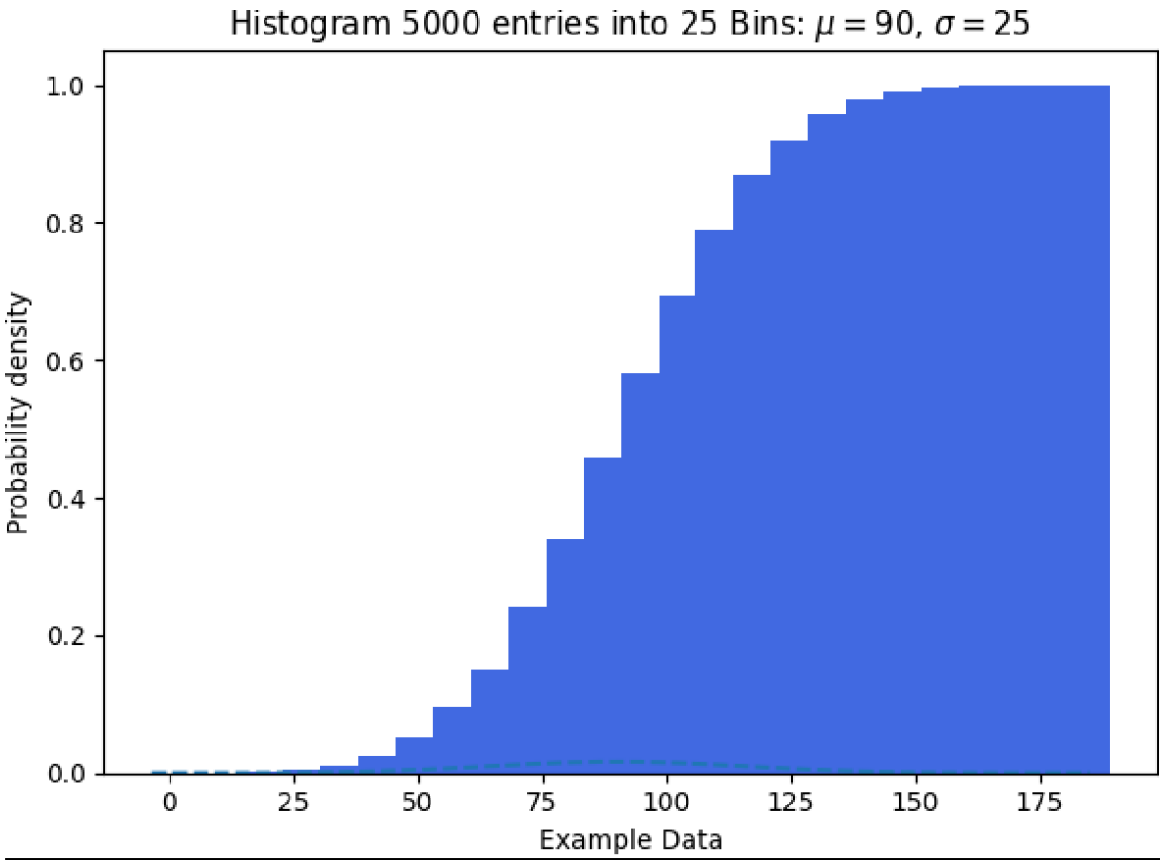
Output:

```
#1 Removing leading or lagging spaces from a data entry
>  Data Science with too many spaces is bad!!!  <
> Data Science with too many spaces is bad!!! <
#2 Removing nonprintable characters from a data entry
Bad Data :  DataScience with funny characters is bad!!!
Clean Data :  DataScience with funny characters is bad!!!
# 3 Reformatting data entry to match specific formatting criteria.
Bad Data :  2019-10-31
Good Data :  31 October 2019
```

3B. Data Binning or Bucketing

```
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
np.random.seed(0)
# example data
mu = 90 # mean of distribution
sigma = 25 # standard deviation of distribution
x = mu + sigma * np.random.randn(5000)
num_bins = 25
fig, ax = plt.subplots()
# the histogram of the data
n, bins, patches = ax.hist(
x,
num_bins,
color="royalblue",
density=1,
alpha=1,
align="right",
orientation="vertical",
cumulative=True,
)
# add a 'best fit' line
y = stats.norm.pdf(bins, mu, sigma)
# mlab.normpdf(bins, mu, sigma)
ax.plot(bins, y, "--")
ax.set_xlabel("Example Data")
ax.set_ylabel("Probability density")
sTitle = (
r"Histogram "
+ str(len(x))
+ " entries into "
+ str(num_bins)
+ " Bins: $\mu="
+ str(mu)
+ "$, $\sigma="
+ str(sigma)
+ "$"
)
ax.set_title(sTitle)
fig.tight_layout()
sPathFig = "D:/OneDrive - South Indian Education Society/Online MSIT
Classes/Practicals/Data Science/Practical 3/3B/DU-Histogram.png"
fig.savefig(sPathFig)
plt.plot()
plt.show()
```

Output:


Histogram 5000 entries into 25 Bins: $\mu = 90$, $\sigma = 25$
Probability density
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
Example Data

3C. Averaging of Data

```
import pandas as pd
#####################################################################
InputFileName = "IP_DATA_CORE.csv"
OutputFileName = "Retrieve_Router_Location.csv"
Base = "D:/VKHCG"
print("#################################")
print("Working Base :", Base, " using ")
print("#################################")
sFileName = Base + "/01-Vermeulen/00-RawData/" + InputFileName
print("Loading :", sFileName)
IP_DATA_ALL = pd.read_csv(
sFileName,
header=0,
low_memory=False,
usecols=["Country", "Place Name", "Latitude", "Longitude"],
encoding="latin-1",
)
IP_DATA_ALL.rename(columns={"Place Name": "Place_Name"}, inplace=True)
AllData = IP_DATA_ALL[["Country", "Place_Name", "Latitude"]]
print(AllData)
MeanData = AllData.groupby(["Country",
"Place_Name"])["Latitude"].mean()
print(MeanData)
#####################################################################
```

Output:

```
################################
Working Base : D:/VKHCG  using
################################
Loading : D:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
     Country Place_Name  Latitude
0         US   New York   40.7528
1         US   New York   40.7528
2         US   New York   40.7528
3         US   New York   40.7528
4         US   New York   40.7528
...      ...        ...       ...
3557      DE     Munich   48.0915
3558      DE     Munich   48.1833
3559      DE     Munich   48.1000
3560      DE     Munich   48.1480
3561      DE     Munich   48.1480

[3562 rows x 3 columns]
Country  Place_Name
DE       Munich        48.143223
GB       London        51.509406
US       New York      40.747044
Name: Latitude, dtype: float64
```

Practical No: 04

Aim: 4A. Data processing using R.

```
library(readr)
IP_DATA_ALL <-
read_csv("D:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
View(IP_DATA_ALL)
spec(IP_DATA_ALL)
library(tibble)
set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
sapply(IP_DATA_ALL_FIX, typeof)
library(data.table)
hist_country = data.table(Country =
unique(IP_DATA_ALL_FIX[is.na(IP_DATA_ALL_FIX['Country']) ==
0,]$Country))
setorder(hist_country, 'Country')
hist_country_with_id = rowid_to_column(hist_country, var =
"RowIDCountry")
View(hist_country_fix)
IP_DATA_COUNTRY_FREQ = data.table(with(IP_DATA_ALL_FIX,
table(Country)))
View(IP_DATA_COUNTRY_FREQ)
hist_latitude = data.table(Latitude =
unique(IP_DATA_ALL_FIX[is.na(IP_DATA_ALL_with_ID['Latitude']) ==
0,]$Latitude))
setkeyv(hist_latitude, 'Latitude')
setorder(hist_latitude)
hist_latitude_with_id = rowid_to_column(hist_latitude, var = "RowID")
View(hist_latitude_with_id)
IP_DATA_Latitude_FREQ = data.table(with(IP_DATA_ALL_FIX,
table(Latitude)))
View(IP_DATA_Latitude_FREQ)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], min, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Country'], min, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], max, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Country'], max, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], mean, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], median, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], range, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], quantile, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], sd, na.rm = TRUE)
sapply(IP_DATA_ALL_FIX[, 'Longitude'], sd, na.rm = TRUE)
```

Output:

```
> sapply(IP_DATA_ALL_FIX[, 'Latitude'], min, na.rm = TRUE)
Latitude
-54.1561
> sapply(IP_DATA_ALL_FIX[, 'Country'], min, na.rm = TRUE)
Country
   "AD"
> sapply(IP_DATA_ALL_FIX[, 'Latitude'], max, na.rm = TRUE)
Latitude
 78.2167
> sapply(IP_DATA_ALL_FIX[, 'Country'], max, na.rm = TRUE)
Country
   "ZW"
> sapply(IP_DATA_ALL_FIX [, 'Latitude'], mean, na.rm = TRUE)
Latitude
39.28514
> sapply(IP_DATA_ALL_FIX [, 'Latitude'], median, na.rm = TRUE)
Latitude
 41.4406
> sapply(IP_DATA_ALL_FIX [, 'Latitude'], range, na.rm = TRUE)
     Latitude
[1,] -54.1561
[2,]  78.2167
> sapply(IP_DATA_ALL_FIX [, 'Latitude'], quantile, na.rm = TRUE)
     Latitude
0%   -54.1561
25%   35.8204
50%   41.4406
75%   48.5613
100%  78.2167
> sapply(IP_DATA_ALL_FIX [, 'Latitude'], sd, na.rm = TRUE)
Latitude
15.88847
> sapply(IP_DATA_ALL_FIX [, 'Longitude'], sd, na.rm = TRUE)
Longitude
  68.0561
```

4B. Program to retrieve different attributes of data

```
import os
import pandas as pd
#####################################################################
Base = "D:/VKHCG"
#####################################################################
sFileName = Base + "/01-Vermeulen/00-RawData/IP_DATA_ALL.csv"
print("Loading:", sFileName)
IP_DATA_ALL = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
#####################################################################
sFileDir = Base + "01-Vermeulen/01-Retrieve/01-EDS/02-Python"
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
print("Rows", IP_DATA_ALL.shape[0])
print("Columns", IP_DATA_ALL.shape[1])
print("### Raw Data Set #########################################")
for i in range(0, len(IP_DATA_ALL.columns)):
print(IP_DATA_ALL.columns[i], type(IP_DATA_ALL.columns[i]))
print("### Fixed Data Set #########################################")
IP_DATA_ALL_FIX = IP_DATA_ALL
for i in range(0, len(IP_DATA_ALL.columns)):
cNameOld = IP_DATA_ALL_FIX.columns[i] + ""
cNameNew = cNameOld.strip().replace(" ", ".")
IP_DATA_ALL_FIX.columns.values[i] = cNameNew
print(IP_DATA_ALL.columns[i], type(IP_DATA_ALL.columns[i]))
#####################################################################
# print (IP_DATA_ALL_FIX.head()
#####################################################################
print("Fixed Data Set with ID")
IP_DATA_ALL_with_ID = IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ["RowID"]
# print (IP_DATA_ALL_FIX_with_ID.head()
sFileName2 = sFileDir + "/Retrieve_IP_DATA.csv"
IP_DATA_ALL_with_ID.to_csv(sFileName2, index=True, encoding="latin-1")
#####################################################################
print("### Done!! ############################################")
#####################################################################
```

Output:

```
Loading: D:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows 1048575
Columns 9
### Raw Data Set #####################################
Unnamed: 0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
### Fixed Data Set #####################################
Unnamed:.0 <class 'str'>
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
### Done!! ############################################
```

4C.Data Pattern

```
library(readr)
library(data.table)
FileName = paste0('D:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv')
IP_DATA_ALL <- read.csv(FileName)
hist_country = data.table(Country = unique(IP_DATA_ALL$Country))
pattern_country = data.table(Country = hist_country$Country,
PatternCountry = hist_country$Country)
oldchar = c(letters, LETTERS)
newchar = replicate(length(oldchar), "A")
for (r in seq(nrow(pattern_country))) {
s = pattern_country[r,]$PatternCountry
for (c in seq(length(oldchar))) {
s = chartr(oldchar[c], newchar[c], s)
}
s = chartr(" ", "b", s)
s = chartr(".", "u", s)
pattern_country[r,]$PatternCountry = s
}
View(pattern_country)
```

Output:

Filter

	Country	PatternCountry
1	BW	AA
2	NE	AA
3	MZ	AA
4	GH	AA
5	DZ	AA
6	EG	AA
7	KE	AA
8	CM	AA
9	SN	AA

Practical No: 05

Aim: Assessing Data

5A. Perform error management on the given data using pandas package

I. Drop the Columns Where All Elements Are Missing Values

```
import os
import sys
import pandas as pd
#################################################################
Base = "D:/VKHCG"
#################################################################
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#################################################################
sInputFileName = "Good-or-Bad.csv"
sOutputFileName = "Good-or-Bad-01.csv"
Company = "01-Vermeulen"
#################################################################
Base = "D:/VKHCG"
#################################################################
sFileDir = Base + "/" + Company + "/02-Assess/01-EDS/02-Python"
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#################################################################
# Import Warehouse
#################################################################
sFileName = Base + "/" + Company + "/00-RawData/" + sInputFileName
print("Loading :", sFileName)
RawData = pd.read_csv(sFileName, header=0)
print("################################")
print("## Raw Data Values")
print("################################")
print(RawData)
print("################################")
print("## Data Profile")
print("################################")
print("Rows :", RawData.shape[0])
print("Columns :", RawData.shape[1])
print("################################")
#################################################################
sFileName = sFileDir + "/" + sInputFileName
RawData.to_csv(sFileName, index=False)
#################################################################
TestData = RawData.dropna(axis=1, how="all")
#################################################################
print("################################")
print("## Test Data Values")
print("################################")
print(TestData)
print("################################")
```

```
print("## Data Profile")
print("#################################")
print("Rows :", TestData.shape[0])
print("Columns :", TestData.shape[1])
print("#################################")
#####################################################################
sFileName = sFileDir + "/" + sOutputFileName
TestData.to_csv(sFileName, index=False)
#####################################################################
print("#################################")
print("### Done!! ####################")
print("#################################")
#####################################################################
```

Output:

```
#################################
Rows : 21
Columns : 8
#################################
#################################
## Test Data Values
#################################
      ID FieldA  FieldB FieldC  FieldD    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0   10241.0       1
1    2.0   Good     NaN   Best   512.0    5121.0       2
2    3.0   Good  Better    NaN   256.0     256.0       3
3    4.0   Good  Better   Best     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0    6411.0       5
5    6.0   Good     NaN   Best    32.0      32.0       6
6    7.0    NaN  Better   Best    16.0    1611.0       7
7    8.0    NaN     NaN   Best     8.0    8111.0       8
8    9.0    NaN     NaN    NaN     4.0      41.0       9
9   10.0      A       B      C     2.0   21111.0      10
10   NaN    NaN     NaN    NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0  102411.0      12
12  10.0   Good     NaN   Best   512.0     512.0      13
13  10.0   Good  Better    NaN   256.0    1256.0      14
14  10.0   Good  Better   Best     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     164.0      16
16  10.0   Good     NaN   Best    32.0     322.0      17
17  10.0    NaN  Better   Best    16.0     163.0      18
18  10.0    NaN     NaN   Best     8.0     844.0      19
19  10.0    NaN     NaN    NaN     4.0    4555.0      20
20  10.0      A       B      C     2.0     111.0      21
#################################
## Data Profile
#################################
Rows : 21
Columns : 7
#################################
#################################
### Done!! ####################
#################################
```

II. Drop the Columns Where Any of the Elements Is Missing Values

```
import sys
import os
import pandas as pd
#################################################################
Base = "D:/VKHCG"
sInputFileName = "Good-or-Bad.csv"
sOutputFileName = "Good-or-Bad-02.csv"
Company = "01-Vermeulen"
#################################################################
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#################################################################
sFileDir = Base + "/" + Company + "/02-Assess/01-EDS/02-Python"
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#################################################################
# Import Warehouse
#################################################################
sFileName = Base + "/" + Company + "/00-RawData/" + sInputFileName
print("Loading :", sFileName)
RawData = pd.read_csv(sFileName, header=0)
print("################################")
print("## Raw Data Values")
print("################################")
print(RawData)
print("################################")
print("## Data Profile")
print("################################")
print("Rows :", RawData.shape[0])
print("Columns :", RawData.shape[1])
print("################################")
#################################################################
sFileName = sFileDir + "/" + sInputFileName
RawData.to_csv(sFileName, index=False)
#################################################################
TestData = RawData.dropna(axis=1, how="any")
#################################################################
print("################################")
print("## Test Data Values")
print("################################")
print(TestData)
print("################################")
print("## Data Profile")
print("################################")
print("Rows :", TestData.shape[0])
print("Columns :", TestData.shape[1])
print("################################")
#################################################################
```

```
sFileName = sFileDir + "/" + sOutputFileName
TestData.to_csv(sFileName, index=False)
########################################################################
print("################################")
print("### Done!! ####################")
print("################################")
########################################################################
```

Output:

```
################################
Working Base : D:/VKHCG  using  win32
################################
Loading : D:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
```

III. Keep Only the Rows That Contain a Maximum of Two Missing Values

```
import sys
import os
import pandas as pd
#################################################################
sInputFileName = "Good-or-Bad.csv"
sOutputFileName = "Good-or-Bad-03.csv"
Company = "01-Vermeulen"
Base = "D:/VKHCG"
#################################################################
print("################################")
print("Working Base :", Base, sys.platform)
print("################################")
#################################################################
sFileDir = Base + "/" + Company + "/02-Assess/01-EDS/02-Python"
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#################################################################
### Import Warehouse
#################################################################
sFileName = Base + "/" + Company + "/00-RawData/" + sInputFileName
print("Loading :", sFileName)
RawData = pd.read_csv(sFileName, header=0)
print("################################")
print("## Raw Data Values")
print("################################")
print(RawData)
print("################################")
print("## Data Profile")
print("################################")
print("Rows :", RawData.shape[0])
print("Columns :", RawData.shape[1])
print("################################")
#################################################################
sFileName = sFileDir + "/" + sInputFileName
RawData.to_csv(sFileName, index=False)
#################################################################
TestData = RawData.dropna(thresh=2)
print("################################")
print("## Test Data Values")
print("################################")
print(TestData)
print("################################")
print("## Data Profile")
print("################################")
print("Rows :", TestData.shape[0])
print("Columns :", TestData.shape[1])
print("################################")
sFileName = sFileDir + "/" + sOutputFileName
TestData.to_csv(sFileName, index=False)
```

```
######################################################################
print("###################################")
print("### Done!! ######################")
print("###################################")
######################################################################
```

Output:

```
################################
Working Base : D:/VKHCG win32
################################
Loading : D:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
## Data Profile
################################
Rows : 21
Columns : 8
################################
################################
## Test Data Values
################################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
################################
## Data Profile
################################
Rows : 20
Columns : 8
################################
################################
### Done!! ######################
################################
```

