RESEARCH IN COMPUTING

Table of Contents

Sr. No	Practical No		Name of the Practical	Signature
1)	1	A	Write a program for obtaining descriptive statistics of data.	
2)		B	Import data from different data sources (from Excel, csv, mysql, sql server, oracle to R/Python/Excel)	
3)	2	A	Design a survey form for a given case study, collect the primary data and analyze it	
4)		B	Perform suitable analysis of given secondary data.	
5)	3	A	Perform testing of hypothesis using one sample t-test.	
6)		B	Perform testing of hypothesis using two sample t-test.	
7)		C	Perform testing of hypothesis using paired t-test.	
8)	4	A	Perform testing of hypothesis using chi-squared goodness-of-fit test.	
9)		B	Perform testing of hypothesis using chi-squared Test of Independence	
10)	5		Perform testing of hypothesis using Z-test.	
11)	6	A	Perform testing of hypothesis using one-way ANOVA.	
12)		B	Perform testing of hypothesis using two-way ANOVA.	
13)		C	Perform testing of hypothesis using multivariate ANOVA (MANOVA).	
14)	7	A	Perform the Random sampling for the given data and analyse it.	
15)		B	Perform the Stratified sampling for the given data and analyse it.	
16)	8		Compute different types of correlation.	
17)	9	A	Perform linear regression for prediction.	
18)		B	Perform polynomial regression for prediction.	
19)	10	A	Perform multiple linear regression.	
20)		B	Perform Logistic regression.	

Practical 1:

A. Write a program for obtaining descriptive statistics of data.

```
#################################################################
#Practical 1A: Write a python program on descriptive statistics analysis.
#################################################################
import pandas as pd
#Create a Dictionary of series
d = {'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}
#Create a DataFrame
df = pd.DataFrame(d)
print(df)
print('############  Sum ########## ')
print (df.sum())
print('############  Mean ########## ')
print (df.mean())
print('############  Standard Deviation ########## ')
print (df.std())
print('############  Descriptive Statistics ########## ')
print (df.describe())
```

Output:

```
    Age  Rating
0    25    4.23
1    26    3.24
2    25    3.98
3    23    2.56
4    30    3.20
5    29    4.60
6    23    3.80
7    34    3.78
8    40    2.98
9    30    4.80
10   51    4.10
11   46    3.65
############  Sum ##########
Age        382.00
Rating      44.92
dtype: float64
############  Mean ##########
Age        31.833333
Rating      3.743333
dtype: float64
############  Standard Deviation ##########
Age        9.232682
Rating     0.661628
dtype: float64
############  Descriptive Statistics ##########
             Age     Rating
count  12.000000  12.000000
mean   31.833333   3.743333
std     9.232682   0.661628
min    23.000000   2.560000
25%    25.000000   3.230000
50%    29.500000   3.790000
75%    35.500000   4.132500
max    51.000000   4.800000
```

Using Excel

Go to File Menu → Options → Add-Ins→ Select Analysis ToolPak→ Press OK

Select the data range from the excel worksheet.

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

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:

```
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!! ############################################')
```

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
>>>
= RESTART: C:/VKHCG/03-Hillman/01-Retrieve/Retrieve-IP_DATA_ALL_2_SQLite.py =
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv
Storing : C:/VKHCG/01-Vermeulen/00-RawData/SQLite/vermeulen.db  Table: IP_DATA_ALL
Loading : C:/VKHCG/01-Vermeulen/00-RawData/SQLite/vermeulen.db  Table: IP_DATA_ALL
################
## Data Values
################
      RowIDCSV  RowID    ID  ...  Longitude  First.IP.Number  Last.IP.Number
0            0      0     1  ...   -73.9725        204276480       204276735
1            1      1     2  ...   -73.9725        301984864       301985791
2            2      2     3  ...   -73.9725        404678736       404679039
3            3      3     4  ...   -73.9725        411592704       411592959
4            4      4     5  ...   -73.9725        416784384       416784639
...        ...    ...   ...  ...        ...              ...             ...
3557      3557   3557  3558  ...    11.5392       1591269504      1591269631
3558      3558   3558  3559  ...    11.7500       1558374784      1558374911
3559      3559   3559  3560  ...    11.4667       1480845312      1480845439
3560      3560   3560  3561  ...    11.7434       1480596992      1480597503
3561      3561   3561  3562  ...    11.7434       1558418432      1558418943

[3562 rows x 10 columns]
################
## Data Profile
################
Rows : 3562
Columns : 10
################
### Done!! ##########################################
>>>
Ln: 107  Col: 4
```

MySQL:

Open MySql

Create a database “DataScience”

Create a python file and add the following code:

```
################ Connection With MySQL #######################
importmysql.connector

conn = mysql.connector.connect(host='localhost',
database='DataScience',
user='root',
password='root')
conn.connect
if(conn.is_connected):
print('###### Connection With MySql Established Successfullly ##### ')
else:
print('Not Connected -- Check Connection Properites')
```

```
>>>
 RESTART: C:/Users/User/AppData/Local/Programs/Python/Python37-32/mysqlconnection.py
###### Connection With MySql Established Successfullly #####
>>>
```

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:

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/VKHCG/04-Clark/01-Retrieve/Retrieve-Country-Currency.py ====
                        Country             Currency CurrencyCode
1                   Afghanistan        Afghan afghani          AFN
2    Akrotiri and Dhekelia (UK)         European euro          EUR
3       Aland Islands (Finland)         European euro          EUR
4                       Albania          Albanian lek          ALL
5                       Algeria        Algerian dinar          DZD
..                          ...                   ...          ...
271           Wake Island (USA)  United States dollar          USD
272  Wallis and Futuna (France)             CFP franc          XPF
274                       Yemen           Yemeni rial          YER
276                      Zambia        Zambian kwacha          ZMW
277                    Zimbabwe  United States dollar          USD

[253 rows x 3 columns]
~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~
>>>
Ln: 20  Col: 4
```

Practical 2

Perform analysis of given secondary data.

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.

World population 2010

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

E4 =-1*B4/$D$22*100

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 3:

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:

```
# -*- coding: utf-8 -*-
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

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								

Using Python

```
importnumpy as np
fromscipy import stats
fromnumpy.random import randn
N = 20
#a = [35,40,12,15,21,14,46,10,28,48,16,30, 32,48,31,22,12,39,19,25]
#b = [2,27,31,38,1,19,1,34,3,1,2,1,3,1,2,1,3,29,37,2]
a = 5 * randn(100) + 50
b = 5 * randn(100) + 51
var_a = a.var(ddof=1)
var_b = b.var(ddof=1)
s = np.sqrt((var_a + var_b)/2)
t = (a.mean() - b.mean())/(s*np.sqrt(2/N))
df = 2*N - 2
#p-value after comparison with the t
p = 1 - stats.t.cdf(t,df=df)
print("t = " + str(t))
print("p = " + str(2*p))
if  t> p :
print('Mean of two distribution are differnt and significant')
else:
print('Mean of two distribution are same and not significant')
```

Output:

```
In [9]: runfile('E:/Research In Computing/Programs/
Practical_04/Program_4B.py', wdir='E:/Research In
Computing/Programs/Practical_04')
t = -1.051463820987354
p = 1.700313560478936
Mean of two distribution are same and not significant

In [10]: runfile('E:/Research In Computing/Programs/
Practical_04/Program_4B.py', wdir='E:/Research In
Computing/Programs/Practical_04')
t = 0.46409515960993775
p = 0.6452274090296801
Mean of two distribution are differnt and significant
```

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:

- **H_0** - The mean difference between sample 1 and sample 2 is equal to 0.
- **H_0** - The mean difference between sample 1 and sample 2 is not equal to 0

Program Code:

```
# -*- coding: utf-8 -*-
Created on Mon Dec 16 19:49:23 2019
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())

#First let’s check for any significant outliers in
#each of the variables.
df[['bp_before', 'bp_after']].plot(kind='box')
# This saves the plot as a png file
plt.savefig('boxplot_outliers.png')

# make a histogram to differences between the two scores.
df['bp_difference'] = df['bp_before'] - df['bp_after']

df['bp_difference'].plot(kind='hist', title= 'Blood Pressure Difference Histogram')
#Again, this saves the plot as a png file
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 4

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 $\sum \frac{(O_i - E_i)^2}{Ei}$ 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

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

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
58	89	M'sila	m	A
59	96	Heliopolis	m	A
60	97	Berrouaghia	m	A
61	98	Sougueur	m	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 $\sum \frac{(O_i - E_i)^2}{Ei}$

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**					

Using Python

```
importnumpy as np
import pandas as pd
importscipy.stats as stats
np.random.seed(10)
stud_grade = np.random.choice(a=["O","A","B","C","D"],
                    p=[0.20, 0.20 ,0.20, 0.20, 0.20], size=100)
stud_gen = np.random.choice(a=["Male","Female"],  p=[0.5, 0.5],  size=100)
mscpart1 = pd.DataFrame({"Grades":stud_grade, "Gender":stud_gen})
print(mscpart1)
stud_tab = pd.crosstab(mscpart1.Grades, mscpart1.Gender, margins=True)
```

```
stud_tab.columns = ["Male", "Female",  "row_totals"]
stud_tab.index = ["O", "A", "B", "C", "D", "col_totals"]
observed = stud_tab.iloc[0:5, 0:2 ]
print(observed)
expected =  np.outer(stud_tab["row_totals"][0:5],
stud_tab.loc["col_totals"][0:2]) / 100
print(expected)
chi_squared_stat = (((observed-expected)**2)/expected).sum().sum()
print('Calculated : ',chi_squared_stat)

crit = stats.chi2.ppf(q=0.95, df=4)
print('Table Value : ',crit)

ifchi_squared_stat>= crit:
print('H0 is Accepted ')
else:
print('H0 is Rejected ')
```

Output :

```
In [1]: runfile('E:/Research In Computing/Programs/
Practical_03/ChiSquaer.py', wdir='E:/Research In
Computing/Programs/Practical_03')
   Grades  Gender
0       C  Female
1       O  Female
2       C    Male
3       C    Male
4       B  Female
..    ...     ...
95      B    Male
96      D  Female
97      B  Female
98      A    Male
99      B    Male

[100 rows x 2 columns]
   Male  Female
O    11      12
A     9      13
B     7      11
C    10       8
D    12       7
[[11.27 11.73]
 [10.78 11.22]
 [ 8.82  9.18]
 [ 8.82  9.18]
 [ 9.31  9.69]]
Calculated :  3.158915138993211
Table Value :  9.487729036781154
H0 is Rejected
```

Practical 5

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

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 16 20:42:17 2019
@author: MyHome
"""
```

```
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")
```

```
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 6

A. Perform testing of hypothesis using One-way ANOVA.

ANOVA ASSUMPTIONS

- The dependent variable (SAT scores in our example) should be continuous.
- The independent variables (districts in our example) should be two or more categorical groups.
- There must be different participants in each group with no participant being in more than one group. In our case, each school cannot be in more than one district.
- The dependent variable should be approximately normally distributed for each category.
- Variances of each group are approximately equal.

From our data exploration, we can see that the average SAT scores are quite different for each district. Since we have five different groups, we cannot use the t-test, use the 1-way ANOVA test anyway just to understand the concepts.

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

$$\mu_1 = \mu_2 = \mu_3 = \mu_4 = \mu_5$$

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

If there is at least one group with a significant difference with another group, the null hypothesis will be rejected.

```
import pandas as pd
importnumpy as np
importmatplotlib.pyplot as plt
importseaborn as sns
fromscipy import stats

data = pd.read_csv("scores.csv")
data.head()
data['Borough'].value_counts()
```

############### There is no total score column, have to create it. ########In addition, find the mean score of the each district across all schools.

```
data['total_score'] = data['Average Score (SAT Reading)'] +  \
data['Average Score (SAT Math)']    +  \
data['Average Score (SAT Writing)']
data = data[['Borough', 'total_score']].dropna()
x = ['Brooklyn', 'Bronx', 'Manhattan', 'Queens', 'Staten Island']
district_dict = {}

#Assigns each test score series to a dictionary key
for district in x:
```

```
district_dict[district] = data[data['Borough'] == district]['total_score']
y = []
yerror = []
#Assigns the mean score and 95% confidence limit to each district
for district in x:
y.append(district_dict[district].mean())
yerror.append(1.96*district_dict[district].std()/np.sqrt(district_dict[district].shap
e[0]))
print(district + '_std : {}'.format(district_dict[district].std()))

sns.set(font_scale=1.8)
fig = plt.figure(figsize=(10,5))
ax = sns.barplot(x, y, yerr=yerror)
ax.set_ylabel('Average Total SAT Score')
plt.show()
```

###################### Perform 1-way ANOVA

```
print(stats.f_oneway(
district_dict['Brooklyn'], district_dict['Bronx'], \
district_dict['Manhattan'], district_dict['Queens'], \
district_dict['Staten Island']
))
districts = ['Brooklyn', 'Bronx', 'Manhattan', 'Queens', 'Staten Island']
ss_b = 0
for d in districts:
ss_b += district_dict[d].shape[0] * \
np.sum((district_dict[d].mean() - data['total_score'].mean())**2)
ss_w = 0
for d in districts:
ss_w += np.sum((district_dict[d] - district_dict[d].mean())**2)
msb = ss_b/4
msw = ss_w/(len(data)-5)
f=msb/msw
print('F_statistic: {}'.format(f))

ss_t = np.sum((data['total_score']-data['total_score'].mean())**2)
eta_squared = ss_b/ss_t
print('eta_squared: {}'.format(eta_squared))
```

Output:

```
In [37]: runfile('E:/Research In Computing/Programs/Practical_05/Annova.py', wdir='E:/Research In
Computing/Programs/Practical_05')
Brooklyn_std : 154.8684270520867
Bronx_std : 150.39390071890668
Manhattan_std : 230.2941395363782
Queens_std : 195.25289850192115
Staten Island_std : 222.30359621222706
```

```
F_onewayResult(statistic=12.733085029201668, pvalue=1.0161974965566023e-09)
F_statistic: 12.733085029201687
eta_squared: 0.12099887621529214
```

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

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.

Program Code:

```
import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.anova import anova_lm
from statsmodels.graphics.factorplots import interaction_plot
import matplotlib.pyplot as plt
from scipy import stats

def eta_squared(aov):
    aov['eta_sq'] = 'NaN'
    aov['eta_sq'] = aov[:-1]['sum_sq']/sum(aov['sum_sq'])
    return aov

def omega_squared(aov):
    mse = aov['sum_sq'][-1]/aov['df'][-1]
    aov['omega_sq'] = 'NaN'
    aov['omega_sq'] = (aov[:-1]['sum_sq']-(aov[:-1]['df']*mse))/(sum(aov['sum_sq'])+mse)
    return aov

datafile = "ToothGrowth.csv"
data = pd.read_csv(datafile)
fig = interaction_plot(data.dose, data.supp, data.len,
            colors=['red','blue'], markers=['D','^'], ms=10)
N = len(data.len)
df_a = len(data.supp.unique()) - 1
df_b = len(data.dose.unique()) - 1
df_axb = df_a*df_b
df_w = N - (len(data.supp.unique())*len(data.dose.unique()))
grand_mean = data['len'].mean()
#Sum of Squares A – supp
ssq_a = sum([(data[data.supp ==l].len.mean()-grand_mean)**2 for l in data.supp])
#Sum of Squares B – supp
ssq_b = sum([(data[data.dose ==l].len.mean()-grand_mean)**2 for l in data.dose])
#Sum of Squares Total
ssq_t = sum((data.len - grand_mean)**2)
vc = data[data.supp == 'VC']
oj = data[data.supp == 'OJ']
vc_dose_means = [vc[vc.dose == d].len.mean() for d in vc.dose]
oj_dose_means = [oj[oj.dose == d].len.mean() for d in oj.dose]
ssq_w = sum((oj.len - oj_dose_means)**2) +sum((vc.len - vc_dose_means)**2)
ssq_axb = ssq_t-ssq_a-ssq_b-ssq_w
ms_a = ssq_a/df_a        #Mean Square A
ms_b = ssq_b/df_b        #Mean Square B
ms_axb = ssq_axb/df_axb        #Mean Square AXB
ms_w = ssq_w/df_w
f_a = ms_a/ms_w
f_b = ms_b/ms_w
```

```
f_axb = ms_axb/ms_w
p_a = stats.f.sf(f_a, df_a, df_w)
p_b = stats.f.sf(f_b, df_b, df_w)
p_axb = stats.f.sf(f_axb, df_axb, df_w)
results = {'sum_sq':[ssq_a, ssq_b, ssq_axb, ssq_w],
           'df':[df_a, df_b, df_axb, df_w],
           'F':[f_a, f_b, f_axb, 'NaN'],
            'PR(>F)':[p_a, p_b, p_axb, 'NaN']}
columns=['sum_sq', 'df', 'F', 'PR(>F)']

aov_table1 = pd.DataFrame(results, columns=columns,
                          index=['supp', 'dose',
                          'supp:dose', 'Residual'])
formula = 'len ~ C(supp) + C(dose) + C(supp):C(dose)'
model = ols(formula, data).fit()
aov_table = anova_lm(model, typ=2)
eta_squared(aov_table)
omega_squared(aov_table)
print(aov_table.round(4))
res = model.resid
fig = sm.qqplot(res, line='s')
plt.show()
```

Output:

```
In [40]: runfile('K:/Research In Computing/Practical Material/Programs/
Practical_06/Annova_2_Way.py', wdir='K:/Research In Computing/Practical
Material/Programs/Practical_06')
                   sum_sq    df       F  PR(>F)  eta_sq  omega_sq
C(supp)           205.3500   1.0  15.572  0.0002  0.0595    0.0555
C(dose)          2426.4343   2.0  92.000  0.0000  0.7029    0.6926
C(supp):C(dose)   108.3190   2.0   4.107  0.0219  0.0314    0.0236
Residual          712.1060  54.0     NaN     NaN     NaN       NaN
```

Using Excel:

Go to Data tab → Data Analysis

Input Range - $A$1:$C$61
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.

B. Perform testing of hypothesis using MANOVA.

Code:

```
import pandas as pd
fromstatsmodels.multivariate.manova import MANOVA
df = pd.read_csv('iris.csv', index_col=0)
df.columns = df.columns.str.replace(".", "_")
df.head()
print('~~~~~~~~ Data Set ~~~~~~~~')
print(df)
maov = MANOVA.from_formula('Sepal_Length + Sepal_Width + \
Petal_Length + Petal_Width  ~ Species', data=df)
print('~~~~~~~~ MANOVA Test Result ~~~~~~~~')
print(maov.mv_test())
```

Output:

```
In [42]: runfile('E:/Research In Computing/Programs/Practical_10/Manova_Test.py', wdir='E:/Research
In Computing/Programs/Practical_10')
~~~~~~~~ Data Set ~~~~~~~~
     Sepal_Length  Sepal_Width  Petal_Length  Petal_Width    Species
1             5.1          3.5           1.4          0.2     setosa
2             4.9          3.0           1.4          0.2     setosa
3             4.7          3.2           1.3          0.2     setosa
4             4.6          3.1           1.5          0.2     setosa
5             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
146           6.7          3.0           5.2          2.3  virginica
147           6.3          2.5           5.0          1.9  virginica
148           6.5          3.0           5.2          2.0  virginica
149           6.2          3.4           5.4          2.3  virginica
150           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
~~~~~~~~ MANOVA Test Result ~~~~~~~~
                    Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0170 4.0000 144.0000 2086.7720 0.0000
         Pillai's trace  0.9830 4.0000 144.0000 2086.7720 0.0000
 Hotelling-Lawley trace 57.9659 4.0000 144.0000 2086.7720 0.0000
    Roy's greatest root 57.9659 4.0000 144.0000 2086.7720 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
        Species          Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0234 8.0000 288.0000  199.1453 0.0000
         Pillai's trace  1.1919 8.0000 290.0000   53.4665 0.0000
 Hotelling-Lawley trace 32.4773 8.0000 203.4024  582.1970 0.0000
    Roy's greatest root 32.1919 4.0000 145.0000 1166.9574 0.0000
================================================================
```

Practical 7

A. Perform the Random sampling for the given data and analyse 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

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

B. Perform the Stratified sampling for the given data and analyse it.

we are to carry out a **hypothetical** housing quality survey across Lagos state, Nigeria. And we looking at a total of 5000 houses (hypothetically). We don't just go to one local government and select 5000 houses, rather we ensure that the 5000 houses are a representative of the whole 20 local government areas Lagos state is comprised of. This is called stratified sampling. The population is divided into homogenous strata and the right number of instances is sampled from each stratum to guarantee that the test-set (which in this case is the 5000 houses) is a representative of the overall population. If we used random sampling, there would be a significant chance of having bias in the survey results.

Program Code:

```
import pandas as pd
importnumpy as np

importmatplotlib
importmatplotlib.pyplot as plt

plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12

importseaborn as sns
color = sns.color_palette()
sns.set_style('darkgrid')

importsklearn
fromsklearn.model_selection import train_test_split

housing =pd.read_csv('housing.csv')
print(housing.head())
print(housing.info())

 #creating a heatmap of the attributes in the dataset
correlation_matrix = housing.corr()
plt.subplots(figsize=(8,6))
sns.heatmap(correlation_matrix, center=0, annot=True, linewidths=.3)


corr =housing.corr()
print(corr['median_house_value'].sort_values(ascending=False))

sns.distplot(housing.median_income)
plt.show()
```

Output:

```
In [28]: runfile('J:/Research In Computing/Practical Material/Programs/Practical_05/
Stratified_Sample.py', wdir='J:/Research In Computing/Practical Material/Programs/Practical_05')
   longitude  latitude  ...  median_house_value  ocean_proximity
0    -122.23     37.88  ...            452600.0         NEAR BAY
1    -122.22     37.86  ...            358500.0         NEAR BAY
2    -122.24     37.85  ...            352100.0         NEAR BAY
3    -122.25     37.85  ...            341300.0         NEAR BAY
4    -122.25     37.85  ...            342200.0         NEAR BAY

[5 rows x 10 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

There's a ton of information we can mine from the heatmap above, a couple of strongly positively correlated features and a couple of negatively correlated features. Take a look at the small bright box right in the middle of the heatmap from total_rooms on the left 'y-axis' till households and note how bright the box is as well as the highly positively correlated attributes,

also note that median_income is the most correlated feature to the target which is median_house_value.

From the image above, we can see that most median incomes are clustered between $20,000 and $50,000 with some outliers going far beyond $60,000 making the distribution skew to the right.

Practical 8

Write a program for computing different correlation.

Positive Correlation:

Code:

```
importnumpy as np
importmatplotlib.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:

Negative Correlation:

```
importnumpy as np
importmatplotlib.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:

```
In [24]: runfile('E:/Research In Computing/Programs/
Practical_06/NegativeCorrelation.py', wdir='E:/Research
In Computing/Programs/Practical_06')
```

No/Weak Correlation:

```
importnumpy as np
importmatplotlib.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:

```
In [25]: runfile('E:/Research In Computing/Programs/
Practical_06/No_or_Weak_Correlation.py', wdir='E:/
Research In Computing/Programs/Practical_06')
```

Practical 9:

A. Write a program to Perform linear regression for prediction.

```
# -*- coding: utf-8 -*-
import Quandl, math
import numpy as np
import pandas as pd
from sklearn import preprocessing, cross_validation, svm
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from matplotlib import style
import datetime
style.use('ggplot')
df = Quandl.get("WIKI/GOOGL")
df = df[['Adj. Open',  'Adj. High',  'Adj. Low',  'Adj. Close', 'Adj. Volume']]
df['HL_PCT'] = (df['Adj. High'] - df['Adj. Low']) / df['Adj. Close'] * 100.0
df['PCT_change'] = (df['Adj. Close'] - df['Adj. Open']) / df['Adj. Open'] * 100.0

df = df[['Adj. Close', 'HL_PCT', 'PCT_change', 'Adj. Volume']]
forecast_col = 'Adj. Close'
df.fillna(value=-99999, inplace=True)
forecast_out = int(math.ceil(0.01 * len(df)))
df['label'] = df[forecast_col].shift(-forecast_out)

X = np.array(df.drop(['label'], 1))
X = preprocessing.scale(X)
X_lately = X[-forecast_out:]
X = X[:-forecast_out]

df.dropna(inplace=True)
y = np.array(df['label'])
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y, test_size=0.2)
clf = LinearRegression(n_jobs=-1)
clf.fit(X_train, y_train)
confidence = clf.score(X_test, y_test)

forecast_set = clf.predict(X_lately)
df['Forecast'] = np.nan

last_date = df.iloc[-1].name
last_unix = last_date.timestamp()
one_day = 86400
next_unix = last_unix + one_day

for i in forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += 86400
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

df['Adj. Close'].plot()
```

```
df['Forecast'].plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

Output:

B. Perform polynomial regression for prediction.

```
importnumpy as np
importmatplotlib.pyplot as plt

defestimate_coef(x, y):
        # number of observations/points
        n = np.size(x)

        # mean of x and y vector
        m_x, m_y = np.mean(x), np.mean(y)

        # calculating cross-deviation and deviation about x
        SS_xy = np.sum(y*x) - n*m_y*m_x
        SS_xx = np.sum(x*x) - n*m_x*m_x

        # calculating regression coefficients
        b_1 = SS_xy / SS_xx
        b_0 = m_y - b_1*m_x

        return(b_0, b_1)

defplot_regression_line(x, y, b):
        # plotting the actual points as scatter plot
        plt.scatter(x, y, color = "m",
                        marker = "o", s = 30)

        # predicted response vector
        y_pred = b[0] + b[1]*x
```

```
        # plotting the regression line
        plt.plot(x, y_pred, color = "g")

        # putting labels
        plt.xlabel('x')
        plt.ylabel('y')

        # function to show plot
        plt.show()

def main():
        # observations
        x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
        y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

        # estimating coefficients
        b = estimate_coef(x, y)
        print("Estimated coefficients:\nb_0 = {}  b_1 = {}".format(b[0], b[1]))

        # plotting regression line
        plot_regression_line(x, y, b)

if __name__ == "__main__":
        main()
```

Output:

```
In [22]: runfile('E:/Research In Computing/Programs/
Practical_07/Practical_7B.py', wdir='E:/Research In
Computing/Programs/Practical_07')
Estimated coefficients:
b_0 = 1.2363636363636363  b_1 = 1.1696969696969697
```

Practical 10:

A. Write a program for multiple linear regression analysis.

Step #1: Data Pre Processing

a) Importing The Libraries.
b) Importing the Data Set.
c) Encoding the Categorical Data.
d) Avoiding the Dummy Variable Trap.
e) Splitting the Data set into Training Set and Test Set.

Step #2: Fitting Multiple Linear Regression to the Training set

Step #3: Predicting the Test set results.

```
importnumpy as np
importmatplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
importmatplotlib.pyplot as plt
defgenerate_dataset(n):
	x = []
	y = []
	random_x1 = np.random.rand()
	random_x2 = np.random.rand()
	fori in range(n):
		x1 = i
		x2 = i/2 + np.random.rand()*n
		x.append([1, x1, x2])
		y.append(random_x1 * x1 + random_x2 * x2 + 1)
	returnnp.array(x), np.array(y)
x, y = generate_dataset(200)
mpl.rcParams['legend.fontsize'] = 12
fig = plt.figure()
ax = fig.gca(projection ='3d')
ax.scatter(x[:, 1], x[:, 2], y, label ='y', s = 5)
ax.legend()
ax.view_init(45, 0)
plt.show()
defmse(coef, x, y):
	returnnp.mean((np.dot(x, coef) - y)**2)/2
def gradients(coef, x, y):
	returnnp.mean(x.transpose()*(np.dot(x, coef) - y), axis = 1)
defmultilinear_regression(coef, x, y, lr, b1 = 0.9, b2 = 0.999, epsilon = 1e-8):
	prev_error = 0
	m_coef = np.zeros(coef.shape)
	v_coef = np.zeros(coef.shape)
	moment_m_coef = np.zeros(coef.shape)
	moment_v_coef = np.zeros(coef.shape)
	t = 0
	while True:
		error = mse(coef, x, y)
		if abs(error - prev_error) <= epsilon:
			break
		prev_error = error
```

```
            grad = gradients(coef, x, y)
            t += 1
            m_coef = b1 * m_coef + (1-b1)*grad
            v_coef = b2 * v_coef + (1-b2)*grad**2
            moment_m_coef = m_coef / (1-b1**t)
            moment_v_coef = v_coef / (1-b2**t)

            delta = ((lr / moment_v_coef**0.5 + 1e-8) *
                        (b1 * moment_m_coef + (1-b1)*grad/(1-b1**t)))
            coef = np.subtract(coef, delta)
        returncoef
coef = np.array([0, 0, 0])
c = multilinear_regression(coef, x, y, 1e-1)
fig = plt.figure()
ax = fig.gca(projection ='3d')
ax.scatter(x[:, 1], x[:, 2], y, label ='y',
                        s = 5, color ="dodgerblue")
ax.scatter(x[:, 1], x[:, 2], c[0] + c[1]*x[:, 1] + c[2]*x[:, 2],
                        label ='regression', s = 5, color ="orange")
ax.view_init(45, 0)
ax.legend()
plt.show()
```

Output:

B. Perform logistic regression analysis.

Logistic regression is a classification method built on the same concept as linear regression. With linear regression, we take linear combination of explanatory variables plus an intercept term to arrive at a prediction.

In this example we will use a logistic regression model to predict survival.

Program Code:

```
import os
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import scipy.stats as stats
from sklearn import linear_model
from sklearn import preprocessing
from sklearn import metrics

matplotlib.style.use('ggplot')
plt.figure(figsize=(9,9))

def sigmoid(t):                    # Define the sigmoid function
   return (1/(1 + np.e**(-t)))

plot_range = np.arange(-6, 6, 0.1)

y_values = sigmoid(plot_range)

# Plot curve
plt.plot(plot_range,   # X-axis range
        y_values,        # Predicted values
        color="red")
titanic_train = pd.read_csv("titanic_train.csv")    # Read the data
char_cabin = titanic_train["Cabin"].astype(str)     # Convert cabin to str
new_Cabin = np.array([cabin[0] for cabin in char_cabin]) # Take first letter

titanic_train["Cabin"] = pd.Categorical(new_Cabin)  # Save the new cabin var

# Impute median Age for NA Age values
new_age_var = np.where(titanic_train["Age"].isnull(), # Logical check
                    28,                    # Value if check is true
                    titanic_train["Age"])     # Value if check is false

titanic_train["Age"] = new_age_var

label_encoder = preprocessing.LabelEncoder()

# Convert Sex variable to numeric
encoded_sex = label_encoder.fit_transform(titanic_train["Sex"])

# Initialize logistic regression model
log_model = linear_model.LogisticRegression()
```

```
# Train the model
log_model.fit(X = pd.DataFrame(encoded_sex),
              y = titanic_train["Survived"])

# Check trained model intercept
print(log_model.intercept_)

# Check trained model coefficients
print(log_model.coef_)

# Make predictions
preds = log_model.predict_proba(X= pd.DataFrame(encoded_sex))
preds = pd.DataFrame(preds)
preds.columns = ["Death_prob", "Survival_prob"]

# Generate table of predictions vs Sex
pd.crosstab(titanic_train["Sex"], preds.ix[:, "Survival_prob"])

# Convert more variables to numeric
encoded_class = label_encoder.fit_transform(titanic_train["Pclass"])
encoded_cabin = label_encoder.fit_transform(titanic_train["Cabin"])

train_features = pd.DataFrame([encoded_class,
                               encoded_cabin,
                               encoded_sex,
                               titanic_train["Age"]]).T

# Initialize logistic regression model
log_model = linear_model.LogisticRegression()

# Train the model
log_model.fit(X = train_features ,
              y = titanic_train["Survived"])

# Check trained model intercept
print(log_model.intercept_)

# Check trained model coefficients
print(log_model.coef_)

# Make predictions
preds = log_model.predict(X= train_features)

# Generate table of predictions vs actual
pd.crosstab(preds,titanic_train["Survived"])

log_model.score(X = train_features ,
                y = titanic_train["Survived"])
```

```
metrics.confusion_matrix(y_true=titanic_train["Survived"],  # True labels
                         y_pred=preds) # Predicted labels

# View summary of common classification metrics
print(metrics.classification_report(y_true=titanic_train["Survived"],
                            y_pred=preds) )

# Read and prepare test data
titanic_test = pd.read_csv("titanic_test.csv")    # Read the data

char_cabin = titanic_test["Cabin"].astype(str)     # Convert cabin to str

new_Cabin = np.array([cabin[0] for cabin in char_cabin]) # Take first letter

titanic_test["Cabin"] = pd.Categorical(new_Cabin)  # Save the new cabin var

# Impute median Age for NA Age values
new_age_var = np.where(titanic_test["Age"].isnull(), # Logical check
                       28,                    # Value if check is true
                       titanic_test["Age"])      # Value if check is false

titanic_test["Age"] = new_age_var

# Convert test variables to match model features
encoded_sex = label_encoder.fit_transform(titanic_test["Sex"])
encoded_class = label_encoder.fit_transform(titanic_test["Pclass"])
encoded_cabin = label_encoder.fit_transform(titanic_test["Cabin"])

test_features = pd.DataFrame([encoded_class,
                            encoded_cabin,encoded_sex,titanic_test["Age"]]).T

# Make test set predictions
test_preds = log_model.predict(X=test_features)

# Create a submission for Kaggle
submission = pd.DataFrame({"PassengerId":titanic_test["PassengerId"],
                           "Survived":test_preds})

# Save submission to CSV
submission.to_csv("tutorial_logreg_submission.csv",
                  index=False)       # Do not save index values

print(pd)
```

Output:

Survival_prob	0.193110906347	0.729443792051
Sex		
female	0	312
male	577	0

The table shows that the model predicted a survival chance of roughly 19% for males and 73% for females.

	precision	recall	f1-score	support
0	0.82	0.85	0.83	549
1	0.74	0.70	0.72	340
avg / total	0.79	0.79	0.79	889

For the Titanic competition, accuracy is the scoring metric used to judge the competition, so we don't have to worry too much about other metrics.

Survived	0	1
row_0		
0	467	103
1	82	237

The table above shows the classes our model predicted vs. true values of the Survived variable.

This logistic regression model has an accuracy score of 0.75598 which is actually worse than the accuracy of the simplistic women survive, men die model (0.76555).

Example 2:

The dataset is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict whether the client will subscribe (1/0) to a term deposit (variable y). The dataset provides the bank customers' information. It includes 41,188 records and 21 fields.

Input variables

1. **age** (numeric)
2. **job :** type of job (categorical: "admin", "blue-collar", "entrepreneur", "housemaid", "management", "retired", "self-employed", "services", "student", "technician", "unemployed", "unknown")
3. **marital :** marital status (categorical: "divorced", "married", "single", "unknown")

4. **education** (categorical: “basic.4y”, “basic.6y”, “basic.9y”, “high.school”, “illiterate”, “professional.course”, “university.degree”, “unknown”)
5. **default:** has credit in default? (categorical: “no”, “yes”, “unknown”)
6. **housing:** has housing loan? (categorical: “no”, “yes”, “unknown”)
7. **loan:** has personal loan? (categorical: “no”, “yes”, “unknown”)
8. **contact:** contact communication type (categorical: “cellular”, “telephone”)
9. **month:** last contact month of year (categorical: “jan”, “feb”, “mar”, …, “nov”, “dec”)
10. **day_of_week:** last contact day of the week (categorical: “mon”, “tue”, “wed”, “thu”, “fri”)
11. **duration:** last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y=’no’). The duration is not known before a call is performed, also, after the end of the call, y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model
12. **campaign:** number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. **pdays:** number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14. **previous:** number of contacts performed before this campaign and for this client (numeric)
15. **poutcome:** outcome of the previous marketing campaign (categorical: “failure”, “nonexistent”, “success”)
16. **emp.var.rate:** employment variation rate — (numeric)
17. **cons.price.idx:** consumer price index — (numeric)
18. **cons.conf.idx**: consumer confidence index — (numeric)
19. **euribor3m:** euribor 3 month rate — (numeric)
20. **nr.employed:** number of employees — (numeric)

Predict variable (desired target):
y — has the client subscribed a term deposit?
(binary: “1”, means “Yes”, “0” means “No”)

Program Code:

```
# -*- coding: utf-8 -*-
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
data = pd.read_csv('bank.csv', header=0)
data = data.dropna()
print(data.shape)
print(list(data.columns))
data['education'].unique()
data['education']=np.where(data['education'] =='basic.9y', 'Basic', data['education'])
```

```
data['education']=np.where(data['education'] =='basic.6y', 'Basic', data['education'])
data['education']=np.where(data['education'] =='basic.4y', 'Basic', data['education'])
data['education'].unique()
data['y'].value_counts()

sns.countplot(x='y', data=data, palette='hls')
plt.show();
plt.savefig('Practical10B-plot.jpeg')

count_no_sub = len(data[data['y']==0])
count_sub = len(data[data['y']==1])
pct_of_no_sub = count_no_sub/(count_no_sub+count_sub)
print("percentage of no subscription is", pct_of_no_sub*100)
pct_of_sub = count_sub/(count_no_sub+count_sub)
print("percentage of subscription", pct_of_sub*100)

data.groupby('y').mean()
data.groupby('job').mean()
data.groupby('marital').mean()
data.groupby('education').mean()

########### Purchase Frequency for Job Title
pd.crosstab(data.job,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Job Title')
plt.xlabel('Job')
plt.ylabel('Frequency of Purchase')
plt.savefig('purchase_fre_job')

#######################  Marital Status vs Purchase
table=pd.crosstab(data.marital,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')
plt.savefig('mariral_vs_pur_stack')

############       Education vs Purchase
table=pd.crosstab(data.education,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Education vs Purchase')
plt.xlabel('Education')
plt.ylabel('Proportion of Customers')
plt.savefig('edu_vs_pur_stack')

pd.crosstab(data.day_of_week,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Frequency of Purchase')
plt.savefig('pur_dayofweek_bar')
```

```
############  Purchase Frequency for Month
pd.crosstab(data.month,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Month')
plt.xlabel('Month')
plt.ylabel('Frequency of Purchase')
plt.savefig('pur_fre_month_bar')

########### Age Purchase frequency pattern
data.age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('hist_age')
```

Output: -

```
In [47]: runfile('K:/Research In Computing/Practical Material/Prograı
Practical_10/Practical_10B.py', wdir='K:/Research In Computing/Pract:
Material/Programs/Practical_10')
(41188, 21)
['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
'previous', 'poutcome', 'emp_var_rate', 'cons_price_idx',
'cons_conf_idx', 'euribor3m', 'nr_employed', 'y']
```

```
percentage of no subscription is 88.73458288821988
percentage of subscription 11.265417111780131
```

PERCENTAGE OF NO SUBSCRIPTION IS 88.73458288821988
PERCENTAGE OF SUBSCRIPTION 11.265417111780131

Our classes are imbalanced, and the ratio of no-subscription to subscription instances is 89:11.

- The average age of customers who bought the term deposit is higher than that of the customers who didn't.
- The pdays (days since the customer was last contacted) is understandably lower for the customers who bought it. The lower the pdays, the better the memory of the last call and hence the better chances of a sale.
- Surprisingly, campaigns (number of contacts or calls made during the current campaign) are lower for customers who bought the term deposit.

The frequency of purchase of the deposit depends a great deal on the job title. Thus, the job title can be a good predictor of the outcome variable.

The marital status does not seem a strong predictor for the outcome variable.

Education seems a good predictor of the outcome variable.

Day of week may not be a good predictor of the outcome.

Month might be a good predictor of the outcome variable.

Most of the customers of the bank in this dataset are in the age range of 30–40.

DATA SCIENCE

Table of Contents

Sr. No	Practical No		Name of the Practical	SIGNATURE
1)	---		Prerequisites to Data Science Practical.	
2)	1		Creating Data Model using Cassandra.	
3)	2		**Conversion from different formats to HOURS format.**	
4)		A.	Text delimited csv format.	
5)		B.	XML	
6)		C.	JSON	
7)		D.	MySQL Database	
8)		E.	Picture (JPEG)	
9)		F.	Video	
10)		G.	Audio	
11)	3		**Utilities and Auditing**	
12)	4		**Retrieving Data**	
13)	5		**Assessing Data**	
14)	6		**Processing Data**	
15)	7		**Transforming Data**	
16)	8		**Organizing Data**	
17)	9		**Generating Reports**	
18)	10		**Data Visualization with Power BI**	

Practical 1:

Creating Data Model using Cassandra.

Go to Cassandra directory
C:\apache-cassandra-3.11.4\bin

Run Cassandra.bat file
Open C:\apache-cassandra-3.11.4\bin\cqlsh.py with python 2.7 and run

Creating a Keyspace using Cqlsh

Create keyspace keyspace1 with replication = {'class':'SimpleStratergy', 'replication_factor': 3};
Use keyspace1;

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 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 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');

```
cqlsh:keyspace1> select * from emp;

 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

(6 rows)
```

```
cqlsh:keyspace1> select * from dept;

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

(3 rows)
```

update dept set dept_name='Human Resource' where dept_id=1003;

```
cqlsh:keyspace1> select * from dept;

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

(3 rows)
```

```
cqlsh:keyspace1> delete from emp where emp_id=1006;
cqlsh:keyspace1> select * from emp;

 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 2:

Write Python / R Program to convert from the following formats to HORUS format:

A. **Text delimited CSVto HORUS format.**

Code:

```
# Utility Start CSV to HORUS ==================================
# Standard Tools
import pandas as pd
# Input Agreement ============================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/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='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
# Utility done ===============================================
```

Output:

```
=============== RESTART: C:\VKHCG\05-DS\9999-Data\CSV2HORUS.py ===============
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
>>>
```

B. **XML to HORUS Format**

Code:

```
# 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='C:/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='C:/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:

```
=============== RESTART: C:\VKHCG\05-DS\9999-Data\XML2HORUS.py ===============
====================================================
Input Data Values ==================================
====================================================
Squeezed text (385 lines).
====================================================
====================================================
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
====================================================
>>>
```

C. JSON to HORUS Format

Code:

```
# Utility Start JSON to HORUS ================================
# Standard Tools
#==============================================================
=
import pandas as pd
# Input Agreement ============================================
sInputFileName='C:/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='c:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
# Utility done ==============================================
```

Output:

```
>>>
============== RESTART: C:\VKHCG\05-DS\9999-Data\JSON2HORUS.py ==============
Input Data Values ===================================
                          Country ISO-2-CODE ISO-3-Code  ISO-M49
0                     Afghanistan         AF        AFG        4
1                   Aland Islands         AX        ALA      248
10                      Argentina         AR        ARG       32
100                       Hungary         HU        HUN      348
101                       Iceland         IS        ISL      352
..                            ...        ...        ...      ...
95                         Guyana         GY        GUY      328
96                          Haiti         HT        HTI      332
97      Heard and Mcdonald Islands         HM        HMD      334
98   Holy Seeÿ(Vatican City State)         VA        VAT      336
99                       Honduras         HN        HND      340

[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
>>> |
```

D. MySql Database to HORUS Format

Code:

```
# Utility Start Database to HORUS ==================================
# Standard Tools
#==============================================================
=
import pandas as pd
import sqlite3 as sq
# Input Agreement ==========================================
```

```
sInputFileName='C:/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='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
# Utility done ================================================
```

Output:

```
>>>
============ RESTART: C:\VKHCG\05-DS\9999-Data\DATABASE2HORUS.py ============
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
```

E. Picture (JPEG) to HORUS Format

Code:

```
# Utility Start Picture to HORUS ==================================
# Standard Tools
#=================================================================
=
from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Input Agreement =============================================
```

```
sInputFileName='C:/VKHCG/05-DS/9999-Data/Angus.jpg'
InputData = imread(sInputFileName, flatten=False, mode='RGBA')

print('Input Data Values ===================================')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('=====================================================')
# Processing Rules ==========================================
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']
ProcessData.columns=sColumns
ProcessData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('=====================================================')
print('Process Data Values ================================')
print('=====================================================')
plt.imshow(InputData)
plt.show()
print('=====================================================')
# Output Agreement ==========================================
OutputData=ProcessData
print('Storing File')
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Picture to HORUS - Done')
print('=====================================================')
```

Output:

F. Video to HORUS Format

Code:

Movie to Frames

```
# Utility Start Movie to HORUS (Part 1) =====================
# Standard Tools
#============================================================
import os
```

```
import shutil
import cv2
#==============================================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/dog.mp4'
sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp'
if os.path.exists(sDataBaseDir):
shutil.rmtree(sDataBaseDir)
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
print('=====================================================')
print('Start Movie to Frames')
print('=====================================================')
vidcap = cv2.VideoCapture(sInputFileName)
success,image = vidcap.read()
count = 0
while success:
success,image = vidcap.read()
sFrame=sDataBaseDir + str('/dog-frame-' + str(format(count, '04d'))+ '.jpg')
print('Extracted: ', sFrame)
cv2.imwrite(sFrame, image)
if os.path.getsize(sFrame) == 0:
count += -1
os.remove(sFrame)
print('Removed: ', sFrame)
if cv2.waitKey(10) == 27: # exit if Escape is hit
break
        count += 1
print('=====================================================')
print('Generated : ', count, ' Frames')
print('=====================================================')
print('Movie to Frames HORUS - Done')
print('=====================================================')
# Utility done ================================================
```

```
>>>
========== RESTART: C:\VKHCG\05-DS\9999-Data\MOVIE2HORUSFrame.py ==========
=====================================================
Start Movie to Frames
=====================================================
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0000.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0001.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0002.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0003.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0004.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0005.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0006.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0007.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0008.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0009.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0010.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0099.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0100.jpg
Extracted:  C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0101.jpg
=====================================================
Generated :  101  Frames
=====================================================
Movie to Frames HORUS - Done
=====================================================
>>>
```

Now frames are created and need to load them into HORUS.

Frames to Horus

```
# Utility Start Movie to HORUS (Part 2) ======================
# Standard Tools
#==============================================================
```

```
from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
# Input Agreement ==========================================
sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp'
f=0
for file in os.listdir(sDataBaseDir):
if file.endswith(".jpg"):
f += 1
sInputFileName=os.path.join(sDataBaseDir, file)
print('Process : ', sInputFileName)
InputData = imread(sInputFileName, flatten=False, mode='RGBA')
print('Input Data Values ===================================')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
        print('=====================================================')
# Processing Rules ==========================================
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
ProcessFrameData['Frame']=file
print('====================================================')
print('Process Data Values ================================')
print('====================================================')
plt.imshow(InputData)
plt.show()
if f == 1:
ProcessData=ProcessFrameData
else:
ProcessData=ProcessData.append(ProcessFrameData)
if f > 0:
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha','FrameName']
ProcessData.columns=sColumns
print('====================================================')
ProcessFrameData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('====================================================')
# Output Agreement ==========================================
OutputData=ProcessData
print('Storing File')
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Movie-Frame.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('====================================================')
print('Processed ; ', f,' frames')
print('====================================================')
```

```
print('Movie to HORUS - Done')
print('=====================================================')
```

Output:

```
=====================================================
Rows:  15667200
Columns : 7
=====================================================
Storing File
=====================================================
Processed ;  102  frames
=====================================================
Movie to HORUS - Done
=====================================================
```

dog-frame-0000.jpeg

dog-frame-0001.jpeg

dog-frame-0100.jpeg

dog-frame-0101.jpeg

Check the files from C:\VKHCG\05-DS\9999-Data\temp
The movie clip is converted into 102 picture frames and then to HORUS format.

G. Audio to HORUS Format

Code:

```
# Utility Start Audio to HORUS ================================
# Standard Tools
#==============================================================
=
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
```

```
import numpy as np
#==============================================================
=
def show_info(aname, a,r):
    print ('----------------')
    print ("Audio:", aname)
    print ('----------------')
    print ("Rate:", r)
    print ('----------------')
    print ("shape:", a.shape)
    print ("dtype:", a.dtype)
    print ("min, max:", a.min(), a.max())
    print ('----------------')
    plot_info(aname, a,r)
#==============================================================
=
def plot_info(aname, a,r):
    sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz'
    plt.title(sTitle)
    sLegend=[]
    for c in range(a.shape[1]):
        sLabel = 'Ch' + str(c+1)
        sLegend=sLegend+[str(c+1)]
        plt.plot(a[:,c], label=sLabel)
    plt.legend(sLegend)
    plt.show()
#==============================================================
=
sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#==============================================================
=
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
```

```
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#=============================================================
=
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("6 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#=============================================================
=
sInputFileName='C:/VKHCG/05-DS/9999-Data/8ch-sound.wav'
print('=====================================================')
print('Processing : ', sInputFileName)
print('=====================================================')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("8 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====================================================')
print('Audio to HORUS - Done')
```

Output:

Practical 3: Utilities and Auditing

A. Fixers Utilities:

Fixers enable your solution to take your existing data and fix a specific quality issue.

```
#---------------------------- Program to Demonstrate Fixers utilities -------------------
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#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:

```
>>>
==================== RESTART: C:/Users/User/Desktop/u1.py ====================
#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   :  Data Science 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
>>>
Ln: 72  Col: 4
```

B. Data Binning or Bucketing

Binning is a data preprocessing technique used to reduce the effects of minor observation errors. Statistical data binning is a way to group a number of more or less continuous values into a smaller number of “bins.”

Code :

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
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, density=1)
# 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='C:/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png'
fig.savefig(sPathFig)
plt.show()
```

Output:

C. Averaging of Data

The use of averaging of features value enables the reduction of data volumes in a control fashion to improve effective data processing.

C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Mean.py

Code:

```
import pandas as pd
#################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/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:

```
>>>
============= RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Mean.py =============
##################################
Working Base : C:/VKHCG  using
##################################
Loading : C:/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
```

Outlier Detection

Outliers are data that is so different from the rest of the data in the data set that it may be caused by an error in the data source. There is a technique called outlier detection that, with good data science, will identify these outliers.

C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py

Code:

```
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import pandas as pd
#################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('##################################')
```

```
print('Working Base :',Base)
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)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data')
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) &
(AllData.Latitude<=UpperBound)]
print(OutliersNot)
#################################################################
```

Output:

```
=========== RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py
===========
################################
Working Base : C:/VKHCG
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
     Country Place_Name  Latitude
1910      GB     London   51.5130
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
...      ...        ...       ...

[1502 rows x 3 columns]
Outliers
Higher than  51.51263550786781
     Country Place_Name  Latitude
1910      GB     London   51.5130
```

```
1911      GB     London   51.5508
1912      GB     London   51.5649
1913      GB     London   51.5895
1914      GB     London   51.5232
1916      GB     London   51.5491
1919      GB     London   51.5161
1920      GB     London   51.5198
1921      GB     London   51.5198
1923      GB     London   51.5237
1924      GB     London   51.5237
Lower than  51.50617687562166
    Country Place_Name  Latitude
1915      GB     London   51.4739
Not Outliers
    Country Place_Name  Latitude
```

D. Logging

Write a Python / R program for basic logging in data science.

C:\VKHCG\77-Yoke\Yoke_Logging.py

Code:

```
import sys
import os
import logging
import uuid
import shutil
import time
##############################################################
   Base='C:/VKHCG'
##############################################################
sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','06-
Report']
sLevels=['debug','info','warning','error']

for sCompany in sCompanies:
   sFileDir=Base + '/' + sCompany
   if not os.path.exists(sFileDir):
      os.makedirs(sFileDir)
   for sLayer in sLayers:
      log = logging.getLogger()  # root logger
      for hdlr in log.handlers[:]:  # remove all old handlers
         log.removeHandler(hdlr)
      #-------------------------------------------------------------------------------
      sFileDir=Base + '/' + sCompany + '/' + sLayer + '/Logging'
      if os.path.exists(sFileDir):
         shutil.rmtree(sFileDir)
      time.sleep(2)
      if not os.path.exists(sFileDir):
         os.makedirs(sFileDir)
      skey=str(uuid.uuid4())
```

```
        sLogFile=Base + '/' + sCompany + '/' + sLayer +
'/Logging/Logging_'+skey+'.log'
        print('Set up:',sLogFile)
        # set up logging to file - see previous section for more details
        logging.basicConfig(level=logging.DEBUG,
                        format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
                        datefmt='%m-%d %H:%M',
                        filename=sLogFile,
                        filemode='w')
        # define a Handler which writes INFO messages or higher to the sys.stderr
        console = logging.StreamHandler()
        console.setLevel(logging.INFO)
        # set a format which is simpler for console use
        formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
        # tell the handler to use this format
        console.setFormatter(formatter)
        # add the handler to the root logger
        logging.getLogger('').addHandler(console)
         # Now, we can log to the root logger, or any other logger. First the root...
        logging.info('Practical Data Science is fun!.')

        for sLevel in sLevels:
            sApp='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel
            logger = logging.getLogger(sApp)
            if sLevel == 'debug':
                logger.debug('Practical Data Science logged a debugging message.')
             if sLevel == 'info':
                logger.info('Practical Data Science logged information message.')
            if sLevel == 'warning':
                logger.warning('Practical Data Science logged a warning message.')
            if sLevel == 'error':
                logger.error('Practical Data Science logged an error message.')
#-----------------------------------------------------------------------------
```

Output:

```
>>>
================= RESTART: C:\VKHCG\77-Yoke\Yoke_Logging.py =================
Set up: C:/VKHCG/01-Vermeulen/01-Retrieve/Logging/Logging_61705603-bb6e-47f0-b5a
9-23d42e267311.log
root         : INFO     Practical Data Science is fun!.
Apllication-01-Vermeulen-01-Retrieve-info: INFO     Practical Data Science logge
d information message.
Apllication-01-Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science lo
gged a warning message.
Apllication-01-Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logg
ed an error message.
Set up: C:/VKHCG/01-Vermeulen/02-Assess/Logging/Logging_a7fecb9b-4d40-474e-bc2d-
994958d85194.log
```

Practical 4

A. Perform the following data processing using R.

Use R-Studio for the following:

```
>library(readr)
Warning message:package ‘readr’ was built under R version 3.4.4
```

Load a table named IP_DATA_ALL.csv.

```
>IP_DATA_ALL <- read_csv("C:/VKHCG/01-Vermeulen/00-
RawData/IP_DATA_ALL.csv")
Parsed with column specification:
cols(
  ID = col_double(),
  Country = col_character(),
  `Place Name` = col_character(),
  `Post Code` = col_double(),
  Latitude = col_double(),
  Longitude = col_double(),
  `First IP Number` = col_double(),
  `Last IP Number` = col_double()
)

>View(IP_DATA_ALL)
>spec(IP_DATA_ALL)
cols(
  ID = col_double(),
  Country = col_character(),
  `Place Name` = col_character(),
  `Post Code` = col_double(),
  Latitude = col_double(),
  Longitude = col_double(),
  `First IP Number` = col_double(),
  `Last IP Number` = col_double()
)
```

This informs you that you have the following eight columns:

- ID of type integer
- Place name of type character
- Post code of type character
- Latitude of type numeric double
- Longitude of type numeric double
- First IP number of type integer
- Last IP number of type integer

```
>library(tibble)
>set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
New names:
Place Name -> Place.Name
Post Code -> Post.Code
```

First IP Number -> First.IP.Number
Last IP Number -> Last.IP.Number
This informs you that four of the field names are not valid and suggests new field names that are valid.You can fix any detected invalid column names by executing
IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = TRUE)
By using command View(IP_DATA_ALL_FIX), you can check that you have fixed the columns. The new table IP_DATA_ALL_FIX.csv will fix the invalid column names with valid names.

```
>sapply(IP_DATA_ALL_FIX, typeof)
          ID        Country      Place.Name       Post.Code        Latitude
    "double"    "character"    "character"       "double"        "double"
   Longitude First.IP.Number  Last.IP.Number
    "double"       "double"       "double"
>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)
```

IP_DATA_COUNTRY_FREQ

Filter

	Country	N
2	GB	1502
3	US	1383
1	DE	677

- The two biggest subset volumes are from the US and GB.
- The US has just over four times the data as GB.

```
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)
```

	Latitude	N
133	51.5092	1478
1	40.6888	130
107	48.15	114
9	40.7143	113

- The two biggest data volumes are from latitudes 51.5092 and 40.6888.
- The spread appears to be nearly equal between the top-two latitudes.

```
>sapply(IP_DATA_ALL_FIX[,'Latitude'], min, na.rm=TRUE)
```

Latitude 40.6888
What does this tell you?
Fact: The range of latitude for the Northern Hemisphere is from 0 to 90. So, if you do not have any latitudes farther south than 40.6888, you can improve your retrieve routine.

```
>sapply(IP_DATA_ALL_FIX[,'Country'], min, na.rm=TRUE)
Country    "DE"
```

Minimum business frequency is from DE – Denmark.

```
>sapply(IP_DATA_ALL_FIX[,'Latitude'], max, na.rm=TRUE)
Latitude
 51.5895
>sapply(IP_DATA_ALL_FIX[,'Country'], max, na.rm=TRUE)
Country
   "US"
```

The result is 51.5895. What does this tell you?
Fact: The range in latitude for the Northern Hemisphere is from 0 to 90. So, if you do not have any latitudes more northerly than 51.5895, you can improve your retrieve routine.

```
>sapply(IP_DATA_ALL_FIX [,'Latitude'], mean, na.rm=TRUE)
Latitude
46.69097
>sapply(IP_DATA_ALL_FIX [,'Latitude'], median, na.rm=TRUE)
Latitude
   48.15
>sapply(IP_DATA_ALL_FIX [,'Latitude'], range, na.rm=TRUE)
     Latitude
[1,]  40.6888
[2,]  51.5895
>sapply(IP_DATA_ALL_FIX [,'Latitude'], quantile, na.rm=TRUE)
     Latitude
0%    40.6888
25%   40.7588
50%   48.1500
75%   51.5092
100%  51.5895
>sapply(IP_DATA_ALL_FIX [,'Latitude'], sd, na.rm=TRUE)
Latitude
4.890387
>sapply(IP_DATA_ALL_FIX [,'Longitude'], sd, na.rm=TRUE)
Longitude
 38.01702
```

B. Program to retrieve different attributes of data.

```
##### C:\ VKHCG\01-Vermeulen\01-Retrieve\Retrive_IP_DATA_ALL.py###
import sys
import os
import pandas as pd
#################################################################
Base='C:/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_with_ID.head())
sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
##################################################################
print('### Done!! ##############################################')
##################################################################
```

C. Data Pattern

To determine a pattern of the data values, Replace all alphabet values with an uppercase case *A*, all numbers with an uppercase *N*, and replace any spaces with a lowercase letter *b* and all other unknown characters with a lowercase *u*. As a result, "Good Book 101" becomes "AAAAbAAAAbNNNu."This pattern creation is beneficial for designing any specific assess rules. This pattern view of data is a quick way to identify common patterns or determine standard layouts.

```
library(readr)
library(data.table)
FileName=paste0('c:/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)
  };
  for (n in seq(0,9,1)){
    s=chartr(as.character(n),"N",s)
  };
  s=chartr(" ","b",s)
  s=chartr(".","u",s)
  pattern_country[r,]$PatternCountry=s;
};
View(pattern_country)
```

Filter

	Country	PatternCountry
1	US	AA
2	DE	AA
3	GB	AA

Example 2: This is a common use of patterns to separate common standards and structures. Pattern can be loaded in separate retrieve procedures. If the same two patterns, NNNNuNNuNN and uuNNuNNuNN, are found, you can send NNNNuNNuNN directly to be converted into a date, while uuNNuNNuNN goes through a quality-improvement process to then route back to the same queue as NNNNuNNuNN, once it complies.

```
library(readr)
library(data.table)
Base='C:/VKHCG'
FileName=paste0(Base,'/01-Vermeulen/00-RawData/IP_DATA_ALL.csv')
IP_DATA_ALL <- read_csv(FileName)
hist_latitude=data.table(Latitude=unique(IP_DATA_ALL$Latitude))
```

```
pattern_latitude=data.table(latitude=hist_latitude$Latitude,
                            Patternlatitude=as.character(hist_latitude$Latitude))
oldchar=c(letters,LETTERS)
newchar=replicate(length(oldchar),"A")
for (r in seq(nrow(pattern_latitude))){
  s=pattern_latitude[r,]$Patternlatitude;
  for (c in seq(length(oldchar))){
    s=chartr(oldchar[c],newchar[c],s)
  };
  for (n in seq(0,9,1)){
    s=chartr(as.character(n),"N",s)
  };
  s=chartr(" ","b",s)
  s=chartr("+","u",s)
  s=chartr("-","u",s)
  s=chartr(".","u",s)
  pattern_latitude[r,]$Patternlatitude=s;
};
setorder(pattern_latitude,latitude)
View(pattern_latitude[1:3])
```

Filter

	latitude	Patternlatitude
1	40.6888	NNuNNNN
2	40.7038	NNuNNNN
3	40.7055	NNuNNNN

D. Loading IP_DATA_ALL:

This data set contains all the IP address allocations in the world. It will help you to locateyour customers when interacting with them online.