5B. Write Python / R program to create the network routing diagram from the given data on routers.

assess-network-routing-company.py

```
########## Assess-Network-Routing-Company.py ####################

import os
import sys
import pandas as pd
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################
Base = "D:/VKHCG"
#####################################################################
print("##################################")
print("Working Base :", Base, sys.platform)
print("##################################")
#####################################################################
sInputFileName1 = "01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv"
sInputFileName2 = "01-Retrieve/01-EDS/02-
Python/Retrieve_Router_Location.csv"
sInputFileName3 = "01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv"
#####################################################################
sOutputFileName = "Assess-Network-Routing-Company.csv"
Company = "01-Vermeulen"
#####################################################################
#####################################################################
### Import Country Data
#####################################################################
sFileName = Base + "/" + Company + "/" + sInputFileName1
print("##################################")
print("Loading :", sFileName)
print("##################################")
CountryData = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
print("Loaded Country:", CountryData.columns.values)
print("##################################")
#####################################################################
## Assess Country Data
#####################################################################
print("##################################")
print("Changed :", CountryData.columns.values)
CountryData.rename(columns={"Country": "Country_Name"}, inplace=True)
CountryData.rename(columns={"ISO-2-CODE": "Country_Code"},
inplace=True)
CountryData.drop("ISO-M49", axis=1, inplace=True)

CountryData.drop("ISO-3-Code", axis=1, inplace=True)
CountryData.drop("RowID", axis=1, inplace=True)
print("To :", CountryData.columns.values)
print("##################################")
```

```
#####################################################################
#####################################################################
### Import Company Data
#####################################################################
sFileName = Base + "/" + Company + "/" + sInputFileName2
print("################################")
print("Loading :", sFileName)
print("################################")
CompanyData = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
print("Loaded Company :", CompanyData.columns.values)
print("################################")
#####################################################################
## Assess Company Data
#####################################################################
print("################################")
print("Changed :", CompanyData.columns.values)
CompanyData.rename(columns={"Country": "Country_Code"}, inplace=True)
print("To :", CompanyData.columns.values)
print("################################")
#####################################################################
#####################################################################
### Import Customer Data
#####################################################################
sFileName = Base + "/" + Company + "/" + sInputFileName3
print("################################")
print("Loading :", sFileName)
print("################################")
CustomerRawData = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
print("################################")
print("Loaded Customer :", CustomerRawData.columns.values)
print("################################")
#####################################################################
CustomerData = CustomerRawData.dropna(axis=0, how="any")
print("################################")
print("Remove Blank Country Code")
print("Reduce Rows from", CustomerRawData.shape[0], " to ",
CustomerData.shape[0])
print("################################")
#####################################################################

print("################################")
print("Changed :", CustomerData.columns.values)
CustomerData.rename(columns={"Country": "Country_Code"}, inplace=True)
print("To :", CustomerData.columns.values)
print("################################")
#####################################################################
print("################################")
print("Merge Company and Country Data")

print("################################")
```

```
CompanyNetworkData = pd.merge(CompanyData, CountryData, how="inner",
on="Country_Code")
#########################################################################
print("################################")
print("Change ", CompanyNetworkData.columns.values)
for i in CompanyNetworkData.columns.values:
j = "Company_" + i
CompanyNetworkData.rename(columns={i: j}, inplace=True)
print("To ", CompanyNetworkData.columns.values)
print("################################")
#########################################################################
#########################################################################
sFileDir = Base + "/" + Company + "/02-Assess/01-EDS/02-Python"
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#########################################################################
sFileName = sFileDir + "/" + sOutputFileName
print("################################")
print("Storing :", sFileName)
print("################################")
CompanyNetworkData.to_csv(sFileName, index=False, encoding="latin-1")
#########################################################################
#########################################################################
print("################################")
print("### Done!! ####################")
print("################################")
#########################################################################
```

Output:

```
##############################
Working Base : D:/VKHCG win32
##############################
##############################
Loading : D:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv
##############################
Loaded Country: ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
##############################
##############################
Changed : ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
To : ['Country_Name' 'Country_Code']
##############################
##############################
Loading : D:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv
##############################
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
##############################
##############################
Changed : ['Country' 'Place_Name' 'Latitude' 'Longitude']
To : ['Country_Code' 'Place_Name' 'Latitude' 'Longitude']
##############################
##############################
Loading : D:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv
##############################
##############################
…
Storing : D:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv
##############################
##############################
### Done!! ####################
##############################
```

assess-network-routing-customer.py

```
import os
import sys
import pandas as pd
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################
Base = "D:/VKHCG"
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#####################################################################
sInputFileName = (Base+ "/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-
Routing-Customer.csv")
#####################################################################
sOutputFileName = "Assess-Network-Routing-Customer.gml"
Company = "01-Vermeulen"
#####################################################################
### Import Country Data
#####################################################################
sFileName = sInputFileName
print("################################")
print("Loading :", sFileName)
print("################################")
CustomerData = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
print("Loaded Country:", CustomerData.columns.values)
print("################################")
print(CustomerData.head())
print("################################")
print("### Done!! ####################")
print("################################")
#####################################################################
```

Output:

```
################################
Working Base : D:/VKHCG  using  win32
################################
################################
Loading : D:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code Customer_Place_Name  Customer_Latitude  Customer_Longitude Customer_Country_Name
0                    BW            Gaborone          -24.6464             25.9119              Botswana
1                    BW         Francistown          -21.1667             27.5167              Botswana
2                    BW                Maun          -19.9833             23.4167              Botswana
3                    BW          Molepolole          -24.4167             25.5333              Botswana
4                    NE              Niamey           13.5167              2.1167                 Niger
################################
### Done!! ####################
################################
```

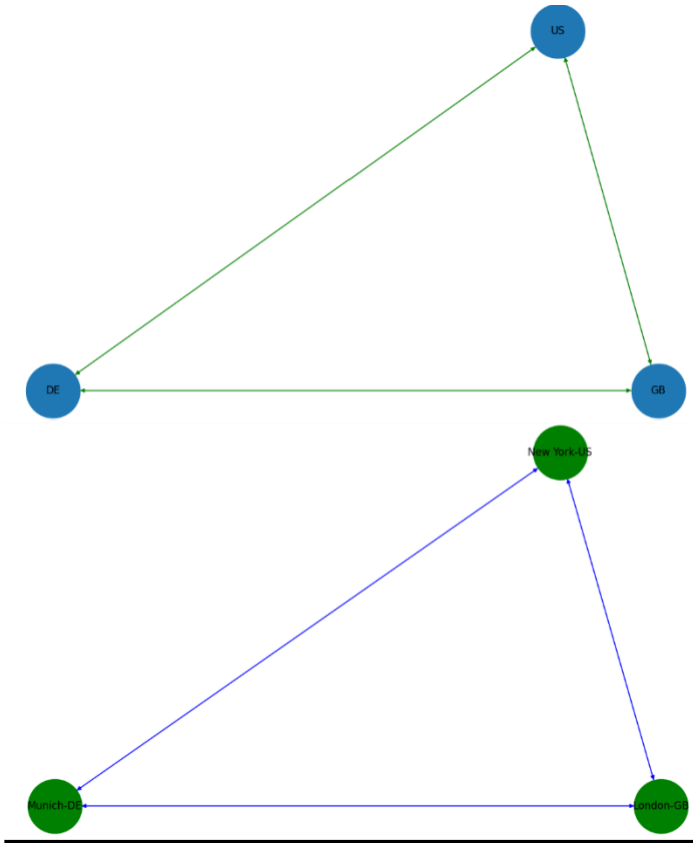
5C. Write a Python / R program to build directed acyclic graph
assess-DAG-location.py

```
import os
import sys
import matplotlib.pyplot as plt
import networkx as nx
import pandas as pd
#####################################################################
Base = "D:/VKHCG"
#####################################################################
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#####################################################################
sInputFileName = "01-Retrieve/01-EDS/02-
Python/Retrieve_Router_Location.csv"
sOutputFileName1 = "Assess-DAG-Company-Country.png"
sOutputFileName2 = "Assess-DAG-Company-Country-Place.png"
Company = "01-Vermeulen"
#####################################################################
#### Import Company Data
#####################################################################
sFileName = Base + "/" + Company + "/" + sInputFileName
print("################################")
print("Loading :", sFileName)
print("################################")
CompanyData = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
print("Loaded Company :", CompanyData.columns.values)
print("################################")
#####################################################################
print(CompanyData)
print("################################")
print("Rows : ", CompanyData.shape[0])
print("################################")
#####################################################################
G1 = nx.DiGraph()
G2 = nx.DiGraph()
#####################################################################
for i in range(CompanyData.shape[0]):
G1.add_node(CompanyData["Country"][i])
sPlaceName = CompanyData["Place_Name"][i] + "-" +
CompanyData["Country"][i]
G2.add_node(sPlaceName)

print("################################")
for n1 in G1.nodes():
for n2 in G1.nodes():
if n1 != n2:
print("Link :", n1, " to ", n2)
```

```
G1.add_edge(n1, n2)
print("#################################")
print("#################################")
print("Nodes of graph: ")
print(G1.nodes())
print("Edges of graph: ")
print(G1.edges())
print("#################################")
#####################################################################
sFileDir = Base + "/" + Company + "/02-Assess/01-EDS/02-Python"
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sFileName = sFileDir + "/" + sOutputFileName1
print("#################################")
print("Storing :", sFileName)
print("#################################")
nx.draw(
G1,
pos=nx.spectral_layout(G1),
edge_color="g",
with_labels=True,
node_size=8000,
font_size=12,
)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
print("#################################")
for n1 in G2.nodes():
for n2 in G2.nodes():
if n1 != n2:
print("Link :", n1, " to ", n2)
G2.add_edge(n1, n2)
print("#################################")
print("#################################")
print("Nodes of graph: ")
print(G2.nodes())
print("Edges of graph: ")
print(G2.edges())

print("#################################")
#####################################################################
sFileDir = Base + "/" + Company + "/02-Assess/01-EDS/02-Python"
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
#####################################################################
sFileName = sFileDir + "/" + sOutputFileName2
print("#################################")
print("Storing :", sFileName)
print("#################################")
```

```
nx.draw(
G2,
pos=nx.spectral_layout(G2),
edge_color="b",
node_color="g",
with_labels=True,
node_size=8000,
font_size=12,
)
plt.savefig(sFileName) # save as png
plt.show()
```

Output:



Practical No: 06

Aim: 6A. Build the time hub, links, and satellites

```
import sys
import os
from datetime import datetime
from datetime import timedelta
from pytz import timezone, all_timezones
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment = None
#################################################################
Base = "D:/VKHCG"
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#################################################################
Company = "01-Vermeulen"
InputDir = "00-RawData"
InputFileName = "VehicleData.csv"
#################################################################
sDataBaseDir = Base + "/" + Company + "/03-Process/SQLite"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName = sDataBaseDir + "/Hillman.db"
conn1 = sq.connect(sDatabaseName)
#################################################################
sDataVaultDir = Base + "/88-DV"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName = sDataVaultDir + "/datavault.db"
conn2 = sq.connect(sDatabaseName)
#################################################################
base = datetime(2018, 1, 1, 0, 0, 0)
numUnits = 10 * 365 * 24
#################################################################
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)]
t = 0
for i in date_list:
now_utc = i.replace(tzinfo=timezone("UTC"))
sDateTime = now_utc.strftime("%Y-%m-%d %H:%M:%S")
print(sDateTime)
sDateTimeKey = sDateTime.replace(" ", "-").replace(":", "-")
t += 1
IDNumber = str(uuid.uuid4())
TimeLine = [
```

```
("ZoneBaseKey", ["UTC"]),
("IDNumber", [IDNumber]),
("nDateTimeValue", [now_utc]),
("DateTimeValue", [sDateTime]),
("DateTimeKey", [sDateTimeKey]),]
if t == 1:
TimeFrame = pd.DataFrame.from_dict(dict(TimeLine))
else:
TimeRow = pd.DataFrame.from_dict(dict(TimeLine))
TimeFrame = TimeFrame.append(TimeRow)
#####################################################################
TimeHub = TimeFrame[["IDNumber", "ZoneBaseKey", "DateTimeKey",
"DateTimeValue"]]
TimeHubIndex = TimeHub.set_index(["IDNumber"], inplace=False)
#####################################################################
TimeFrame.set_index(["IDNumber"], inplace=True)
#####################################################################
sTable = "Process-Time"
print("Storing :", sDatabaseName, " Table:", sTable)
TimeHubIndex.to_sql(sTable, conn1, if_exists="replace")
#####################################################################
sTable = "Hub-Time"
print("Storing :", sDatabaseName, " Table:", sTable)
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
active_timezones = all_timezones
z = 0
for zone in active_timezones:
t = 0
for j in range(TimeFrame.shape[0]):
now_date = TimeFrame["nDateTimeValue"][j]
DateTimeKey = TimeFrame["DateTimeKey"][j]
now_utc = now_date.replace(tzinfo=timezone("UTC"))
sDateTime = now_utc.strftime("%Y-%m-%d %H:%M:%S")
now_zone = now_utc.astimezone(timezone(zone))
sZoneDateTime = now_zone.strftime("%Y-%m-%d %H:%M:%S")
print(sZoneDateTime)
t += 1
z += 1
IDZoneNumber = str(uuid.uuid4())
TimeZoneLine = [
("ZoneBaseKey", ["UTC"]),
("IDZoneNumber", [IDZoneNumber]),
("DateTimeKey", [DateTimeKey]),
("UTCDateTimeValue", [sDateTime]),
("Zone", [zone]),
("DateTimeValue", [sZoneDateTime]),]
if t == 1:
TimeZoneFrame = pd.DataFrame.from_dict(dict(TimeZoneLine))
else:
TimeZoneRow = pd.DataFrame.from_dict(dict(TimeZoneLine))
```

```
TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)
TimeZoneFrameIndex = TimeZoneFrame.set_index(["IDZoneNumber"],
inplace=False)
sZone = zone.replace("/", "-").replace(" ", "")
#################################################################
sTable = "Process-Time-" + sZone
print("Storing :", sDatabaseName, " Table:", sTable)
TimeZoneFrameIndex.to_sql(sTable, conn1, if_exists="replace")
#################################################################
sTable = "Satellite-Time-" + sZone
print("Storing :", sDatabaseName, " Table:", sTable)
TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print("#################")
print("Vacuum Databases")
sSQL = "VACUUM;"
sql.execute(sSQL, conn1)
sql.execute(sSQL, conn2)
print("#################")
print("### Done!! ############################################")
#################################################################
```

Output:

```
2017-02-26 12:00:00
2017-02-26 11:00:00
2017-02-26 10:00:00
2017-02-26 09:00:00
2017-02-26 08:00:00
2017-02-26 07:00:00
2017-02-26 06:00:00
2017-02-26 05:00:00
2017-02-26 04:00:00
2017-02-26 03:00:00
2017-02-26 02:00:00
2017-02-26 01:00:00
2017-02-26 00:00:00
2017-02-25 23:00:00
2017-02-25 22:00:00
2017-02-25 21:00:00
2017-02-25 20:00:00
```

6B. Write a program to load the vehicle data for Hillman Ltd into the data vault

```
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment = None
#################################################################
Base = "D:/VKHCG"
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#################################################################
Company = "03-Hillman"
InputDir = "00-RawData"
InputFileName = "VehicleData.csv"
#################################################################
sDataBaseDir = Base + "/" + Company + "/03-Process/SQLite"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName = sDataBaseDir + "/Hillman.db"
conn1 = sq.connect(sDatabaseName)
#################################################################
sDataVaultDir = Base + "/88-DV"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName = sDataVaultDir + "/datavault.db"
conn2 = sq.connect(sDatabaseName)
#################################################################
sFileName = Base + "/" + Company + "/" + InputDir + "/" +
InputFileName
print("###########")
print("Loading :", sFileName)
VehicleRaw = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
#################################################################
sTable = "Process_Vehicles"
print("Storing :", sDatabaseName, " Table:", sTable)
VehicleRaw.to_sql(sTable, conn1, if_exists="replace")
#################################################################
VehicleRawKey = VehicleRaw[["Make", "Model"]].copy()
VehicleKey = VehicleRawKey.drop_duplicates()
#################################################################
VehicleKey["ObjectKey"] = VehicleKey.apply(
lambda row: str(
"("
+ str(row["Make"]).strip().replace(" ", "-").replace("/", "-
```

```
").lower()
+ ")-("
+ (str(row["Model"]).strip().replace(" ", "-").replace(" ", "-
").lower())
+ ")"
),
axis=1,
)
#####################################################################
VehicleKey["ObjectType"] = VehicleKey.apply(lambda row: "vehicle",
axis=1)
#####################################################################
VehicleKey["ObjectUUID"] = VehicleKey.apply(lambda row:
str(uuid.uuid4()), axis=1)
#####################################################################
### Vehicle Hub
#####################################################################
#
VehicleHub = VehicleKey[["ObjectType", "ObjectKey",
"ObjectUUID"]].copy()
VehicleHub.index.name = "ObjectHubID"
sTable = "Hub-Object-Vehicle"
print("Storing :", sDatabaseName, " Table:", sTable)
VehicleHub.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
### Vehicle Satellite
#####################################################################
#
VehicleSatellite = VehicleKey[
["ObjectType", "ObjectKey", "ObjectUUID", "Make", "Model"]
].copy()
VehicleSatellite.index.name = "ObjectSatelliteID"
sTable = "Satellite-Object-Make-Model"
print("Storing :", sDatabaseName, " Table:", sTable)
VehicleSatellite.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
### Vehicle Dimension
#####################################################################
sView = "Dim-Object"
print("Storing :", sDatabaseName, " View:", sView)
sSQL = "CREATE VIEW IF NOT EXISTS [" + sView + "] AS"
sSQL = sSQL + " SELECT DISTINCT"
sSQL = sSQL + " H.ObjectType,"
sSQL = sSQL + " H.ObjectKey AS VehicleKey,"
sSQL = sSQL + " TRIM(S.Make) AS VehicleMake,"
sSQL = sSQL + " TRIM(S.Model) AS VehicleModel"
sSQL = sSQL + " FROM"
sSQL = sSQL + " [Hub-Object-Vehicle] AS H"
sSQL = sSQL + " JOIN"
sSQL = sSQL + " [Satellite-Object-Make-Model] AS S"
sSQL = sSQL + " ON"
```

```
sSQL = sSQL + " H.ObjectType=S.ObjectType"
sSQL = sSQL + " AND"
sSQL = sSQL + " H.ObjectUUID=S.ObjectUUID;"
sql.execute(sSQL, conn2)
print("#################")
print("Loading :", sDatabaseName, " Table:", sView)
sSQL = " SELECT DISTINCT"
sSQL = sSQL + " VehicleMake,"
sSQL = sSQL + " VehicleModel"
sSQL = sSQL + " FROM"
sSQL = sSQL + " [" + sView + "]"
sSQL = sSQL + " ORDER BY"
sSQL = sSQL + " VehicleMake"
sSQL = sSQL + " AND"
sSQL = sSQL + " VehicleMake;"
DimObjectData = pd.read_sql_query(sSQL, conn2)
DimObjectData.index.name = "ObjectDimID"
DimObjectData.sort_values(["VehicleMake", "VehicleModel"],
inplace=True, ascending=True)
print("#################")
print(DimObjectData)
#########################################################################
print("#################")
print("Vacuum Databases")
sSQL = "VACUUM;"
sql.execute(sSQL, conn1)
sql.execute(sSQL, conn2)
print("#################")
#########################################################################
conn1.close()
conn2.close()
#########################################################################
print("### Done!! ################################################")
#########################################################################
```