Create a new Python script file and save it as Retrieve-IP_DATA_ALL.py in directory C:\VKHCG\01-Vermeulen\01-Retrieve.

```
##############Retrieve-IP_DATA_ALL.py#######################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
#################################################################
Base='C:/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_with_ID.head())

sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
#####################################################################
print('### Done!! ##########################################')
#####################################################################
```

```
>>>
===== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-IP_DATA_ALL.py =====
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 3562
Columns: 8
### Raw Data Set ####################################
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 ##################################
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!! ##########################################
>>>
```

Start your Python editor and create a text file named Retrieve-IP_Routing.py in directory. C:\VKHCG\01-Vermeulen\01-Retrieve.

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
import os
import pandas as pd
from math import radians, cos, sin, asin, sqrt
#####################################################################
def haversine(lon1, lat1, lon2, lat2,stype):
```

```
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    if stype == 'km':
        r = 6371 # Radius of earth in kilometers
    else:
        r = 3956 # Radius of earth in miles
    d=round(c * r,3)
    return d
##################################################################
Base='C:/VKHCG'
##################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_CORE.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
##################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
##################################################################
IP_DATA = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
IP_DATA.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA1 = IP_DATA
IP_DATA1.insert(0, 'K', 1)
IP_DATA2 = IP_DATA1
##################################################################
print(IP_DATA1.shape)
##################################################################
IP_CROSS=pd.merge(right=IP_DATA1,left=IP_DATA2,on='K')
IP_CROSS.drop('K', axis=1, inplace=True)
IP_CROSS.rename(columns={'Longitude_x': 'Longitude_from', 'Longitude_y':
'Longitude_to'}, inplace=True)
IP_CROSS.rename(columns={'Latitude_x': 'Latitude_from', 'Latitude_y': 'Latitude_to'},
inplace=True)
IP_CROSS.rename(columns={'Place_Name_x': 'Place_Name_from', 'Place_Name_y':
'Place_Name_to'}, inplace=True)
IP_CROSS.rename(columns={'Country_x': 'Country_from', 'Country_y': 'Country_to'},
inplace=True)
##################################################################
IP_CROSS['DistanceBetweenKilometers'] = IP_CROSS.apply(lambda row:
    haversine(
```

```
        row['Longitude_from'],
        row['Latitude_from'],
        row['Longitude_to'],
        row['Latitude_to'],
        'km')
        ,axis=1)
##################################################################
IP_CROSS['DistanceBetweenMiles'] = IP_CROSS.apply(lambda row:
   haversine(
        row['Longitude_from'],
        row['Latitude_from'],
        row['Longitude_to'],
        row['Latitude_to'],
        'miles')
        ,axis=1)
print(IP_CROSS.shape)
sFileName2=sFileDir + '/Retrieve_IP_Routing.csv'
IP_CROSS.to_csv(sFileName2, index = False, encoding="latin-1")
##################################################################
print('### Done!! ##############################################')
##################################################################
```

Output:

See the file named Retrieve_IP_Routing.csv in C:\VKHCG\01-Vermeulen\01-Retrieve\01-EDS\02-Python.

1	Country_from	Place_Name_from	Latitude_from	Longitude_from	Country_to	Place_Name_to	Latitude_to	Longitude_to	DistanceBetweenKilometers	DistanceBetweenMiles
2	US	New York	40.7528	-73.9725	US	New York	40.7528	-73.9725	0	0
3	US	New York	40.7528	-73.9725	US	New York	40.7214	-74.0052	4.448	2.762
4	US	New York	40.7528	-73.9725	US	New York	40.7662	-73.9862	1.885	1.17
5	US	New York	40.7528	-73.9725	US	New York	40.7449	-73.9782	1.001	0.622
6	US	New York	40.7528	-73.9725	US	New York	40.7605	-73.9933	1.95	1.211
7	US	New York	40.7528	-73.9725	US	New York	40.7588	-73.968	0.767	0.476
8	US	New York	40.7528	-73.9725	US	New York	40.7637	-73.9727	1.212	0.753
9	US	New York	40.7528	-73.9725	US	New York	40.7553	-73.9924	1.699	1.055
10	US	New York	40.7528	-73.9725	US	New York	40.7308	-73.9975	3.228	2.004
11	US	New York	40.7528	-73.9725	US	New York	40.7694	-73.9609	2.088	1.297

Total Records: 22501

So, the distance between a router in New York (40.7528, -73.9725) to anoher router in New York (40.7214, -74.0052) is 4.448 kilometers, or 2.762 miles.

Building a Diagram for the Scheduling of Jobs

Start your Python editor and create a text file named Retrieve-Router-Location.py in directory.

C:\VKHCG\01-Vermeulen\01-Retrieve.

```
###################    Retrieve-Router-Location.py   #######################
# -*- coding: utf-8 -*-
##################################################################
import sys
import os
import pandas as pd
##################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
##################################################################
Base='C:/VKHCG'
```

```
#################################################################
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)
#################################################################
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)

print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])

sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False, encoding="latin-1")
#################################################################
print('### Done!! ##############################################')
#################################################################
```

Output:

```
>>>
=== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-Router-Location.py ===
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
Rows : 150
Columns : 4
### Done!! ##############################################
```

See the file named Retrieve_Router_Location.csv in C:\VKHCG\01-Vermeulen\01-Retrieve\01-EDS\02-Python.

1	Country	Place_Name	Latitude	Longitude
2	US	New York	40.7528	-73.9725
3	US	New York	40.7214	-74.0052
4	US	New York	40.7662	-73.9862
5	US	New York	40.7449	-73.9782
6	US	New York	40.7605	-73.9933
7	US	New York	40.7588	-73.968
8	US	New York	40.7637	-73.9727
9	US	New York	40.7553	-73.9924
10	US	New York	40.7308	-73.9975

Krennwallner AG

The company has two main jobs in need of your attention:

- *Picking content for billboards*: I will guide you through the data science required to pick advertisements for each billboard in the company.
- *Understanding your online visitor data*: I will guide you through the evaluation of the web traffic to the billboard's online web servers.

Picking Content for Billboards
Start your Python editor and create a text file named Retrieve-DE-Billboard-Locations.py in directory.
C:\VKHCG\02-Krennwallner\01-Retrieve.

```
################# Retrieve-DE-Billboard-Locations.py    ###############
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
################################################################
InputFileName='DE_Billboard_Locations.csv'
OutputFileName='Retrieve_DE_Billboard_Locations.csv'
Company='02-Krennwallner'
################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
Base='C:/VKHCG'
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','PlaceName','Latitude','Longitude'])

IP_DATA_ALL.rename(columns={'PlaceName': 'Place_Name'}, inplace=True)
################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

ROUTERLOC = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)

print('Rows :',ROUTERLOC.shape[0])
print('Columns :',ROUTERLOC.shape[1])

sFileName2=sFileDir + '/' + OutputFileName
ROUTERLOC.to_csv(sFileName2, index = False)

################################################################
print('### Done!! ############################################')
################################################################
```

```
>>>
 RESTART: C:\VKHCG\02-Krennwallner\01-Retrieve\Retrieve-DE-Billboard-Locations.py
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/02-Krennwallner/00-RawData/DE_Billboard_Locations.csv
Rows : 8873
Columns : 4
### Done!! ############################################
```

See the file named Retrieve_Router_Location.csv in C:\VKHCG\02-Krennwallner\01-Retrieve\01-EDS\02-Python.

1	Country	Place_Name	Latitude	Longitude
2	US	New York	40.7528	-73.9725
3	US	New York	40.7214	-74.0052
4	US	New York	40.7662	-73.9862
5	US	New York	40.7449	-73.9782
6	US	New York	40.7605	-73.9933
7	US	New York	40.7588	-73.968
8	US	New York	40.7637	-73.9727
9	US	New York	40.7553	-73.9924
10	US	New York	40.7308	-73.9975

Understanding Your Online Visitor Data

Let's retrieve the visitor data for the billboard we have in Germany.

Several times it was found that common and important information is buried somewhere in the company's various data sources. Investigating any direct suppliers or consumers' upstream or downstream data sources attached to the specific business process is necessary. That is part of your skills that you are applying to data science. Numerous insightful fragments of information was found in the data sources surrounding a customer's business processes.

Start your Python editor and create a file named Retrieve-Online-Visitor.py in directory C:\VKHCG\02-Krennwallner\01-Retrieve.

```
################################################################
# -*- coding: utf-8 -*-
################################################################
import sys
import os
import pandas as pd
import gzip as gz
################################################################
InputFileName='IP_DATA_ALL.csv'
OutputFileName='Retrieve_Online_Visitor'
CompanyIn= '01-Vermeulen'
CompanyOut= '02-Krennwallner'
################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Base='C:/VKHCG'
sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName
```

```
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
  usecols=['Country','Place Name','Latitude','Longitude','First IP Number','Last IP Number'])

IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA_ALL.rename(columns={'First IP Number': 'First_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Last IP Number': 'Last_IP_Number'}, inplace=True)
#################################################################
sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

visitordata = IP_DATA_ALL.drop_duplicates(subset=None, keep='first', inplace=False)
visitordata10=visitordata.head(10)

print('Rows :',visitordata.shape[0])
print('Columns :',visitordata.shape[1])

print('Export CSV')
sFileName2=sFileDir + '/' + OutputFileName + '.csv'
visitordata.to_csv(sFileName2, index = False)
print('Store All:',sFileName2)

sFileName3=sFileDir + '/' + OutputFileName + '_10.csv'
visitordata10.to_csv(sFileName3, index = False)
print('Store 10:',sFileName3)

for z in ['gzip', 'bz2', 'xz']:
    if z == 'gzip':
        sFileName4=sFileName2 + '.gz'
    else:
        sFileName4=sFileName2 + '.' + z
    visitordata.to_csv(sFileName4, index = False, compression=z)
    print('Store :',sFileName4)
#################################################################
print('Export JSON')
for sOrient in ['split','records','index', 'columns','values','table']:
    sFileName2=sFileDir + '/' + OutputFileName + '_' + sOrient + '.json'
    visitordata.to_json(sFileName2,orient=sOrient,force_ascii=True)
    print('Store All:',sFileName2)

    sFileName3=sFileDir + '/' + OutputFileName + '_10_' + sOrient + '.json'
    visitordata10.to_json(sFileName3,orient=sOrient,force_ascii=True)
    print('Store 10:',sFileName3)

    sFileName4=sFileName2 + '.gz'
    file_in = open(sFileName2, 'rb')
    file_out = gz.open(sFileName4, 'wb')
    file_out.writelines(file_in)
    file_in.close()
```

```
    file_out.close()
    print('Store GZIP All:',sFileName4)

    sFileName5=sFileDir + '/' + OutputFileName + '_' + sOrient + '_UnGZip.json'
    file_in = gz.open(sFileName4, 'rb')
    file_out = open(sFileName5, 'wb')
    file_out.writelines(file_in)
    file_in.close()
    file_out.close()
    print('Store UnGZIP All:',sFileName5)
#####################################################################
print('### Done!! ##############################################')
#####################################################################
```

Output:

```
== RESTART: C:\VKHCG\02-Krennwallner\01-Retrieve\Retrieve-Online-Visitor.py ==
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows : 3562
Columns : 6
Export CSV
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10.csv
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.gz
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.bz2
Store : C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv.xz
Export JSON
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_split.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_split_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_records.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_records_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_index.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_index_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns.json
Store 10: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_10_columns.json
Store GZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns.json.gz
Store UnGZIP All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_columns_UnGZip.json
Store All: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor_values.json
```

See the file named Retrieve_Online_Visitor.csv in C:\VKHCG\02-Krennwallner\01-Retrieve\01-EDS\02-Python.

	A	B	C	D	E	F
1	Country	Place_Name	Latitude	Longitude	First_IP_Number	Last_IP_Number
2	US	New York	40.6888	-74.0203	400887248	400887263
3	US	New York	40.6888	-74.0203	400904512	400904543
4	US	New York	40.6888	-74.0203	401402080	401402095
5	US	New York	40.6888	-74.0203	402261072	402261087
6	US	New York	40.6888	-74.0203	402288032	402288047
7	US	New York	40.6888	-74.0203	641892352	641900543
8	US	New York	40.6888	-74.0203	644464896	644465151
9	US	New York	40.6888	-74.0203	758770912	758770927
10	US	New York	40.6888	-74.0203	1075972352	1075975167

You can also see the following JSON files of only ten records.

XML processing.
Start Python editor and create a file named Retrieve-Online-Visitor-XML.py indirectory C:\VKHCG\02-Krennwallner\01-Retrieve.

```
#####################################################################
# -*- coding: utf-8 -*-
#####################################################################
import sys
```

```
import os
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)
#################################################################
InputFileName='IP_DATA_ALL.csv'
OutputFileName='Retrieve_Online_Visitor.xml'
CompanyIn= '01-Vermeulen'
CompanyOut= '02-Krennwallner'
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
#################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#################################################################
sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False)

IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
IP_DATA_ALL.rename(columns={'First IP Number': 'First_IP_Number'}, inplace=True)
```

```
IP_DATA_ALL.rename(columns={'Last IP Number': 'Last_IP_Number'}, inplace=True)
IP_DATA_ALL.rename(columns={'Post Code': 'Post_Code'}, inplace=True)
#################################################################
sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

visitordata = IP_DATA_ALL.head(10000)

print('Original Subset Data Frame')
print('Rows :',visitordata.shape[0])
print('Columns :',visitordata.shape[1])
print(visitordata)

print('Export XML')
sXML=df2xml(visitordata)

sFileName=sFileDir + '/' + OutputFileName
file_out = open(sFileName, 'wb')
file_out.write(sXML)
file_out.close()
print('Store XML:',sFileName)
xml_data = open(sFileName).read()
unxmlrawdata=xml2df(xml_data)
print('Raw XML Data Frame')
print('Rows :',unxmlrawdata.shape[0])
print('Columns :',unxmlrawdata.shape[1])
print(unxmlrawdata)
unxmldata = unxmlrawdata.drop_duplicates(subset=None, keep='first', inplace=False)
print('Deduplicated XML Data Frame')
print('Rows :',unxmldata.shape[0])
print('Columns :',unxmldata.shape[1])
print(unxmldata)
#print('### Done!! ##############################################')
```

Output:
See a file named Retrieve_Online_Visitor.xml in C:\VKHCG\02-Krennwallner\01-Retrieve\01-EDS\02-Python.
This enables you to deliver XML format data as part of the retrieve step.

```
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Original Subset Data Frame
Rows : 3562
Columns : 8
      ID Country Place_Name  ... Longitude  First_IP_Number  Last_IP_Number
0      1     US   New York  ...  -73.9725       204276480       204276735
1      2     US   New York  ...  -73.9725       301984864       301985791
2      3     US   New York  ...  -73.9725       404678736       404679039
3      4     US   New York  ...  -73.9725       411592704       411592959
4      5     US   New York  ...  -73.9725       416784384       416784639
...   ...    ...      ... ...      ...            ...             ...
3557  3558    DE     Munich  ...   11.5392      1591269504      1591269631
3558  3559    DE     Munich  ...   11.7500      1558374784      1558374911
3559  3560    DE     Munich  ...   11.4667      1480845312      1480845439
3560  3561    DE     Munich  ...   11.7434      1480596992      1480597503
3561  3562    DE     Munich  ...   11.7434      1558418432      1558418943

[3562 rows x 8 columns]
Export XML
Store XML: C:/VKHCG/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.xml
Raw XML Data Frame
Rows : 3562
Columns : 8
      ID Country Place_Name  ... Longitude First_IP_Number Last_IP_Number
0      1     US   New York  ...  -73.9725       204276480       204276735
1      2     US   New York  ...  -73.9725       301984864       301985791
2      3     US   New York  ...  -73.9725       404678736       404679039
3      4     US   New York  ...  -73.9725       411592704       411592959
4      5     US   New York  ...  -73.9725       416784384       416784639
...   ...    ...      ... ...      ...            ...             ...
3557  3558    DE     Munich  ...   11.5392      1591269504      1591269631
3558  3559    DE     Munich  ...     11.75      1558374784      1558374911
3559  3560    DE     Munich  ...   11.4667      1480845312      1480845439
3560  3561    DE     Munich  ...   11.7434      1480596992      1480597503
3561  3562    DE     Munich  ...   11.7434      1558418432      1558418943

[3562 rows x 8 columns]
Deduplicated XML Data Frame
Rows : 3562
Columns : 8
      ID Country Place_Name  ... Longitude First_IP_Number Last_IP_Number
0      1     US   New York  ...  -73.9725       204276480       204276735
1      2     US   New York  ...  -73.9725       301984864       301985791
2      3     US   New York  ...  -73.9725       404678736       404679039
3      4     US   New York  ...  -73.9725       411592704       411592959
4      5     US   New York  ...  -73.9725       416784384       416784639
...   ...    ...      ... ...      ...            ...             ...
3557  3558    DE     Munich  ...   11.5392      1591269504      1591269631
3558  3559    DE     Munich  ...     11.75      1558374784      1558374911
3559  3560    DE     Munich  ...   11.4667      1480845312      1480845439
3560  3561    DE     Munich  ...   11.7434      1480596992      1480597503
3561  3562    DE     Munich  ...   11.7434      1558418432      1558418943

[3562 rows x 8 columns]
>>>
```

C:\VKHCG\02-Krennwallner\01-Retrieve\01-EDS\...


```
- <root>
  - <entry>
      <ID>1</ID>
      <ID>1</ID>
      <Country>US</Country>
      <Country>US</Country>
      <Place_Name>New York</Place_Name>
      <Place_Name>New York</Place_Name>
      <Post_Code>10017</Post_Code>
      <Post_Code>10017</Post_Code>
      <Latitude>40.7528</Latitude>
      <Latitude>40.7528</Latitude>
      <Longitude>-73.9725</Longitude>
      <Longitude>-73.9725</Longitude>
      <First_IP_Number>204276480</First_IP_Number>
      <First_IP_Number>204276480</First_IP_Number>
      <Last_IP_Number>204276735</Last_IP_Number>
      <Last_IP_Number>204276735</Last_IP_Number>
    </entry>
```


Hillman Ltd

Start yourPython editor and create a file named Retrieve-Incoterm-EXW.py in directory C:\VKHCG\03-Hillman\01-Retrieve.

import os

```
import sys
import pandas as pd
IncoTerm='EXW'
InputFileName='Incoterm_2010.csv'
OutputFileName='Retrieve_Incoterm_' + IncoTerm + '_RuleSet.csv'
Company='03-Hillman'
#################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#################################################################
### Import Incoterms
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('############')
print('Loading :',sFileName)
IncotermGrid=pd.read_csv(sFileName,header=0,low_memory=False)
IncotermRule=IncotermGrid[IncotermGrid.Shipping_Term == IncoTerm]
print('Rows :',IncotermRule.shape[0])
print('Columns :',IncotermRule.shape[1])
print('############')
print(IncotermRule)
sFileName=sFileDir + '/' + OutputFileName
IncotermRule.to_csv(sFileName, index = False)
print('### Done!! ##############################################')
```

Output

See the file named Retrieve_Incoterm_EXW.csv in C:\VKHCG\03-Hillman\01-Retrieve\01-EDS\02-Python. Open this file,

```
>>>
===== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Incoterm-EXW.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
############
Loading : C:/VKHCG/03-Hillman/00-RawData/Incoterm_2010.csv
Rows : 1
Columns : 9
############
  Shipping_Term  Seller Carrier Port_From  ... Port_To Terminal Named_Place  Buyer
0           EXW  Seller   Buyer     Buyer  ...   Buyer    Buyer       Buyer  Buyer

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

FCA—Free Carrier (Named Place of Delivery)

```
import os
import sys
import pandas as pd
#################################################################
IncoTerm='FCA'
InputFileName='Incoterm_2010.csv'
OutputFileName='Retrieve_Incoterm_' + IncoTerm + '_RuleSet.csv'
Company='03-Hillman'
#################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#################################################################
### Import Incoterms
#################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('###########')
print('Loading :',sFileName)
IncotermGrid=pd.read_csv(sFileName,header=0,low_memory=False)
IncotermRule=IncotermGrid[IncotermGrid.Shipping_Term == IncoTerm]
print('Rows :',IncotermRule.shape[0])
print('Columns :',IncotermRule.shape[1])
print('###########')
print(IncotermRule)
sFileName=sFileDir + '/' + OutputFileName
IncotermRule.to_csv(sFileName, index = False)
print('### Done!! ############################################')
```

Output:

```
>>>
===== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Incoterm-FCA.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
###########
Loading : C:/VKHCG/03-Hillman/00-RawData/Incoterm_2010.csv
Rows : 1
Columns : 9
###########
  Shipping_Term  Seller Carrier Port_From  ... Port_To Terminal Named_Place  Buyer
1           FCA  Seller  Seller     Buyer  ...   Buyer    Buyer       Buyer  Buyer

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

CPT—Carriage Paid To (Named Place of Destination)

C:\VKHCG\03-Hillman\01-Retrieve.

CIP—Carriage and Insurance Paid To (Named Place of Destination)

DAT—Delivered at Terminal (Named Terminal at Port or Place of Destination)
DAP—Delivered at Place (Named Place of Destination)

DDP—Delivered Duty Paid (Named Place of Destination)
By this term, the seller is responsible for delivering the goods to the named place in the country of the buyer and pays all costs in bringing the goods to the destination, including import duties and taxes. The seller is not responsible for unloading. This term places the maximum obligations on the seller and minimum obligations on the buyer. No risk or responsibility is transferred to the buyer until delivery of the goods at the named place of destination.

Possible Shipping Routes
There are numerous potential shipping routes available to the company. The retrieve step can generate the potential set, by using a route combination generator. This will give you a set of routes, but it is highly unlikely that you will ship along all of them. It is simply a population of routes that can be used by the data science to find the optimum solution.
Start your Python editor and create a file named Retrieve-Warehouse-Incoterm-Chains.py in directory C:\VKHCG\03-Hillman\01-Retrieve.

Adopt New Shipping Containers
Adopting the best packing option for shipping in containers will require that I introduce a new concept. Shipping of containers is based on a concept reducing the packaging you use down to an optimum set of sizes having the following requirements:

- The product must fit within the box formed by the four sides of a cube.
- The product can be secured using packing foam, which will fill any void volume in the packaging.
- Packaging must fit in shipping containers with zero space gaps.
- Containers can only hold product that is shipped to a single warehouse, shop, or customer.

Start your Python editor and create a text file named Retrieve-Container-Plan.py in directory . C:\VKHCG\03-Hillman\01-Retrieve.
***** Replace pd.DataFrame.from_items with pd.DataFrame.from_dict**

```
import sys
import os
import pandas as pd
##################################################################
ContainerFileName='Retrieve_Container.csv'
BoxFileName='Retrieve_Box.csv'
ProductFileName='Retrieve_Product.csv'
Company='03-Hillman'
##################################################################
Base='C:/VKHCG'
##################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
##################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
```

```
    os.makedirs(sFileDir)
#################################################################
### Create the Containers
#################################################################
containerLength=range(1,21)
containerWidth=range(1,10)
containerHeigth=range(1,6)
containerStep=1
c=0
for l in containerLength:
    for w in containerWidth:
        for h in containerHeigth:
            containerVolume=(l/containerStep)*(w/containerStep)*(h/containerStep)
            c=c+1
            ContainerLine=[('ShipType', ['Container']),
                    ('UnitNumber', ('C'+format(c,"06d"))),
                    ('Length',(format(round(l,3),".4f"))),
                    ('Width',(format(round(w,3),".4f"))),
                    ('Height',(format(round(h,3),".4f"))),
                    ('ContainerVolume',(format(round(containerVolume,6),".6f")))]
            if c==1:
              ContainerFrame = pd.DataFrame.from_dict(ContainerLine)
            else:
                ContainerRow = pd.DataFrame.from_dict(ContainerLine)
                ContainerFrame = ContainerFrame.append(ContainerRow)
            ContainerFrame.index.name = 'IDNumber'

print('################')
print('## Container')
print('################')
print('Rows :',ContainerFrame.shape[0])
print('Columns :',ContainerFrame.shape[1])
print('################')
#################################################################
sFileContainerName=sFileDir + '/' + ContainerFileName
ContainerFrame.to_csv(sFileContainerName, index = False)
#################################################################
## Create valid Boxes with packing foam
#################################################################
boxLength=range(1,21)
boxWidth=range(1,21)
boxHeigth=range(1,21)
packThick=range(0,6)
boxStep=10
b=0
for l in boxLength:
    for w in boxWidth:
        for h in boxHeigth:
            for t in packThick:
                boxVolume=round((l/boxStep)*(w/boxStep)*(h/boxStep),6)
```

```
            productVolume=round(((l-t)/boxStep)*((w-t)/boxStep)*((h-t)/boxStep),6)
            if productVolume > 0:
               b=b+1
               BoxLine=[('ShipType', ['Box']),
                  ('UnitNumber', ('B'+format(b,"06d"))),
                  ('Length',(format(round(l/10,6),".6f"))),
                  ('Width',(format(round(w/10,6),".6f"))),
                  ('Height',(format(round(h/10,6),".6f"))),
                  ('Thickness',(format(round(t/5,6),".6f"))),
                  ('BoxVolume',(format(round(boxVolume,9),".9f"))),
                  ('ProductVolume',(format(round(productVolume,9),".9f")))]
               if b==1:
                  BoxFrame = pd.DataFrame.from_dict(BoxLine)
               else:
                   BoxRow = pd.DataFrame.from_dict(BoxLine)
                   BoxFrame = BoxFrame.append(BoxRow)
               BoxFrame.index.name = 'IDNumber'
print('#################')
print('## Box')
print('#################')
print('Rows :',BoxFrame.shape[0])
print('Columns :',BoxFrame.shape[1])
print('#################')
#################################################################
sFileBoxName=sFileDir + '/' + BoxFileName
BoxFrame.to_csv(sFileBoxName, index = False)
#################################################################
## Create valid Product
#################################################################
productLength=range(1,21)
productWidth=range(1,21)
productHeigth=range(1,21)
productStep=10
p=0
for l in productLength:
   for w in productWidth:
      for h in productHeigth:
         productVolume=round((l/productStep)*(w/productStep)*(h/productStep),6)
         if productVolume > 0:
            p=p+1
            ProductLine=[('ShipType', ['Product']),
               ('UnitNumber', ('P'+format(p,"06d"))),
               ('Length',(format(round(l/10,6),".6f"))),
               ('Width',(format(round(w/10,6),".6f"))),
               ('Height',(format(round(h/10,6),".6f"))),
               ('ProductVolume',(format(round(productVolume,9),".9f")))]
            if p==1:
               ProductFrame = pd.DataFrame.from_dict(ProductLine)
            else:
               ProductRow = pd.DataFrame.from_dict(ProductLine)
```

```
            ProductFrame = ProductFrame.append(ProductRow)
        BoxFrame.index.name = 'IDNumber'
print('##################')
print('## Product')
print('##################')
print('Rows :',ProductFrame.shape[0])
print('Columns :',ProductFrame.shape[1])
print('##################')
#################################################################
sFileProductName=sFileDir + '/' + ProductFileName
ProductFrame.to_csv(sFileProductName, index = False)
#################################################################
##################################################################
print('### Done!! ##############################################')
##################################################################
```

Output:

Your second simulation is the cardboard boxes for the packing of the products. The requirement is for boxes having a dimension of 100 centimeters × 100 centimeters × 100 centimeters to 2.1 meters × 2.1 meters × 2.1 meters. You can also use between zero and 600 centimeters of packing foam to secure any product in the box.

See the container data file Retrieve_Container.csv and Retrieve_Box.csv in C:\VKHCG\03-Hillman\01-Retrieve\01-EDS\02-Python.

Create a Delivery Route

The model enables you to generate a complex routing plan for the shipping routes of the company. Start your Python editor and create a text file named Retrieve-Route-Plan.py in directory .
C:\VKHCG\03-Hillman\01-Retrieve.

```
import os
import sys
import pandas as pd
from geopy.distance import vincenty
```

```
##################################################################
InputFileName='GB_Postcode_Warehouse.csv'
OutputFileName='Retrieve_GB_Warehouse.csv'
Company='03-Hillman'
##################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
##################################################################
sFileDir=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
##################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('###########')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
WarehouseClean=Warehouse[Warehouse.latitude != 0]
WarehouseGood=WarehouseClean[WarehouseClean.longitude != 0]
WarehouseGood.drop_duplicates(subset='postcode', keep='first', inplace=True)
WarehouseGood.sort_values(by='postcode', ascending=1)
##################################################################
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)

WarehouseLoop = WarehouseGood.head(20)

for i in range(0,WarehouseLoop.shape[0]):
    print('Run :',i,' =======>>>>>>>>>>>',WarehouseLoop['postcode'][i])
    WarehouseHold = WarehouseGood.head(10000)
    WarehouseHold['Transaction']=WarehouseHold.apply(lambda row:
            'WH-to-WH'
            ,axis=1)
    OutputLoopName='Retrieve_Route_' + 'WH-' + WarehouseLoop['postcode'][i] +
'_Route.csv'

    WarehouseHold['Seller']=WarehouseHold.apply(lambda row:
            'WH-' + WarehouseLoop['postcode'][i]
            ,axis=1)

    WarehouseHold['Seller_Latitude']=WarehouseHold.apply(lambda row:
            WarehouseHold['latitude'][i],axis=1)
    WarehouseHold['Seller_Longitude']=WarehouseHold.apply(lambda row:
            WarehouseLoop['longitude'][i],axis=1)

    WarehouseHold['Buyer']=WarehouseHold.apply(lambda row:
            'WH-' + row['postcode'],axis=1)

    WarehouseHold['Buyer_Latitude']=WarehouseHold.apply(lambda row:
```

```
            row['latitude'],axis=1)
    WarehouseHold['Buyer_Longitude']=WarehouseHold.apply(lambda row:
            row['longitude'],axis=1)

    WarehouseHold['Distance']=WarehouseHold.apply(lambda row: round(
       vincenty((WarehouseLoop['latitude'][i],WarehouseLoop['longitude'][i]),
            (row['latitude'],row['longitude'])).miles,6),axis=1)

    WarehouseHold.drop('id', axis=1, inplace=True)
    WarehouseHold.drop('postcode', axis=1, inplace=True)
    WarehouseHold.drop('latitude', axis=1, inplace=True)
    WarehouseHold.drop('longitude', axis=1, inplace=True)
    ##################################################################
    sFileLoopName=sFileDir + '/' + OutputLoopName
    WarehouseHold.to_csv(sFileLoopName, index = False)
##################################################################
print('### Done!! ##############################################')
##################################################################
```

Output:

```
====== RESTART: C:\VKHCG\03-Hillman\01-Retrieve\Retrieve-Route-Plan.py ======
################################
Working Base : C:/VKHCG  using  win32
################################
###########
Loading : C:/VKHCG/03-Hillman/00-RawData/GB_Postcode_Warehouse.csv
Run : 0  =======>>>>>>>>>>> AB10
Run : 1  =======>>>>>>>>>>> AB11
Run : 2  =======>>>>>>>>>>> AB12
Run : 3  =======>>>>>>>>>>> AB13
Run : 4  =======>>>>>>>>>>> AB14
Run : 5  =======>>>>>>>>>>> AB15
Run : 6  =======>>>>>>>>>>> AB16
Run : 7  =======>>>>>>>>>>> AB21
Run : 8  =======>>>>>>>>>>> AB22
Run : 9  =======>>>>>>>>>>> AB23
Run : 10  =======>>>>>>>>>>> AB24
Run : 19  =======>>>>>>>>>>> AB37
### Done!! ##############################################
>>>
```

See the collection of files similar in format to Retrieve_Route_WH-AB11_Route.csv in C:\VKHCG\03-Hillman\01-Retrieve\01-EDS\02-Python.

1	Transaction	Seller	Seller_Latitude	Seller_Longitude	Buyer	Buyer_Latitude	Buyer_Longitude	Distance
2	WH-to-WH	WH-AB11	57.13875	-2.09089	WH-AB10	57.13514	-2.11731	1.024915
3	WH-to-WH	WH-AB11	57.13875	-2.09089	WH-AB11	57.13875	-2.09089	0
4	WH-to-WH	WH-AB11	57.13875	-2.09089	WH-AB12	57.101	-2.1106	2.715503
5	WH-to-WH	WH-AB11	57.13875	-2.09089	WH-AB13	57.10801	-2.23776	5.922893

Global Post Codes

Open RStudio and use R to process the following R script:

Retrieve-Postcode-Global.r.

```
library(readr)
All_Countries <- read_delim("C:/VKHCG/03-Hillman/00-RawData/All_Countries.txt",
                    "\t", col_names = FALSE,
                    col_types = cols(
                      X12 = col_skip(),
                      X6 = col_skip(),
                      X7 = col_skip(),
                      X8 = col_skip(),
                      X9 = col_skip()),
                    na = "null", trim_ws = TRUE)
write.csv(All_Countries,
        file = "C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-
R/Retrieve_All_Countries.csv")
```

Output:

The program will successfully uploaded a new file named Retrieve_All_Countries.csv, after removing column No. 6, 7, 8, 9 and 12 from All_Countries.txt

	A	B	C	D	E	F	G	H
1		X1	X2	X3	X4	X5	X10	X11
2	1	x1	x2	x3	x4	x5	x10	x11
3	2	AD	AD100	Canillo			42.5833	1.6667
4	3	AD	AD200	Encamp			42.5333	1.6333
5	4	AD	AD300	Ordino			42.6	1.55
6	5	AD	AD400	La Massana			42.5667	1.4833
7	6	AD	AD500	Andorra la Vella			42.5	1.5
8	7	AD	AD600	Sant Juli<c3> de Lòria			42.4667	1.5
9	8	AD	AD700	Escaldes-Engordany			42.5	1.5667
10	9	AR	3636	POZO CERCADO (EL CHORRO (F), DPTO. RIVADAVIA (S))	Salta	A	-23.4933	-61.9267
11	10	AR	4123	LAS SALADAS	Salta	A	-25.7833	-64.5

Clark Ltd

Clark is the financial powerhouse of the group. It must process all the money-related data sources.

Forex-The first financial duty of the company is to perform any foreign exchange trading.

Forex Base Data-Previously, you found a single data source (Euro_ExchangeRates.csv) for forex rates in Clark. Earlier in the chapter, I helped you to create the load, as part of your R processing.

The relevant file is Retrieve_Retrieve_Euro_ExchangeRates.csv in directory C:\ VKHCG\04-Clark\01-Retrieve\01-EDS\01-R. So, that data is ready.

Financials - Clark generates the financial statements for all the group's companies.

Financial Base Data - You found a single data source (Profit_And_Loss.csv) in Clark for financials and, as mentioned previously, a single data source (Euro_ExchangeRates.csv) for forex rates. The file relevant file is Retrieve_Profit_And_Loss.csv in directory C:\VKHCG\04-Clark\01-Retrieve\ 01-EDS\01-R.

Person Base Data

Start Python editor and create a file named Retrieve-PersonData.py in directory . C:\VKHCG\04-Clark\01-Retrieve.

```
#######################################################################
# -*- coding: utf-8 -*-
```

```
import sys
import os
import shutil
import zipfile
import pandas as pd
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
Company='04-Clark'
ZIPFiles=['Data_female-names','Data_male-names','Data_last-names']
for ZIPFile in ZIPFiles:
    InputZIPFile=Base+'/'+Company+'/00-RawData/' + ZIPFile + '.zip'
    OutputDir=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/' + ZIPFile
    OutputFile=Base+'/'+Company+'/01-Retrieve/01-EDS/02-Python/Retrieve-'+ZIPFile+'.csv'
    zip_file = zipfile.ZipFile(InputZIPFile, 'r')
    zip_file.extractall(OutputDir)
    zip_file.close()
    t=0
    for dirname, dirnames, filenames in os.walk(OutputDir):
        for filename in filenames:
            sCSVFile = dirname + '/' + filename
            t=t+1
            if t==1:
                NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
                NameData=NameRawData
            else:
                NameRawData=pd.read_csv(sCSVFile,header=None,low_memory=False)
                NameData=NameData.append(NameRawData)
    NameData.rename(columns={0 : 'NameValues'},inplace=True)
    NameData.to_csv(OutputFile, index = False)
    shutil.rmtree(OutputDir)
    print('Process: ',InputZIPFile)
print('### Done!! ##############################################')
```

This generates three files named

Retrieve-Data_female-names.csv
Retrieve-Data_male-names.csv
Retrieve-Data_last-names.csv

Connecting to other Data Sources

A. Program to connect to different data sources.

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!! ##############################################')
```

```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
= RESTART: C:/VKHCG/03-Hillman/01-Retrieve/Retrieve-IP_DATA_ALL_2_SQLite.py =
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv
Storing : C:/VKHCG/01-Vermeulen/00-RawData/SQLite/vermeulen.db  Table: IP_DATA_ALL
Loading : C:/VKHCG/01-Vermeulen/00-RawData/SQLite/vermeulen.db  Table: IP_DATA_ALL
################
## Data Values
################
      RowIDCSV  RowID    ID  ... Longitude First.IP.Number Last.IP.Number
0            0      0     1  ...  -73.9725       204276480      204276735
1            1      1     2  ...  -73.9725       301984864      301985791
2            2      2     3  ...  -73.9725       404678736      404679039
3            3      3     4  ...  -73.9725       411592704      411592959
4            4      4     5  ...  -73.9725       416784384      416784639
...        ...    ...   ...  ...       ...             ...            ...
3557      3557   3557  3558  ...   11.5392      1591269504     1591269631
3558      3558   3558  3559  ...   11.7500      1558374784     1558374911
3559      3559   3559  3560  ...   11.4667      1480845312     1480845439
3560      3560   3560  3561  ...   11.7434      1480596992     1480597503
3561      3561   3561  3562  ...   11.7434      1558418432     1558418943

[3562 rows x 10 columns]
################
## Data Profile
################
Rows : 3562
Columns : 10
################
### Done!! ########################################
>>>
Ln: 107  Col: 4
```

MySQL:

Open MySql

Create a database “DataScience”

Create a python file and add the following code:

```
################ Connection With MySQL #######################
import mysql.connector

conn = mysql.connector.connect(host='localhost',
                               database='DataScience',
                               user='root',
                               password='root')
conn.connect
if(conn.is_connected):
   print('###### Connection With MySql Established Successfullly ##### ')
else:
   print('Not Connected -- Check Connection Properites')
```

```
>>>
 RESTART: C:/Users/User/AppData/Local/Programs/Python/Python37-32/mysqlconnection.py
###### Connection With MySql Established Successfullly #####
>>>
```

Microsoft Excel

```
##################Retrieve-Country-Currency.py
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import os
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:

```
                    Country                Currency CurrencyCode
1                Afghanistan            Afghan afghani        AFN
2   Akrotiri and Dhekelia (UK)           European euro        EUR
3      Aland Islands (Finland)           European euro        EUR
4                    Albania            Albanian lek        ALL
5                    Algeria          Algerian dinar        DZD
..                       ...                     ...        ...
271          Wake Island (USA)  United States dollar        USD
272  Wallis and Futuna (France)              CFP franc        XPF
274                      Yemen            Yemeni rial        YER
276                     Zambia          Zambian kwacha        ZMW
277                   Zimbabwe  United States dollar        USD

[253 rows x 3 columns]
~~~~~~ Data from Excel Sheet Retrived Successfully ~~~~~~~
>>>
```

Practical 5:
Assessing Data

Assess Superstep

Data quality refers to the condition of a set of qualitative or quantitative variables. Dataquality is a multidimensional measurement of the acceptability of specific data sets. Inbusiness, data quality is measured to determine whether data can be used as a basis forreliable intelligence extraction for supporting organizational decisions. Data profiling involves observing in your data sources all the viewpoints that theinformation offers. The main goal is to determine if individual viewpoints are accurateand complete. The Assess superstep determines what additional processing to apply tothe entries that are noncompliant.

Errors

Typically, one of four things can be done with an error to the data.

1. Accept the Error
2. Reject the Error
3. Correct the Error
4. Create a Default Value

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

Python pandas package enables several automatic error-management features.

File Location: C:\VKHCG\01-Vermeulen\02-Assess

Missing Values in Pandas:

i. Drop the Columns Where All Elements Are Missing Values

	A	B	C	D	E	F	G	H
1	ID	FieldA	FieldB	FieldC	FieldD	FieldE	FieldF	FieldG
2	1	Good	Better	Best	1024		10241	1
3	2	Good		Best	512		5121	2
4	3	Good	Better		256		256	3
5	4	Good	Better	Best			211	4
6	5	Good	Better		64		6411	5
7	6	Good		Best	32		32	6
8	7		Better	Best	16		1611	7
9	8			Best	8		8111	8
10	9				4		41	9
11	10	A	B	C	2		21111	10
12								11
13	10	Good	Better	Best	1024		102411	12
14	10	Good		Best	512		512	13
15	10	Good	Better		256		1256	14
16	10	Good	Better	Best				15
17	10	Good	Better		64		164	16
18	10	Good		Best	32		322	17
19	10		Better	Best	16		163	18
20	10			Best	8		844	19
21	10				4		4555	20
22	10	A	B	C	2		111	21

Code :

```
################### Assess-Good-Bad-01.py#########################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
#################################################################
Base='C:/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='C:/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:

```
>>>
```

```
======= RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-01.py
=======
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/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
```

All of column E has been deleted, owing to the fact that all values in that column were missing values/errors.

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

```
################## Assess-Good-Bad-02.py##########################
import sys
import os
import pandas as pd
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-02.csv'
Company='01-Vermeulen'
################################################################
Base='C:/VKHCG'
################################################################
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('################################')
    ##################################################################
>>>
======= RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-02.py
=======
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/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
################################
## Data Profile
################################
Rows : 21
Columns : 8
################################
################################
## Test Data Values
################################
   FieldG
0       1
1       2
################################
## Data Profile
################################
Rows : 21
Columns : 1
################################
################################
### Done!! ####################
################################
>>>
```

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

```
#####################    Assess-Good-Bad-03.py  ################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
#################################################################
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-03.csv'
Company='01-Vermeulen'
Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using Windows ~~~~')
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('##################################')
#################################################################
```

	A	B	C	D	E	F	G	H
1	ID	FieldA	FieldB	FieldC	FieldD	FieldE	FieldF	FieldG
2	1	Good	Better	Best	1024		10241	1
3	2	Good		Best	512		5121	2
4	3	Good	Better		256		256	3
5	4	Good	Better	Best			211	4
6	5	Good	Better		64		6411	5
7	6	Good		Best	32		32	6
8	7		Better	Best	16		1611	7
9	8			Best	8		8111	8
10	9				4		41	9
11	10	A	B	C	2		21111	10
12								11
13	10	Good	Better	Best	1024		102411	12
14	10	Good		Best	512		512	13
15	10	Good	Better		256		1256	14
16	10	Good	Better	Best				15
17	10	Good	Better		64		164	16
18	10	Good		Best	32		322	17
19	10		Better	Best	16		163	18
20	10			Best	8		844	19
21	10				4		4555	20
22	10	A	B	C	2		111	21

Before After

	A	B	C	D	E	F	G	H
1	ID	FieldA	FieldB	FieldC	FieldD	FieldE	FieldF	FieldG
2	1	Good	Better	Best	1024		10241	1
3	2	Good		Best	512		5121	2
4	3	Good	Better		256		256	3
5	4	Good	Better	Best			211	4
6	5	Good	Better		64		6411	5
7	6	Good		Best	32		32	6
8	7		Better	Best	16		1611	7
9	8			Best	8		8111	8
10	9				4		41	9
11	10	A	B	C	2		21111	10
12	10	Good	Better	Best	1024		102411	12
13	10	Good		Best	512		512	13
14	10	Good	Better		256		1256	14
15	10	Good	Better	Best				15
16	10	Good	Better		64		164	16
17	10	Good		Best	32		322	17
18	10		Better	Best	16		163	18
19	10			Best	8		844	19
20	10				4		4555	20
21	10	A	B	C	2		111	21

Row with more than two missing values got deleted.

The next step along the route is to generate a full network routing solution for the company, to resolve the data issues in the retrieve data.

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

```
########## Assess-Network-Routing-Company.py ####################
import sys
import os
import pandas as pd
#################################################################
pd.options.mode.chained_assignment = None
#################################################################
Base='C:/VKHCG'
#################################################################
print('##################################')
print('Working Base :',Base, ' using Windows')
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:
Go to C:\VKHCG\01-Vermeulen\02-Assess\01-EDS\02-Python folder and open Assess-Network-Routing-Company.csv

	A	B	C	D	E
1	ny_Countr	Company_Place_Name	Company_Latitude	Company_Longitude	Company_Country_Name
2	US	New York	40.7528	-73.9725	United States of America
3	US	New York	40.7214	-74.0052	United States of America
4	US	New York	40.7662	-73.9862	United States of America
5	US	New York	40.7449	-73.9782	United States of America
6	US	New York	40.7605	-73.9933	United States of America
7	US	New York	40.7588	-73.968	United States of America
8	US	New York	40.7637	-73.9727	United States of America
9	US	New York	40.7553	-73.9924	United States of America

Next, Access the the customers location using network router location.

```
####################Assess-Network-Routing-Customer.py #######################
import sys
import os
import pandas as pd
#################################################################
pd.options.mode.chained_assignment = None
#################################################################
Base='C:/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

Assess-Network-Routing-Customer.csv

	A	B	C	D	E
1	ıer_Countr	tomer_Place_Na	Customer_Latitude	Customer_Longitude	Customer_Country_Name
2	BW	Gaborone	-24.6464	25.9119	Botswana
3	BW	Francistown	-21.1667	27.5167	Botswana
4	BW	Maun	-19.9833	23.4167	Botswana
5	BW	Molepolole	-24.4167	25.5333	Botswana
6	NE	Niamey	13.5167	2.1167	Niger
7	MZ	Maputo	-25.9653	32.5892	Mozambique
8	MZ	Tete	-16.1564	33.5867	Mozambique
9	MZ	Quelimane	-17.8786	36.8883	Mozambique
10	MZ	Chimoio	-19.1164	33.4833	Mozambique
11	MZ	Matola	-25.9622	32.4589	Mozambique
12	MZ	Pemba	-12.9608	40.5078	Mozambique
13	MZ	Lichinga	-13.3128	35.2406	Mozambique
14	MZ	Maxixe	-23.8597	35.3472	Mozambique
15	MZ	Chibuto	-24.6867	33.5306	Mozambique
16	MZ	Ressano Garcia	-25.4428	31.9953	Mozambique
17	GH	Tema	5.6167	-0.0167	Ghana
18	GH	Kumasi	6.6833	-1.6167	Ghana
19	GH	Takoradi	4.8833	-1.75	Ghana
20	GH	Accra	5.55	-0.2167	Ghana

Assess-Network-Routing-Node.py

```
#################################################################
import sys
import os
import pandas as pd
#################################################################
pd.options.mode.chained_assignment = None
#################################################################
Base='C:/VKHCG'
#################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
#################################################################
sOutputFileName='Assess-Network-Routing-Node.csv'
Company='01-Vermeulen'
#################################################################
### Import IP Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
IPData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded IP :', IPData.columns.values)
print('##################################')
#################################################################
print('##################################')
print('Changed :',IPData.columns.values)
IPData.drop('RowID', axis=1, inplace=True)
IPData.drop('ID', axis=1, inplace=True)
IPData.rename(columns={'Country': 'Country_Code'}, inplace=True)
IPData.rename(columns={'Place.Name': 'Place_Name'}, inplace=True)
IPData.rename(columns={'Post.Code': 'Post_Code'}, inplace=True)
```

```
IPData.rename(columns={'First.IP.Number': 'First_IP_Number'}, inplace=True)
IPData.rename(columns={'Last.IP.Number': 'Last_IP_Number'}, inplace=True)
print('To :',IPData.columns.values)
print('################################')
#################################################################
print('################################')
print('Change ',IPData.columns.values)
for i in IPData.columns.values:
    j='Node_'+i
    IPData.rename(columns={i: j}, inplace=True)
print('To ', IPData.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('################################')
IPData.to_csv(sFileName, index = False, encoding="latin-1")
#################################################################
print('################################')
print('### Done!! ####################')
print('################################')
#################################################################
```

Output:
C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Node.csv

	A	B	C	D	E	F	G
1	Node_Country_Code	Node_Place_Name	Node_Post_Code	Node_Latitude	Node_Longitude	ode_First_IP_Numb	Node_Last_IP_Number
2	BW	Gaborone		-24.6464	25.9119	692781056	692781567
3	BW	Gaborone		-24.6464	25.9119	692781824	692783103
4	BW	Gaborone		-24.6464	25.9119	692909056	692909311
5	BW	Gaborone		-24.6464	25.9119	692909568	692910079
6	BW	Gaborone		-24.6464	25.9119	693051392	693052415
7	BW	Gaborone		-24.6464	25.9119	693078272	693078527
8	BW	Gaborone		-24.6464	25.9119	693608448	693616639
9	BW	Gaborone		-24.6464	25.9119	696929792	696930047
10	BW	Gaborone		-24.6464	25.9119	700438784	700439039
11	BW	Gaborone		-24.6464	25.9119	702075904	702076927
12	BW	Gaborone		-24.6464	25.9119	702498816	702499839
13	BW	Gaborone		-24.6464	25.9119	702516224	702517247
14	BW	Gaborone		-24.6464	25.9119	774162663	774162667
15	BW	Gaborone		-24.6464	25.9119	1401887232	1401887743
16	BW	Gaborone		-24.6464	25.9119	1754209024	1754209279
17	NE	Niamey		13.5167	2.1167	696918528	696919039
18	NE	Niamey		13.5167	2.1167	696922112	696924159
19	NE	Niamey		13.5167	2.1167	701203456	701203711
20	NE	Niamey		13.5167	2.1167	758886912	758887167
21	NE	Niamey		13.5167	2.1167	1347294153	1347294160
22	NE	Niamey		13.5167	2.1167	1755108096	1755108351
23	NE	Niamey		13.5167	2.1167	1755828480	1755828735
24	MZ	Maputo		-25.9653	32.5892	692883456	692883967
25	MZ	Maputo		-25.9653	32.5892	692944896	692946943

Directed Acyclic Graph (DAG)
A directed acyclic graph is a specific graph that only has one path through the graph.

C. Write a Python / R program to build directed acyclic graph.

Open your python editor and create a file named Assess-DAG-Location.py in directory C:\VKHCG\01-Vermeulen\02-Assess

```
##################################################################
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
##################################################################
Base='C:/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),
        nodecolor='r',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),
        nodecolor='r',edge_color='b',
        with_labels=True,node_size=8000,
        font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####################################################################
```

Output:

```
##################################
Rows :  150
##################################
##################################
Link : US  to  DE
Link : US  to  GB
Link : DE  to  US
Link : DE  to  GB
Link : GB  to  US
Link : GB  to  DE
##################################
##################################
Nodes of graph:
['US', 'DE', 'GB']
Edges of graph:
[('US', 'DE'), ('US', 'GB'), ('DE', 'US'), ('DE', 'GB'), ('GB', 'US'), ('GB', 'DE')]
##################################
```

Customer Location DAG

```
################### Assess-DAG-Location.py###################
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
#################################################################
Base='C:/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),
        nodecolor='r',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),
        nodecolor='r',edge_color='b',
        with_labels=True,node_size=8000,
        font_size=12)
plt.savefig(sFileName) # save as png
```

```
plt.show() # display
#####################################################################
```

Output:

```
#################################
Link : New York-US  to  Munich-DE
Link : New York-US  to  London-GB
Link : Munich-DE  to  New York-US
Link : Munich-DE  to  London-GB
Link : London-GB  to  New York-US
Link : London-GB  to  Munich-DE
#################################
#################################
Nodes of graph:
['New York-US', 'Munich-DE', 'London-GB']
Edges of graph:
[('New York-US', 'Munich-DE'), ('New York-US', 'London-GB'), ('Munich-DE', 'New York-US'), ('Munich-DE', 'London-GB'), ('London-GB', 'New York-US'), ('London-GB', 'Munich-DE')]
```

Open your Python editor and create a file named Assess-DAG-GPS.py in directory C:\VKHCG\01-Vermeulen\02-Assess.

```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName='Assess-DAG-Company-GPS.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('################################')
G=nx.Graph()
for i in range(CompanyData.shape[0]):
    nLatitude=round(CompanyData['Latitude'][i],2)
    nLongitude=round(CompanyData['Longitude'][i],2)

    if nLatitude < 0:
        sLatitude = str(nLatitude*-1) + ' S'
    else:
        sLatitude = str(nLatitude) + ' N'

    if nLongitude < 0:
        sLongitude = str(nLongitude*-1) + ' W'
    else:
        sLongitude = str(nLongitude) + ' E'

    sGPS= sLatitude + '-' + sLongitude
    G.add_node(sGPS)

print('################################')
for n1 in G.nodes():
    for n2 in G.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G.add_edge(n1,n2)
print('################################')

print('################################')
print("Nodes of graph: ")
print(G.number_of_nodes())
print("Edges of graph: ")
print(G.number_of_edges())
print('################################')
```

Output:

```
=== RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-DAG-GPS-unsmoothed.py
===
################################
Working Base : C:/VKHCG  using  win32
################################
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-
Python/Retrieve_Router_Location.csv
################################
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
################################
```

```
   Country Place_Name  Latitude  Longitude
0       US   New York   40.7528   -73.9725
1       US   New York   40.7214   -74.0052
-
-
```

D. Write a Python / R program to pick the content for Bill Boards from the given data.

Picking Content for Billboards

The basic process required is to combine two sets of data and then calculate the number of visitors per day from the range of IP addresses that access the billboards in Germany.

Bill Board Location: Rows - 8873

Access Visitors: Rows - 75999

Access Location Record: Rows – 1,81,235

Open Python editor and create a file named **Assess-DE-Billboard.py** in directory C:\VKHCG\02-Krennwallner\02-Assess

```
################# Assess-DE-Billboard.py######################
import sys
import os
import sqlite3 as sq
import pandas as pd
#################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#################################################################
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
sOutputFileName='Assess-DE-Billboard-Visitor.csv'
Company='02-Krennwallner'
#################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
```

```
##################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
##################################################################
### Import Billboard Data
##################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('################################')
print('Loading :',sFileName)
print('################################')
BillboardRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-
1")
BillboardRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BillboardData=BillboardRawData
print('Loaded Company :',BillboardData.columns.values)
print('################################')
##################################################################
print('################')
sTable='Assess_BillboardData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardData.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
print(BillboardData.head())
print('################################')
print('Rows : ',BillboardData.shape[0])
print('################################')
##################################################################
### Import Billboard Data
##################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName)
print('################################')
VisitorRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData[VisitorRawData.Country=='DE']
print('Loaded Company :',VisitorData.columns.values)
print('################################')
##################################################################
print('################')
sTable='Assess_VisitorData'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
print(VisitorData.head())
print('################################')
print('Rows : ',VisitorData.shape[0])
print('################################')
```

```
#################################################################
print('################')
sTable='Assess_BillboardVisitorData'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.Country AS BillboardCountry,"
sSQL=sSQL+ " A.Place_Name AS BillboardPlaceName,"
sSQL=sSQL+ " A.Latitude AS BillboardLatitude, "
sSQL=sSQL+ " A.Longitude AS BillboardLongitude,"
sSQL=sSQL+ " B.Country AS VisitorCountry,"
sSQL=sSQL+ " B.Place_Name AS VisitorPlaceName,"
sSQL=sSQL+ " B.Latitude AS VisitorLatitude, "
sSQL=sSQL+ " B.Longitude AS VisitorLongitude,"
sSQL=sSQL+ " (B.Last_IP_Number - B.First_IP_Number) * 365.25 * 24 * 12 AS
VisitorYearRate"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardData as A"
sSQL=sSQL+ " JOIN "
sSQL=sSQL+ " Assess_VisitorData as B"
sSQL=sSQL+ " ON "
sSQL=sSQL+ " A.Country = B.Country"
sSQL=sSQL+ " AND "
sSQL=sSQL+ " A.Place_Name = B.Place_Name;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn)
print('################')
#################################################################
print('################')
sTable='Assess_BillboardVistorsData'
print('Storing :',sDatabaseName,' Table:',sTable)
BillboardVistorsData.to_sql(sTable, conn, if_exists="replace")
print('################')
#################################################################
print(BillboardVistorsData.head())
print('################################')
print('Rows : ',BillboardVistorsData.shape[0])
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
print('################################')
print('Storing :', sFileName)
print('################################')
sFileName=sFileDir + '/' + sOutputFileName
BillboardVistorsData.to_csv(sFileName, index = False)
print('################################')
#################################################################
print('### Done!! ##########################################')
#################################################################
```

Output:
C:\VKHCG\02-Krennwallner\01-Retrieve\01-EDS\02-Python\Retrieve_Online_Visitor.csv containing, 10,48,576**(Ten lack Forty Eight Thousand Five Hundred and Seventy Six)**rows.

	A	B	C	D	E	F
1	Country	Place_Name	Latitude	Longitude	First_IP_Number	Last_IP_Number
2	BW	Gaborone	-24.6464	25.9119	692781056	692781567
3	BW	Gaborone	-24.6464	25.9119	692781824	692783103
4	BW	Gaborone	-24.6464	25.9119	692909056	692909311
1048556	NL	Amsterdam	52.3556	4.9136	385939968	385940479
1048557	NL	Amsterdam	52.3556	4.9136	385942528	385943551
1048558	NL	Amsterdam	52.3556	4.9136	385957888	385961983
1048559	NL	Amsterdam	52.3556	4.9136	386003200	386003967
1048560	NL	Amsterdam	52.3556	4.9136	386012160	386012671
1048561	NL	Amsterdam	52.3556	4.9136	386013184	386013695
1048562	NL	Amsterdam	52.3556	4.9136	386015232	386015487
1048563	NL	Amsterdam	52.3556	4.9136	386020352	386021375
1048564	NL	Amsterdam	52.3556	4.9136	386035712	386039807
1048565	NL	Amsterdam	52.3556	4.9136	386060288	386068479
1048566	NL	Amsterdam	52.3556	4.9136	386073344	386073599
1048567	NL	Amsterdam	52.3556	4.9136	386074112	386074623
1048568	NL	Amsterdam	52.3556	4.9136	386076416	386076671
1048569	NL	Amsterdam	52.3556	4.9136	386088960	386089983
1048570	NL	Amsterdam	52.3556	4.9136	386095616	386096127
1048571	NL	Amsterdam	52.3556	4.9136	386109440	386113535
1048572	NL	Amsterdam	52.3556	4.9136	386191360	386195455
1048573	NL	Amsterdam	52.3556	4.9136	386201600	386203135
1048574	NL	Amsterdam	52.3556	4.9136	386215936	386220031
1048575	NL	Amsterdam	52.3556	4.9136	386228224	386232319
1048576	NL	Amsterdam	52.3556	4.9136	386244608	386244863

Retrieve_Online_Visitor

SQLite Visitor's Database
C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db Table:

	BillboardCountry	BillboardPlaceName	...	VisitorLongitude	VisitorYearRate
0	DE	Lake	...	8.5667	26823960.0
1	DE	Horb	...	8.6833	26823960.0
2	DE	Horb	...	8.6833	53753112.0
3	DE	Horb	...	8.6833	107611416.0
4	DE	Horb	...	8.6833	13359384.0

	A	B	C	D	E	F	G	H	I
1	illboardCountr	BillboardPlaceName	oardLatit	oardLongi	itorCount	VisitorPlaceName	itorLatitu	itorLongitu	VisitorYearRate
2	DE	Lake	51.7833	8.5667	DE	Lake	51.7833	8.5667	26823960
3	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	26823960
4	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	53753112
5	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	107611416
6	DE	Horb	48.4333	8.6833	DE	Horb	48.4333	8.6833	13359384
7	DE	Horb	48.4333	8.6833	DE	Horb	48.4889	8.6734	26823960
8	DE	Horb	48.4333	8.6833	DE	Horb	48.4889	8.6734	53753112
9	DE	Hardenberg	51.1	7.7333	DE	Hardenberg	51.1	7.7333	26823960
181221	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1167	8.6833	1157112
181222	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1167	8.6833	24299352
181223	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1167	8.6833	807769368
181224	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	53753112
181225	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	26823960
181226	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	107611416
181227	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1172	8.7281	1577880
181228	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1184	8.6095	15042456
181229	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1184	8.6095	10834776
181230	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1319	8.6838	736344
181231	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1319	8.6838	0
181232	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1327	8.7668	736344
181233	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1492	8.7097	1723360536
181234	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1492	8.7097	430761240
181235	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1528	8.745	26823960
181236	DE	Frankfurt	50.1327	8.7668	DE	Frankfurt	50.1878	8.6632	1577880

Assess-DE-Billboard-Visitor

E. Write a Python / R program to generate GML file from the given csv file.
Understanding Your Online Visitor Data
Online visitors have to be mapped to their closest billboard, to ensure we understand where and what they can access.

Open your Python editor and create a file called Assess-Billboard_2_Visitor.py in directory
C:\VKHCG\ 02-Krennwallner\02-Assess.