Output:

```
################################
Working Base : D:/VKHCG  using  win32
################################
###########
Loading : D:/VKHCG/03-Hillman/00-RawData/VehicleData.csv
Storing : D:/VKHCG/88-DV/datavault.db  Table: Process_Vehicles
Storing : D:/VKHCG/88-DV/datavault.db  Table: Hub-Object-Vehicle
Storing : D:/VKHCG/88-DV/datavault.db  Table: Satellite-Object-Make-Model
Storing : D:/VKHCG/88-DV/datavault.db  View: Dim-Object
################
Loading : D:/VKHCG/88-DV/datavault.db  Table: Dim-Object
################
                VehicleMake                       VehicleModel
ObjectDimID
2213             AM General                 DJ Po Vehicle 2WD
2212             AM General                  FJ8c Post Office
129              AM General               Post Office DJ5 2WD
131              AM General               Post Office DJ8 2WD
2869       ASC Incorporated                               GNX
...                     ...                               ...
1996                 smart                fortwo convertible
1997                 smart                      fortwo coupe
2622                 smart   fortwo electric drive cabriolet
2833                 smart  fortwo electric drive convertible
2623                 smart        fortwo electric drive coupe

[3885 rows x 2 columns]
################
Vacuum Databases
################
### Done!! ############################################
```

6C. Write a program to build relationship between Process-Location and Hub-Location

```
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
#####################################################################
Base = "D:/VKHCG"
print("##################################")
print("Working Base :", Base, " using ", sys.platform)
print("##################################")
#####################################################################
Company = "01-Vermeulen"
InputAssessGraphName = "Assess_All_Animals.gml"
EDSAssessDir =  "02-Assess/01-EDS"
InputAssessDir = EDSAssessDir + "/02-Python"
#####################################################################
sFileAssessDir = Base + "/" + Company + "/" + InputAssessDir
if not os.path.exists(sFileAssessDir):
os.makedirs(sFileAssessDir)
#####################################################################
sDataBaseDir = Base + "/" + Company + "/03-Process/SQLite"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName = sDataBaseDir + "/Vermeulen.db"
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataVaultDir = Base + "/88-DV"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName = sDataVaultDir + "/datavault.db"
conn2 = sq.connect(sDatabaseName)
t = 0
tMax = 360 * 180
#####################################################################
for Longitude in range(-180, 180, 10):
for Latitude in range(-90, 90, 10):
t += 1
IDNumber = str(uuid.uuid4())
LocationName = ("L"+ format(round(Longitude, 3) * 1000, "+07d")
+ "-"+ format(round(Latitude, 3) * 1000, "+07d"))
print("Create:", t, " of ", tMax, ":", LocationName)
LocationLine = [
("ObjectBaseKey", ["GPS"]),
("IDNumber", [IDNumber]),
("LocationNumber", [str(t)]),
("LocationName", [LocationName]),
```

```
("Longitude", [Longitude]),
("Latitude", [Latitude]),]
if t == 1:
LocationFrame = pd.DataFrame.from_dict(dict(LocationLine))
else:
LocationRow = pd.DataFrame.from_dict(dict(LocationLine))
LocationFrame = LocationFrame.append(LocationRow)
#################################################################
LocationHubIndex = LocationFrame.set_index(["IDNumber"],
inplace=False)
#################################################################
sTable = "Process-Location"
print("Storing :", sDatabaseName, " Table:", sTable)
LocationHubIndex.to_sql(sTable, conn1, if_exists="replace")
#################################################################
sTable = "Hub-Location"
print("Storing :", sDatabaseName, " Table:", sTable)
LocationHubIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print("################")
print("Vacuum Databases")
sSQL = "VACUUM;"
sql.execute(sSQL, conn1)
sql.execute(sSQL, conn2)
print("################")
#################################################################
print("### Done!! ################################################")
#################################################################
```

Output:

```
Create: 637  of  64800 : L+170000--030000
Create: 638  of  64800 : L+170000--020000
Create: 639  of  64800 : L+170000--010000
Create: 640  of  64800 : L+170000-+000000
Create: 641  of  64800 : L+170000-+010000
Create: 642  of  64800 : L+170000-+020000
Create: 643  of  64800 : L+170000-+030000
Create: 644  of  64800 : L+170000-+040000
Create: 645  of  64800 : L+170000-+050000
Create: 646  of  64800 : L+170000-+060000
Create: 647  of  64800 : L+170000-+070000
Create: 648  of  64800 : L+170000-+080000
Storing : D:/VKHCG/88-DV/datavault.db  Table: Process-Location
Storing : D:/VKHCG/88-DV/datavault.db  Table: Hub-Location
################
Vacuum Databases
################
### Done!! ############################################
```

Practical No: 07

Aim: 7A. Write a program to perform Transform Superstep on given data

```
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
#################################################################
Base = "D:/VKHCG"
print("##################################")
print("Working Base :", Base, " using ", sys.platform)
print("##################################")
#################################################################
Company = "01-Vermeulen"
InputDir = "00-RawData"
InputFileName = "VehicleData.csv"
#################################################################
sDataBaseDir = Base + "/" + Company + "/04-Transform/SQLite"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName = sDataBaseDir + "/Vermeulen.db"
conn1 = sq.connect(sDatabaseName)
#################################################################
sDataVaultDir = Base + "/88-DV"
if not os.path.exists(sDataVaultDir):
os.makedirs(sDataVaultDir)
#################################################################
sDatabaseName = sDataVaultDir + "/datavault.db"
conn2 = sq.connect(sDatabaseName)
#################################################################
sDataWarehouseDir = Base + "/99-DW"
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
#################################################################
sDatabaseName = sDataWarehouseDir + "/datawarehouse.db"
conn3 = sq.connect(sDatabaseName)
#################################################################
print("\n##################################")
print("Time Category")
print("UTC Time")
BirthDateUTC = datetime(1960, 12, 20, 10, 15, 0)
BirthDateZoneUTC = BirthDateUTC.replace(tzinfo=timezone("UTC"))
BirthDateZoneStr = BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr = BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S
(%Z) (%z)")
```

```
print(BirthDateZoneUTCStr)
print("#################################")
print("Birth Date in Reykjavik :")
BirthZone = "Atlantic/Reykjavik"
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr = BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal = BirthDate.strftime("%Y-%m-%d %H:%M:%S")
print(BirthDateStr)
print("#################################")
#####################################################################
IDZoneNumber = str(uuid.uuid4())
sDateTimeKey = BirthDateZoneStr.replace(" ", "-").replace(":", "-")
TimeLine = [
("ZoneBaseKey", ["UTC"]),
("IDNumber", [IDZoneNumber]),
("DateTimeKey", [sDateTimeKey]),
("UTCDateTimeValue", [BirthDateZoneUTC]),
("Zone", [BirthZone]),
("DateTimeValue", [BirthDateStr]),
]
TimeFrame = pd.DataFrame.from_dict(dict(TimeLine))
#####################################################################
TimeHub = TimeFrame[["IDNumber", "ZoneBaseKey", "DateTimeKey",
"DateTimeValue"]]
TimeHubIndex = TimeHub.set_index(["IDNumber"], inplace=False)
#####################################################################
sTable = "Hub-Time-Gunnarsson"
print("\n#################################")
print("Storing :", sDatabaseName, "\n Table:", sTable)
print("\n#################################")
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = "Dim-Time-Gunnarsson"
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
TimeSatellite = TimeFrame[["IDNumber", "DateTimeKey", "Zone",
"DateTimeValue"]]
TimeSatelliteIndex = TimeSatellite.set_index(["IDNumber"],
inplace=False)

#####################################################################
BirthZoneFix = BirthZone.replace(" ", "-").replace("/", "-")
sTable = "Satellite-Time-" + BirthZoneFix + "-Gunnarsson"
print("\n#################################")
print("Storing :", sDatabaseName, "\n Table:", sTable)
print("\n#################################")
TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = "Dim-Time-" + BirthZoneFix + "-Gunnarsson"
TimeSatelliteIndex.to_sql(sTable, conn3, if_exists="replace")
#####################################################################
print("\n#################################")
print("Person Category")
```

```
FirstName = "Guðmundur"
LastName = "Gunnarsson"
print("Name:", FirstName, LastName)
print("Birth Date:", BirthDateLocal)
print("Birth Zone:", BirthZone)
print("UTC Birth Date:", BirthDateZoneStr)
print("#################################")
#################################################################
IDPersonNumber = str(uuid.uuid4())
PersonLine = [
("IDNumber", [IDPersonNumber]),
("FirstName", [FirstName]),
("LastName", [LastName]),
("Zone", ["UTC"]),
("DateTimeValue", [BirthDateZoneStr]),
]
PersonFrame = pd.DataFrame.from_dict(dict(PersonLine))
#################################################################
TimeHub = PersonFrame
TimeHubIndex = TimeHub.set_index(["IDNumber"], inplace=False)
#################################################################
sTable = "Hub-Person-Gunnarsson"
print("\n#################################")
print("Storing :", sDatabaseName, "\n Table:", sTable)
print("\n#################################")
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = "Dim-Person-Gunnarsson"
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#################################################################
```

Output:

```
################################
Working Base : D:/VKHCG  using  win32
################################

################################
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
################################
Birth Date in Reykjavik :
1960-12-20 09:15:00 (-01) (-0100)
################################

################################
Storing : D:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Time-Gunnarsson

################################

################################
Storing : D:/VKHCG/99-DW/datawarehouse.db
 Table: Satellite-Time-Atlantic-Reykjavik-Gunnarsson

################################

################################
Person Category
Name: Guðmundur Gunnarsson
Birth Date: 1960-12-20 09:15:00
Birth Zone: Atlantic/Reykjavik
UTC Birth Date: 1960-12-20 10:15:00
################################

################################
Storing : D:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Person-Gunnarsson

################################
```

7B. Write a program to build dimension Person, dimension Time, and factPersonBornAtTime

```
import os
import sqlite3 as sq
import sys
import uuid
from datetime import datetime
import pandas as pd
from pytz import timezone
pd.options.mode.chained_assignment = None
#####################################################################
Base = "D:/VKHCG"
print("##################################")
print("Working Base :", Base, " using ", sys.platform)
print("##################################")
#####################################################################
Company = "01-Vermeulen"
#####################################################################
sDataBaseDir = Base + "/" + Company + "/04-Transform/SQLite"
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
#####################################################################
sDatabaseName = sDataBaseDir + "/Vermeulen.db"
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDataWarehousetDir = Base + "/99-DW"
if not os.path.exists(sDataWarehousetDir):
os.makedirs(sDataWarehousetDir)
#####################################################################
sDatabaseName = sDataWarehousetDir + "/datawarehouse.db"
conn2 = sq.connect(sDatabaseName)
#####################################################################
print("\n#################################")
print("Time Dimension")
BirthZone = "Atlantic/Reykjavik"
BirthDateUTC = datetime(1960, 12, 20, 10, 15, 0)
BirthDateZoneUTC = BirthDateUTC.replace(tzinfo=timezone("UTC"))
BirthDateZoneStr = BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr = BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S
(%Z) (%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr = BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal = BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#####################################################################
IDTimeNumber = str(uuid.uuid4())
TimeLine = [
("TimeID", [IDTimeNumber]),
("UTCDate", [BirthDateZoneStr]),
("LocalTime", [BirthDateLocal]),
("TimeZone", [BirthZone]),
]
```

```
TimeFrame = pd.DataFrame.from_dict(dict(TimeLine))
#####################################################################
DimTime = TimeFrame
DimTimeIndex = DimTime.set_index(["TimeID"], inplace=False)
#####################################################################
sTable = "Dim-Time"
print("\n#################################")
print("Storing :", sDatabaseName, "\n Table:", sTable)
print("\n#################################")
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print("\n#################################")
print("Dimension Person")
print("\n#################################")
FirstName = "Guðmundur"
LastName = "Gunnarsson"
#####################################################################
IDPersonNumber = str(uuid.uuid4())
PersonLine = [
("PersonID", [IDPersonNumber]),
("FirstName", [FirstName]),
("LastName", [LastName]),
("Zone", ["UTC"]),
("DateTimeValue", [BirthDateZoneStr]),
]
PersonFrame = pd.DataFrame.from_dict(dict(PersonLine))
#####################################################################
DimPerson = PersonFrame
DimPersonIndex = DimPerson.set_index(["PersonID"], inplace=False)
#####################################################################
sTable = "Dim-Person"
print("\n#################################")
print("Storing :", sDatabaseName, "\n Table:", sTable)
print("\n#################################")
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
print("\n#################################")
print("Fact - Person - time")
print("\n#################################")
IDFactNumber = str(uuid.uuid4())
PersonTimeLine = [
("IDNumber", [IDFactNumber]),
("IDPersonNumber", [IDPersonNumber]),
("IDTimeNumber", [IDTimeNumber]),
]
PersonTimeFrame = pd.DataFrame.from_dict(dict(PersonTimeLine))
#####################################################################
FctPersonTime = PersonTimeFrame
FctPersonTimeIndex = FctPersonTime.set_index(["IDNumber"],
```

```
inplace=False)
#####################################################################
sTable = "Fact-Person-Time"
print("\n#################################")
print("Storing :", sDatabaseName, "\n Table:", sTable)
print("\n#################################")
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####################################################################
```

Output:

```
################################
Working Base : D:/VKHCG  using  win32
################################

################################
Time Dimension

################################
Storing : D:/VKHCG/99-DW/datawarehouse.db
 Table: Dim-Time

################################

################################
Dimension Person

################################

################################
Storing : D:/VKHCG/99-DW/datawarehouse.db
 Table: Dim-Person

################################

################################
Fact - Person - time

################################

################################
Storing : D:/VKHCG/99-DW/datawarehouse.db
 Table: Fact-Person-Time

################################
```

Practical No: 08

Aim: 8A. Write a program to perform horizontal-style slicing orsubsetting of the data warehouse

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#####################################################################
Base = "D:/VKHCG"
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#####################################################################
#####################################################################
Company = "01-Vermeulen"
#####################################################################
sDataWarehouseDir = Base + "/99-DW"
if not os.path.exists(sDataWarehouseDir):
os.makedirs(sDataWarehouseDir)
#####################################################################
sDatabaseName = sDataWarehouseDir + "/datawarehouse.db"
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName = sDataWarehouseDir + "/datamart.db"
conn2 = sq.connect(sDatabaseName)
#####################################################################
print("################")
sTable = "Dim-BMI"
print("Loading :", sDatabaseName, " Table:", sTable)
sSQL = "SELECT * FROM [Dim-BMI];"
PersonFrame0 = pd.read_sql_query(sSQL, conn1)
print("################")
sTable = "Dim-BMI"
print("Loading :", sDatabaseName, " Table:", sTable)
sSQL = "SELECT PersonID, \
Height,\
Weight,\
bmi,\
Indicator\
FROM [Dim-BMI]\
WHERE \
Height > 1.5 \
and Indicator = 1 \
ORDER BY \
Height,\
Weight;"

PersonFrame1 = pd.read_sql_query(sSQL, conn1)
#####################################################################
DimPerson = PersonFrame1
DimPersonIndex = DimPerson.set_index(["PersonID"], inplace=False)
```

```
#################################################################
sTable = "Dim-BMI"
print("\n#################################")
print("Storing :", sDatabaseName, "\n Table:", sTable)
print("\n#################################")
# DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print("#################")
sTable = "Dim-BMI"
print("Loading :", sDatabaseName, " Table:", sTable)
sSQL = "SELECT * FROM [Dim-BMI];"
PersonFrame2 = pd.read_sql_query(sSQL, conn2)
print("Full Data Set (Rows):", PersonFrame0.shape[0])
print("Full Data Set (Columns):", PersonFrame0.shape[1])
print("Horizontal Data Set (Rows):", PersonFrame2.shape[0])
print("Horizontal Data Set (Columns):", PersonFrame2.shape[1])
```

Output:

```
################################
Working Base : D:/VKHCG  using  win32
################################
#################
Loading : D:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
#################
Loading : D:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

################################
Storing : D:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI

################################
#################
Loading : D:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
```

8B. Write a program to perform association rule mining on the given data

```
import os
import sys
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
#####################################################################
Base = "D:/VKHCG"
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
Company = "01-Vermeulen"
InputFileName = "Online-Retail-Billboard.xlsx"
EDSAssessDir = "02-Assess/01-EDS"
InputAssessDir = EDSAssessDir + "/02-Python"
#####################################################################
sFileAssessDir = Base + "/" + Company + "/" + InputAssessDir
if not os.path.exists(sFileAssessDir):
os.makedirs(sFileAssessDir)
#####################################################################
sFileName = Base + "/" + Company + "/00-RawData/" + InputFileName
#####################################################################
df = pd.read_excel(sFileName)
print(df.shape)
#####################################################################
df["Description"] = df["Description"].str.strip()
df.dropna(axis=0, subset=["InvoiceNo"], inplace=True)
df["InvoiceNo"] = df["InvoiceNo"].astype("str")
df = df[~df["InvoiceNo"].str.contains("C")]
basket = (
df[df["Country"] == "France"]
.groupby(["InvoiceNo", "Description"])["Quantity"]
.sum()
.unstack()
.reset_index()
.fillna(0)
.set_index("InvoiceNo"))
#####################################################################
def encode_units(x):
if x <= 0:
return 0
if x >= 1:
return 1
#####################################################################
basket_sets = basket.applymap(encode_units)
basket_sets.drop("POSTAGE", inplace=True, axis=1)
frequent_itemsets = apriori(basket_sets, min_support=0.07,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift",
min_threshold=1)
print(rules.head())
```

```
rules[(rules["lift"] >= 6) & (rules["confidence"] >= 0.8)]
#######################################################################
sProduct1 = "ALARM CLOCK BAKELIKE GREEN"
print(sProduct1)
print(basket[sProduct1].sum())
sProduct2 = "ALARM CLOCK BAKELIKE RED"
print(sProduct2)
print(basket[sProduct2].sum())
#######################################################################
basket2 = (df[df["Country"] == "Germany"]
.groupby(["InvoiceNo", "Description"])["Quantity"]
.sum()
.unstack()
.reset_index()
.fillna(0)
.set_index("InvoiceNo"))
basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop("POSTAGE", inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05,
use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift",
min_threshold=1)
print(rules2[(rules2["lift"] >= 4) & (rules2["confidence"] >= 0.5)])
#######################################################################
print("### Done!! ############################################")
#######################################################################
```

Output:

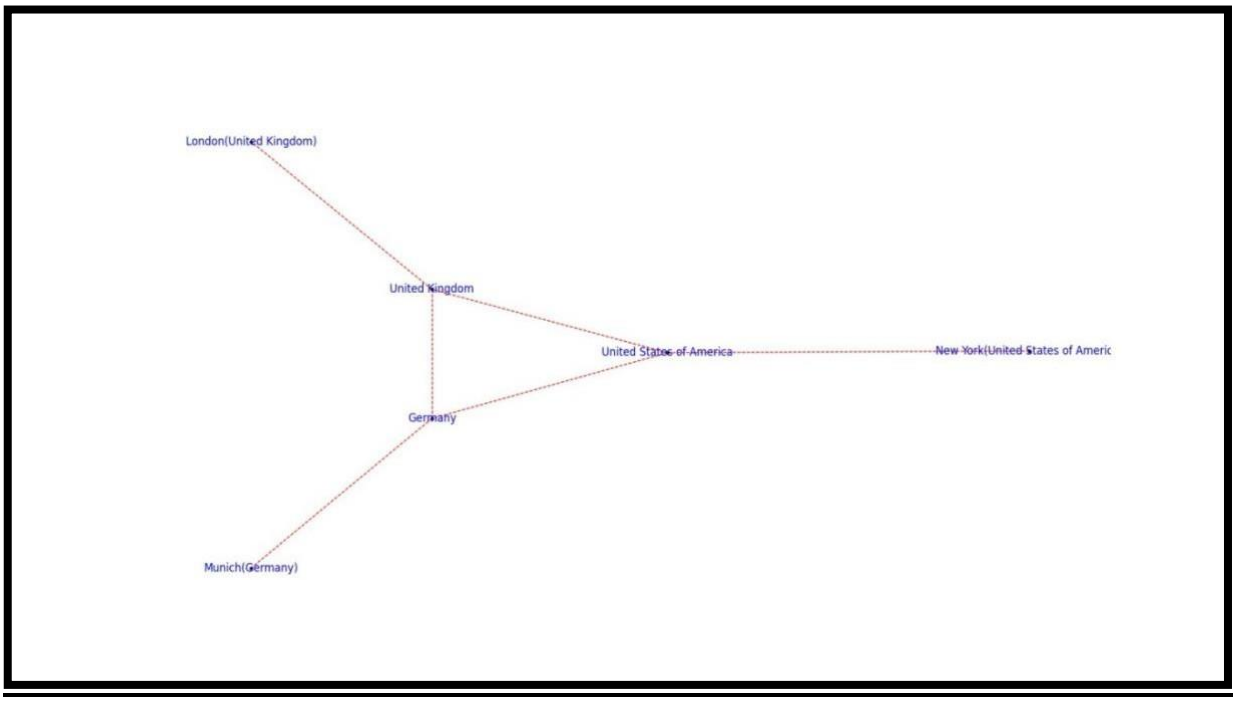
```
##############################
Working Base : D:/VKHCG  using  win32
##############################
(541909, 8)
                 antecedents                  consequents  antecedent support  consequent support  ...  confidence      lift  leverage  conviction
0   (ALARM CLOCK BAKELIKE PINK)  (ALARM CLOCK BAKELIKE GREEN)            0.102041            0.096939  ...    0.725000  7.478947  0.064088    3.283859
1  (ALARM CLOCK BAKELIKE GREEN)   (ALARM CLOCK BAKELIKE PINK)            0.096939            0.102041  ...    0.763158  7.478947  0.064088    3.791383
2  (ALARM CLOCK BAKELIKE GREEN)    (ALARM CLOCK BAKELIKE RED)            0.096939            0.094388  ...    0.815789  8.642959  0.069932    4.916181
3    (ALARM CLOCK BAKELIKE RED)  (ALARM CLOCK BAKELIKE GREEN)            0.094388            0.096939  ...    0.837838  8.642959  0.069932    5.568878
4   (ALARM CLOCK BAKELIKE PINK)    (ALARM CLOCK BAKELIKE RED)            0.102041            0.094388  ...    0.725000  7.681081  0.064348    3.293135

[5 rows x 9 columns]
ALARM CLOCK BAKELIKE GREEN
340.0
ALARM CLOCK BAKELIKE RED
316.0
                     antecedents                       consequents  antecedent support  ...      lift  leverage  conviction
1   (PLASTERS IN TIN CIRCUS PARADE)  (PLASTERS IN TIN WOODLAND ANIMALS)            0.115974  ...  4.242887  0.051846    2.076984
7        (PLASTERS IN TIN SPACEBOY)  (PLASTERS IN TIN WOODLAND ANIMALS)            0.107221  ...  4.145125  0.046488    2.011670
10    (RED RETROSPOT CHARLOTTE BAG)           (WOODLAND CHARLOTTE BAG)            0.070822  ...  6.648168  0.050194    5.587746

[3 rows x 9 columns]
### Done!! ############################################
```

8C. Write a program to create a Network Routing Diagram using given data.

```
import sys
import matplotlib.pyplot as plt
import networkx as nx
import pandas as pd
#####################################################################
pd.options.mode.chained_assignment = None
#####################################################################
Base = "D:/VKHCG"
#####################################################################
print("#################################")
print("Working Base :", Base, " using ", sys.platform)
print("#################################")
#####################################################################
sInputFileName = "02-Assess/01-EDS/02-Python/Assess-Network-Routing-
Company.csv"
#####################################################################
sOutputFileName1 = "05-Organise/01-EDS/02-Python/Organise-Network-
Routing-Company.gml"
sOutputFileName2 = "05-Organise/01-EDS/02-Python/Organise-Network-
Routing-Company.png"
Company = "01-Vermeulen"
#####################################################################
#####################################################################
### Import Country Data
#####################################################################
sFileName = Base + "/" + Company + "/" + sInputFileName
print("#################################")
print("Loading :", sFileName)
print("#################################")
CompanyData = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
print("#################################")
#####################################################################
print(CompanyData.head())
print(CompanyData.shape)
#####################################################################
G = nx.Graph()
for i in range(CompanyData.shape[0]):
for j in range(CompanyData.shape[0]):
Node0 = CompanyData["Company_Country_Name"][i]
Node1 = CompanyData["Company_Country_Name"][j]

if Node0 != Node1:
G.add_edge(Node0, Node1)
for i in range(CompanyData.shape[0]):
Node0 = CompanyData["Company_Country_Name"][i]
Node1 = (
CompanyData["Company_Place_Name"][i]
+ "("
+ CompanyData["Company_Country_Name"][i]
```

```
+ ")"
)
if Node0 != Node1:
G.add_edge(Node0, Node1)
print("Nodes:", G.number_of_nodes())
print("Edges:", G.number_of_edges())
#####################################################################
sFileName = Base + "/" + Company + "/" + sOutputFileName1
print("################################")
print("Storing :", sFileName)
print("################################")
nx.write_gml(G, sFileName)
#####################################################################
sFileName = Base + "/" + Company + "/" + sOutputFileName2
print("################################")
print("Storing Graph Image:", sFileName)
print("################################")
plt.figure(figsize=(15, 15))
pos = nx.spectral_layout(G, dim=2)
nx.draw_networkx_nodes(G, pos, node_color="k", node_size=10,
alpha=0.8)
nx.draw_networkx_edges(G, pos, edge_color="r", arrows=False,
style="dashed")
nx.draw_networkx_labels(G, pos, font_size=12, font_family="sans-
serif", font_color="b")
plt.axis("off")
plt.savefig(sFileName, dpi=600)
plt.show()
#####################################################################
print("################################")
print("### Done!! ####################")
print("################################")
#####################################################################
```

Output:


London(United Kingdom)
United Kingdom
United States of America
New York(United States of Americ
Germany
Munich(Germany)

Practical No: 09

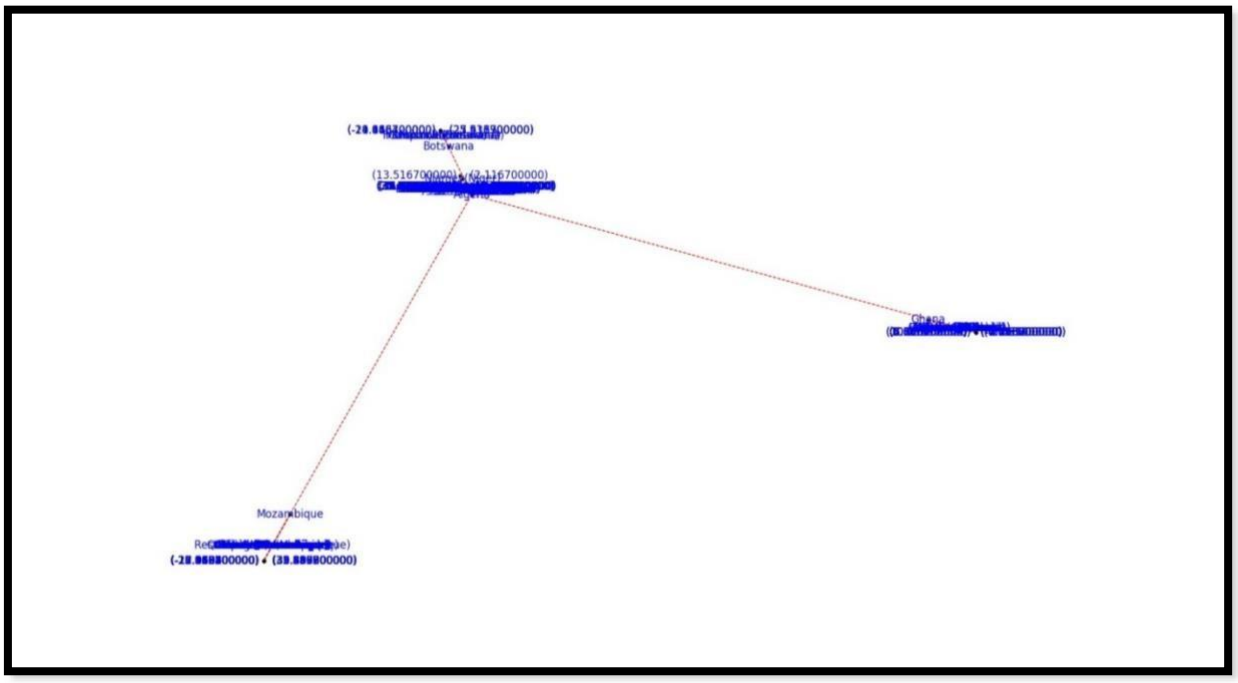
Aim: Generating Data
9A. Write a program to perform Report Superstep on Vermeulen PLC

```
import sys
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
##################################################################
pd.options.mode.chained_assignment = None
##################################################################
Base = "D:/VKHCG"
##################################################################
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
##################################################################
sInputFileName = "02-Assess/01-EDS/02-Python/Assess-Network-Routing-
Customer.csv"
##################################################################
sOutputFileName1 = "06-Report/01-EDS/02-Python/Report-Network-Routing-
Customer.gml"
sOutputFileName2 = "06-Report/01-EDS/02-Python/Report-Network-Routing-
Customer.png"
Company = "01-Vermeulen"
##################################################################
##################################################################
### Import Country Data
##################################################################
sFileName = Base + "/" + Company + "/" + sInputFileName
print("################################")
print("Loading :", sFileName)
print("################################")
CustomerDataRaw = pd.read_csv(sFileName, header=0, low_memory=False,
encoding="latin-1")
CustomerData = CustomerDataRaw.head(100)
print("Loaded Country:", CustomerData.columns.values)
print("################################")
##################################################################
print(CustomerData.head())
##################################################################
G = nx.Graph()
for i in range(CustomerData.shape[0]):
for j in range(CustomerData.shape[0]):
Node0 = CustomerData["Customer_Country_Name"][i]
Node1 = CustomerData["Customer_Country_Name"][j]
if Node0 != Node1:
G.add_edge(Node0, Node1)
for i in range(CustomerData.shape[0]):
Node0 = CustomerData["Customer_Country_Name"][i]
```

```
Node1 = (
CustomerData["Customer_Place_Name"][i]
+ "("
+ CustomerData["Customer_Country_Name"][i]
+ ")"
)
Node2 = (
"("
+ "{:.9f}".format(CustomerData["Customer_Latitude"][i])
+ ")\
("
+ "{:.9f}".format(CustomerData["Customer_Longitude"][i])
+ ")"
)
if Node0 != Node1:
G.add_edge(Node0, Node1)
if Node1 != Node2:
G.add_edge(Node1, Node2)
print("Nodes:", G.number_of_nodes())
print("Edges:", G.number_of_edges())
#####################################################################
sFileName = Base + "/" + Company + "/" + sOutputFileName1
print("################################")
print("Storing :", sFileName)
print("################################")
nx.write_gml(G, sFileName)
#####################################################################
sFileName = Base + "/" + Company + "/" + sOutputFileName2
print("################################")
print("Storing Graph Image:", sFileName)
print("################################")
plt.figure(figsize=(25, 25))
pos = nx.spectral_layout(G, dim=2)
nx.draw_networkx_nodes(G, pos, node_color="k", node_size=10,
alpha=0.8)
nx.draw_networkx_edges(G, pos, edge_color="r", arrows=False,
style="dashed")

nx.draw_networkx_labels(G, pos, font_size=12, font_family="sans-
serif", font_color="b")
plt.axis("off")
plt.savefig(sFileName, dpi=600)
plt.show()
print("################################")
print("### Done!! ####################")
print("################################")
```

Output:

```
################################
Working Base : D:/VKHCG  using  win32
################################
################################
Loading : D:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code Customer_Place_Name  Customer_Latitude  Customer_Longitude Customer_Country_Name
0                    BW            Gaborone          -24.6464             25.9119              Botswana
1                    BW         Francistown          -21.1667             27.5167              Botswana
2                    BW               Maun          -19.9833             23.4167              Botswana
3                    BW          Molepolole          -24.4167             25.5333              Botswana
4                    NE              Niamey           13.5167              2.1167                 Niger
Nodes: 205
Edges: 204
################################
Storing : D:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml
################################
################################
Storing Graph Image: D:/VKHCG/01-Vermeulen/06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png
################################
################################
### Done!! ######################
################################
```



9B. Write a program for Hillman Ltd to convert all numbers on thesides of containers into digits.

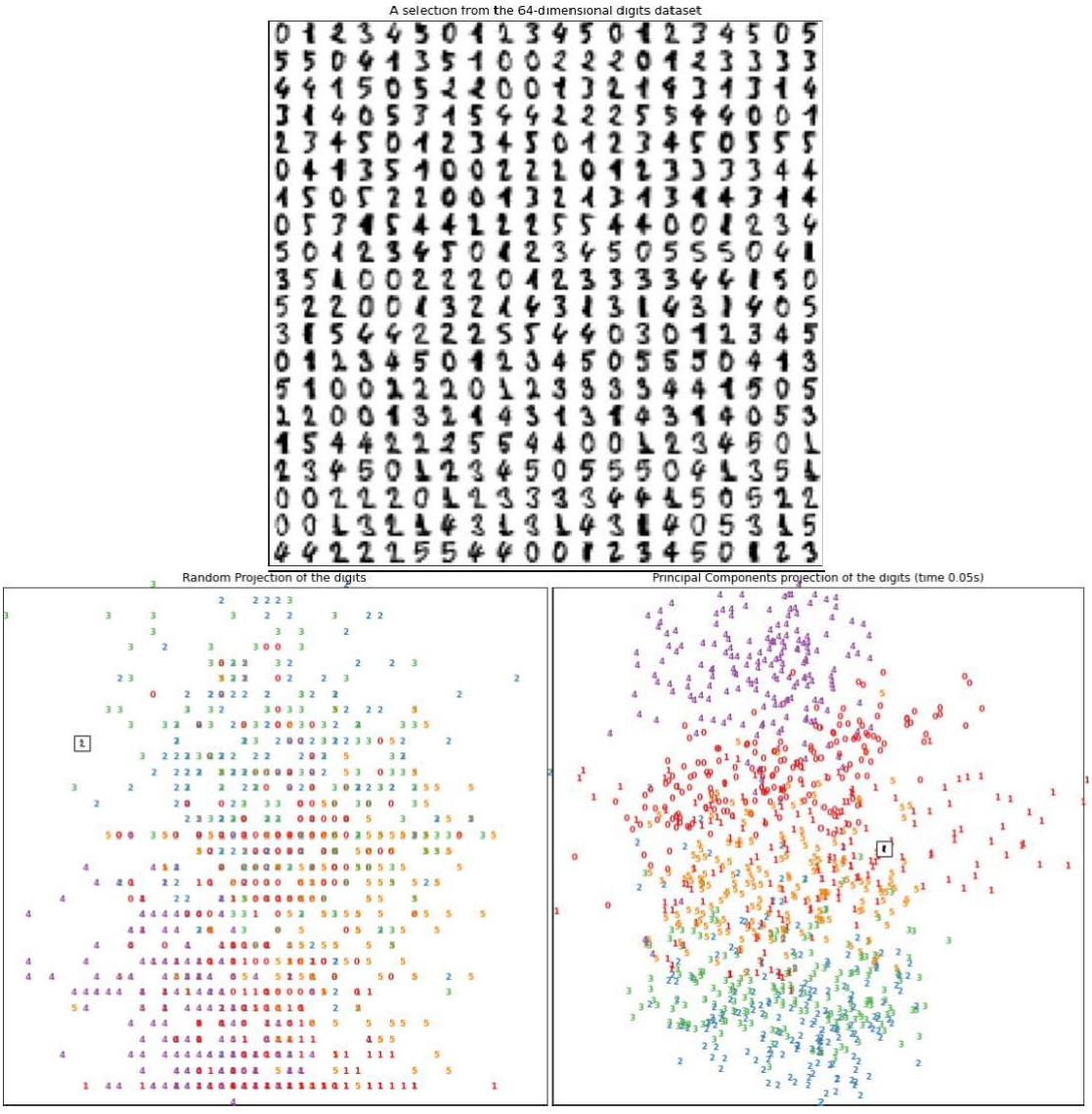
```
from time import time
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import offsetbox
from sklearn import (datasets, decomposition, discriminant_analysis,
ensemble,
manifold, random_projection)
digits = datasets.load_digits(n_class=6)
X = digits.data
y = digits.targetō
n_samples, n_features = X.shape
n_neighbors = 30
def plot_embedding(X, title=None):
x_min, x_max = np.min(X, 0), np.max(X, 0)
X = (X - x_min) / (x_max - x_min)
plt.figure(figsize=(10, 10))
ax = plt.subplot(111)
for i in range(X.shape[0]):
plt.text(
X[i, 0],
X[i, 1],
str(digits.target[i]),
color=plt.cm.Set1(y[i] / 10.0),
fontdict={"weight": "bold", "size": 9},
)
if hasattr(offsetbox, "AnnotationBbox"):
# only print thumbnails with matplotlib > 1.0
shown_images = np.array([[1.0, 1.0]]) # just something big
for i in range(digits.data.shape[0]):
dist = np.sum((X[i] - shown_images) ** 2, 1)
if np.min(dist) < 4e-3:
# don't show points that are too close
continue
shown_images = np.r_[shown_images, [X[i]]]
imagebox = offsetbox.AnnotationBbox(
offsetbox.OffsetImage(digits.images[i],
cmap=plt.cm.gray_r), X[i]
)
ax.add_artist(imagebox)
plt.xticks([]), plt.yticks([])
if title is not None:
plt.title(title)