```
import networkx as nx
import sys
import os
import sqlite3 as sq
import pandas as pd
from geopy.distance import vincenty
#################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
Company='02-Krennwallner'
sTable='Assess_BillboardVisitorData'
sOutputFileName='Assess-DE-Billboard-Visitor.gml'
#################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
#################################################################
print('################')
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select "
sSQL=sSQL+ " A.BillboardCountry,"
sSQL=sSQL+ " A.BillboardPlaceName,"
sSQL=sSQL+ " ROUND(A.BillboardLatitude,3) AS BillboardLatitude, "
sSQL=sSQL+ " ROUND(A.BillboardLongitude,3) AS BillboardLongitude,"
sSQL=sSQL+ " (CASE WHEN A.BillboardLatitude < 0 THEN "
sSQL=sSQL+ "  'S' || ROUND(ABS(A.BillboardLatitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'N' || ROUND(ABS(A.BillboardLatitude),3)"
sSQL=sSQL+ " END ) AS sBillboardLatitude,"
sSQL=sSQL+ " (CASE WHEN A.BillboardLongitude < 0 THEN "
sSQL=sSQL+ "  'W' || ROUND(ABS(A.BillboardLongitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'E' || ROUND(ABS(A.BillboardLongitude),3)"
sSQL=sSQL+ " END ) AS sBillboardLongitude,"
sSQL=sSQL+ " A.VisitorCountry,"
sSQL=sSQL+ " A.VisitorPlaceName,"
```

```
sSQL=sSQL+ " ROUND(A.VisitorLatitude,3) AS VisitorLatitude, "
sSQL=sSQL+ " ROUND(A.VisitorLongitude,3) AS VisitorLongitude,"
sSQL=sSQL+ " (CASE WHEN A.VisitorLatitude < 0 THEN "
sSQL=sSQL+ "  'S' || ROUND(ABS(A.VisitorLatitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'N' ||ROUND(ABS(A.VisitorLatitude),3)"
sSQL=sSQL+ " END ) AS sVisitorLatitude,"
sSQL=sSQL+ " (CASE WHEN A.VisitorLongitude < 0 THEN "
sSQL=sSQL+ "  'W' || ROUND(ABS(A.VisitorLongitude),3)"
sSQL=sSQL+ " ELSE "
sSQL=sSQL+ "  'E' || ROUND(ABS(A.VisitorLongitude),3)"
sSQL=sSQL+ " END ) AS sVisitorLongitude,"
sSQL=sSQL+ " A.VisitorYearRate"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_BillboardVistorsData AS A;"
BillboardVistorsData=pd.read_sql_query(sSQL, conn)
print('################')
#################################################################
BillboardVistorsData['Distance']=BillboardVistorsData.apply(lambda row:
   round(
   vincenty((row['BillboardLatitude'],row['BillboardLongitude']),
            (row['VisitorLatitude'],row['VisitorLongitude'])).miles
         ,4)
     ,axis=1)
#################################################################
G=nx.Graph()
#################################################################
for i in range(BillboardVistorsData.shape[0]):
   sNode0='MediaHub-' + BillboardVistorsData['BillboardCountry'][i]
   sNode1='B-'+ BillboardVistorsData['sBillboardLatitude'][i] + '-'
   sNode1=sNode1 + BillboardVistorsData['sBillboardLongitude'][i]
   G.add_node(sNode1,
           Nodetype='Billboard',
           Country=BillboardVistorsData['BillboardCountry'][i],
           PlaceName=BillboardVistorsData['BillboardPlaceName'][i],
           Latitude=round(BillboardVistorsData['BillboardLatitude'][i],3),
           Longitude=round(BillboardVistorsData['BillboardLongitude'][i],3))

   sNode2='M-'+ BillboardVistorsData['sVisitorLatitude'][i] + '-'
   sNode2=sNode2 + BillboardVistorsData['sVisitorLongitude'][i]
   G.add_node(sNode2,
           Nodetype='Mobile',
           Country=BillboardVistorsData['VisitorCountry'][i],
           PlaceName=BillboardVistorsData['VisitorPlaceName'][i],
           Latitude=round(BillboardVistorsData['VisitorLatitude'][i],3),
           Longitude=round(BillboardVistorsData['VisitorLongitude'][i],3))

   print('Link Media Hub :',sNode0,' to Billboard : ', sNode1)
   G.add_edge(sNode0,sNode1)
   print('Link Post Code :',sNode1,' to GPS : ', sNode2)
```

```
    G.add_edge(sNode1,sNode2,distance=round(BillboardVistorsData['Distance'][i]))
##################################################################
print('#################################')
print("Nodes of graph: ",nx.number_of_nodes(G))
print("Edges of graph: ",nx.number_of_edges(G))
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('#################################')
nx.write_gml(G,sFileName)
sFileName=sFileName +'.gz'
nx.write_gml(G,sFileName)
##################################################################
##################################################################
print('### Done!! ##############################################')
##################################################################
```

Output:

This will produce a set of demonstrated values onscreen, plus a graph data file named Assess-DE-Billboard-Visitor.gml.

(It takes a long time to complete the process, after completion the gml file can be viewed in text editor)

Hence, we have applied formulae to extract features, such as the distance between the billboard and the visitor.

Planning an Event for Top-Ten Customers

Open Python editor and create a file named Assess-Visitors.py in directory C:\VKHCG\02-Krennwallner\02-Assess

```
##################################################################
import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
##################################################################
Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
##################################################################
Company='02-Krennwallner'
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Online_Visitor.csv'
##################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
```

```
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/krennwallner.db'
conn = sq.connect(sDatabaseName)
#################################################################
### Import Country Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#################################')
print('Loading :',sFileName)
print('#################################')
VisitorRawData=pd.read_csv(sFileName,
                           header=0,
                           low_memory=False,
                           encoding="latin-1",
                           skip_blank_lines=True)
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData=VisitorRawData
print('Loaded Company :',VisitorData.columns.values)
print('#################################')
#################################################################
print('################')
sTable='Assess_Visitor'
print('Storing :',sDatabaseName,' Table:',sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('################')
#################################################################
print(VisitorData.head())
print('#################################')
print('Rows : ',VisitorData.shape[0])
print('#################################')
#################################################################
print('################')
sView='Assess_Visitor_UseIt'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " A.Country,"
sSQL=sSQL+ " A.Place_Name,"
sSQL=sSQL+ " A.Latitude,"
sSQL=sSQL+ " A.Longitude,"
sSQL=sSQL+ " (A.Last_IP_Number - A.First_IP_Number) AS UsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor as A"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " Country is not null"
sSQL=sSQL+ " AND"
```

```
sSQL=sSQL+ " Place_Name is not null;"
sql.execute(sSQL,conn)
#################################################################
print('################')
sView='Assess_Total_Visitors_Location'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Country,"
sSQL=sSQL+ " Place_Name,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt"
sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Country,"
sSQL=sSQL+ " Place_Name"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC"
sSQL=sSQL+ " LIMIT 10;"
sql.execute(sSQL,conn)
#################################################################
print('################')
sView='Assess_Total_Visitors_GPS'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " Latitude,"
sSQL=sSQL+ " Longitude,"
sSQL=sSQL+ " SUM(UsesIt) AS TotalUsesIt"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Visitor_UseIt"
sSQL=sSQL+ " GROUP BY"
sSQL=sSQL+ " Latitude,"
sSQL=sSQL+ " Longitude"
sSQL=sSQL+ " ORDER BY"
sSQL=sSQL+ " TotalUsesIt DESC"
sSQL=sSQL+ " LIMIT 10;"
sql.execute(sSQL,conn)
#################################################################
sTables=['Assess_Total_Visitors_Location', 'Assess_Total_Visitors_GPS']
for sTable in sTables:
    print('################')
    print('Loading :',sDatabaseName,' Table:',sTable)
    sSQL=" SELECT "
```

```
    sSQL=sSQL+ " *"
    sSQL=sSQL+ " FROM"
    sSQL=sSQL+ " " + sTable + ";"
    TopData=pd.read_sql_query(sSQL, conn)
    print('################')
    print(TopData)
    print('################')
    print('#################################')
    print('Rows : ',TopData.shape[0])
    print('#################################')
##################################################################
print('### Done!! ##############################################')
##################################################################
```

Output:

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
****************
Loading : C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db  Table: Assess_Total_Visitors_Location
****************
  Country     Place_Name  TotalUsesIt
0      CN        Beijing     53139475
1      US      Palo Alto     33682341
2      US  Fort Huachuca     33472427
3      JP          Tokyo     31404799
4      US      Cambridge     25598851
5      US      San Diego     17751367
6      CN      Guangzhou     17563744
7      US         Newark     17270604
8      US        Raleigh     17167484
9      US         Durham     16914033
****************
********************************
Rows :  10
********************************
****************
Loading : C:/VKHCG/02-Krennwallner/02-Assess/SQLite/krennwallner.db  Table: Assess_Total_Visitors_GPS
****************
   Latitude  Longitude  TotalUsesIt
0   39.9289   116.3883     53139732
1   37.3762  -122.1826     33551404
2   31.5273  -110.3607     33472427
3   35.6427   139.7677     31439772
4   23.1167   113.2500     17577053
5   42.3646   -71.1028     16890698
6   40.7355   -74.1741     16813373
7   42.3223   -83.1763     16777212
8   35.7977   -78.6253     16761084
9   32.8072  -117.1649     16747680
****************
```

F. Write a Python / R program to plan the locations of the warehouses from the given data.

Planning the Locations of the Warehouses

Planning the location of the warehouses requires the assessment of the GPS locations of these warehouses against the requirements for Hillman's logistics needs.

Open your editor and create a file named Assess-Warehouse-Address.py in directory C:\VKHCG\03-Hillman\02-Assess.

```
################### Assess-Warehouse-Address.py  ####################
import os
import pandas as pd
from geopy.geocoders import Nominatim
geolocator = Nominatim()
##################################################################
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_GB_Postcode_Warehouse.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_GB_Warehouse_Address.csv'
Company='03-Hillman'
##################################################################
Base='C:/VKHCG'
print('################################')
```

```
print('Working Base :',Base, ' using Windows')
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('###########')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False)
Warehouse.sort_values(by='postcode', ascending=1)
#################################################################
## Limited to 10 due to service limit on Address Service.
#################################################################
WarehouseGoodHead=Warehouse[Warehouse.latitude != 0].head(5)
WarehouseGoodTail=Warehouse[Warehouse.latitude != 0].tail(5)
#################################################################
WarehouseGoodHead['Warehouse_Point']=WarehouseGoodHead.apply(lambda row:
         (str(row['latitude'])+','+str(row['longitude']))
         ,axis=1)
WarehouseGoodHead['Warehouse_Address']=WarehouseGoodHead.apply(lambda row:
         geolocator.reverse(row['Warehouse_Point']).address
        ,axis=1)
WarehouseGoodHead.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodHead.drop('id', axis=1, inplace=True)
WarehouseGoodHead.drop('postcode', axis=1, inplace=True)
#################################################################
WarehouseGoodTail['Warehouse_Point']=WarehouseGoodTail.apply(lambda row:
         (str(row['latitude'])+','+str(row['longitude']))
         ,axis=1)
WarehouseGoodTail['Warehouse_Address']=WarehouseGoodTail.apply(lambda row:
         geolocator.reverse(row['Warehouse_Point']).address
        ,axis=1)
WarehouseGoodTail.drop('Warehouse_Point', axis=1, inplace=True)
WarehouseGoodTail.drop('id', axis=1, inplace=True)
WarehouseGoodTail.drop('postcode', axis=1, inplace=True)
#################################################################
WarehouseGood=WarehouseGoodHead.append(WarehouseGoodTail, ignore_index=True)
print(WarehouseGood)
#################################################################
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)
##################################################################
print('### Done!! ##############################################')
##################################################################
```

Output:

```
################################
Working Base : C:/VKHCG  using Windows
################################
###########
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcode_Warehouse.csv
    latitude  longitude                                Warehouse_Address
0  57.135140  -2.117310  35, Broomhill Road, Broomhill, Aberdeen, Aberd...
1  57.138750  -2.090890  South Esplanade West, Torry, Aberdeen, Aberdee...
2  57.101000  -2.110600  A92, Cove and Altens, Aberdeen, Aberdeen City,...
3  57.108010  -2.237760  Colthill Circle, Milltimber, Countesswells, Ab...
4  57.100760  -2.270730  Johnston Gardens East, Peterculter, South Last...
5  53.837717  -1.780013  HM Revenue and Customs, Riverside Estate, Temp...
6  53.794470  -1.766539  Listerhills Road Norcroft Street, Listerhills ...
7  51.518556  -0.714794  Sorting Office, Stafferton Way, Fishery, Maide...
8  54.890923  -2.943847  Royal Mail (Delivery Office), Junction Street,...
9  57.481338  -4.223951  Inverness Sorting & Delivery Office, Strothers...
### Done!! ##########################################
>>> |
```

G. Write a Python / R program using data science via clustering to determine new warehouses using the given data.

Global New Warehouse: Hillman wants to add extra global warehouses, and you are required to assess wherethey should be located. We only have to collect the possible locations for warehouses.

The following example will show you how to modify the data columns you read in that are totally ambiguous. Open Python editor and create a file named Assess-Warehouse-Global.py in directory
C:\VKHCG\03-Hillman\02-Assess

```
################# Assess-Warehouse-Global.py##############
import sys
import os
import pandas as pd
#################################################################
Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
Company='03-Hillman'
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_All_Countries.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_All_Warehouse.csv'
#################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
```

```
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('###########')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#################################################################
sColumns={'X1' : 'Country',
        'X2' : 'PostCode',
        'X3' : 'PlaceName',
        'X4' : 'AreaName',
        'X5' : 'AreaCode',
        'X10' : 'Latitude',
        'X11' : 'Longitude'}
Warehouse.rename(columns=sColumns,inplace=True)
WarehouseGood=Warehouse
#################################################################
sFileName=sFileDir + '/' + OutputFileName
WarehouseGood.to_csv(sFileName, index = False)
##################################################################
print('### Done!! ##############################################')
##################################################################
```

This will produce a set of demonstrated values onscreen, plus a graph data file named Assess_All_Warehouse.csv.

Output:

```
>>>
===== RESTART: C:\VKHCG\03-Hillman\02-Assess\Assess-Warehouse-Global.py =====
################################
Working Base : C:/VKHCG  using  win32
################################
###########
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_All_Countries.csv
### Done!! ##########################################
>>>
                                                                    Ln: 33  Col: 4
```

Open Assess0_All_Warehose.csv from C:\VKHCG\03-Hillman\02-Assess\01-EDS\02-Python

	A	B	C	D	E	F	G	H
1	Unnamed: 0	Country	PostCode	PlaceName	AreaName	AreaCode	Latitude	Longitude
2	1	AD	AD100	Canillo			42.5833	1.6667
3	2	AD	AD200	Encamp			42.5333	1.6333
4	3	AD	AD300	Ordino			42.6	1.55
5	4	AD	AD400	La Massana			42.5667	1.4833
6	5	AD	AD500	Andorra la Vella			42.5	1.5
31621	31620	AT	4925	Gumpling	OberÃ¶sterreich	4	48.1555	13.4802
31622	31621	AT	4925	Windischhub	OberÃ¶sterreich	4	48.1555	13.4802
31623	31622	AT	4926	Obereselbach	OberÃ¶sterreich	4	48.1917	13.5784
31624	31623	AT	4926	Jetzing	OberÃ¶sterreich	4	48.1555	13.4802
31625	31624	AT	4926	Pilgersham	OberÃ¶sterreich	4	48.1772	13.5855
31626	31625	AT	4926	Grausgrub	OberÃ¶sterreich	4	48.1555	13.4802
31627	31626	AT	4926	Marienkirchen am Ha	OberÃ¶sterreich	4	48.1828	13.577
31628	31627	AT	4926	Stocket	OberÃ¶sterreich	4	48.1555	13.4802
31629	31628	AT	4926	Baching	OberÃ¶sterreich	4	48.1555	13.4802
31630	31629	AT	4926	Kern	OberÃ¶sterreich	4	48.1555	13.4802
31631	31630	AT	4926	Manaberg	OberÃ¶sterreich	4	48.1555	13.4802
31632	31631	AT	4926	Untereselbach	OberÃ¶sterreich	4	48.1555	13.4802
31633	31632	AT	4926	Hatting	OberÃ¶sterreich	4	48.1555	13.4802
31634	31633	AT	4926	Unering	OberÃ¶sterreich	4	48.1555	13.4802
31635	31634	AT	4926	Kleinbach	OberÃ¶sterreich	4	48.1555	13.4802
31636	31635	AT	4926	Lehen	OberÃ¶sterreich	4	48.1555	13.4802
31637	31636	AT	4926	Hof	OberÃ¶sterreich	4	48.1555	13.4802
31638	31637	AT	4931	GroÃŸweiffendorf	OberÃ¶sterreich	4	48.15	13.3333
31639	31638	AT	4931	Neulendt	OberÃ¶sterreich	4	48.1697	13.3531

Assess_All_Warehouse

H. Using the given data, write a Python / R program to plan the shipping routes for best-fit international logistics.

Hillman requires an international logistics solution to support all the required shippingroutes.

Open Python editor and create a file named Assess-Best-Fit-Logistics.py in directory C:\VKHCG\03-Hillman\02-Assess

```
import sys
import os
import pandas as pd
import networkx as nx
from geopy.distance import vincenty
import sqlite3 as sq
from pandas.io import sql
##################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
##################################################################
Company='03-Hillman'
InputDir='01-Retrieve/01-EDS/01-R'
InputFileName='Retrieve_All_Countries.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_Best_Logistics.gml'
##################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
```

```
##################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
##################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
##################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn = sq.connect(sDatabaseName)
##################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName
print('###########')
print('Loading :',sFileName)
Warehouse=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
##################################################################
sColumns={'X1' : 'Country',
        'X2' : 'PostCode',
        'X3' : 'PlaceName',
        'X4' : 'AreaName',
        'X5' : 'AreaCode',
        'X10' : 'Latitude',
        'X11' : 'Longitude'}
Warehouse.rename(columns=sColumns,inplace=True)
WarehouseGood=Warehouse
#print(WarehouseGood.head())
##################################################################
RoutePointsCountry=pd.DataFrame(WarehouseGood.groupby(['Country'])[['Latitude','Longitude']].mean())
#print(RoutePointsCountry.head())
print('################')
sTable='Assess_RoutePointsCountry'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsCountry.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
RoutePointsPostCode=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode'])[['Latitude','Longitude']].mean())
#print(RoutePointsPostCode.head())
print('################')
sTable='Assess_RoutePointsPostCode'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPostCode.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
RoutePointsPlaceName=pd.DataFrame(WarehouseGood.groupby(['Country',
'PostCode','PlaceName'])[['Latitude','Longitude']].mean())
#print(RoutePointsPlaceName.head())
print('################')
sTable='Assess_RoutePointsPlaceName'
print('Storing :',sDatabaseName,' Table:',sTable)
RoutePointsPlaceName.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
### Fit Country to Country
##################################################################
print('################')
sView='Assess_RouteCountries'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
```

```
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry,"
sSQL=sSQL+ "   S.Latitude AS SourceLatitude,"
sSQL=sSQL+ "   S.Longitude AS SourceLongitude,"
sSQL=sSQL+ "   T.Country AS TargetCountry,"
sSQL=sSQL+ "   T.Latitude AS TargetLatitude,"
sSQL=sSQL+ "   T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS S"
sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS T"
sSQL=sSQL+ " WHERE S.Country <> T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)

print('################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
RouteCountries=pd.read_sql_query(sSQL, conn)

RouteCountries['Distance']=RouteCountries.apply(lambda row:
   round(
   vincenty((row['SourceLatitude'],row['SourceLongitude']),
            (row['TargetLatitude'],row['TargetLongitude'])).miles,4),axis=1)

print(RouteCountries.head(5))
#################################################################
### Fit Country to Post Code
#################################################################
print('################')
sView='Assess_RoutePostCode'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry,"
sSQL=sSQL+ "   S.Latitude AS SourceLatitude,"
sSQL=sSQL+ "   S.Longitude AS SourceLongitude,"
sSQL=sSQL+ "   T.Country AS TargetCountry,"
sSQL=sSQL+ "   T.PostCode AS TargetPostCode,"
sSQL=sSQL+ "   T.Latitude AS TargetLatitude,"
sSQL=sSQL+ "   T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsCountry AS S"
sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsPostCode AS T"
sSQL=sSQL+ " WHERE S.Country = T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
```

```
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)

print('################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
RoutePostCode=pd.read_sql_query(sSQL, conn)

RoutePostCode['Distance']=RoutePostCode.apply(lambda row:
   round(
   vincenty((row['SourceLatitude'],row['SourceLongitude']),
            (row['TargetLatitude'],row['TargetLongitude'])).miles
         ,4)
     ,axis=1)

print(RoutePostCode.head(5))
#################################################################
### Fit Post Code to Place Name
#################################################################
print('################')
sView='Assess_RoutePlaceName'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ "   S.Country AS SourceCountry,"
sSQL=sSQL+ "   S.PostCode AS SourcePostCode,"
sSQL=sSQL+ "   S.Latitude AS SourceLatitude,"
sSQL=sSQL+ "   S.Longitude AS SourceLongitude,"
sSQL=sSQL+ "   T.Country AS TargetCountry,"
sSQL=sSQL+ "   T.PostCode AS TargetPostCode,"
sSQL=sSQL+ "   T.PlaceName AS TargetPlaceName,"
sSQL=sSQL+ "   T.Latitude AS TargetLatitude,"
sSQL=sSQL+ "   T.Longitude AS TargetLongitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ "   Assess_RoutePointsPostCode AS S"
sSQL=sSQL+ "   ,"
sSQL=sSQL+ "   Assess_RoutePointsPLaceName AS T"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " S.Country = T.Country"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.PostCode = T.PostCode"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " S.Country in ('GB','DE','BE','AU','US','IN')"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " T.Country in ('GB','DE','BE','AU','US','IN');"
sql.execute(sSQL,conn)

print('################')
print('Loading :',sDatabaseName,' Table:',sView)
sSQL=" SELECT "
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sView + ";"
```

```
RoutePlaceName=pd.read_sql_query(sSQL, conn)

RoutePlaceName['Distance']=RoutePlaceName.apply(lambda row:
   round(
   vincenty((row['SourceLatitude'],row['SourceLongitude']),
             (row['TargetLatitude'],row['TargetLongitude'])).miles
          ,4)
     ,axis=1)

print(RoutePlaceName.head(5))
#################################################################
G=nx.Graph()
#################################################################
print('Countries:',RouteCountries.shape)
for i in range(RouteCountries.shape[0]):
   sNode0='C-' + RouteCountries['SourceCountry'][i]
   G.add_node(sNode0,
            Nodetype='Country',
            Country=RouteCountries['SourceCountry'][i],
            Latitude=round(RouteCountries['SourceLatitude'][i],4),
            Longitude=round(RouteCountries['SourceLongitude'][i],4))

   sNode1='C-' + RouteCountries['TargetCountry'][i]
   G.add_node(sNode1,
            Nodetype='Country',
            Country=RouteCountries['TargetCountry'][i],
            Latitude=round(RouteCountries['TargetLatitude'][i],4),
            Longitude=round(RouteCountries['TargetLongitude'][i],4))
   G.add_edge(sNode0,sNode1,distance=round(RouteCountries['Distance'][i],3))
   #print(sNode0,sNode1)
#################################################################
print('Post Code:',RoutePostCode.shape)
for i in range(RoutePostCode.shape[0]):
   sNode0='C-' + RoutePostCode['SourceCountry'][i]
   G.add_node(sNode0,
            Nodetype='Country',
            Country=RoutePostCode['SourceCountry'][i],
            Latitude=round(RoutePostCode['SourceLatitude'][i],4),
            Longitude=round(RoutePostCode['SourceLongitude'][i],4))

   sNode1='P-' + RoutePostCode['TargetPostCode'][i]  + '-' + RoutePostCode['TargetCountry'][i]
   G.add_node(sNode1,
            Nodetype='PostCode',
            Country=RoutePostCode['TargetCountry'][i],
            PostCode=RoutePostCode['TargetPostCode'][i],
            Latitude=round(RoutePostCode['TargetLatitude'][i],4),
            Longitude=round(RoutePostCode['TargetLongitude'][i],4))
   G.add_edge(sNode0,sNode1,distance=round(RoutePostCode['Distance'][i],3))
   #print(sNode0,sNode1)
#################################################################
print('Place Name:',RoutePlaceName.shape)
for i in range(RoutePlaceName.shape[0]):
   sNode0='P-' + RoutePlaceName['TargetPostCode'][i]  + '-'
   sNode0=sNode0 + RoutePlaceName['TargetCountry'][i]
   G.add_node(sNode0,
            Nodetype='PostCode',
            Country=RoutePlaceName['SourceCountry'][i],
            PostCode=RoutePlaceName['TargetPostCode'][i],
            Latitude=round(RoutePlaceName['SourceLatitude'][i],4),
            Longitude=round(RoutePlaceName['SourceLongitude'][i],4))
```

```
    sNode1='L-' + RoutePlaceName['TargetPlaceName'][i]  + '-'
    sNode1=sNode1 + RoutePlaceName['TargetPostCode'][i]  + '-'
    sNode1=sNode1 + RoutePlaceName['TargetCountry'][i]
    G.add_node(sNode1,
            Nodetype='PlaceName',
            Country=RoutePlaceName['TargetCountry'][i],
            PostCode=RoutePlaceName['TargetPostCode'][i],
            PlaceName=RoutePlaceName['TargetPlaceName'][i],
            Latitude=round(RoutePlaceName['TargetLatitude'][i],4),
            Longitude=round(RoutePlaceName['TargetLongitude'][i],4))
    G.add_edge(sNode0,sNode1,distance=round(RoutePlaceName['Distance'][i],3))
    #print(sNode0,sNode1)
################################################################
sFileName=sFileDir + '/' + OutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
nx.write_gml(G,sFileName)
sFileName=sFileName +'.gz'
nx.write_gml(G,sFileName)
################################################################
print('################################')
print('Path:', nx.shortest_path(G,source='P-SW1-GB',target='P-01001-US',weight='distance'))
print('Path length:', nx.shortest_path_length(G,source='P-SW1-GB',target='P-01001-US',weight='distance'))
print('Path length (1):', nx.shortest_path_length(G,source='P-SW1-GB',target='C-GB',weight='distance'))
print('Path length (2):', nx.shortest_path_length(G,source='C-GB',target='C-US',weight='distance'))
print('Path length (3):', nx.shortest_path_length(G,source='C-US',target='P-01001-US',weight='distance'))
print('################################')
print('Routes from P-SW1-GB < 2: ', nx.single_source_shortest_path(G,source='P-SW1-GB' ,cutoff=1))
print('Routes from P-01001-US < 2: ', nx.single_source_shortest_path(G,source='P-01001-US' ,cutoff=1))
print('################################')
################################################################
print('################')
print('Vacuum Database')
sSQL="VACUUM;"
sql.execute(sSQL,conn)
print('################')
################################################################
print('### Done!! ##########################################')
################################################################
```

Output:

You can now query features out of a graph, such as shortage pathsbetween locations and paths from a given location, using Assess_Best_Logistics.gml with appropirate application.

I. Write a Python / R program to decide the best packing option to ship in container from the given data.

Hillman wants to introduce new shipping containers into its logistics strategy. This program will through a process of assessing the possible container sizes.This example introduces features with ranges or tolerances.