n_img_per_row = 20
img = np.zeros((10 * n_img_per_row, 10 * n_img_per_row))
for i in range(n_img_per_row):
ix = 10 * i + 1
for j in range(n_img_per_row):
iy = 10 * j + 1
```

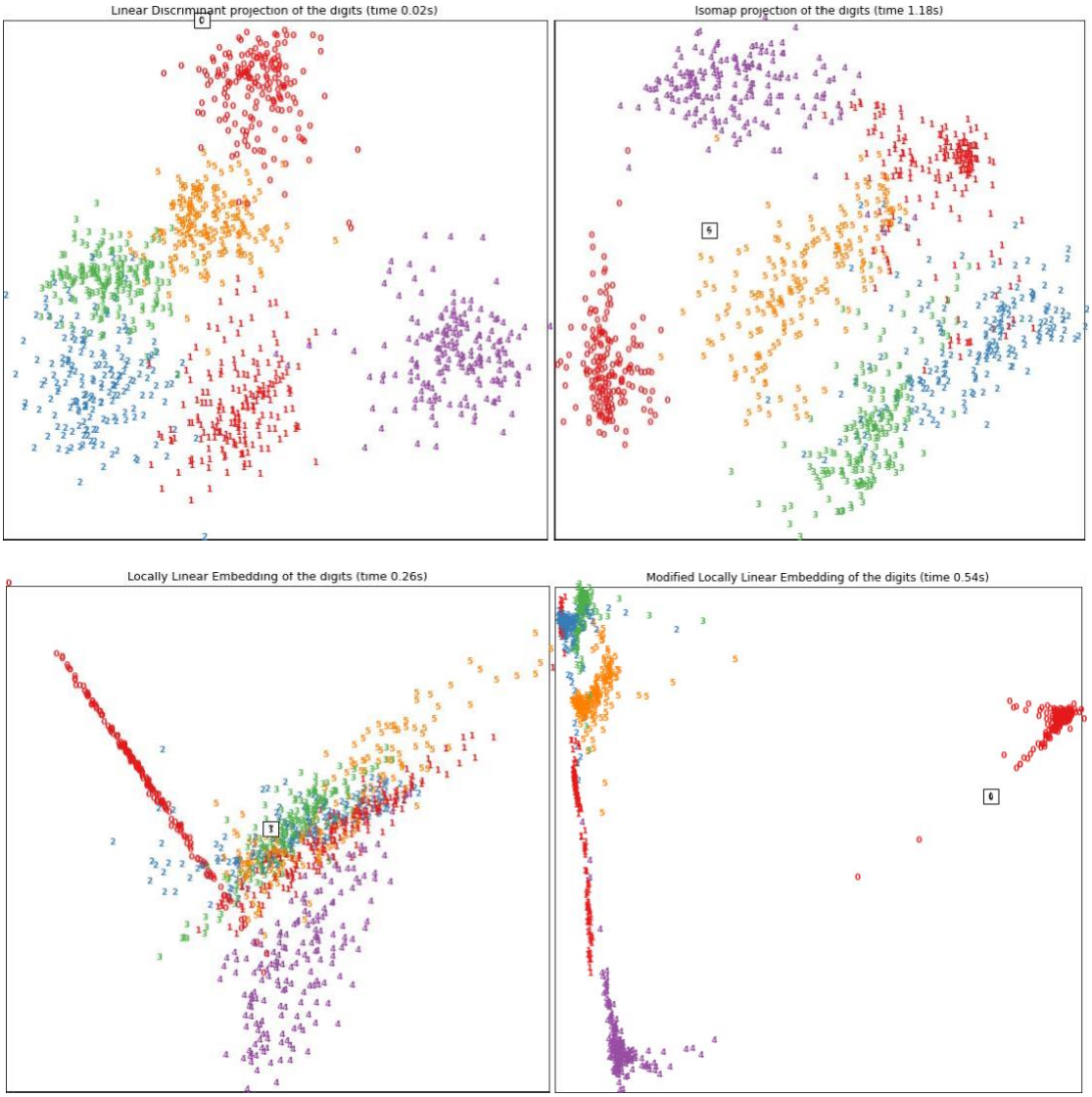
```
img[ix : ix + 8, iy : iy + 8] = X[i * n_img_per_row +
j].reshape((8, 8))
plt.figure(figsize=(10, 10))
plt.imshow(img, cmap=plt.cm.binary)
plt.xticks([])
plt.yticks([])
plt.title("A selection from the 64-dimensional digits dataset")
print("Computing random projection")
rp = random_projection.SparseRandomProjection(n_components=2,
random_state=42)
X_projected = rp.fit_transform(X)
plot_embedding(X_projected, "Random Projection of the digits")
print("Computing PCA projection")
t0 = time()
X_pca = decomposition.TruncatedSVD(n_components=2).fit_transform(X)
plot_embedding(
X_pca, "Principal Components projection of the digits (time
%.2fs)" % (time() - t0)
)
print("Computing Linear Discriminant Analysis projection")
X2 = X.copy()
X2.flat[:: X.shape[1] + 1] += 0.01 # Make X invertible
t0 = time()
X_lda =
discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_t
ransform(
X2, y
)
plot_embedding(
X_lda, "Linear Discriminant projection of the digits (time %.2fs)"
% (time() - t0)
)
print("Computing Isomap embedding")
t0 = time()
X_iso = manifold.Isomap(n_neighbors, n_components=2).fit_transform(X)
print("Done.")
plot_embedding(X_iso, "Isomap projection of the digits (time %.2fs)" %
(time() - t0))
print("Computing LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,
method="standard")
t0 = time()
X_lle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(
X_lle, "Locally Linear Embedding of the digits (time %.2fs)" %
(time() - t0)
)
print("Computing modified LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,
method="modified")
```

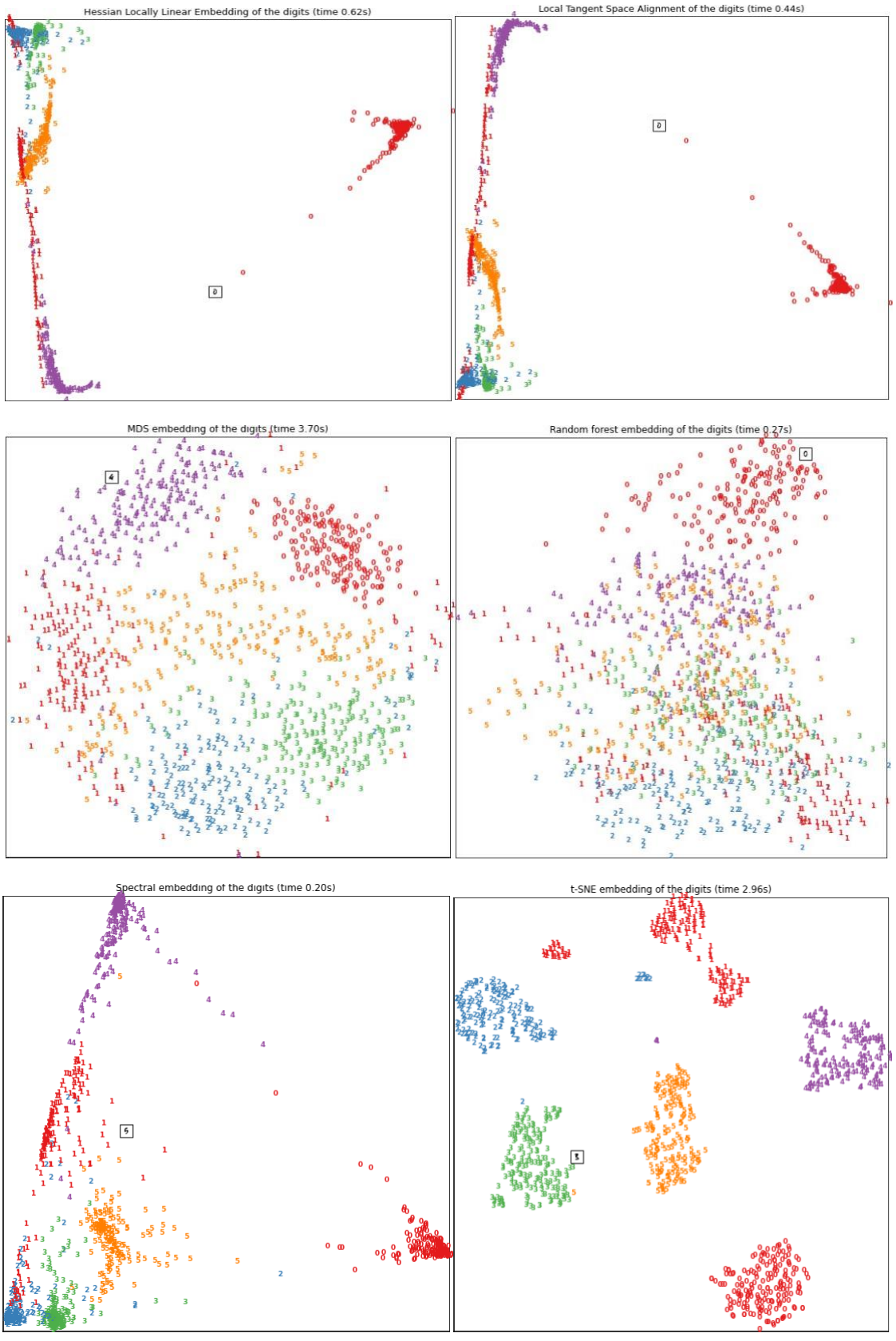
```
t0 = time()
X_mlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(
X_mlle,
"Modified Locally Linear Embedding of the digits (time %.2fs)" %
(time() - t0),
)
print("Computing Hessian LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,
method="hessian")
t0 = time()
X_hlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(
X_hlle,
"Hessian Locally Linear Embedding of the digits (time %.2fs)" %
(time() - t0),
)
print("Computing LTSA embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,
method="ltsa")
t0 = time()
X_ltsa = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(
X_ltsa, "Local Tangent Space Alignment of the digits (time %.2fs)"
% (time() - t0)
)
print("Computing MDS embedding")
clf = manifold.MDS(n_components=2, n_init=1, max_iter=100)
t0 = time()
X_mds = clf.fit_transform(X)
print("Done. Stress: %f" % clf.stress_)
plot_embedding(X_mds, "MDS embedding of the digits (time %.2fs)" %
(time() - t0))
print("Computing Totally Random Trees embedding")
hasher = ensemble.RandomTreesEmbedding(n_estimators=200,
random_state=0, max_depth=5)
t0 = time()
X_transformed = hasher.fit_transform(X)
pca = decomposition.TruncatedSVD(n_components=2)
X_reduced = pca.fit_transform(X_transformed)
plot_embedding(
X_reduced, "Random forest embedding of the digits (time %.2fs)" %
(time() - t0)
)
print("Computing Spectral embedding")
embedder = manifold.SpectralEmbedding(
n_components=2, random_state=0, eigen_solver="arpack"
)
```

```
t0 = time()
X_se = embedder.fit_transform(X)
plot_embedding(X_se, "Spectral embedding of the digits (time %.2fs)" %
(time() - t0))
print("Computing t-SNE embedding")
tsne = manifold.TSNE(n_components=2, init="pca", random_state=0)
t0 = time()
X_tsne = tsne.fit_transform(X)
plot_embedding(X_tsne, "t-SNE embedding of the digits (time %.2fs)" %
(time() - t0))
plt.show()
```

Output:



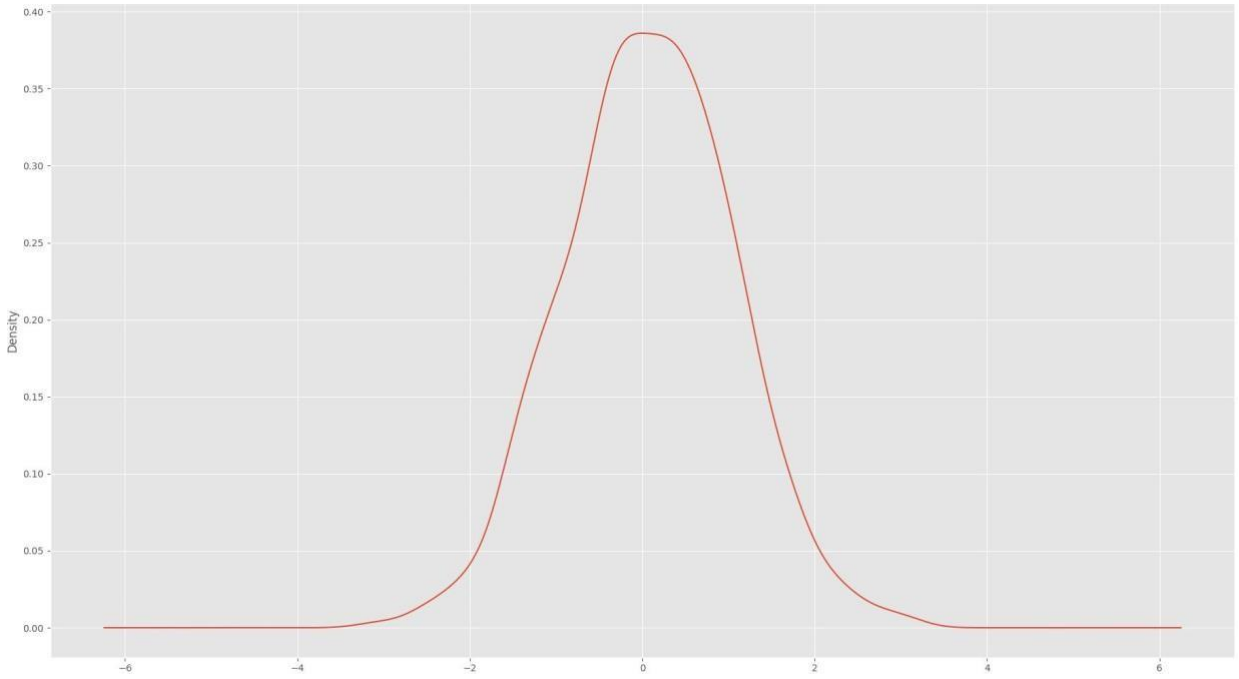

Linear Discriminant projection of the digits (time 0.02s)
Isomap projection of the digits (time 1.18s)
Locally Linear Embedding of the digits (time 0.26s)
Modified Locally Linear Embedding of the digits (time 0.54s)

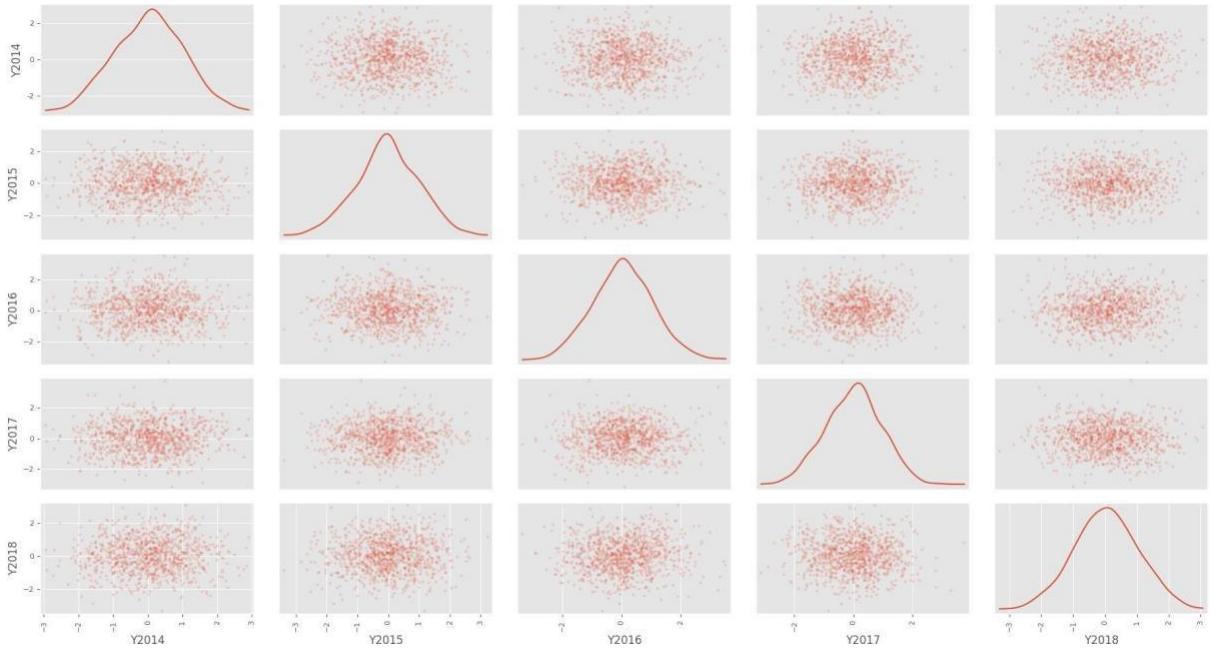

Hessian Locally Linear Embedding of the digits (time 0.62s)
Local Tangent Space Alignment of the digits (time 0.44s)
MDS embedding of the digits (time 3.70s)
Random forest embedding of the digits (time 0.27s)
Spectral embedding of the digits (time 0.20s)
t-SNE embedding of the digits (time 2.96s)

9C. Write a program to perform Kernel Density

```
import sys
import pandas as pd
import matplotlib as ml
import numpy as np
from matplotlib import pyplot as plt
#####################################################################
Base = "D:/VKHCG"
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#####################################################################
ml.style.use("ggplot")
fig1 = plt.figure(figsize=(10, 10))
ser = pd.Series(np.random.randn(1000))
ser.plot(figsize=(10, 10), kind="kde")
sPicNameOut1 = Base + "/01-Vermeulen/06-Report/01-EDS/02-
Python/kde.png"
plt.savefig(sPicNameOut1, dpi=600)
plt.tight_layout()
plt.show()
fig2 = plt.figure(figsize=(10, 10))
from pandas.plotting import scatter_matrix
df = pd.DataFrame(
np.random.randn(1000, 5), columns=["Y2014", "Y2015", "Y2016",
"Y2017", "Y2018"]
)
scatter_matrix(df, alpha=0.2, figsize=(10, 10), diagonal="kde")
sPicNameOut2 = Base + "/01-Vermeulen/06-Report/01-EDS/02-
Python/scatter_matrix.png"
plt.savefig(sPicNameOut2, dpi=600)
plt.tight_layout()
plt.show()
```

Output:

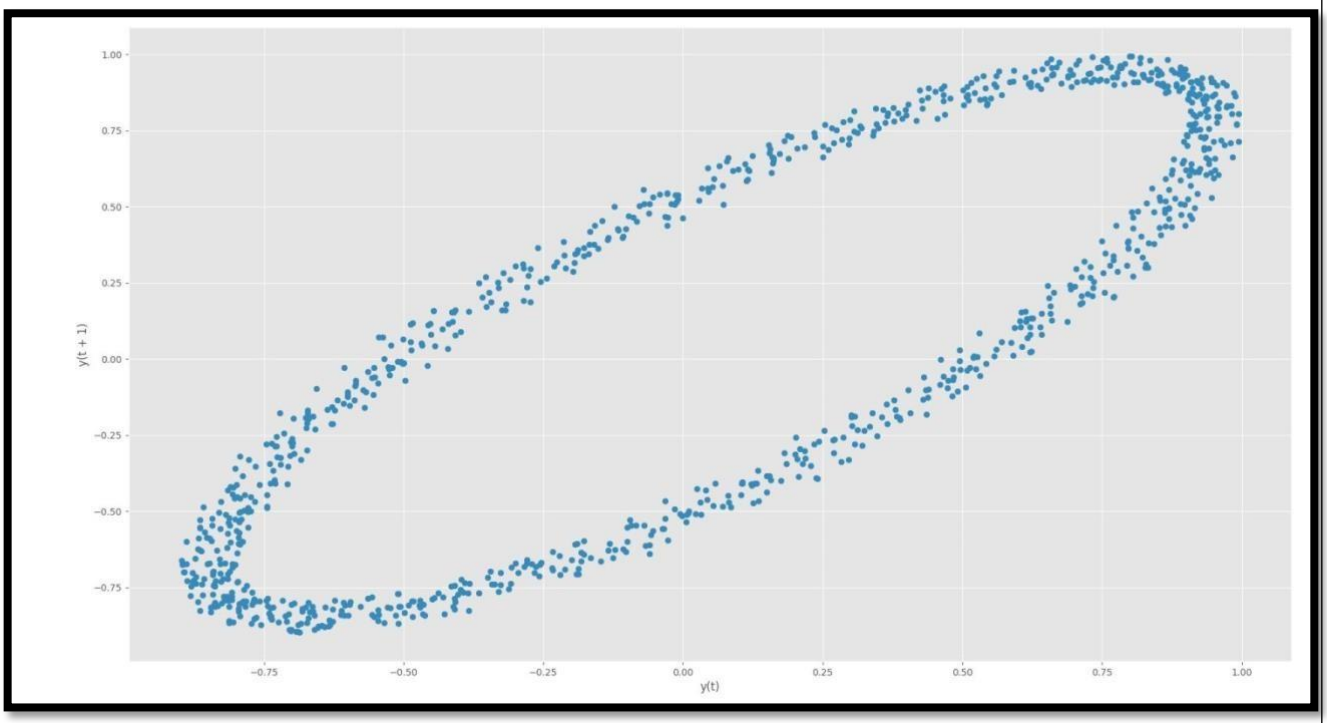

Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−6
−4
−2
0
2
4
6

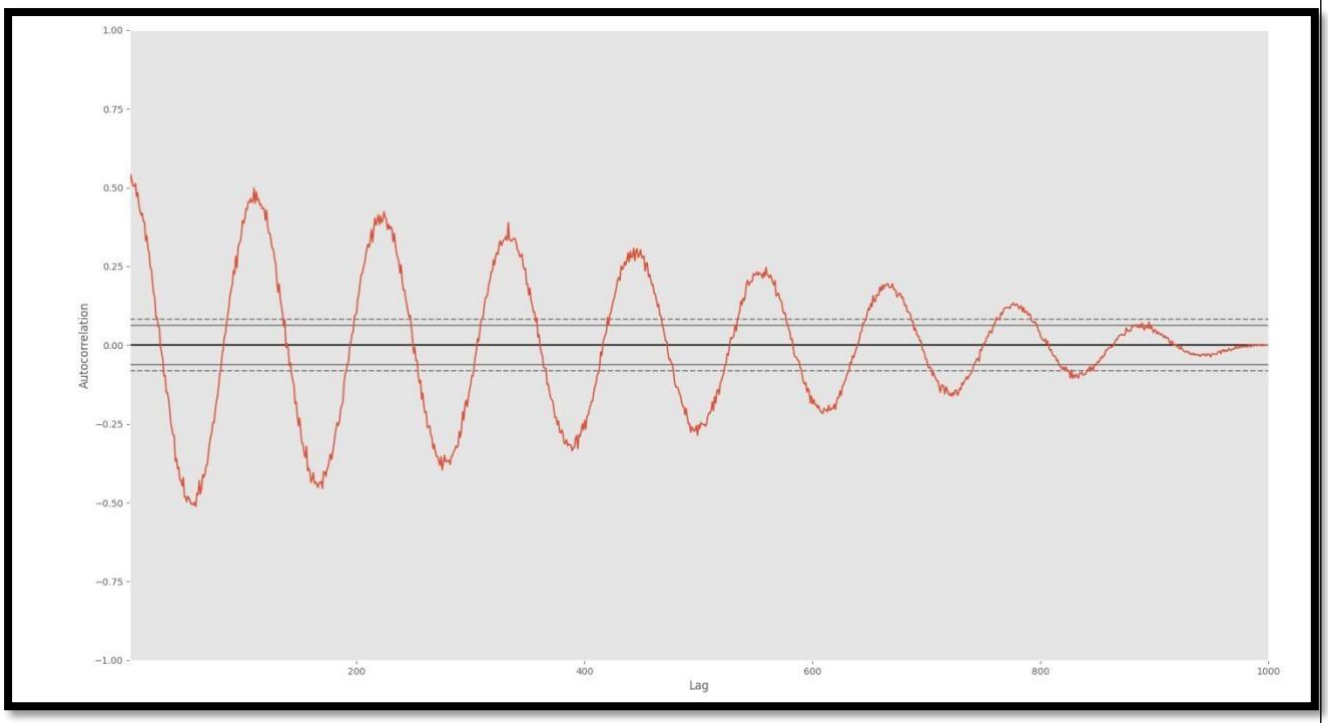

Y2014
Y2015
Y2016
Y2017
Y2018

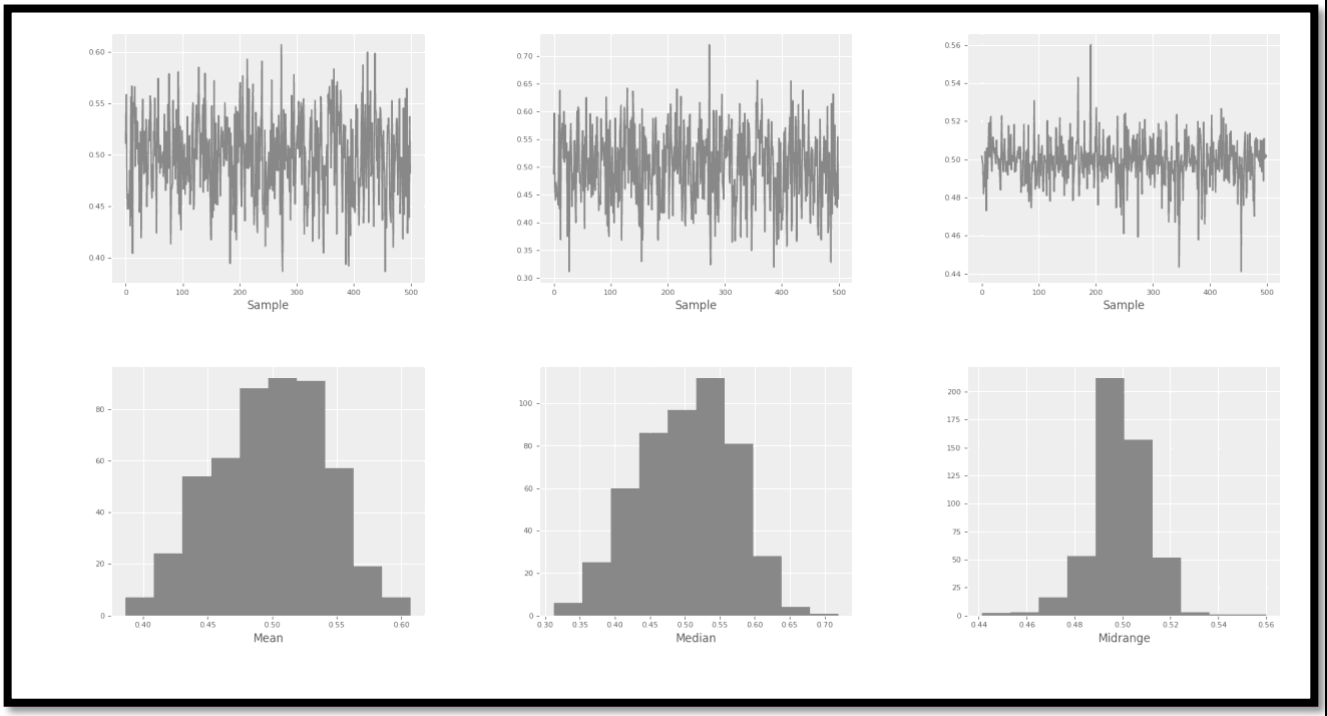
9D. Write a program to plot Lag Plot, Autocorrelation, and Bootstrap Plot

```
import sys
import pandas as pd
from matplotlib import style
from matplotlib import pyplot as plt
import numpy as np
#####################################################################
Base = "D:/VKHCG"
print("################################")
print("Working Base :", Base, " using ", sys.platform)
print("################################")
#####################################################################
style.use("ggplot")
from pandas.plotting import lag_plot
plt.figure(figsize=(10, 10))
data = pd.Series(
0.1 * np.random.rand(1000)
+ 0.9 * np.sin(np.linspace(-99 * np.pi, 99 * np.pi, num=1000))
)
lag_plot(data)
sPicNameOut1 = Base + "/01-Vermeulen/06-Report/01-EDS/02-
Python/lag_plot.png"
plt.savefig(sPicNameOut1, dpi=600)
plt.tight_layout()
plt.show()
from pandas.plotting import autocorrelation_plot
plt.figure(figsize=(10, 10))
data = pd.Series(
0.7 * np.random.rand(1000)
+ 0.3 * np.sin(np.linspace(-9 * np.pi, 9 * np.pi, num=1000))
)
autocorrelation_plot(data)
sPicNameOut2 = (
Base + "/01-Vermeulen/06-Report/01-EDS/02-
Python/autocorrelation_plot.png"
)