Open Python editor and create a file named Assess-Shipping-Containers.py in directory C:\VKHCG\03-Hillman\02-Assess

```
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
```

```
##################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
##################################################################
Company='03-Hillman'
InputDir='01-Retrieve/01-EDS/02-Python'
InputFileName1='Retrieve_Product.csv'
InputFileName2='Retrieve_Box.csv'
InputFileName3='Retrieve_Container.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName='Assess_Shipping_Containers.csv'
##################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
##################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
##################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
##################################################################
sDatabaseName=sDataBaseDir + '/hillman.db'
conn = sq.connect(sDatabaseName)
##################################################################
##################################################################
### Import Product Data
##################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName1
print('###########')
print('Loading :',sFileName)
ProductRawData=pd.read_csv(sFileName,
                header=0,
                low_memory=False,
                encoding="latin-1"
                )
ProductRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ProductRawData.index.name = 'IDNumber'
ProductData=ProductRawData[ProductRawData.Length <= 0.5].head(10)
print('Loaded Product :',ProductData.columns.values)
print('################################')
##################################################################
print('################')
sTable='Assess_Product'
print('Storing :',sDatabaseName,' Table:',sTable)
ProductData.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
print(ProductData.head())
print('################################')
print('Rows : ',ProductData.shape[0])
print('################################')
##################################################################
##################################################################
### Import Box Data
```

```
##################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName2
print('###########')
print('Loading :',sFileName)
BoxRawData=pd.read_csv(sFileName,
                header=0,
                low_memory=False,
                encoding="latin-1"
                )
BoxRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BoxRawData.index.name = 'IDNumber'
BoxData=BoxRawData[BoxRawData.Length <= 1].head(1000)
print('Loaded Product :',BoxData.columns.values)
print('################################')
##################################################################
print('################')
sTable='Assess_Box'
print('Storing :',sDatabaseName,' Table:',sTable)
BoxData.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
print(BoxData.head())
print('################################')
print('Rows : ',BoxData.shape[0])
print('################################')
##################################################################
##################################################################
### Import Container Data
##################################################################
sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName3
print('###########')
print('Loading :',sFileName)
ContainerRawData=pd.read_csv(sFileName,
                header=0,
                low_memory=False,
                encoding="latin-1"
                )
ContainerRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ContainerRawData.index.name = 'IDNumber'
ContainerData=ContainerRawData[ContainerRawData.Length <= 2].head(10)
print('Loaded Product :',ContainerData.columns.values)
print('################################')
##################################################################
print('################')
sTable='Assess_Container'
print('Storing :',sDatabaseName,' Table:',sTable)
BoxData.to_sql(sTable, conn, if_exists="replace")
print('################')
##################################################################
print(ContainerData.head())
print('################################')
print('Rows : ',ContainerData.shape[0])
print('################################')
##################################################################
##################################################################
### Fit Product in Box
##################################################################
print('################')
sView='Assess_Product_in_Box'
print('Creating :',sDatabaseName,' View:',sView)
```

```
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " P.UnitNumber AS ProductNumber,"
sSQL=sSQL+ " B.UnitNumber AS BoxNumber,"
sSQL=sSQL+ " (B.Thickness * 1000) AS PackSafeCode,"
sSQL=sSQL+ " (B.BoxVolume - P.ProductVolume) AS PackFoamVolume,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 167 AS Air_Dimensional_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 333 AS Road_Dimensional_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 1000 AS Sea_Dimensional_Weight,"
sSQL=sSQL+ " P.Length AS Product_Length,"
sSQL=sSQL+ " P.Width AS Product_Width,"
sSQL=sSQL+ " P.Height AS Product_Height,"
sSQL=sSQL+ " P.ProductVolume AS Product_cm_Volume,"
sSQL=sSQL+ " ((P.Length*10) * (P.Width*10) * (P.Height*10)) AS Product_ccm_Volume,"
sSQL=sSQL+ " (B.Thickness * 0.95) AS Minimum_Pack_Foam,"
sSQL=sSQL+ " (B.Thickness * 1.05) AS Maximum_Pack_Foam,"
sSQL=sSQL+ " B.Length - (B.Thickness * 1.10) AS Minimum_Product_Box_Length,"
sSQL=sSQL+ " B.Length - (B.Thickness * 0.95) AS Maximum_Product_Box_Length,"
sSQL=sSQL+ " B.Width - (B.Thickness * 1.10) AS Minimum_Product_Box_Width,"
sSQL=sSQL+ " B.Width - (B.Thickness * 0.95) AS Maximum_Product_Box_Width,"
sSQL=sSQL+ " B.Height - (B.Thickness * 1.10) AS Minimum_Product_Box_Height,"
sSQL=sSQL+ " B.Height - (B.Thickness * 0.95) AS Maximum_Product_Box_Height,"
sSQL=sSQL+ " B.Length AS Box_Length,"
sSQL=sSQL+ " B.Width AS Box_Width,"
sSQL=sSQL+ " B.Height AS Box_Height,"
sSQL=sSQL+ " B.BoxVolume AS Box_cm_Volume,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) AS Box_ccm_Volume,"
sSQL=sSQL+ " (2 * B.Length * B.Width) + (2 * B.Length * B.Height) + (2 * B.Width * B.Height) AS
Box_sqm_Area,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) *  3.5 AS Box_A_Max_Kg_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) *  7.7 AS Box_B_Max_Kg_Weight,"
sSQL=sSQL+ " ((B.Length*10) * (B.Width*10) * (B.Height*10)) * 10.0 AS Box_C_Max_Kg_Weight"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Product as P"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Box as B"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " P.Length >= (B.Length - (B.Thickness * 1.10))"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Width >= (B.Width - (B.Thickness * 1.10))"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Height >= (B.Height - (B.Thickness * 1.10))"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Length <= (B.Length - (B.Thickness * 0.95))"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Width <= (B.Width - (B.Thickness * 0.95))"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " P.Height <= (B.Height - (B.Thickness * 0.95))"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " (B.Height - B.Thickness) >= 0"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " (B.Width - B.Thickness) >= 0"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " (B.Height - B.Thickness) >= 0"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " B.BoxVolume >= P.ProductVolume;"
sql.execute(sSQL,conn)
```

```
#################################################################
### Fit Box in Pallet
#################################################################
t=0
for l in range(2,8):
   for w in range(2,8):
      for h in range(4):
         t += 1
         PalletLine=[('IDNumber',[t]),
                  ('ShipType', ['Pallet']),
                  ('UnitNumber', ('L-'+format(t,"06d"))),
                  ('Box_per_Length',(format(2**l,"4d"))),
                  ('Box_per_Width',(format(2**w,"4d"))),
                  ('Box_per_Height',(format(2**h,"4d")))]
         if t==1:
           PalletFrame = pd.DataFrame.from_items(PalletLine)
         else:
            PalletRow = pd.DataFrame.from_items(PalletLine)
            PalletFrame = PalletFrame.append(PalletRow)
PalletFrame.set_index(['IDNumber'],inplace=True)
#################################################################
PalletFrame.head()
print('################################')
print('Rows : ',PalletFrame.shape[0])
print('################################')
#################################################################
### Fit Box on Pallet
#################################################################
print('################')
sView='Assess_Box_on_Pallet'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT DISTINCT"
sSQL=sSQL+ " P.UnitNumber AS PalletNumber,"
sSQL=sSQL+ " B.UnitNumber AS BoxNumber,"
sSQL=sSQL+ " round(B.Length*P.Box_per_Length,3) AS Pallet_Length,"
sSQL=sSQL+ " round(B.Width*P.Box_per_Width,3) AS Pallet_Width,"
sSQL=sSQL+ " round(B.Height*P.Box_per_Height,3) AS Pallet_Height,"
sSQL=sSQL+ " P.Box_per_Length * P.Box_per_Width * P.Box_per_Height AS Pallet_Boxes"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Box as B"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Pallet as P"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " round(B.Length*P.Box_per_Length,3) <= 20"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(B.Width*P.Box_per_Width,3) <= 9"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " round(B.Height*P.Box_per_Height,3) <= 5;"
sql.execute(sSQL,conn)
#################################################################
sTables=['Assess_Product_in_Box','Assess_Box_on_Pallet']
for sTable in sTables:
   print('################')
   print('Loading :',sDatabaseName,' Table:',sTable)
   sSQL=" SELECT "
   sSQL=sSQL+ " *"
```

```
    sSQL=sSQL+ " FROM"
    sSQL=sSQL+ " " + sTable + ";"
    SnapShotData=pd.read_sql_query(sSQL, conn)
    print('################')
    sTableOut=sTable + '_SnapShot'
    print('Storing :',sDatabaseName,' Table:',sTable)
    SnapShotData.to_sql(sTableOut, conn, if_exists="replace")
    print('################')
#################################################################
### Fit Pallet in Container
#################################################################
sTables=['Length','Width','Height']
for sTable in sTables:

    sView='Assess_Pallet_in_Container_' + sTable
    print('Creating :',sDatabaseName,' View:',sView)
    sSQL="DROP VIEW IF EXISTS " + sView + ";"
    sql.execute(sSQL,conn)

    sSQL="CREATE VIEW " + sView + " AS"
    sSQL=sSQL+ " SELECT DISTINCT"
    sSQL=sSQL+ " C.UnitNumber AS ContainerNumber,"
    sSQL=sSQL+ " P.PalletNumber,"
    sSQL=sSQL+ " P.BoxNumber,"
    sSQL=sSQL+ " round(C." + sTable + "/P.Pallet_" + sTable + ",0)"
    sSQL=sSQL+ " AS Pallet_per_" + sTable + ","
    sSQL=sSQL+ " round(C." + sTable + "/P.Pallet_" + sTable + ",0)"
    sSQL=sSQL+ " * P.Pallet_Boxes AS Pallet_" + sTable + "_Boxes,"
    sSQL=sSQL+ " P.Pallet_Boxes"
    sSQL=sSQL+ " FROM"
    sSQL=sSQL+ " Assess_Container as C"
    sSQL=sSQL+ " ,"
    sSQL=sSQL+ " Assess_Box_on_Pallet_SnapShot as P"
    sSQL=sSQL+ " WHERE"
    sSQL=sSQL+ " round(C.Length/P.Pallet_Length,0) > 0"
    sSQL=sSQL+ " AND"
    sSQL=sSQL+ " round(C.Width/P.Pallet_Width,0) > 0"
    sSQL=sSQL+ " AND"
    sSQL=sSQL+ " round(C.Height/P.Pallet_Height,0) > 0;"
    sql.execute(sSQL,conn)

    print('################')
    print('Loading :',sDatabaseName,' Table:',sView)
    sSQL=" SELECT "
    sSQL=sSQL+ " *"
    sSQL=sSQL+ " FROM"
    sSQL=sSQL+ " " + sView + ";"
    SnapShotData=pd.read_sql_query(sSQL, conn)
    print('################')
    sTableOut= sView + '_SnapShot'
    print('Storing :',sDatabaseName,' Table:',sTableOut)
    SnapShotData.to_sql(sTableOut, conn, if_exists="replace")
    print('################')
#################################################################
print('################')
sView='Assess_Pallet_in_Container'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)
```

```
sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " CL.ContainerNumber,"
sSQL=sSQL+ " CL.PalletNumber,"
sSQL=sSQL+ " CL.BoxNumber,"
sSQL=sSQL+ " CL.Pallet_Boxes AS Boxes_per_Pallet,"
sSQL=sSQL+ " CL.Pallet_per_Length,"
sSQL=sSQL+ " CW.Pallet_per_Width,"
sSQL=sSQL+ " CH.Pallet_per_Height,"
sSQL=sSQL+ " CL.Pallet_Length_Boxes * CW.Pallet_Width_Boxes * CH.Pallet_Height_Boxes AS
Container_Boxes"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Pallet_in_Container_Length_SnapShot as CL"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " Assess_Pallet_in_Container_Width_SnapShot as CW"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " CL.ContainerNumber = CW.ContainerNumber"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " CL.PalletNumber = CW.PalletNumber"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " CL.BoxNumber = CW.BoxNumber"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " Assess_Pallet_in_Container_Height_SnapShot as CH"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " CL.ContainerNumber = CH.ContainerNumber"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " CL.PalletNumber = CH.PalletNumber"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " CL.BoxNumber = CH.BoxNumber;"
sql.execute(sSQL,conn)
#################################################################
sTables=['Assess_Product_in_Box','Assess_Pallet_in_Container']
for sTable in sTables:
    print('################')
    print('Loading :',sDatabaseName,' Table:',sTable)
    sSQL=" SELECT "
    sSQL=sSQL+ " *"
    sSQL=sSQL+ " FROM"
    sSQL=sSQL+ " " + sTable + ";"
    PackData=pd.read_sql_query(sSQL, conn)
    print('################')
    print(PackData)
    print('################')
    print('################################')
    print('Rows : ',PackData.shape[0])
    print('################################')
    sFileName=sFileDir + '/' + sTable + '.csv'
    print(sFileName)
    PackData.to_csv(sFileName, index = False)
print('### Done!! ##########################################')
#################################################################
```

J. Write a Python program to create a delivery route using the given data.

Creating a Delivery Route

Hillman requires the complete grid plan of the delivery routes for the company, to ensure the suppliers, warehouses, shops, and customers can be reached by its new strategy. This new plan will enable the optimum routes between suppliers, warehouses, shops, and customers.

Open Python editor and create a file named Assess-Shipping-Routes.py in directory C:\VKHCG\03-Hillman\02-Assess.

```
#####################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import networkx as nx
from geopy.distance import vincenty
#####################################################################
nMax=3
nMaxPath=10
nSet=False
nVSet=False
#####################################################################
Base='C:/VKHCG'
#####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
#####################################################################
Company='03-Hillman'
InputDir1='01-Retrieve/01-EDS/01-R'
InputDir2='01-Retrieve/01-EDS/02-Python'
InputFileName1='Retrieve_GB_Postcode_Warehouse.csv'
InputFileName2='Retrieve_GB_Postcodes_Shops.csv'
EDSDir='02-Assess/01-EDS'
OutputDir=EDSDir + '/02-Python'
OutputFileName1='Assess_Shipping_Routes.gml'
OutputFileName2='Assess_Shipping_Routes.txt'
#####################################################################
sFileDir=Base + '/' + Company + '/' + EDSDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#####################################################################
sFileDir=Base + '/' + Company + '/' + OutputDir
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
#####################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
```

```
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#################################################################
sDatabaseName=sDataBaseDir + '/hillman.db'
conn = sq.connect(sDatabaseName)
#################################################################
#################################################################
### Import Warehouse Data
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir1 + '/' + InputFileName1
print('###########')
print('Loading :',sFileName)
WarehouseRawData=pd.read_csv(sFileName,
                header=0,
                low_memory=False,
                encoding="latin-1"
                )
WarehouseRawData.drop_duplicates(subset=None, keep='first', inplace=True)
WarehouseRawData.index.name = 'IDNumber'
WarehouseData=WarehouseRawData.head(nMax)
WarehouseData=WarehouseData.append(WarehouseRawData.tail(nMax))
WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode=='KA13'])
if nSet==True:

WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode=='SW1W'])
WarehouseData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Loaded Warehouses :',WarehouseData.columns.values)
print('################################')
#################################################################
print('################')
sTable='Assess_Warehouse_UK'
print('Storing :',sDatabaseName,' Table:',sTable)
WarehouseData.to_sql(sTable, conn, if_exists="replace")
print('################')
#################################################################
print(WarehouseData.head())
print('################################')
print('Rows : ',WarehouseData.shape[0])
print('################################')
#################################################################
### Import Shop Data
#################################################################
sFileName=Base + '/' + Company + '/' + InputDir1 + '/' + InputFileName2
print('###########')
print('Loading :',sFileName)
ShopRawData=pd.read_csv(sFileName,
                header=0,
                low_memory=False,
                encoding="latin-1"
                )
ShopRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ShopRawData.index.name = 'IDNumber'
ShopData=ShopRawData
print('Loaded Shops :',ShopData.columns.values)
print('################################')
#################################################################
print('################')
sTable='Assess_Shop_UK'
print('Storing :',sDatabaseName,' Table:',sTable)
ShopData.to_sql(sTable, conn, if_exists="replace")
```

```
print('################')
#################################################################
print(ShopData.head())
print('################################')
print('Rows : ',ShopData.shape[0])
print('################################')
#################################################################
### Connect HQ
#################################################################
print('################')
sView='Assess_HQ'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " W.postcode AS HQ_PostCode,"
sSQL=sSQL+ " 'HQ-' || W.postcode AS HQ_Name,"
sSQL=sSQL+ " round(W.latitude,6) AS HQ_Latitude,"
sSQL=sSQL+ " round(W.longitude,6) AS HQ_Longitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " TRIM(W.postcode) in ('KA13','SW1W');"
sql.execute(sSQL,conn)
#################################################################
### Connect Warehouses
#################################################################
print('################')
sView='Assess_Warehouse'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " W.postcode AS Warehouse_PostCode,"
sSQL=sSQL+ " 'WH-' || W.postcode AS Warehouse_Name,"
sSQL=sSQL+ " round(W.latitude,6) AS Warehouse_Latitude,"
sSQL=sSQL+ " round(W.longitude,6) AS Warehouse_Longitude"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W;"
sql.execute(sSQL,conn)
#################################################################
### Connect Warehouse to Shops by PostCode
-#################################################################
print('################')
sView='Assess_Shop'
print('Creating :',sDatabaseName,' View:',sView)
sSQL="DROP VIEW IF EXISTS " + sView + ";"
sql.execute(sSQL,conn)

sSQL="CREATE VIEW " + sView + " AS"
sSQL=sSQL+ " SELECT"
sSQL=sSQL+ " TRIM(S.postcode) AS Shop_PostCode,"
sSQL=sSQL+ " 'SP-' || TRIM(S.FirstCode) || '-' || TRIM(S.SecondCode) AS Shop_Name,"
sSQL=sSQL+ " TRIM(S.FirstCode) AS Warehouse_PostCode,"
sSQL=sSQL+ " round(S.latitude,6) AS Shop_Latitude,"
sSQL=sSQL+ " round(S.longitude,6) AS Shop_Longitude"
```

```
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " Assess_Warehouse_UK as W"
sSQL=sSQL+ " JOIN"
sSQL=sSQL+ " Assess_Shop_UK as S"
sSQL=sSQL+ " ON"
sSQL=sSQL+ " TRIM(W.postcode) = TRIM(S.FirstCode);"
sql.execute(sSQL,conn)
#################################################################
#################################################################
G=nx.Graph()
#################################################################
print('################')
sTable = 'Assess_HQ'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
print('################')
#################################################################
print(RouteData.head())
print('################################')
print('HQ Rows : ',RouteData.shape[0])
print('################################')
#################################################################
for i in range(RouteData.shape[0]):
    sNode0=RouteData['HQ_Name'][i]
    G.add_node(sNode0,
            Nodetype='HQ',
            PostCode=RouteData['HQ_PostCode'][i],
            Latitude=round(RouteData['HQ_Latitude'][i],6),
            Longitude=round(RouteData['HQ_Longitude'][i],6))
#################################################################
print('################')
sTable = 'Assess_Warehouse'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
print('################')
#################################################################
print(RouteData.head())
print('################################')
print('Warehouse Rows : ',RouteData.shape[0])
print('################################')
for i in range(RouteData.shape[0]):
    sNode0=RouteData['Warehouse_Name'][i]
    G.add_node(sNode0,
            Nodetype='Warehouse',
            PostCode=RouteData['Warehouse_PostCode'][i],
            Latitude=round(RouteData['Warehouse_Latitude'][i],6),
            Longitude=round(RouteData['Warehouse_Longitude'][i],6))
print('################')
sTable = 'Assess_Shop'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL=" SELECT DISTINCT"
sSQL=sSQL+ " *"
```

```
sSQL=sSQL+ " FROM"
sSQL=sSQL+ " " + sTable + ";"
RouteData=pd.read_sql_query(sSQL, conn)
print('################')
print(RouteData.head())
print('################################')
print('Shop Rows : ',RouteData.shape[0])
print('################################')
for i in range(RouteData.shape[0]):
   sNode0=RouteData['Shop_Name'][i]
   G.add_node(sNode0,
            Nodetype='Shop',
            PostCode=RouteData['Shop_PostCode'][i],
            WarehousePostCode=RouteData['Warehouse_PostCode'][i],
            Latitude=round(RouteData['Shop_Latitude'][i],6),
            Longitude=round(RouteData['Shop_Longitude'][i],6))
#################################################################
## Create Edges
#################################################################
print('################################')
print('Loading Edges')
print('################################')

for sNode0 in nx.nodes_iter(G):
   for sNode1 in nx.nodes_iter(G):
      if G.node[sNode0]['Nodetype']=='HQ' and \
         G.node[sNode1]['Nodetype']=='HQ' and \
         sNode0 != sNode1:
            distancemeters=round(\
                        vincenty(\
                              (\
                               G.node[sNode0]['Latitude'],\
                               G.node[sNode0]['Longitude']\
                               ),\
                               (\
                                G.node[sNode1]['Latitude']\
                                ,\
                                G.node[sNode1]['Longitude']\
                                )\
                                ).meters\
                        ,0)
            distancemiles=round(\
                        vincenty(\
                              (\
                               G.node[sNode0]['Latitude'],\
                               G.node[sNode0]['Longitude']\
                               ),\
                               (\
                                G.node[sNode1]['Latitude']\
                                ,\
                                G.node[sNode1]['Longitude']\
                                )\
                                ).miles\
                        ,3)

            if distancemiles >= 0.05:
               cost = round(150+(distancemiles * 2.5),6)
               vehicle='V001'
            else:
               cost = round(2+(distancemiles * 0.10),6)
```

```
            vehicle='ForkLift'

        G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                DistanceMiles=distancemiles, \
                Cost=cost,Vehicle=vehicle)
        if nVSet==True:
            print('Edge-H-H:',sNode0,' to ', sNode1, \
              ' Distance:',distancemeters,'meters',\
              distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

    if G.node[sNode0]['Nodetype']=='HQ' and \
        G.node[sNode1]['Nodetype']=='Warehouse' and \
        sNode0 != sNode1:
            distancemeters=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).meters\
                        ,0)
            distancemiles=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).miles\
                        ,3)
            if distancemiles >= 10:
                cost = round(50+(distancemiles * 2),6)
                vehicle='V002'
            else:
                cost = round(5+(distancemiles * 1.5),6)
                vehicle='V003'
            if distancemiles <= 50:
                G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                        DistanceMiles=distancemiles, \
                        Cost=cost,Vehicle=vehicle)
                if nVSet==True:
                    print('Edge-H-W:',sNode0,' to ', sNode1, \
                      ' Distance:',distancemeters,'meters',\
                      distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

    if nSet==True and \
        G.node[sNode0]['Nodetype']=='Warehouse' and \
        G.node[sNode1]['Nodetype']=='Warehouse' and \
        sNode0 != sNode1:
            distancemeters=round(\
                        vincenty(\
```

```
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).meters\
                        ,0)
            distancemiles=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).miles\
                        ,3)
            if distancemiles >= 10:
                cost = round(50+(distancemiles * 1.10),6)
                vehicle='V004'
            else:
                cost = round(5+(distancemiles * 1.05),6)
                vehicle='V005'

            if distancemiles <= 20:
                G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                        DistanceMiles=distancemiles, \
                        Cost=cost,Vehicle=vehicle)
                if nVSet==True:
                    print('Edge-W-W:',sNode0,' to ', sNode1, \
                     ' Distance:',distancemeters,'meters',\
                     distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

        if G.node[sNode0]['Nodetype']=='Warehouse' and \
            G.node[sNode1]['Nodetype']=='Shop' and \
            G.node[sNode0]['PostCode']==G.node[sNode1]['WarehousePostCode'] and \
            sNode0 != sNode1:

            distancemeters=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).meters\
                        ,0)
            distancemiles=round(\
                        vincenty(\
```

```
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).miles\
                        ,3)
        if distancemiles >= 10:
            cost = round(50+(distancemiles * 1.50),6)
            vehicle='V006'
        else:
            cost = round(5+(distancemiles * 0.75),6)
            vehicle='V007'
        if distancemiles <= 10:
            G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                    DistanceMiles=distancemiles, \
                    Cost=cost,Vehicle=vehicle)
            if nVSet==True:
                print('Edge-W-S:',sNode0,' to ', sNode1, \
                  ' Distance:',distancemeters,'meters',\
                  distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

    if nSet==True and \
        G.node[sNode0]['Nodetype']=='Shop' and \
        G.node[sNode1]['Nodetype']=='Shop' and \
        G.node[sNode0]['WarehousePostCode']==G.node[sNode1]['WarehousePostCode'] and \
        sNode0 != sNode1:

            distancemeters=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).meters\
                        ,0)
            distancemiles=round(\
                        vincenty(\
                                (\
                                 G.node[sNode0]['Latitude'],\
                                 G.node[sNode0]['Longitude']\
                                 ),\
                                 (\
                                  G.node[sNode1]['Latitude']\
                                  ,\
                                  G.node[sNode1]['Longitude']\
                                  )\
                                  ).miles\
                        ,3)

            if distancemiles >= 0.05:
```

```
            cost = round(5+(distancemiles * 0.5),6)
            vehicle='V008'
        else:
            cost = round(1+(distancemiles * 0.1),6)
            vehicle='V009'

        if distancemiles <= 0.075:
            G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                    DistanceMiles=distancemiles, \
                    Cost=cost,Vehicle=vehicle)
            if nVSet==True:
                print('Edge-S-S:',sNode0,' to ', sNode1, \
                  ' Distance:',distancemeters,'meters',\
                  distancemiles,'miles','Cost', cost,'Vehicle',vehicle)

    if nSet==True and \
        G.node[sNode0]['Nodetype']=='Shop' and \
        G.node[sNode1]['Nodetype']=='Shop' and \
        G.node[sNode0]['WarehousePostCode']!=G.node[sNode1]['WarehousePostCode'] and \
        sNode0 != sNode1:

        distancemeters=round(\
                    vincenty(\
                        (\
                         G.node[sNode0]['Latitude'],\
                         G.node[sNode0]['Longitude']\
                         ),\
                         (\
                          G.node[sNode1]['Latitude']\
                          ,\
                          G.node[sNode1]['Longitude']\
                          )\
                          ).meters\
                    ,0)
        distancemiles=round(\
                    vincenty(\
                        (\
                         G.node[sNode0]['Latitude'],\
                         G.node[sNode0]['Longitude']\
                         ),\
                         (\
                          G.node[sNode1]['Latitude']\
                          ,\
                          G.node[sNode1]['Longitude']\
                          )\
                          ).miles\
                    ,3)

        cost = round(1+(distancemiles * 0.1),6)
        vehicle='V010'

        if distancemiles <= 0.025:
            G.add_edge(sNode0,sNode1,DistanceMeters=distancemeters, \
                    DistanceMiles=distancemiles, \
                    Cost=cost,Vehicle=vehicle)
            if nVSet==True:
                print('Edge-S-S:',sNode0,' to ', sNode1, \
                  ' Distance:',distancemeters,'meters',\
                  distancemiles,'miles','Cost', cost,'Vehicle',vehicle)
sFileName=sFileDir + '/' + OutputFileName1
```

```
print('################################')
print('Storing :', sFileName)
print('################################')
nx.write_gml(G,sFileName)
sFileName=sFileName +'.gz'
nx.write_gml(G,sFileName)
print('Nodes:',nx.number_of_nodes(G))
print('Edges:',nx.number_of_edges(G))
sFileName=sFileDir + '/' + OutputFileName2
print('################################')
print('Storing :', sFileName)
print('################################')
## Create Paths
print('################################')
print('Loading Paths')
print('################################')
f = open(sFileName,'w')
l=0
sline = 'ID|Cost|StartAt|EndAt|Path|Measure'
if nVSet==True: print ('0', sline)
f.write(sline+ '\n')
for sNode0 in nx.nodes_iter(G):
    for sNode1 in nx.nodes_iter(G):
        if sNode0 != sNode1 and \
            nx.has_path(G, sNode0, sNode1)==True and \
            nx.shortest_path_length(G, \
              source=sNode0, \
              target=sNode1, \
              weight='DistanceMiles') < nMaxPath:
                l+=1
                sID='{:.0f}'.format(l)
                spath = ','.join(nx.shortest_path(G, \
                  source=sNode0, \
                  target=sNode1, \
                  weight='DistanceMiles'))
                slength= '{:.6f}'.format(\
                  nx.shortest_path_length(G, \
                  source=sNode0, \
                  target=sNode1, \
                  weight='DistanceMiles'))
                sline = sID + '|"DistanceMiles"|"' + sNode0 + '"|"' \
                + sNode1 + '"|"' + spath + '"|' + slength
                if nVSet==True: print (sline)
                f.write(sline + '\n')
                l+=1
                sID='{:.0f}'.format(l)
                spath = ','.join(nx.shortest_path(G, \
                  source=sNode0, \
                  target=sNode1, \
                  weight='DistanceMeters'))
                slength= '{:.6f}'.format(\
                  nx.shortest_path_length(G, \
                  source=sNode0, \
                  target=sNode1, \
                  weight='DistanceMeters'))
                sline = sID + '|"DistanceMeters"|"' + sNode0 + '"|"' \
                + sNode1 + '"|"' + spath + '"|' + slength
                if nVSet==True: print (sline)
                f.write(sline + '\n')
                l+=1
```

```
            sID='{:.0f}'.format(l)
            spath = ','.join(nx.shortest_path(G, \
              source=sNode0, \
              target=sNode1, \
              weight='Cost'))
            slength= '{:.6f}'.format(\
              nx.shortest_path_length(G, \
              source=sNode0, \
              target=sNode1, \
              weight='Cost'))
            sline = sID + '|"Cost"|"' + sNode0 + '"|"' \
            + sNode1 + '"|"' + spath + '"|' + slength
            if nVSet==True: print (sline)
            f.write(sline + '\n')
f.close()
print('Nodes:',nx.number_of_nodes(G))
print('Edges:',nx.number_of_edges(G))
print('Paths:',sID)
print('################')
print('Vacuum Database')
sSQL="VACUUM;"
sql.execute(sSQL,conn)
print('################')
print('### Done!! ##############################################')
```

Clark Ltd

Clark Ltd is the accountancy company that handles everything related to the VKHCG's finances and personnel. Let's investigate Clark with new knowledge.

K. Write a Python program to create Simple forex trading planner from the given data.

Simple Forex Trading Planner

Clark requires the assessment of the group's forex data, for processing and data qualityissues. I will guide you through an example of a forex solution.

Open your Python editor and create a file named Assess-Forex.py in directory C:\VKHCG\04-Clark\02-Assess.

```
#################################################################
import sys
import os
import sqlite3 as sq
import pandas as pd
#################################################################
```

```
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='04-Clark'
sInputFileName1='01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve-Country-Currency.csv'
sInputFileName2='04-Clark/01-Retrieve/01-EDS/01-R/Retrieve_Euro_EchangeRates.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
################################################################
### Import Country Data
################################################################
sFileName1=Base + '/' + sInputFileName1
print('################################')
print('Loading :',sFileName1)
print('################################')
CountryRawData=pd.read_csv(sFileName1,header=0,low_memory=False, encoding="latin-1")
CountryRawData.drop_duplicates(subset=None, keep='first', inplace=True)
CountryData=CountryRawData
print('Loaded Company :',CountryData.columns.values)
print('################################')
################################################################
print('################')
sTable='Assess_Country'
print('Storing :',sDatabaseName,' Table:',sTable)
CountryData.to_sql(sTable, conn, if_exists="replace")
print('################')
################################################################
print(CountryData.head())
print('################################')
print('Rows : ',CountryData.shape[0])
print('################################')
################################################################
### Import Forex Data
################################################################
sFileName2=Base + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName2)
print('################################')
ForexRawData=pd.read_csv(sFileName2,header=0,low_memory=False, encoding="latin-1")
ForexRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ForexData=ForexRawData.head(5)
print('Loaded Company :',ForexData.columns.values)
print('################################')
################################################################
print('################')
sTable='Assess_Forex'
print('Storing :',sDatabaseName,' Table:',sTable)
ForexData.to_sql(sTable, conn, if_exists="replace")
print('################')
################################################################
print(ForexData.head())
print('################################')
print('Rows : ',ForexData.shape[0])
```

```
print('################################')
#################################################################
print('################')
sTable='Assess_Forex'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.CodeIn"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_Forex as A;"
CodeData=pd.read_sql_query(sSQL, conn)
print('################')
#################################################################

for c in range(CodeData.shape[0]):
    print('################')
    sTable='Assess_Forex & 2x Country > ' + CodeData['CodeIn'][c]
    print('Loading :',sDatabaseName,' Table:',sTable)
    sSQL="select distinct"
    sSQL=sSQL+ " A.Date,"
    sSQL=sSQL+ " A.CodeIn,"
    sSQL=sSQL+ " B.Country as CountryIn,"
    sSQL=sSQL+ " B.Currency as CurrencyNameIn,"
    sSQL=sSQL+ " A.CodeOut,"
    sSQL=sSQL+ " C.Country as CountryOut,"
    sSQL=sSQL+ " C.Currency as CurrencyNameOut,"
    sSQL=sSQL+ " A.Rate"
    sSQL=sSQL+ " from"
    sSQL=sSQL+ " Assess_Forex as A"
    sSQL=sSQL+ " JOIN"
    sSQL=sSQL+ " Assess_Country as B"
    sSQL=sSQL+ " ON A.CodeIn = B.CurrencyCode"
    sSQL=sSQL+ " JOIN"
    sSQL=sSQL+ " Assess_Country as C"
    sSQL=sSQL+ " ON A.CodeOut = C.CurrencyCode"
    sSQL=sSQL+ " WHERE"
    sSQL=sSQL+ " A.CodeIn ='" + CodeData['CodeIn'][c] + "';"
    ForexData=pd.read_sql_query(sSQL, conn).head(1000)
    print('################')
    print(ForexData)
    print('################')
    sTable='Assess_Forex_' + CodeData['CodeIn'][c]
    print('Storing :',sDatabaseName,' Table:',sTable)
    ForexData.to_sql(sTable, conn, if_exists="replace")
    print('################')
    print('################################')
    print('Rows : ',ForexData.shape[0])
    print('################################')
#################################################################
print('### Done!! ############################################')
#################################################################
```

Output:

This will produce a set of demonstrated values onscreen by removing duplicate records and other related data processing.

L. Write a Python program to process the balance sheet to ensure that only good data is processing.

Financials
Clark requires you to process the balance sheet for the VKHCG group companies. Go through a sample balance sheet data assessment, to ensure that only the good data is processed.

Open Python editor and create a file named Assess-Financials.py in directory C:\VKHCG\04-Clark\02-Assess.

```
import sys
import os
import sqlite3 as sq
import pandas as pd
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
Company='04-Clark'
sInputFileName='01-Retrieve/01-EDS/01-R/Retrieve_Profit_And_Loss.csv'
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
### Import Financial Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
FinancialRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
FinancialData=FinancialRawData
print('Loaded Company :',FinancialData.columns.values)
print('##################################')
print('################')
sTable='Assess-Financials'
print('Storing :',sDatabaseName,' Table:',sTable)
FinancialData.to_sql(sTable, conn, if_exists="replace")
print('################')
print(FinancialData.head())
print('##################################')
print('Rows : ',FinancialData.shape[0])
print('##################################')
print('### Done!! ##############################################')
```

Write a Python program to store all master records for the financial calendar

Financial Calendar

Clark stores all the master records for the financial calendar. So we import thecalendar from the retrieve step's data storage.

Open Python editor and create a file named Assess-Calendar.py in directory C:\VKHCG\04-Clark\02-Assess.

```
################################################################
import sys
import os
import sqlite3 as sq
import pandas as pd
################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='04-Clark'
################################################################
sDataBaseDirIn=Base + '/' + Company + '/01-Retrieve/SQLite'
if not os.path.exists(sDataBaseDirIn):
   os.makedirs(sDataBaseDirIn)
sDatabaseNameIn=sDataBaseDirIn + '/clark.db'
connIn = sq.connect(sDatabaseNameIn)
################################################################
sDataBaseDirOut=Base + '/' + Company + '/01-Retrieve/SQLite'
if not os.path.exists(sDataBaseDirOut):
   os.makedirs(sDataBaseDirOut)
sDatabaseNameOut=sDataBaseDirOut + '/clark.db'
connOut = sq.connect(sDatabaseNameOut)
################################################################
sTableIn='Retrieve_Date'
sSQL='select * FROM ' + sTableIn + ';'
print('################')
sTableOut='Assess_Time'
print('Loading :',sDatabaseNameIn,' Table:',sTableIn)
dateRawData=pd.read_sql_query(sSQL, connIn)
dateData=dateRawData
################################################################
print('################################')
print('Load Rows : ',dateRawData.shape[0], ' records')
print('################################')
dateData.drop_duplicates(subset='FinDate', keep='first', inplace=True)
################################################################
print('################')
sTableOut='Assess_Date'
print('Storing :',sDatabaseNameOut,' Table:',sTableOut)
dateData.to_sql(sTableOut, connOut, if_exists="replace")
print('################')
################################################################
print('################################')
print('Store Rows : ',dateData.shape[0], ' records')
print('################################')
################################################################
################################################################
sTableIn='Retrieve_Time'
```

```
sSQL='select * FROM ' + sTableIn + ';'
print('################')
sTableOut='Assess_Time'
print('Loading :',sDatabaseNameIn,' Table:',sTableIn)
timeRawData=pd.read_sql_query(sSQL, connIn)
timeData=timeRawData
#################################################################
print('################################')
print('Load Rows : ',timeData.shape[0], ' records')
print('################################')
timeData.drop_duplicates(subset=None, keep='first', inplace=True)
#################################################################
print('################')
sTableOut='Assess_Time'
print('Storing :',sDatabaseNameOut,' Table:',sTableOut)
timeData.to_sql(sTableOut, connOut, if_exists="replace")
print('################')
#################################################################
print('################################')
print('Store Rows : ',timeData.shape[0], ' records')
print('################################')
#################################################################
print('### Done!! ##############################################')
#################################################################
```

M. Write a Python program to generate payroll from the given data.

People

Clark Ltd generates the payroll, so it holds all the staff records. Clark also handles all payments to suppliers and receives payments from customers' details on all companies.

Open Python editor and create a file named Assess-People.py in directory C:\VKHCG\04-Clark\02-Assess.

```
#################################################################
import sys
import os
import sqlite3 as sq
import pandas as pd
#################################################################
```

```
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='04-Clark'
sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve-Data_female-names.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve-Data_male-names.csv'
sInputFileName3='01-Retrieve/01-EDS/02-Python/Retrieve-Data_last-names.csv'
sOutputFileName1='Assess-Staff.csv'
sOutputFileName2='Assess-Customers.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
################################################################
### Import Female Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('################################')
print('Loading :',sFileName)
print('################################')
print(sFileName)
FemaleRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
FemaleRawData.rename(columns={'NameValues' : 'FirstName'},inplace=True)
FemaleRawData.drop_duplicates(subset=None, keep='first', inplace=True)
FemaleData=FemaleRawData.sample(100)
print('################################')
################################################################
print('################')
sTable='Assess_FemaleName'
print('Storing :',sDatabaseName,' Table:',sTable)
FemaleData.to_sql(sTable, conn, if_exists="replace")
print('################')
print('################################')
print('Rows : ',FemaleData.shape[0], ' records')
print('################################')
################################################################
### Import Male Data
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('################################')
print('Loading :',sFileName)
print('################################')
MaleRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
MaleRawData.rename(columns={'NameValues' : 'FirstName'},inplace=True)
MaleRawData.drop_duplicates(subset=None, keep='first', inplace=True)
MaleData=MaleRawData.sample(100)
print('################################')
sTable='Assess_MaleName'
print('Storing :',sDatabaseName,' Table:',sTable)
MaleData.to_sql(sTable, conn, if_exists="replace")
print('################')
################################################################
print('################################')
print('Rows : ',MaleData.shape[0], ' records')
print('################################')
################################################################
```

```
### Import Surname Data
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('################################')
print('Loading :',sFileName)
print('################################')
SurnameRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
SurnameRawData.rename(columns={'NameValues' : 'LastName'},inplace=True)
SurnameRawData.drop_duplicates(subset=None, keep='first', inplace=True)
SurnameData=SurnameRawData.sample(200)
print('################')
sTable='Assess_Surname'
print('Storing :',sDatabaseName,' Table:',sTable)
SurnameData.to_sql(sTable, conn, if_exists="replace")
print('################')
print('################################')
print('Rows : ',SurnameData.shape[0], ' records')
print('################################')
print('################')
sTable='Assess_FemaleName & Assess_MaleName'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " 'Female' as Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FemaleName as A"
sSQL=sSQL+ " UNION"
sSQL=sSQL+ " select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " 'Male' as Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_MaleName as A;"
FirstNameData=pd.read_sql_query(sSQL, conn)
print('################')
#################################################################
#print('################')
sTable='Assess_FirstName'
print('Storing :',sDatabaseName,' Table:',sTable)
FirstNameData.to_sql(sTable, conn, if_exists="replace")
print('################')
################################################################
################################################################
print('################')
sTable='Assess_FirstName x2 & Assess_Surname'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " B.FirstName AS SecondName,"
sSQL=sSQL+ " C.LastName,"
sSQL=sSQL+ " A.Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FirstName as A"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_FirstName as B"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Surname as C"
sSQL=sSQL+ " WHERE"
sSQL=sSQL+ " A.Gender = B.Gender"
sSQL=sSQL+ " AND"
sSQL=sSQL+ " A.FirstName <> B.FirstName;"
PeopleRawData=pd.read_sql_query(sSQL, conn)
```

```
People1Data=PeopleRawData.sample(10000)

sTable='Assess_FirstName & Assess_Surname'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="select distinct"
sSQL=sSQL+ " A.FirstName,"
sSQL=sSQL+ " " AS SecondName,"
sSQL=sSQL+ " B.LastName,"
sSQL=sSQL+ " A.Gender"
sSQL=sSQL+ " from"
sSQL=sSQL+ " Assess_FirstName as A"
sSQL=sSQL+ " ,"
sSQL=sSQL+ " Assess_Surname as B;"
PeopleRawData=pd.read_sql_query(sSQL, conn)
People2Data=PeopleRawData.sample(10000)
PeopleData=People1Data.append(People2Data)
print(PeopleData)
print('################')
#print('################')
sTable='Assess_People'
print('Storing :',sDatabaseName,' Table:',sTable)
PeopleData.to_sql(sTable, conn, if_exists="replace")
print('################')
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
sOutputFileName = sTable+'.csv'
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
PeopleData.to_csv(sFileName, index = False)
print('################################')
#################################################################
print('### Done!! ##############################################')
#################################################################
```

OUTPUT:

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
=========== RESTART: C:\VKHCG\04-Clark\02-Assess\Assess-People.py ===========
Loading : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_female-names.csv
C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_female-names.csv
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FemaleName
Rows :  100  records
Loading : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_male-names.csv
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_MaleName
Rows :  100  records
Loading : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve-Data_last-names.csv
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_Surname
Rows :  200  records
Loading : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FemaleName & Assess_MaleName
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FirstName
Loading : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FirstName x2 & Assess_Surname
Loading : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_FirstName & Assess_Surname
        FirstName SecondName  LastName  Gender
2471856    Miguel      Efren    Ortega    Male
3466902    Tommye    Coretta   Roberts  Female
3336496      Stan     Xavier     Costa    Male
1151796   Faviola       Gene     Heard  Female
893614     Dorene     Joelle   Mccloud  Female
...           ...        ...       ...     ...
31958    Santiago               Crook    Male
32635     Shaunte             Ferreira  Female
2436       Bernie               Dubose    Male
39951     Zoraida               Cherry  Female
7702       Dannie                Foret    Male

[20000 rows x 4 columns]
Storing : C:/VKHCG/04-Clark/02-Assess/SQLite/clark.db  Table: Assess_People
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
################################
```

Practical 6:

Processing Data

A. Build the time hub, links, and satellites.

Open your Python editor and create a file named Process_Time.py. Save it into directory C:\VKHCG\01-Vermeulen\03-Process.

```
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
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/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_items(TimeLine)
   else:
       TimeRow = pd.DataFrame.from_items(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_items(TimeZoneLine)
        else:
            TimeZoneRow = pd.DataFrame.from_items(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!! ##############################################')
```

You have built your first hub and satellites for time in the data vault.

The data vault has been built in directory ..\ VKHCG\88-DV\datavault.db. You can access it with your SQLite tools

Golden Nominal

A golden nominal record is a single person's record, with distinctive references for use by all systems. This gives the system a single view of the person. I use first name, other names, last name, and birth date as my golden nominal. The data we have in the assess directory requires a birth date to become a golden nominal. The proram will generate a golden nominal using our sample data set.

Open your Python editor and create a file called Process-People.py in the ..

```
C:\VKHCG\04-Clark\03-Process directory.
################################################################
import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
from datetime import datetime, timedelta
from pytz import timezone, all_timezones
from random import randint
import uuid
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
Company='04-Clark'
sInputFileName='02-Assess/01-EDS/02-Python/Assess_People.csv'
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
   os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/clark.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)
### Import Female Data
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
print(sFileName)
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)

start_date = datetime(1900,1,1,0,0,0)
start_date_utc=start_date.replace(tzinfo=timezone('UTC'))

HoursBirth=100*365*24
RawData['BirthDateUTC']=RawData.apply(lambda row:
            (start_date_utc + timedelta(hours=randint(0, HoursBirth)))
            ,axis=1)
zonemax=len(all_timezones)-1
RawData['TimeZone']=RawData.apply(lambda row:
            (all_timezones[randint(0, zonemax)])
            ,axis=1)
RawData['BirthDateISO']=RawData.apply(lambda row:
            row["BirthDateUTC"].astimezone(timezone(row['TimeZone']))
            ,axis=1)
RawData['BirthDateKey']=RawData.apply(lambda row:
            row["BirthDateUTC"].strftime("%Y-%m-%d %H:%M:%S")
            ,axis=1)
RawData['BirthDate']=RawData.apply(lambda row:
            row["BirthDateISO"].strftime("%Y-%m-%d %H:%M:%S")
            ,axis=1)
RawData['PersonID']=RawData.apply(lambda row:
```

```
            str(uuid.uuid4())
            ,axis=1)
Data=RawData.copy()
Data.drop('BirthDateUTC', axis=1,inplace=True)
Data.drop('BirthDateISO', axis=1,inplace=True)
indexed_data = Data.set_index(['PersonID'])
print('################################')
print('################')
sTable='Process_Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_data.to_sql(sTable, conn1, if_exists="replace")
print('################')
PersonHubRaw=Data[['PersonID','FirstName','SecondName','LastName','BirthDateKey']]
PersonHubRaw['PersonHubID']=RawData.apply(lambda row:
            str(uuid.uuid4())
            ,axis=1)
PersonHub=PersonHubRaw.drop_duplicates(subset=None, \
                                       keep='first',\
                                       inplace=False)
indexed_PersonHub = PersonHub.set_index(['PersonHubID'])
sTable = 'Hub-Person'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonHub.to_sql(sTable, conn2, if_exists="replace")
PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondName','LastName'\
                    ,'BirthDateKey','Gender']]
PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:
            str(uuid.uuid4())
            ,axis=1)
PersonSatelliteGender=PersonSatelliteGenderRaw.drop_duplicates(subset=None, \
                                                 keep='first', \
                                                 inplace=False)
indexed_PersonSatelliteGender = PersonSatelliteGender.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Gender'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteGender.to_sql(sTable, conn2, if_exists="replace")
################################################################
PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondName','LastName',\
                     'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row:
            str(uuid.uuid4())
            ,axis=1)
PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop_duplicates(subset=None, \
                                                   keep='first',\
                                                   inplace=False)
indexed_PersonSatelliteBirthday = PersonSatelliteBirthday.set_index(['PersonSatelliteID'])
sTable = 'Satellite-Person-Names'
print('Storing :',sDatabaseName,' Table:',sTable)
indexed_PersonSatelliteBirthday.to_sql(sTable, conn2, if_exists="replace")
sFileDir=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir):
   os.makedirs(sFileDir)
sOutputFileName = sTable + '.csv'
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
print('################')
print('Vacuum Databases')
```

```
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
print('### Done!! ##############################################')
```

Output :
It will apply golden nominal rules by assuming nobody born before January 1, 1900, droping to two ISO complex date time structures, as the code does not translate into SQLite's data types and saves your new golden nominal to a CSV file.

Load the person into the data vault

```
========== RESTART: C:\VKHCG\04-Clark\03-Process\Process-People.py ==========
Working Base : C:/VKHCG  using  win32
Loading : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
Storing : C:/VKHCG/88-DV/datavault.db  Table: Process_Person
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Gender
Storing : C:/VKHCG/88-DV/datavault.db  Table: Satellite-Person-Names
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Satellite-Person-Names.csv
Vacuum Databases
################
### Done!! ##############################################
```

Vehicles
The international classification of vehicles is a complex process. There are standards, but these are not universally applied or similar between groups or countries.
Let's load the vehicle data for Hillman Ltd into the data vault, as we will need it later. Create a new file named Process-Vehicle-Logistics.py in the Python editor in directory ..\VKHCG\03-Hillman\03-Process.

```
# -*- coding: utf-8 -*-
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
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/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!! ##############################################')
#####################################################################
```

Human-Environment Interaction

In the Python editor, open a new file named Process_Location.py in directory ..\VKHCG\01-Vermeulen\03-Process.

```
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
Base='C:/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_items(LocationLine)
      else:
         LocationRow = pd.DataFrame.from_items(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!! ##############################################')
```

Forecasting

Forecasting is the ability to project a possible future, by looking at historical data. The datavault enables these types of investigations, owing to the complete history it collects as itprocesses the source's systems data. A data scientist supply answers to such questions as the following:

- What should we buy?
- What should we sell?
- Where will our next business come from?

People want to know what you calculate to determine what is about to happen.
Open a new file in your Python editor and save it as Process-Shares-Data.py in directory C: \VKHCG\04-Clark\03-Process. I will guide you through this process. You will require a library called quandl

type pip install quandl in cmd

```
################################################################
import sys
import os
import sqlite3 as sq
import quandl
import pandas as pd
################################################################
Base='C:/VKHCG'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
Company='04-Clark'
sInputFileName='00-RawData/VKHCG_Shares.csv'
sOutputFileName='Shares.csv'
################################################################
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
################################################################
sFileDir1=Base + '/' + Company + '/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir1):
    os.makedirs(sFileDir1)
################################################################
sFileDir2=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir2):
    os.makedirs(sFileDir2)
################################################################
sFileDir3=Base + '/' + Company + '/03-Process/01-EDS/02-Python'
if not os.path.exists(sFileDir3):
    os.makedirs(sFileDir3)
################################################################
sDatabaseName=sDataBaseDir + '/clark.db'
conn = sq.connect(sDatabaseName)
################################################################
### Import Share Names Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
RawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
RawData.drop_duplicates(subset=None, keep='first', inplace=True)
print('Rows   :',RawData.shape[0])
print('Columns:',RawData.shape[1])
print('################')
################################################################
```

```
sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
################################################################
sFileName=sFileDir2 + '/Assess_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
################################################################
sFileName=sFileDir3 + '/Process_' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
RawData.to_csv(sFileName, index = False)
print('################################')
################################################################
### Import Shares Data Details
nShares=RawData.shape[0]
#nShares=6
for sShare in range(nShares):
    sShareName=str(RawData['Shares'][sShare])
    ShareData = quandl.get(sShareName)
    UnitsOwn=RawData['Units'][sShare]
    ShareData['UnitsOwn']=ShareData.apply(lambda row:(UnitsOwn),axis=1)
    ShareData['ShareCode']=ShareData.apply(lambda row:(sShareName),axis=1)
    print('################')
    print('Share  :',sShareName)
    print('Rows   :',ShareData.shape[0])
    print('Columns:',ShareData.shape[1])
    print('################')
    ##################################################################
    print('################')
    sTable=str(RawData['sTable'][sShare])
    print('Storing :',sDatabaseName,' Table:',sTable)
    ShareData.to_sql(sTable, conn, if_exists="replace")
    print('################')
    ################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir1 + '/Retrieve_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    print('################################')
    ################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir2 + '/Assess_' + sOutputFileName
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    print('################################')
    ################################################################
    sOutputFileName = sTable.replace("/","-") + '.csv'
    sFileName=sFileDir3 + '/Process_' + sOutputFileName
```

```
    print('################################')
    print('Storing :', sFileName)
    print('################################')
    ShareData.to_csv(sFileName, index = False)
    print('################################')
print('### Done!! ##############################################')
##################################################################
```

Output:

```
======== RESTART: C:\VKHCG\04-Clark\03-Process\Process-Shares-Data.py ========
Working Base : C:/VKHCG  using  win32
Loading : C:/VKHCG/04-Clark/00-RawData/VKHCG_Shares.csv
Rows   : 10
Columns: 3
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_Shares.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_Shares.csv
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Process_Shares.csv
Share  : WIKI/GOOGL
Rows   : 3424
Columns: 14
Storing : C:/VKHCG/04-Clark/03-Process/SQLite/clark.db  Table: WIKI_Google
Storing : C:/VKHCG/04-Clark/01-Retrieve/01-EDS/02-Python/Retrieve_WIKI_Google.csv
Storing : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_WIKI_Google.
```

Practical 7:
Transforming Data

Transform Superstep

C: \VKHCG\01-Vermeulen\04-Transform.

```
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='C:/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_items(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_items(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 : Guðmundur Gunnarsson was born on December 20, 1960, at 9:15 in Landspítali,Hringbraut 101, 101 Reykjavík, Iceland.

```
>>>
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson_is_Born.py
Working Base : C:/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 : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Time-Gunnarsson
##################################
##################################
Storing : C:/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 : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Person-Gunnarsson
##################################
```

You must build three items: **dimension Person**, **dimension Time**, and **factPersonBornAtTime**.
Open your Python editor and create a file named Transform-Gunnarsson-Sun-Model.py in directory C:\VKHCG\01-Vermeulen\04-Transform.

```
#####################################################################
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
################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/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_items(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_items(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_items(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:

Building a Data Warehouse

Open the Transform-Sun-Models.py file from directory C:\VKHCG\01-Vermeulen\04-Transform.

```
#######################################################################
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
################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + '/VKHCG'
else:
   Base='C:/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)
################################################################
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)
################################################################
sSQL=" SELECT DateTimeValue FROM [Hub-Time];"
DateDataRaw=pd.read_sql_query(sSQL, conn2)
DateData=DateDataRaw.head(1000)
print(DateData)
################################################################
print('\n#################################')
print('Time Dimension')
print('\n#################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
   BirthZone = ('Atlantic/Reykjavik','Europe/London','UCT')
   for j in range(len(BirthZone)):
      t+=1
      print(t,mt*3)
      BirthDateUTC = datetime.strptime(DateData['DateTimeValue'][i],"%Y-%m-%d %H:%M:%S")
      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[j]))
      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', [str(IDTimeNumber)]),
            ('UTCDate', [str(BirthDateZoneStr)]),
            ('LocalTime', [str(BirthDateLocal)]),
            ('TimeZone', [str(BirthZone)])]
      if t==1:
         TimeFrame = pd.DataFrame.from_items(TimeLine)
      else:
         TimeRow = pd.DataFrame.from_items(TimeLine)
         TimeFrame=TimeFrame.append(TimeRow)
################################################################
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, conn3, if_exists="replace")
```

```
#################################################################
sSQL=" SELECT " + \
     " FirstName," + \
     " SecondName," + \
     " LastName," + \
     " BirthDateKey " + \
     " FROM [Hub-Person];"
PersonDataRaw=pd.read_sql_query(sSQL, conn2)
PersonData=PersonDataRaw.head(1000)
#################################################################
print('\n#################################')
print('Dimension Person')
print('\n#################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
   t+=1
   print(t,mt)
   FirstName = str(PersonData["FirstName"])
   SecondName = str(PersonData["SecondName"])
   if len(SecondName) > 0:
      SecondName=""
   LastName = str(PersonData["LastName"])
   BirthDateKey = str(PersonData["BirthDateKey"])
   ################################################################
   IDPersonNumber=str(uuid.uuid4())
   PersonLine=[('PersonID', [str(IDPersonNumber)]),
               ('FirstName', [FirstName]),
               ('SecondName', [SecondName]),
               ('LastName', [LastName]),
               ('Zone', [str('UTC')]),
               ('BirthDate', [BirthDateKey])]
   if t==1:
      PersonFrame = pd.DataFrame.from_items(PersonLine)
   else:
      PersonRow = pd.DataFrame.from_items(PersonLine)
      PersonFrame = PersonFrame.append(PersonRow)
#################################################################
DimPerson=PersonFrame
print(DimPerson)
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, conn3, if_exists="replace")
#################################################################
```

Output:

You have successfully performed data vault to data warehouse transformation.

Simple Linear Regression

Linear regression is used if there is a relationship or significant association between the variables. This can be checked by scatterplots. If no linear association appears between the variables, fitting a linear regression model to the data will not provide a useful model.

A linear regression line has equations in the following form:

$Y = a + bX$,

Where, X = explanatory variable and

Y = dependent variable
b = slope of the line
a = intercept (the value of y when x = 0)

```
#################################################################
# -*- coding: utf-8 -*-
#################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
import matplotlib.pyplot as plt
import numpy as np

from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
#################################################################
Base='C:/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)
#################################################################
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)
#################################################################
t=0
tMax=((300-100)/10)*((300-30)/5)
for heightSelect in range(100,300,10):
    for weightSelect in range(30,300,5):
        height = round(heightSelect/100,3)
        weight = int(weightSelect)
        bmi = weight/(height*height)
        if bmi <= 18.5:
            BMI_Result=1
        elif bmi > 18.5 and bmi < 25:
            BMI_Result=2
        elif bmi > 25 and bmi < 30:
            BMI_Result=3
        elif bmi > 30:
```

```
        BMI_Result=4
    else:
        BMI_Result=0
    PersonLine=[('PersonID', [str(t)]),
            ('Height', [height]),
            ('Weight', [weight]),
            ('bmi', [bmi]),
            ('Indicator', [BMI_Result])]
    t+=1
    print('Row:',t,'of',tMax)
    if t==1:
        PersonFrame = pd.DataFrame.from_items(PersonLine)
    else:
        PersonRow = pd.DataFrame.from_items(PersonLine)
        PersonFrame = PersonFrame.append(PersonRow)
################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
################################################################
sTable = 'Transform-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
################################################################
################################################################
sTable = 'Person-Satellite-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
################################################################
sTable = 'Dim-BMI'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
################################################################
fig = plt.figure()
PlotPerson=DimPerson[DimPerson['Indicator']==1]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, ".")
PlotPerson=DimPerson[DimPerson['Indicator']==2]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "o")
PlotPerson=DimPerson[DimPerson['Indicator']==3]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "+")
PlotPerson=DimPerson[DimPerson['Indicator']==4]
x=PlotPerson['Height']
y=PlotPerson['Weight']
plt.plot(x, y, "^")
plt.axis('tight')
plt.title("BMI Curve")
plt.xlabel("Height(meters)")
plt.ylabel("Weight(kg)")
```

```
plt.plot()


# Load the diabetes dataset
diabetes = datasets.load_diabetes()

# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-30]
diabetes_X_test = diabetes_X[-50:]
diabetes_y_train = diabetes.target[:-30]
diabetes_y_test = diabetes.target[-50:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f"
    % mean_squared_error(diabetes_y_test, diabetes_y_pred))
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))
plt.scatter(diabetes_X_test, diabetes_y_test, color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.axis('tight')
plt.title("Diabetes")
plt.xlabel("BMI")
plt.ylabel("Age")
plt.show()
```

Output:

```
Row: 1077 of 1080.0
Row: 1078 of 1080.0
Row: 1079 of 1080.0
Row: 1080 of 1080.0

##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Transform-BMI

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

##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Person-Satellite-BMI

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

##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Dim-BMI

##################################
>>>
```

Practical 8:

Organizing Data

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Horizontal.py

```
################################################################
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
Base='C:/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:

Vertical Style

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Vertical.py

```
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
Base='C:/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)
print('################################')
sSQL="SELECT \
      Height,\
      Weight,\
      Indicator\
  FROM [Dim-BMI];"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
################################################################
sTable = 'Dim-BMI-Vertical'
print('\n################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
print('################')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('################################')
################################################################
```

Output:

```
====== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Vertical.py ======
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################

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

################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Vertical
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 1080
Horizontal Data Set (Columns): 3
################################
```

Island Style

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Island.py

```
import sys
import os
import pandas as pd
import sqlite3 as sq
################################################################
Base='C:/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 \
      Height,\
      Weight,\
      Indicator\
  FROM [Dim-BMI]\
  WHERE Indicator > 2\
  ORDER BY  \
      Height,\
      Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
################################################################
sTable = 'Dim-BMI-Vertical'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
################################################################
print('################################')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
print('################################')
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
################################################################
print('################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('################################')
################################################################
```

Output:

Secure Vault Style

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Secure-Vault.py

```
import sys
import os
import pandas as pd
import sqlite3 as sq
#################################################################
Base='C:/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 \
      Height,\
      Weight,\
      Indicator,\
      CASE Indicator\
      WHEN 1 THEN 'Pip'\
      WHEN 2 THEN 'Norman'\
      WHEN 3 THEN 'Grant'\
      ELSE 'Sam'\
```

```
    END AS Name\
  FROM [Dim-BMI]\
  WHERE Indicator > 2\
  ORDER BY  \
    Height,\
    Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#################################################################
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#################################################################
sTable = 'Dim-BMI-Secure'
print('\n##################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n##################################')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#################################################################
print('##################################')
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
print('##################################')
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#################################################################
print('##################################')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('##################################')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('Only Sam Data')
print(PersonFrame2.head())
print('##################################')
#################################################################
```

Output:

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
>>>
==== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Secure-Vault.py ====
################################
Working Base : C:/VKHCG  using  win32
################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

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

################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Secure
################################
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 692
Horizontal Data Set (Columns): 4
Only Sam Data
   Indicator  Height  Weight Name
0          4     1.0      35  Sam
1          4     1.0      40  Sam
2          4     1.0      45  Sam
3          4     1.0      50  Sam
4          4     1.0      55  Sam
################################
```

Association Rule Mining

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Association-Rule.py

```
import sys
import os
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

```
#################################################################
Base='C:/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:

```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Association-Rule.py ==
##############################
Working Base : C:/VKHCG  using  win32
##############################
(541909, 8)
                    antecedents  ... conviction
0   (ALARM CLOCK BAKELIKE PINK)  ...   3.283859
1  (ALARM CLOCK BAKELIKE GREEN)  ...   3.791383
2  (ALARM CLOCK BAKELIKE GREEN)  ...   4.916181
3    (ALARM CLOCK BAKELIKE RED)  ...   5.568878
4   (ALARM CLOCK BAKELIKE PINK)  ...   3.293135

[5 rows x 9 columns]
ALARM CLOCK BAKELIKE GREEN
340.0
ALARM CLOCK BAKELIKE RED
316.0
                         antecedents  ... conviction
0   (PLASTERS IN TIN CIRCUS PARADE)  ...   2.076984
7        (PLASTERS IN TIN SPACEBOY)  ...   2.011670
11    (RED RETROSPOT CHARLOTTE BAG)  ...   5.587746

[3 rows x 9 columns]
### Done!! ##############################################
>>> |
Ln: 28  Col: 4
```

Create a Network Routing Diagram

I will guide you through a possible solution for the requirement, by constructing an island-style Organize superstep that uses a graph data model to reduce the records and the columns on the data set.

C:\VKHCG\01-Vermeulen\05-Organise\ Organise-Network-Routing-Company.py

```
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
#################################################################
pd.options.mode.chained_assignment = None
#################################################################
Base='C:/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('#################################')
```

Picking Content for Billboards

C:\VKHCG\02-Krennwallner\05-Organise\ Organise-billboards.py

```
################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
################################################################
pd.options.mode.chained_assignment = None
################################################################
Base='C:/VKHCG'
################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-DE-Billboard-Visitor.csv'
################################################################
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Billboards.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Billboards.png'
Company='02-Krennwallner'
################################################################
################################################################
### Import Company Data
################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
BillboardDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('################################')
################################################################
print(BillboardDataRaw.head())
print(BillboardDataRaw.shape)
BillboardData=BillboardDataRaw
sSample=list(np.random.choice(BillboardData.shape[0],20))
################################################################
G=nx.Graph()
for i in sSample:
    for j in sSample:
        Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['BillboardCountry'][i] + ')'
        Node1=BillboardData['BillboardPlaceName'][j] + '('+ BillboardData['BillboardCountry'][i] + ')'
        if Node0 != Node1:
            G.add_edge(Node0,Node1)

for i in sSample:
    Node0=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorPlaceName'][i] + ')'
    Node1=BillboardData['BillboardPlaceName'][i] + '('+ BillboardData['VisitorCountry'][i] + ')'
    if Node0 != Node1:
        G.add_edge(Node0,Node1)

print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
################################################################
sFileName=Base + '/02-Krennwallner/' + sOutputFileName1
print('################################')
print('Storing :',sFileName)
```

```
print('################################')
nx.write_gml(G, sFileName)
#################################################################
sFileName=Base + '/02-Krennwallner/' + sOutputFileName2
print('################################')
print('Storing Graph Image:',sFileName)
print('################################')
plt.figure(figsize=(15, 15))
pos=nx.circular_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=150, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='solid')
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 :

Create a Delivery Route C:\VKHCG\03-Hillman\05-Organise\Organise-Routes.py

```
import sys
import os
import pandas as pd
#################################################################
Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.txt'
#################################################################
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Routes.csv'
Company='03-Hillman'
#################################################################
#################################################################
### Import Routes Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
```

```
print('Loading :',sFileName)
print('################################')
RouteDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, sep='|', encoding="latin-1")
print('################################')
#################################################################
RouteStart=RouteDataRaw[RouteDataRaw['StartAt']=='WH-KA13']
#################################################################
RouteDistance=RouteStart[RouteStart['Cost']=='DistanceMiles']
RouteDistance=RouteDistance.sort_values(by=['Measure'], ascending=False)
#################################################################
RouteMax=RouteStart["Measure"].max()
RouteMaxCost=round((((RouteMax/1000)*1.5*2)),2)
print('################################')
print('Maximum (£) per day:')
print(RouteMaxCost)
print('################################')
#################################################################
RouteMean=RouteStart["Measure"].mean()
RouteMeanMonth=round((((RouteMean/1000)*2*30)),6)
print('################################')
print('Mean per Month (Miles):')
print(RouteMeanMonth)
print('################################')
```

Output:

Clark Ltd

Our financial services company has been tasked to investigate the options to convert1 million pounds sterling into extra income. Mr. Clark Junior suggests using the simplevariance in the daily rate between the British pound sterling and the US dollar, togenerate extra income from trading. Your chief financial officer wants to know if this isfeasible?