plt.savefig(sPicNameOut2, dpi=600)
plt.tight_layout()
plt.show()
from pandas.plotting import bootstrap_plot
data = pd.Series(np.random.rand(1000))
plt.figure(figsize=(10, 10))
bootstrap_plot(data, size=50, samples=500, color="grey")
sPicNameOut3 = Base + "/01-Vermeulen/06-Report/01-EDS/02-
Python/bootstrap_plot.png"
plt.savefig(sPicNameOut3, dpi=600)
plt.tight_layout()
plt.show()
```

Output:


y(t + 1)
y(t)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


Autocorrelation
Lag
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
200
400
600
800
1000


Sample
Sample
Sample
Mean
Median
Midrange

Practical No: 10

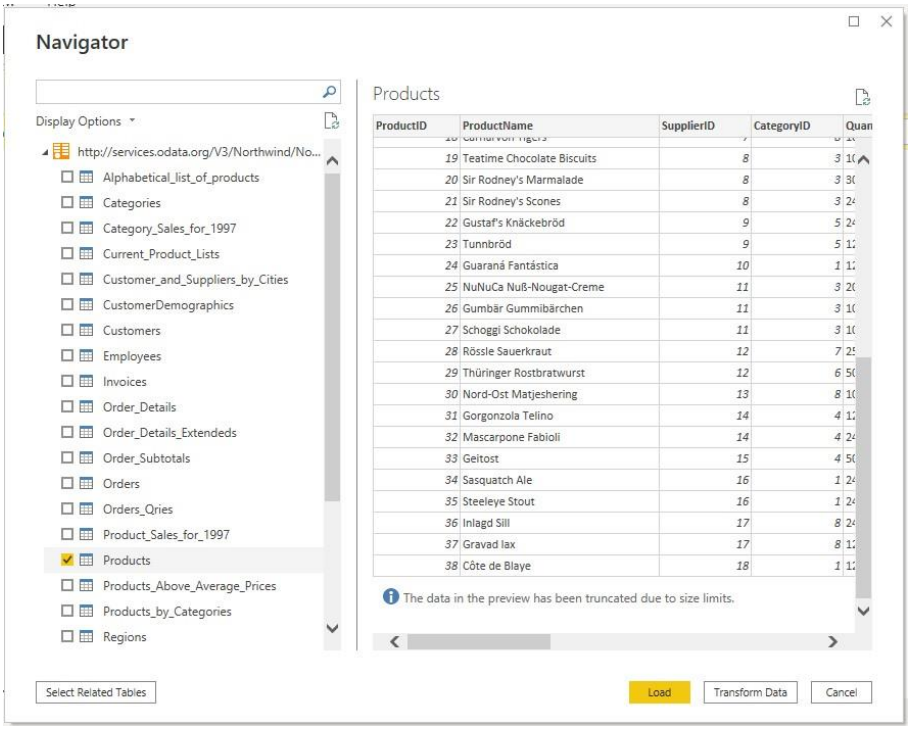
Aim: Data Visualization with Power BI

Importing Data from OData Feed

In this task, you'll bring in order data. This step represents connecting to a sales system. You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below:
http://services.odata.org/V3/Northwind/Northwind.svc/

Connect to an OData feed:

1. From the Home ribbon tab in Query Editor, select Get Data.
2. Browse to the OData Feed data source.
3. In the OData Feed dialog box, paste the URL for the Northwind OData feed.
4. Select OK.
5. In the Navigator pane, select the Orders table, and then select Edit.



ETL Process in Power BI

1. Remove other columns to only display columns of interest

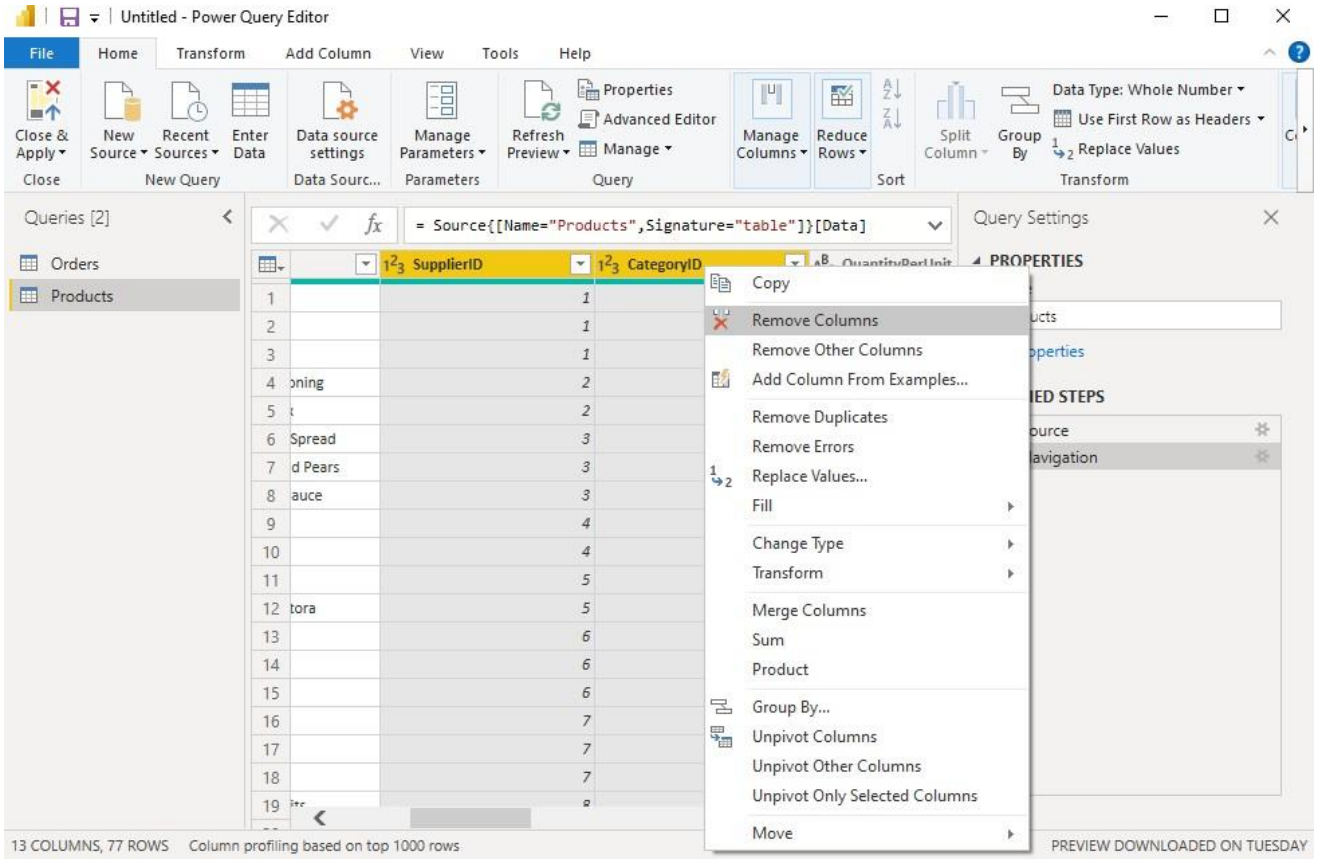
In this step you remove all columns except **ProductID**, **ProductName**, **UnitsInStock**, and

QuantityPerUnit

Power BI Desktop includes Query Editor, which is where you shape and transform your data connections. Query Editor opens automatically when you select **Edit** from Navigator. You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

1. In **Query Editor**, select the **ProductID, ProductName**, **QuantityPerUnit,** and **UnitsInStock** columns (use **Ctrl+Click** to select more than one column, or **Shift+Click** to select columns that are beside each other).

2. Select **Remove Columns > Remove** Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.



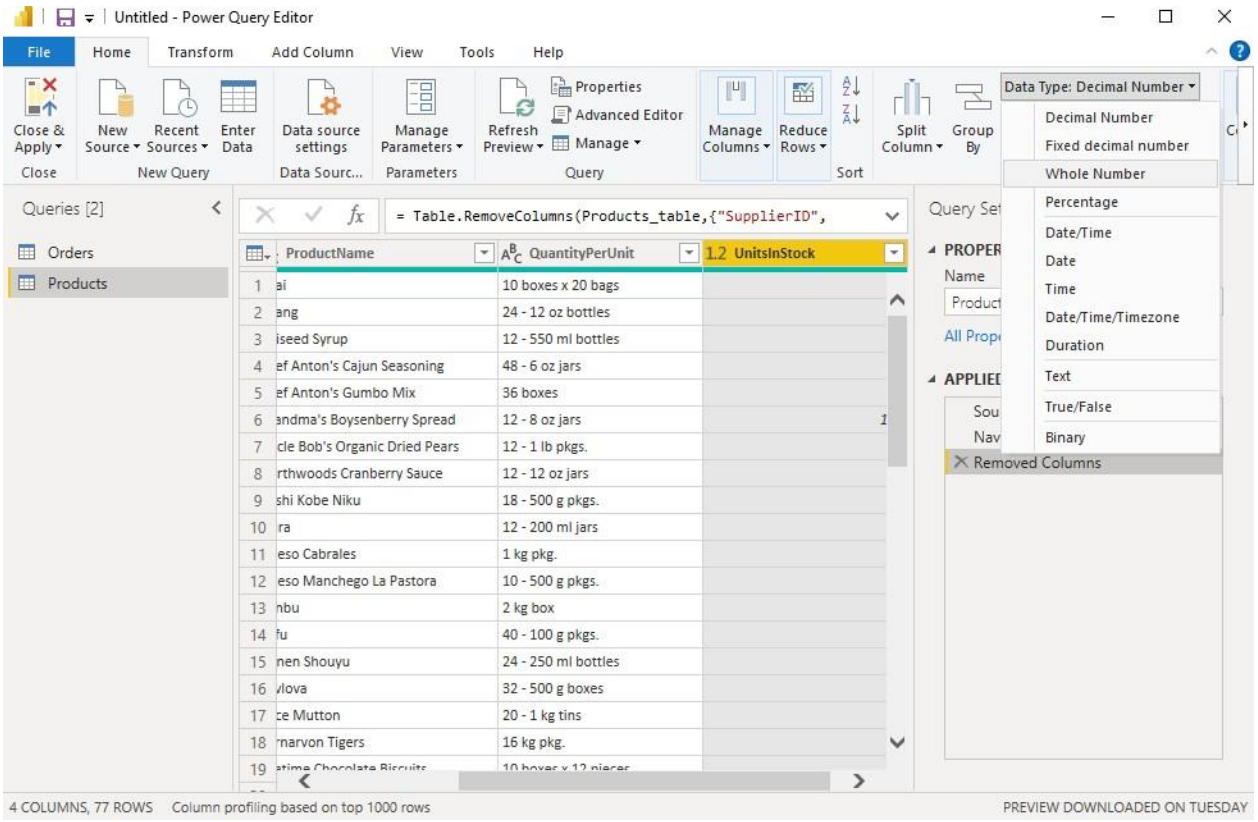
2. Change the data type of the UnitsInStock column

When Query Editor connects to data, it reviews each field and to determine the best data type.For the Excel workbook, products in stock will always be a whole number, so in this step youconfirm the **UnitsInStock** column's datatype is Whole Number.

1. Select the **UnitsInStock** column.

2. Select the **Data Type drop-down button** in the **Home ribbon**.

3. If not already a Whole Number, select **Whole Number** for data type from the drop down(the Data Type: button also displays the data type for the current selection).



3. Expand the Order_Details table

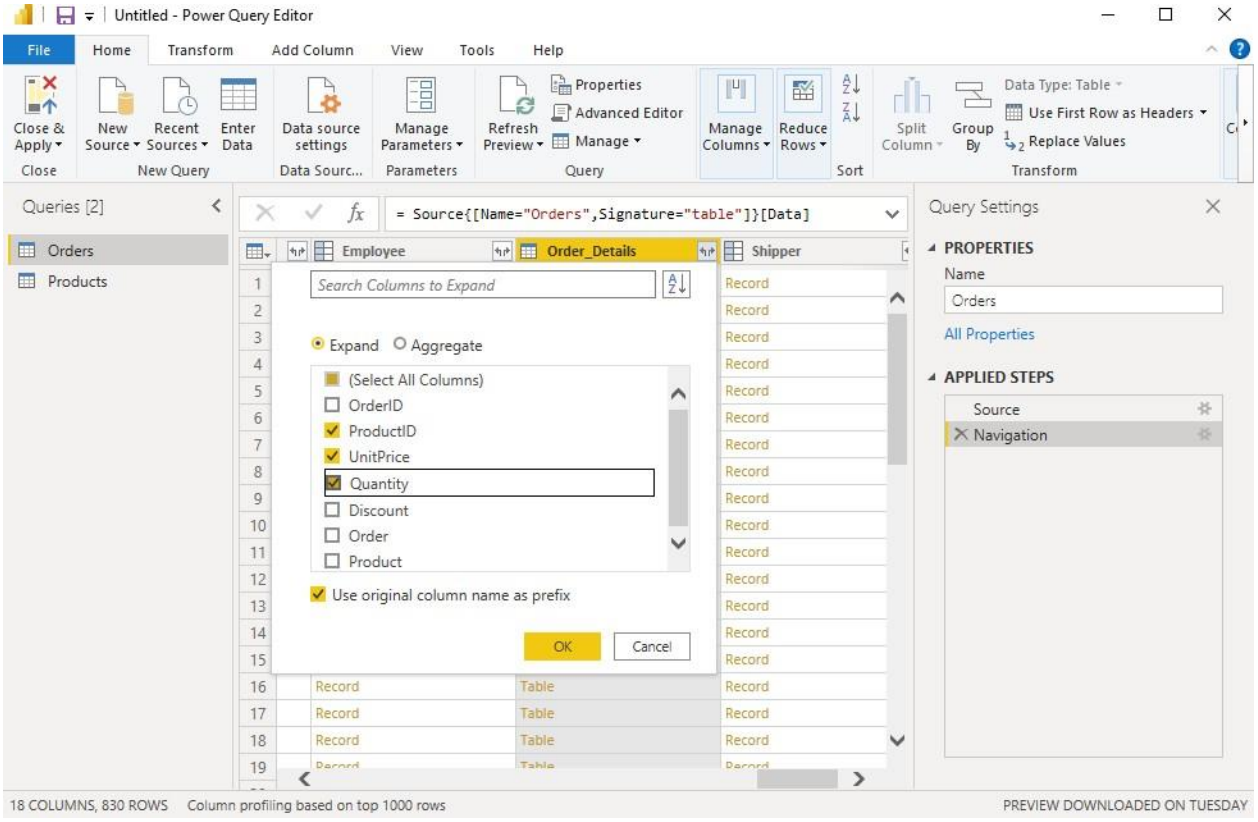
The Orders table contains a reference to a Details table, which contains the individual productsthat were included in each Order. When you connect to data sources with multiples tables (suchas a relational database) you can use these references to build up your query. In this step, you expand the **Order_Details** table that is related to the Orders table, to combinethe **ProductID**, **UnitPrice**, and **Quantity** columns from **Order_Details** into the **Orders table**.This is a representation of the data in these tables:

The Expand operation combines columns from a related table into a subject table.

When the query runs, rows from the related table (**Order_Details**) are combined into rows from the subject table (**Orders**).

After you expand the Order_Details table, three new columns and additional rows are addedto the Orders table, one for each row in the nested or related table.

1. In the Query View, scroll to the Order_Details column.
2. In the Order_Details column, select the expand icon ( ).
3. In the Expand drop-down:
    a. Select (Select All Columns) to clear all columns.
    b. Select ProductID, UnitPrice, and Quantity.
    c. Click OK.

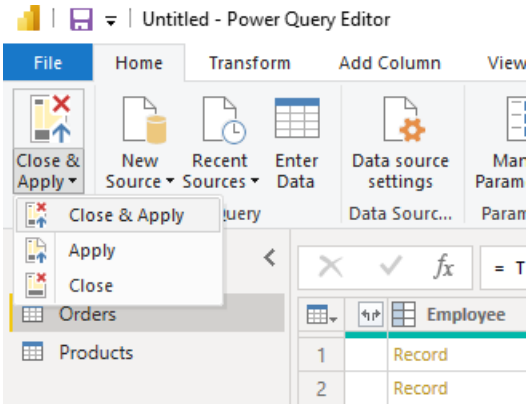


4. Calculate the line total for each Order_Details row

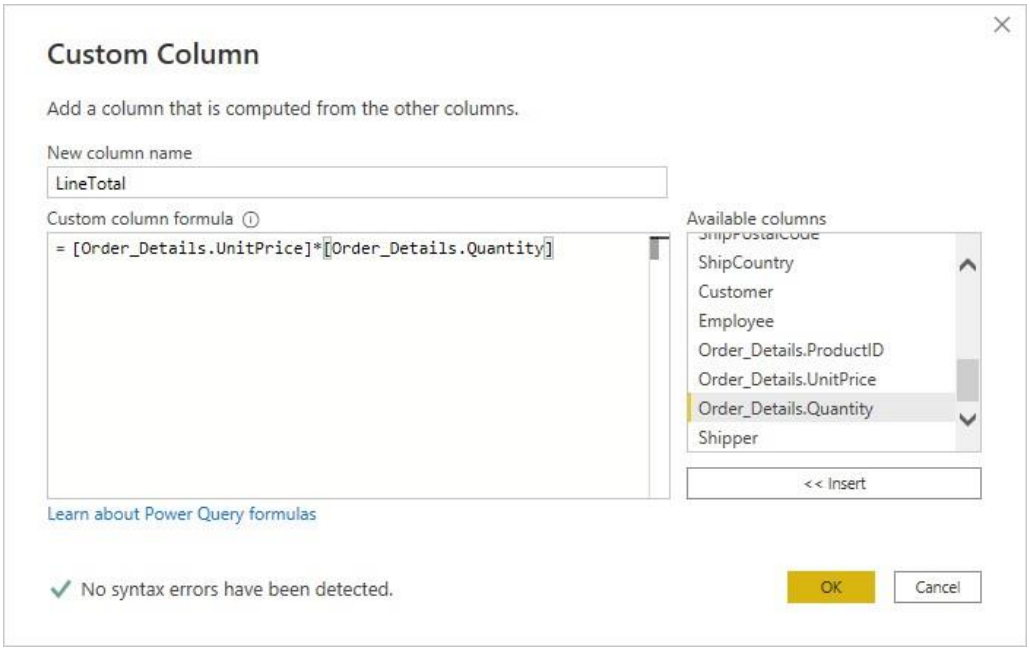
Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order_Details row.

Calculate the line total for each Order_Details row:

1. In the Add Column ribbon tab, click Add Custom Column.



2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter[Order_Details.UnitPrice] * [Order_Details.Quantity].
3. In the New column name textbox, enter LineTotal.
4. Click OK.

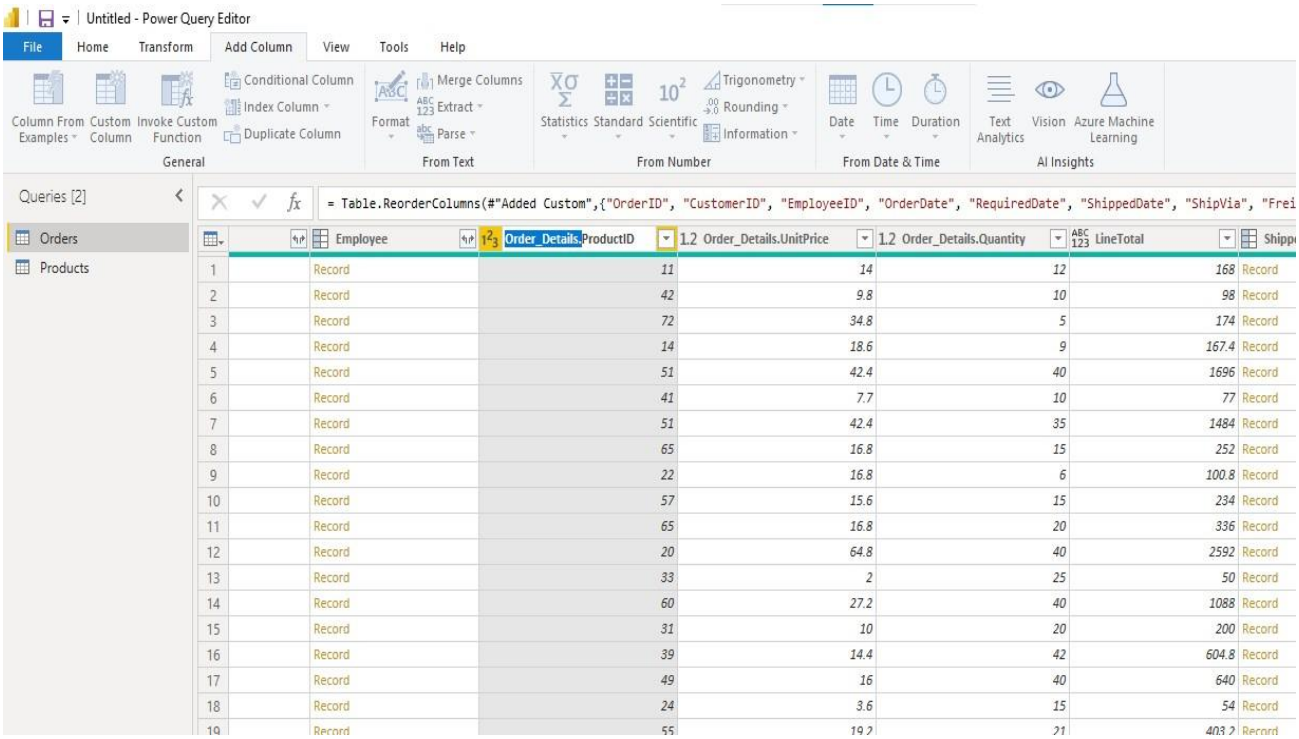


5. Rename and reorder columns in the query

In this step you finish making the model easy to work with when creating reports, byrenaming the final columns and changing their order.

1. In Query Editor, drag the LineTotal column to the left, after ShipCountry.
2. Remove the Order_Details. prefix from the Order_Details.ProductID, Order_Details.UnitPrice and Order_Details.Quantity columns, by double-

clicking on eachcolumn header, and then deleting that text from the column name.



6. Combine the Products and Total Sales queries

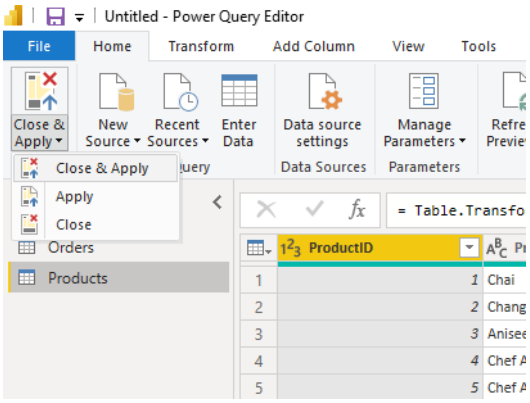
Power BI Desktop does not require you to combine queries to report on them. Instead, you can create Relationships between datasets. These relationships can be created on any columnthat is common to your datasets

we have Orders and Products data that share a common 'ProductID' field, so we need to ensure there's a relationship between them in the model we're using with Power BI Desktop.Simply specify in Power BI Desktop that the columns from each table are related (i.e. columns that have the same values). Power BI Desktop works out the direction and cardinality of the relationship for you. In some cases, it will even detect the relationships automatically.

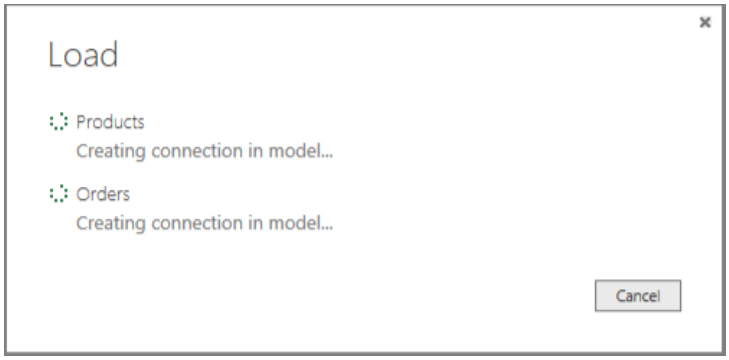
In this task, you confirm that a relationship is established in Power BI Desktop between theProducts and Total Sales queries

Step 1: Confirm the relationship between Products and Total Sales

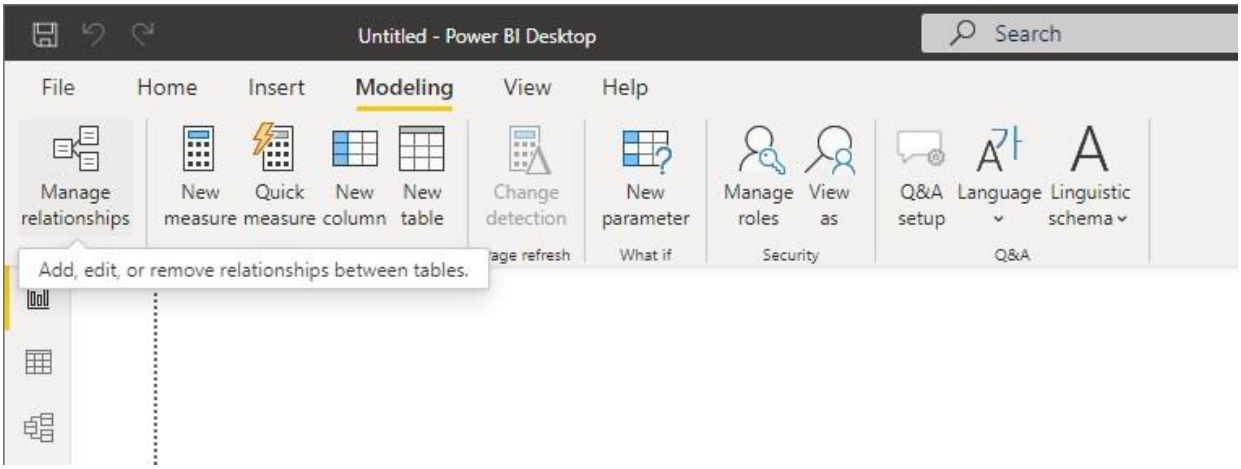
First, we need to load the model that we created in Query Editor into Power BIDesktop. From the Home ribbon of Query Editor, select Close & Load.



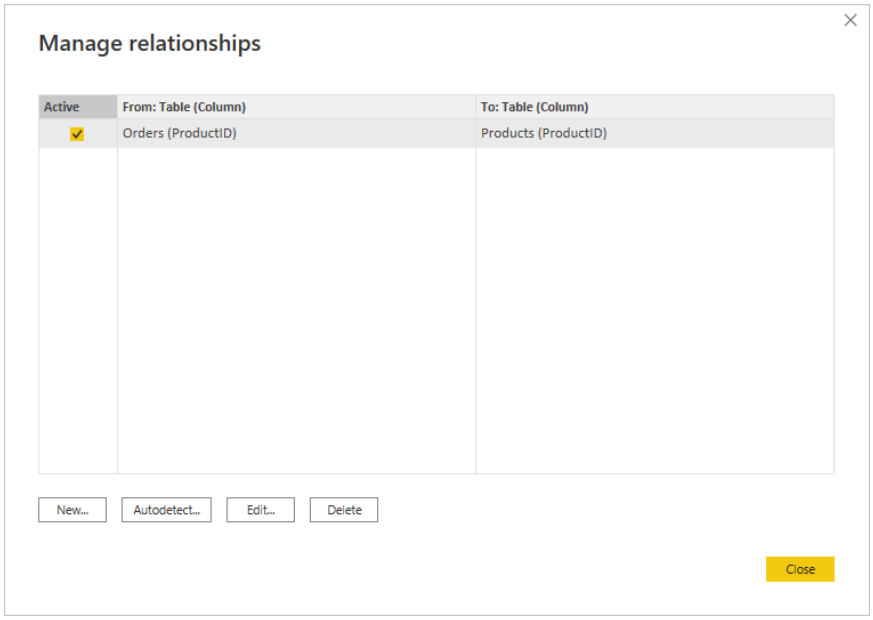
Power BI Desktop loads the data from the two queries.



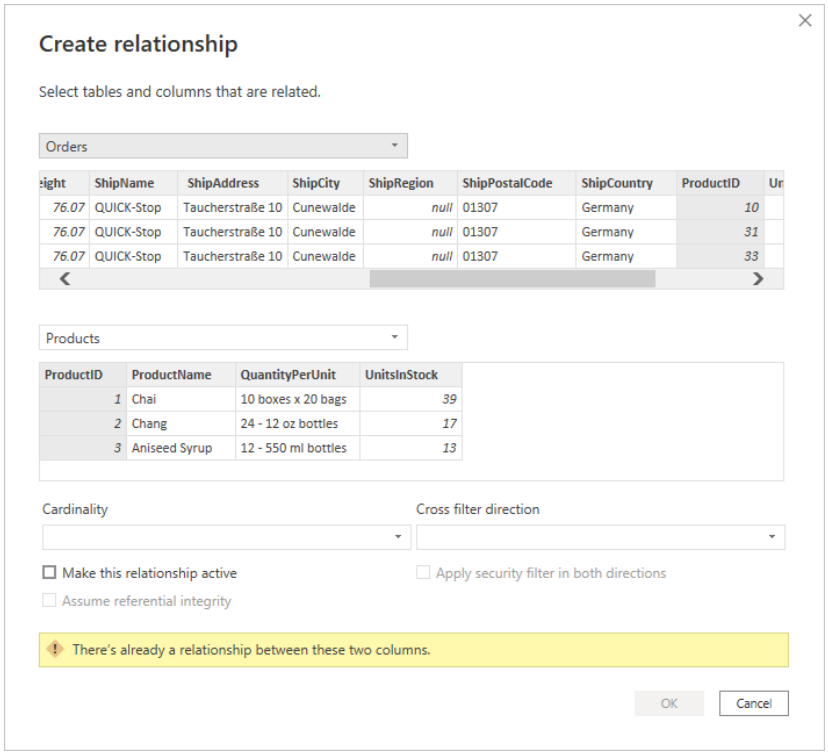
Once the data is loaded, select the Manage Relationships button Home ribbon.



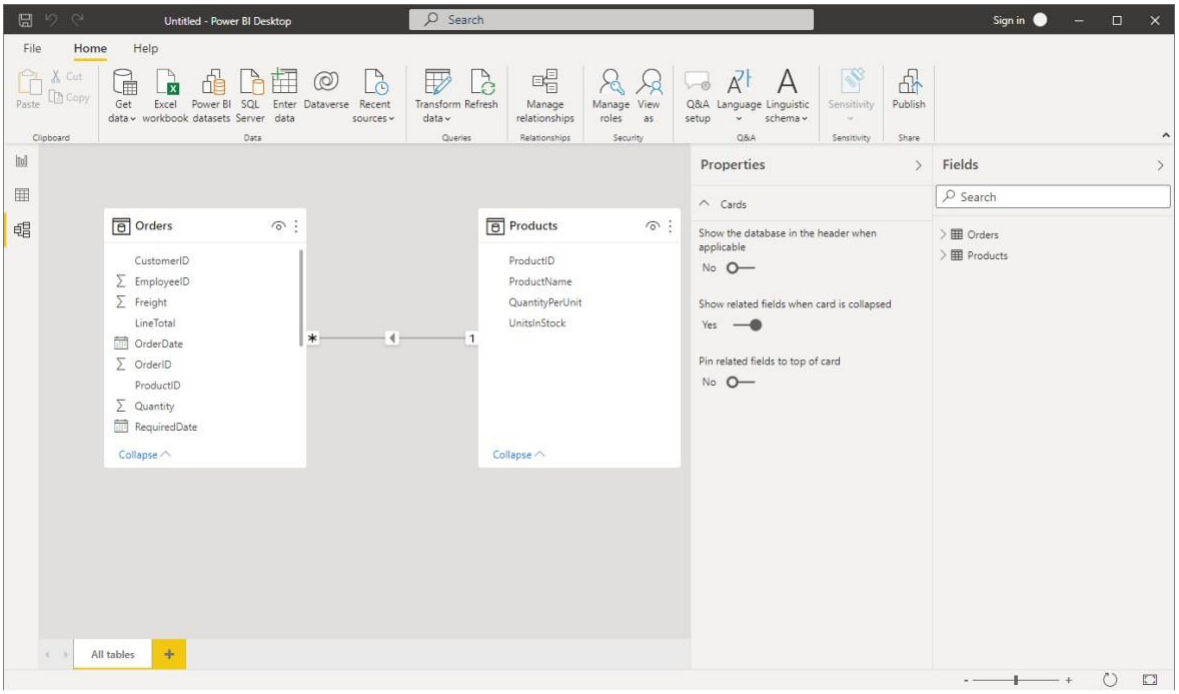
1. Select the New… button



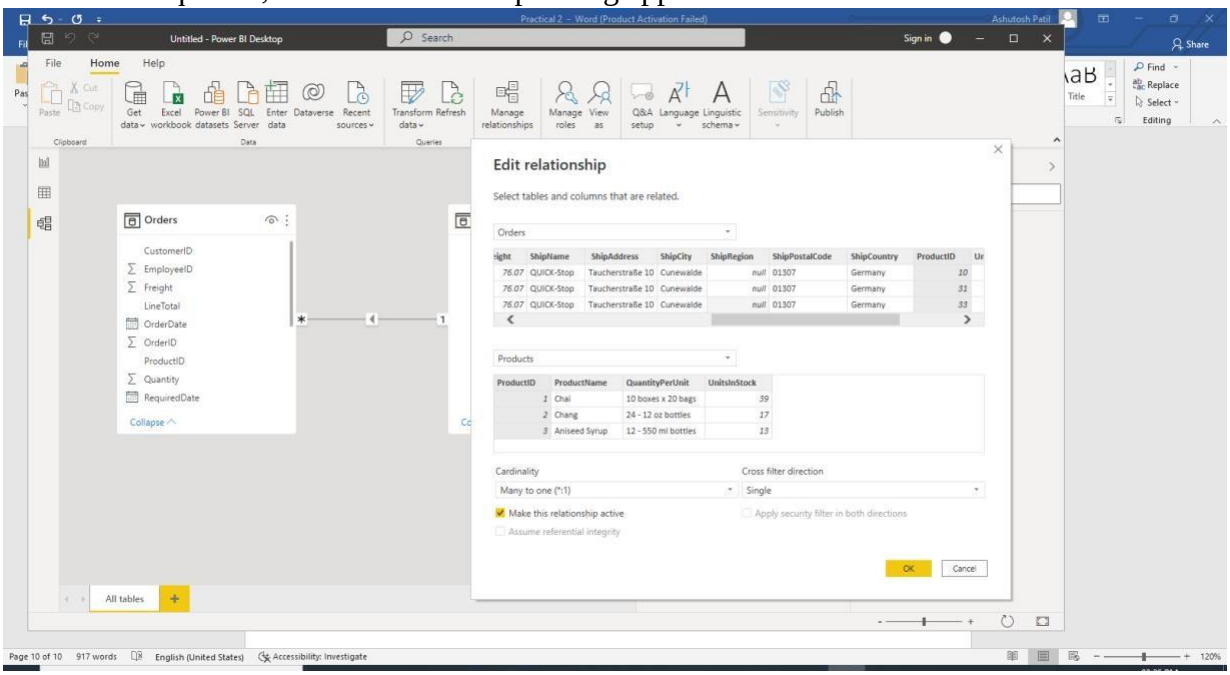
2. When we attempt to create the relationship, we see that one already exists! As shown in theCreate Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.



3. Select Cancel, and then select Relationship view in Power BI Desktop.



4. We see the following, which visualizes the relationship between the queries
5. When you double-click the arrow on the line that connects the to queries, an EditRelationship dialog appears.



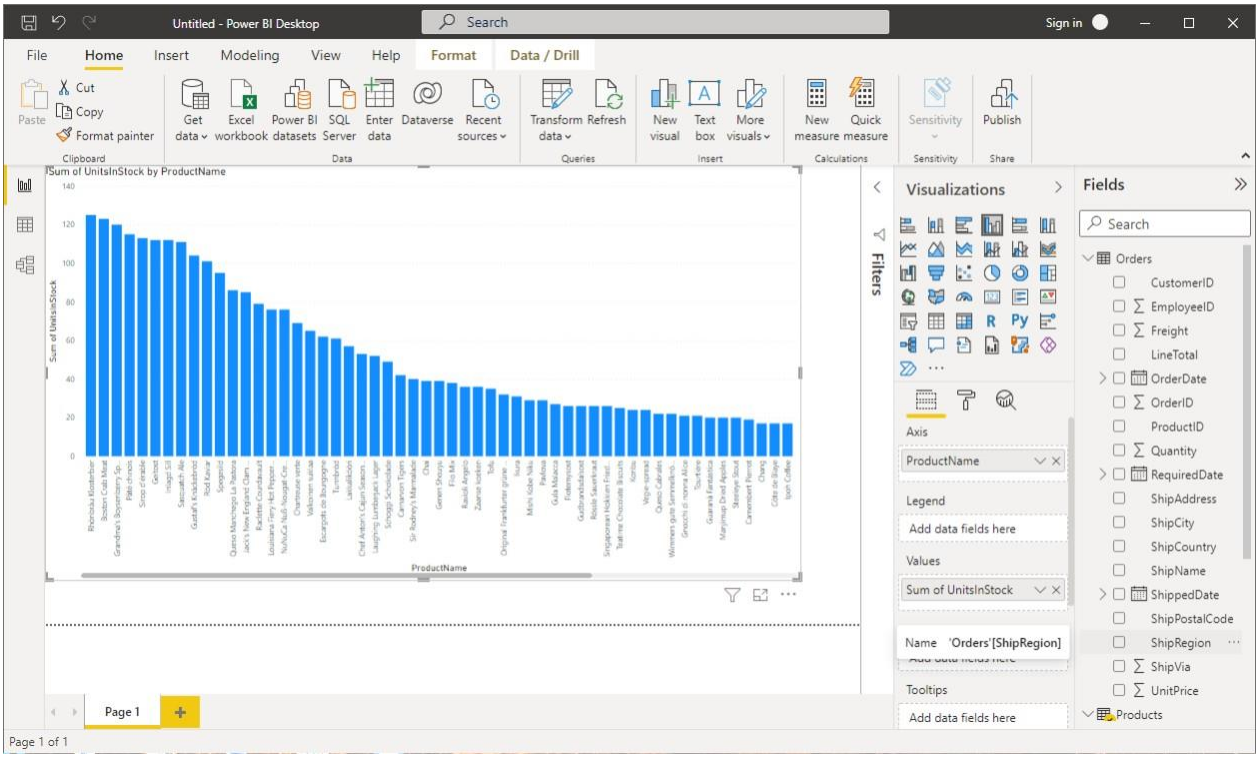
6. No need to make any changes, so we'll just select Cancel toclose the EditRelationship dialog.

Power BI Desktop lets you create a variety of visualizations to gain insights from your data. You can build reports with multiple pages and each page can have multiple visuals. You caninteract with your visualizations to help analyze and understand your data

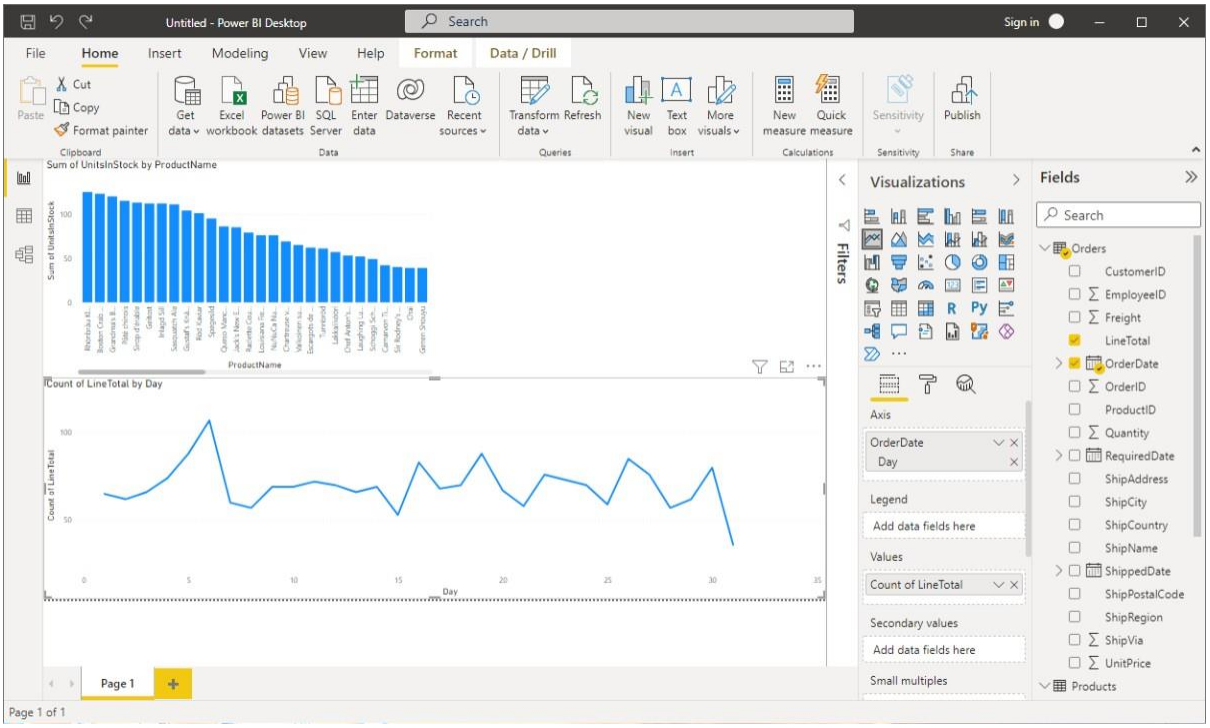
In this task, you create a report based on the data previously loaded. You use the Fields paneto select the columns from which you create the visualizations.

Step 1: Create charts showing Units in Stock by Product and Total Sales by Year

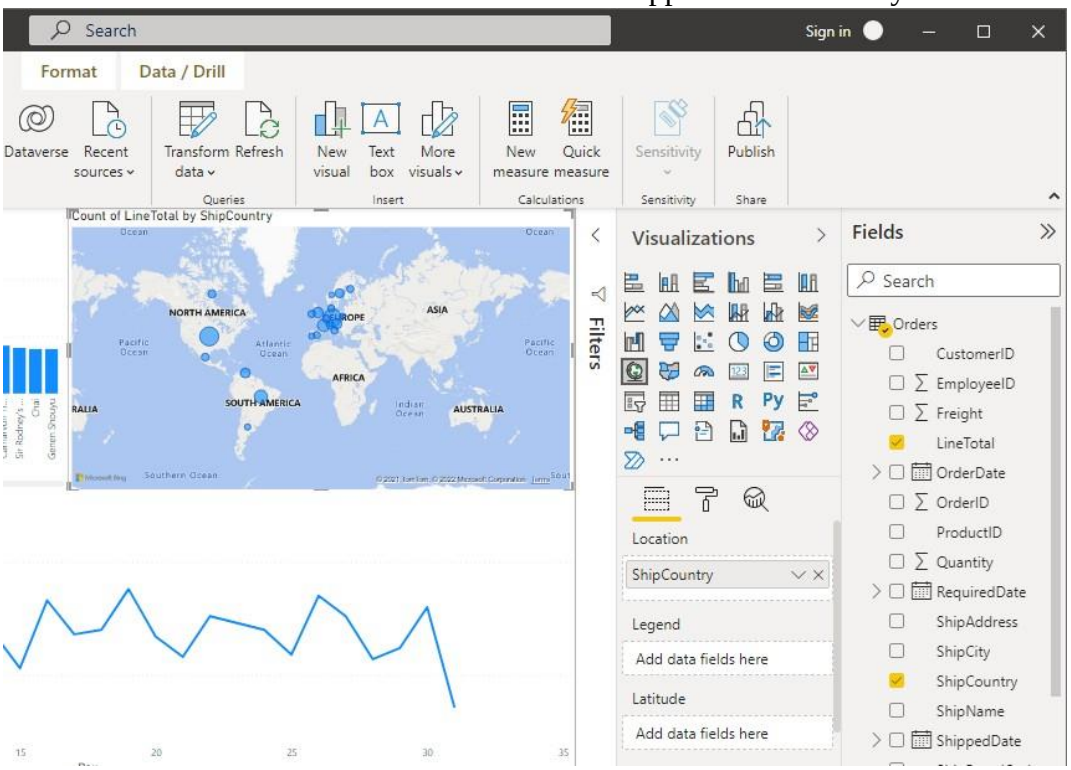
1. Drag UnitsInStock from the Field pane (the Fields pane is along the right of the screen) onto a blank space on the canvas. A Table visualization is created. Next, dragProductName to the Axis box, found in the bottom half of the Visualizations pane. Then we then select Sort By > UnitsInStock using the skittles in the top right corer ofthe visualization.



2. Drag OrderDate to the canvas beneath the first chart, then drag LineTotal (again, fromthe Fields pane) onto the visual, then select Line Chart. The following visualization iscreated.



3. Next, drag ShipCountry to a space on the canvas in the top right. Because you selected a geographic field, a map was created automatically. Now drag LineTotal to the Valuesfield; the circles on the map for each country are now relative in size to the LineTotal for orders shipped to that country.



Step 2: Interact with your report visuals to analyze further

Power BI Desktop lets you interact with visuals that cross-highlight and filter each other touncover further trends.

1. Click on the light blue circle centered in Canada. Note how the other visuals are filtered to show Stock (ShipCountry) and Total Orders (LineTotal) just for Canada.