Simple Forex Trading Planner

Your challenge is to take 1 million US dollars or just over six hunderd thou sand pounds sterling and, by simply converting it between pounds sterling and US dollars, achieve a profit. Are you up to this challenge?

The Program will help you how to model this problem and achieve a positive outcome. The forex data has been collected on a daily basis by Clark's accounting department, from previous overseas transactions.

C:\VKHCG\04-Clark\05-Organise\Organise-Forex.py

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
#################################################################
Base='C:/VKHCG'
#################################################################
```

```
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
sInputFileName='03-Process/01-EDS/02-Python/Process_ExchangeRates.csv'
#################################################################
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Forex.csv'
Company='04-Clark'
#################################################################
sDatabaseName=Base + '/' + Company + '/05-Organise/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
#################################################################
#################################################################
### Import Forex Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('################################')
#################################################################
ForexDataRaw.index.names = ['RowID']
sTable='Forex_All'
print('Storing :',sDatabaseName,' Table:',sTable)
ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
#################################################################
sSQL="SELECT 1 as Bag\
      , CAST(min(Date) AS VARCHAR(10)) as Date \
      ,CAST(1000000.0000000 as NUMERIC(12,4)) as Money \
      ,'USD' as Currency \
      FROM Forex_All \
      ;"
sSQL=re.sub("\s\s+", " ", sSQL)
nMoney=pd.read_sql_query(sSQL, conn)

#################################################################
nMoney.index.names = ['RowID']
sTable='MoneyData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
#################################################################
sTable='TransactionData'
print('Storing :',sDatabaseName,' Table:',sTable)
nMoney.to_sql(sTable, conn, if_exists="replace")
#################################################################
ForexDay=pd.read_sql_query("SELECT Date FROM Forex_All GROUP BY Date;", conn)
#################################################################
t=0
for i in range(ForexDay.shape[0]):
   sDay1=ForexDay['Date'][i]
   sDay=str(sDay1)
   sSQL='\
   SELECT M.Bag as Bag, \
         F.Date as Date, \
         round(M.Money * F.Rate,6) AS Money, \
         F.CodeIn AS PCurrency, \
         F.CodeOut AS Currency \
   FROM MoneyData AS M \
```

```
    JOIN \
    ( \
       SELECT  CodeIn, CodeOut, Date, Rate   FROM  Forex_All WHERE
      CodeIn = "USD" AND CodeOut = "GBP" \
       UNION \
       SELECT  CodeOut AS CodeIn, CodeIn AS CodeOut,  Date, (1/Rate) AS Rate  FROM \
       Forex_All   WHERE   CodeIn = "USD" AND CodeOut = "GBP" \
    ) AS F \
    ON \
    M.Currency=F.CodeIn \
    AND \
    F.Date ="' +sDay + '";'
    sSQL=re.sub("\s\s+", " ", sSQL)

    ForexDayRate=pd.read_sql_query(sSQL, conn)
    for j in range(ForexDayRate.shape[0]):
       sBag=str(ForexDayRate['Bag'][j])
       nMoney=str(round(ForexDayRate['Money'][j],2))
       sCodeIn=ForexDayRate['PCurrency'][j]
       sCodeOut=ForexDayRate['Currency'][j]

    sSQL='UPDATE MoneyData SET Date= '" + sDay + "', '
    sSQL= sSQL + ' Money = ' + nMoney + ', Currency="' + sCodeOut + '"'
    sSQL= sSQL + ' WHERE Bag=' + sBag + ' AND Currency="' + sCodeIn + '";'

    sSQL=re.sub("\s\s+", " ", sSQL)
    cur = conn.cursor()
    cur.execute(sSQL)
    conn.commit()
    t+=1
    print('Trade :', t, sDay, sCodeOut, nMoney)

    sSQL=' \
    INSERT INTO TransactionData ( \
                        RowID, \
                        Bag, \
                        Date, \
                        Money, \
                        Currency \
                     )  \
    SELECT ' + str(t) + ' AS RowID, \
      Bag, \
      Date, \
      Money, \
      Currency \
    FROM MoneyData  ;'
    sSQL=re.sub("\s\s+", " ", sSQL)
    cur = conn.cursor()
    cur.execute(sSQL)
    conn.commit()
#################################################################
sSQL="SELECT RowID, Bag, Date, Money, Currency FROM TransactionData ORDER BY RowID;"
sSQL=re.sub("\s\s+", " ", sSQL)
TransactionData=pd.read_sql_query(sSQL, conn)
OutputFile=Base + '/' + Company + '/' + sOutputFileName
TransactionData.to_csv(OutputFile, index = False)
#################################################################
```

Output:

Save the Assess-Forex.py file, then compile and execute with your Python compiler.
This will produce a set of demonstrated values onscreen.

Practical 9

Generating Data

Report Superstep

The Report superstep is the step in the ecosystem that enhances the data science findings with the art of storytelling and data visualization. You can perform the best data science, but if you cannot execute a respectable and trustworthy Report step by turning your data science into actionable business insights, you have achieved no advantage for your business.

Vermeulen PLC

Vermeulen requires a map of all their customers' data links. Can you provide a report to deliver this? I will guide you through an example that delivers this requirement.

C:\VKHCG\01-Vermeulen\06-Report\Raport-Network-Routing-Customer.py

```
#####################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
#################################################################
pd.options.mode.chained_assignment = None
#################################################################
if sys.platform == 'linux':
   Base=os.path.expanduser('~') + 'VKHCG'
else:
   Base='C:/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())
print(CustomerData.shape)
#################################################################
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('################################')
```

Krennwallner AG

The Krennwallner marketing department wants to deploy the locations of the billboards onto the company web server. Can you prepare three versions of the locations' web pages?

- Locations clustered into bubbles when you zoom out
- Locations as pins
- Locations as heat map

Picking Content for Billboards

C:\VKHCG\02-Krennwallner\06-Report\Report_Billboard.py

```
import sys
import os
import pandas as pd
```

```
from folium.plugins import FastMarkerCluster, HeatMap
from folium import Marker, Map
import webbrowser
##################################################################
Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
##################################################################
sFileName=Base+'/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
df = pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
df.fillna(value=0, inplace=True)
print(df.shape)
##################################################################
t=0
for i in range(df.shape[0]):
    try:
        sLongitude=df["Longitude"][i]
        sLongitude=float(sLongitude)
    except Exception:
        sLongitude=float(0.0)
   try:
        sLatitude=df["Latitude"][i]
        sLatitude=float(sLatitude)
    except Exception:
        sLatitude=float(0.0)
try:
        sDescription=df["Place_Name"][i] + ' (' + df["Country"][i]+')'
    except Exception:
        sDescription='VKHCG'
    if sLongitude != 0.0 and sLatitude != 0.0:
        DataClusterList=list([sLatitude, sLongitude])
        DataPointList=list([sLatitude, sLongitude, sDescription])
        t+=1
        if t==1:
            DataCluster=[DataClusterList]
            DataPoint=[DataPointList]
        else:
            DataCluster.append(DataClusterList)
            DataPoint.append(DataPointList)
data=DataCluster
pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
##################################################################
stops_map1 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
marker_cluster = FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard1.html'
stops_map1.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
##################################################################
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
for name, row in pins.iloc[:100].iterrows():
    Marker([row["Latitude"],row["Longitude"]], popup=row["Description"]).add_to(stops_map2)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard2.html'
stops_map2.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
##################################################################
stops_heatmap = Map(location=[48.1459806, 11.4985484], zoom_start=5)
stops_heatmap.add_child(HeatMap([[row["Latitude"], row["Longitude"]] for name, row in
pins.iloc[:100].iterrows()]))
```

```
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-Python/Billboard_heatmap.html'
stops_heatmap.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
##################################################################
print('### Done!! ##############################################')
##################################################################
```

Output:

Hillman Ltd
Dr. Hillman Sr. has just installed a camera system that enables the company to capture video and, therefore, indirectly, images of all containers that enter or leave the warehouse. Can you convert the number on the side of the containers into digits?

Reading the Containers
C:\VKHCG\03-Hillman\06-Report\ Report_Reading_Container.py

```
from time import time
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble, discriminant_analysis, random_projection)
digits = datasets.load_digits(n_class=6)
X = digits.data
y = digits.target
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.),
                fontdict={'weight': 'bold', 'size': 9})
    if hasattr(offsetbox, 'AnnotationBbox'):
        # only print thumbnails with matplotlib > 1.0
        shown_images = np.array([[1., 1.]]) # 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_transform(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()
```

A selection from the 64-dimensional digits dataset

Linear Discriminant projection of the digits (time 0.07s)

Isomap projection of the digits (time 1.83s)

Locally Linear Embedding of the digits (time 0.80s)

Modified Locally Linear Embedding of the digits (time 2.10s)

Clark Ltd

The financial company in VKHCG is the Clark accounting firm that VKHCG owns with a 60% stake. The accountants are the financial advisers to the group and handle everything to do with the complex work of international accounting.

Financials

The VKHCG companies did well last year, and the teams at Clark must prepare a balance sheet for each company in the group. The companies require a balance sheet for each company, to be produced using the template (Balance-Sheet-Template.xlsx) that can be found in the example directory (..\VKHCG\04-Clark\00-RawData).

The Program will guide you through a process that will enable you to merge the data science with preformatted Microsoft Excel template, to produce a balance sheet for each of the VKHCG companies.

C:\VKHCG\04-Clark\06-Report\Report-Balance-Sheet.py

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
from openpyxl import load_workbook
#################################################################
Base='C:/VKHCG'
#################################################################
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
#################################################################
sInputTemplateName='00-RawData/Balance-Sheet-Template.xlsx'
#################################################################
sOutputFileName='06-Report/01-EDS/02-Python/Report-Balance-Sheet'
```

```
Company='04-Clark'
##################################################################
sDatabaseName=Base + '/' + Company + '/06-Report/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
##################################################################
### Import Balance Sheet Data
##################################################################
for y in range(1,13):
   sInputFileName='00-RawData/BalanceSheets' + str(y).zfill(2) + '.csv'
   sFileName=Base + '/' + Company + '/' + sInputFileName
   print('#################################')
   print('Loading :',sFileName)
   print('#################################')
   ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
   print('#################################')
   ##################################################################
   ForexDataRaw.index.names = ['RowID']
   sTable='BalanceSheets'
   print('Storing :',sDatabaseName,' Table:',sTable)
   if y == 1:
      print('Load Data')
      ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
   else:
      print('Append Data')
      ForexDataRaw.to_sql(sTable, conn, if_exists="append")
##################################################################
sSQL="SELECT \
         Year, \
         Quarter, \
         Country, \
         Company, \
         CAST(Year AS INT) || 'Q' || CAST(Quarter AS INT) AS sDate, \
         Company || ' (' || Country || ')' AS sCompanyName , \
         CAST(Year AS INT) || 'Q' || CAST(Quarter AS INT) || '-' ||\
         Company || '-' || Country AS sCompanyFile \
      FROM BalanceSheets \
      GROUP BY \
         Year, \
         Quarter, \
         Country, \
         Company \
      HAVING Year is not null \
     ;"
sSQL=re.sub("\s\s+", " ", sSQL)
sDatesRaw=pd.read_sql_query(sSQL, conn)
print(sDatesRaw.shape)
sDates=sDatesRaw.head(5)
##################################################################
## Loop Dates
##################################################################
for i in range(sDates.shape[0]):
   sFileName=Base + '/' + Company + '/' + sInputTemplateName
   wb = load_workbook(sFileName)
   ws=wb.get_sheet_by_name("Balance-Sheet")
   sYear=sDates['sDate'][i]
   sCompany=sDates['sCompanyName'][i]
   sCompanyFile=sDates['sCompanyFile'][i]
   sCompanyFile=re.sub("\s+", "", sCompanyFile)
```

```
ws['D3'] = sYear
ws['D5'] = sCompany

sFields = pd.DataFrame(
        [
       ['Cash','D16', 1],
       ['Accounts_Receivable','D17', 1],
       ['Doubtful_Accounts','D18', 1],
       ['Inventory','D19', 1],
       ['Temporary_Investment','D20', 1],
       ['Prepaid_Expenses','D21', 1],
       ['Long_Term_Investments','D24', 1],
       ['Land','D25', 1],
       ['Buildings','D26', 1],
       ['Depreciation_Buildings','D27', -1],
       ['Plant_Equipment','D28', 1],
       ['Depreciation_Plant_Equipment','D29', -1],
       ['Furniture_Fixtures','D30', 1],
       ['Depreciation_Furniture_Fixtures','D31', -1],
       ['Accounts_Payable','H16', 1],
       ['Short_Term_Notes','H17', 1],
       ['Current_Long_Term_Notes','H18', 1],
       ['Interest_Payable','H19', 1],
       ['Taxes_Payable','H20', 1],
       ['Accrued_Payroll','H21', 1],
       ['Mortgage','H24', 1],
       ['Other_Long_Term_Liabilities','H25', 1],
       ['Capital_Stock','H30', 1]
       ]
        )

nYear=str(int(sDates['Year'][i]))
nQuarter=str(int(sDates['Quarter'][i]))
sCountry=str(sDates['Country'][i])
sCompany=str(sDates['Company'][i])

sFileName=Base + '/' + Company + '/' + sOutputFileName + \
'-' + sCompanyFile + '.xlsx'

print(sFileName)

for j in range(sFields.shape[0]):

    sSumField=sFields[0][j]
    sCellField=sFields[1][j]
    nSumSign=sFields[2][j]

    sSQL="SELECT  \
          Year, \
          Quarter, \
          Country, \
          Company, \
          SUM(" + sSumField + ") AS nSumTotal \
        FROM BalanceSheets \
        GROUP BY \
          Year, \
          Quarter, \
          Country, \
          Company \
        HAVING \
```

```
            Year=" + nYear + " \
        AND \
            Quarter=" + nQuarter + " \
        AND \
            Country='" + sCountry + "' \
        AND \
            Company='" + sCompany + "' \
      ;"
    sSQL=re.sub("\s\s+", " ", sSQL)
    sSumRaw=pd.read_sql_query(sSQL, conn)
    ws[sCellField] = sSumRaw["nSumTotal"][0] * nSumSign
    print('Set cell',sCellField,' to ', sSumField,'Total')
  wb.save(sFileName)
```

Output:

Graphics

This section will now guide you through a number of visualizations that particularly useful in presenting data to my customers.

Pie Graph

Double Pie

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py

Line Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py

Bar Graph / Horizontal Bar Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py

Area Graph
C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py

SCATTER GRAPH : VKHCG/03-HILLMAN/06-REPORT/REPORT-SCATTERPLOT-WITH-ENCIRCLING.R

Hexbin:
Program : C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py

Kernel Density Estimation (KDE) Graph
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py

Scatter Matrix Graph
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py

Andrews' Curves
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py

Parallel Coordinates
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py

RADVIZ Method
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py

Lag Plot
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py

Autocorrelation Plot
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py

Bootstrap Plot
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py

Contour Graphs
C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_G.py

3D Graphs
C:\VKHCG\01-Vermeulen\06-Report\Report_PCA_IRIS.py

Practical 10

Data Visualization with Power BI

Case Study : Sales Data

Step 1: Connect to an Excel workbook

1. Launch Power BI Desktop.
2. From the Home ribbon, select **Get Data**. Excel is one of the **Most Common** data connections, so you can select it directly from the **Get Data** menu.

3. If you select the Get Data button directly, you can also select **File > Excel** and select **Connect**.
4. In the **Open File** dialog box, select the **Products.xlsx** file.

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

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.

Step 3: Change the data type of the UnitsInStock column

For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the UnitsInStock column's datatype is Whole Number.

1. Select the UnitsInStock column.
2. 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).*

Task 2: Import order data from an OData feed

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/

Step 1: 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**.

Step 2: Expand the Order_Details table

Expand the **Order_Details** table that is related to the **Orders** table, to combine the **ProductID**, **UnitPrice**, and **Quantity** columns from **Order_Details** into the **Orders** table. 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 added to 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.
   Select **ProductID**, **UnitPrice**, and **Quantity**.
   click **OK**.

Step 3: Remove other columns to only display columns of interest

In this step you remove all columns except **OrderDate, ShipCity**, **ShipCountry**, **Order_Details.ProductID**, **Order_Details.UnitPrice**, and **Order_Details.Quantity** columns. In the previous task, you used **Remove Other Columns**. For this task, you remove selected columns.

In the **Query View**, select all columns by completing a.

a. Click the first column (**OrderID**).
b. Shift+Click the last column (**Shipper**).
c. Now that all columns are selected, use Ctrl+Click to unselect the following columns: OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity.

Now that only the columns we want to remove are selected, right-click on any selected column header and click Remove Columns.

Step 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.

Click **OK**.

Step 5: Set the datatype of the LineTotal field

1. Right click the **LineTotal** column.
2. Select **Change Type** and choose **Decimal Number.**

Step 6: Rename and reorder columns in the query

1. In **Query Editor**, drag the **LineTotal** column to the left, after **ShipCountry**.
2. Remove

2. Remove the *Order_Details.* prefix from the **Order_Details.ProductID**, **Order_Details.UnitPrice** and **Order_Details.Quantity** columns, by double-clicking on each column header, and then deleting that text from the column name.

Task 3: Combine the Products and Total Sales queries

2. Power BI Desktop loads the data from the two queries
3. Once the data is loaded, select the Manage Relationships button Home ribbon
4. Select the New… button

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

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

Task 4: Build visuals using your data

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

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 Values field; 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

~~~~~*****~~~~~
~~~~~

CLOUD COMPUTING

TABLE OF CONTENTS

No	Topic	Sign
1	Write a program for implementing Client Server communication model using TCP	
a	A client server based program using TCP to find if the number entered isprime.	
b	A client server TCP based chatting application.	
2	Write a program for implementing Client Server communication model using UDP	
a	A client server based program using UDP to find if the number entered is evenor odd.	
b	A client server based program using UDP to find the factorial of the enterednumber.	
c	A program to implement simple calculator operations like addition,subtraction, multiplication and division.	
d	A program that finds the square, square root, cube and cube root of the enterednumber.	
3	A multicast Socket example	
4	Write a program to show the object communication using RMI.	
a	A RMI based application program to display current date and time.	
b	A RMI based application program that converts digits to words, e.g. 123 willbe converted to one two three.	
5	Show the implementation of web services.	
6	Implement Xen virtualization and manage with Xen Center	
7	Implement virtualization using VMWare ESXi Server and managing with vCenter	
8	Implement Windows Hyper V virtualization	
9	Develop application for Microsoft Azure.	
10	Develop application for Google App Engine	

Practical No: 01

Aim: Write a program for implementing Client Server communication model using TCP.

Practical 1A: A client server based program using TCP to find if the number entered isprime.

Code

1. **tcpServerPrime.java**

```
import java.net.*;
import java.io.*;
classtcpServerPrime
{
	public static void main(String args[])
	{
		try
		{
			ServerSocketss = new ServerSocket(8001);
			System.out.println("Server Started.....");
			Socket s = ss.accept();
			DataInputStream in = new DataInputStream(s.getInputStream());
			int x= in.readInt();
			DataOutputStreamotc = new DataOutputStream(s.getOutputStream());
			int y = x/2;
			if(x ==1 || x ==2 || x ==3)
			{
				otc.writeUTF(x + "is Prime");
				System.exit(0);
			}
			for(inti=2; i<=y; i++)
			{
				if(x%i != 0)
				{
					otc.writeUTF(x + " is Prime");
				}
				else
				{
					otc.writeUTF(x + " is not Prime");
				}
			}
		}
		catch(Exception e)
		{
			System.out.println(e.toString());
		}
	}
}
```

2. **tcpClientPrime.java**

```
import java.net.*;
import java.io.*;
classtcpClientPrime
{
	public static void main(String args[])
	{
		try
		{
			Socket cs = new Socket("LocalHost",8001);
			BufferedReaderinfu = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Enter a number : ");
			int a = Integer.parseInt(infu.readLine());
			DataOutputStream out = new
DataOutputStream(cs.getOutputStream());
			out.writeInt(a);
			DataInputStream in = new
			DataInputStream(cs.getInputStream());
			System.out.println(in.readUTF()); cs.close();
		}
		catch(Exception e)
		{
			System.out.println(e.toString());
		}
	}
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac tcpServerPrime.java

E:\Ds_Yugi>java tcpServerPrime
Server Started...............

E:\Ds_Yugi>javac tcpServerPrime.java

E:\Ds_Yugi>java tcpServerPrime
Server Started...............
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac tcpClientPrime.java

E:\Ds_Yugi>java tcpClientPrime
Enter a number :
7
7 is Prime

E:\Ds_Yugi>javac tcpClientPrime.java

E:\Ds_Yugi>java tcpClientPrime
Enter a number :
8
8 is not Prime
```

Practical 1B: A client server TCP based chatting application.
Code:

1. **ChatServer.java**

```
import java.net.*;
import java.io.*;
classChatServer
{
	public static void main(String args[])
	{
		try
		{
			ServerSocketss = new ServerSocket(8000);
			System.out.println("Waiting for client to connect..");
			Socket s = ss.accept();
			BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
			DataOutputStream out = new DataOutputStream(s.getOutputStream());
			DataInputStream in = new DataInputStream(s.getInputStream());
			String receive, send;
			while((receive = in.readLine()) != null)
			{
				if(receive.equals("STOP"))
					break;
				System.out.println("Client Says : "+receive);
				System.out.print("Server Says : ");
				send = br.readLine();
				out.writeBytes(send+"\n");
			}
			br.close();
			in.close();
			out.close();
			s.close();
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

2. **ChatClient.java**

```
import java.net.*;
import java.io.*;
classChatClient

{
	public static void main(String args[])
```

```
        {
                try
                {
                        Socket s = new Socket("Localhost",8000);
                        BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
                        DataOutputStream out = new DataOutputStream(s.getOutputStream());
                        DataInputStream in = new DataInputStream(s.getInputStream());
                        String msg;
                        System.out.println("To stop chatting with server type
                        STOP"); System.out.print("Client Says: "); while((msg =
                        br.readLine()) != null)
                        {
                                out.writeBytes(msg+"\n");
                                if(msg.equals("STOP"))
                                        break;
                                System.out.println("Server Says : "+in.readLine());
                                System.out.print("Client Says : ");
                        }
                        br.close();
                        in.close();
                        out.close();
                        s.close();
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Output:

Server:

```
E:\Ds_Yugi>java ChatServer
Waiting for client to connect..
Client Says : Hii...
Server Says : Hello
Client Says : How are you?
Server Says : Fine, thank you..
```

```
E:\Ds_Yugi>java ChatClient
To stop chatting with server type STOP
Client Says: Hii...
Server Says : Hello
Client Says : How are you?
Server Says : Fine, thank you..
Client Says : STOP
```

Practical No: 02

Aim: Write a program for implementing Client Server communication model using UDP.

Practical 2A:A client server based program using UDP to find if the number entered is evenor odd.

Code:

1. **udpServerEO.java**

```
/*Program which finds entered number is even or odd */
import java.io.*;
import java.net.*;
public class udpServerEO
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(2000);
			byte b[] = new byte[1024];
			DatagramPacketdp = new DatagramPacket(b,b.length);
			ds.receive(dp);
			String str = new String(dp.getData(),0,dp.getLength());
			System.out.println(str);
			int a= Integer.parseInt(str);
			String s= new String();
			if (a%2 == 0)
				s = "Number is even";
			else
				s = "Number is odd";

			byte b1[] = new byte[1024];
			b1 = s.getBytes();
			DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1000);
			ds.send(dp1);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

2. **udpClientEO.java**

```
/*Program which finds entered number is even or odd*/
import java.io.*;
import java.net.*;
public class udpClientEO
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(1000);
			BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Enter a number : ");
			String num = br.readLine();
			byte b[] = new byte[1024];
			b=num.getBytes();
			DatagramPacketdp = new
DatagramPacket(b,b.length,InetAddress.getLocalHost(),2000);
			ds.send(dp);
			byte b1[] = new byte[1024];
			DatagramPacket dp1 = new
			DatagramPacket(b1,b1.length); ds.receive(dp1);
			String str = new String(dp1.getData(),0,dp1.getLength());
			System.out.println(str);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac udpClientEO.java

E:\Ds_Yugi>java udpClientEO
Enter a number :
24
Number is even

E:\Ds_Yugi>java udpClientEO
Enter a number :
11
Number is odd
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac udpServerEO.java

E:\Ds_Yugi>java udpServerEO
24

E:\Ds_Yugi>java udpServerEO
11
```

Practical 2B: A client server based program using UDP to find the factorial of the enterednumber.

Code:

1. **udpServerFact.java**

```
/*Program which calculate factorial of a number*/
import java.io.*;
import java.net.*;
public class udpServerFact
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(2000);
			byte b[] = new byte[1024];
			DatagramPacketdp = new DatagramPacket(b,b.length);
			ds.receive(dp);
			String str = new String(dp.getData(),0,dp.getLength());
			System.out.println(str);
			int a= Integer.parseInt(str);
			int f = 1, i;
			String s= new String();
			for(i=1;i<=a;i++)
			{
				f=f*i;
			}
			s=Integer.toString(f);

			String str1 = "The Factorial of " + str + " is : " + f;
			byte b1[] = new byte[1024]; b1 = str1.getBytes();

			DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1000);
			ds.send(dp1);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}

	}
}
```

2. **udpClientFact.java**

```
/*Program which calculate factorial of a number*/

import java.io.*;
import java.net.*;
public class udpClientFact
{
	public static void main(String args[])
	{
		try
		{
			DatagramSocket ds = new DatagramSocket(1000);
			BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
			System.out.println("Enter a number : ");
			String num = br.readLine();
			byte b[] = new byte[1024];
			b=num.getBytes();
			DatagramPacketdp = new
DatagramPacket(b,b.length,InetAddress.getLocalHost(),2000);
			ds.send(dp);
			byte b1[] = new byte[1024];
			DatagramPacket dp1 = new DatagramPacket(b1,b1.length);
			ds.receive(dp1);
			String str = new String(dp1.getData(),0,dp1.getLength());
			System.out.println(str);
		}
		catch(Exception e)
		{
			e.printStackTrace();
		}
	}
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac udpServerFact.java

E:\Ds_Yugi>java udpServerFact
9

E:\Ds_Yugi>java udpServerFact
5
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac udpClientFact.java

E:\Ds_Yugi>java udpClientFact
Enter a number :
9
The Factorial of 9 is : 362880

E:\Ds_Yugi>java udpClientFact
Enter a number :
5
The Factorial of 5 is : 120
```

Practical 2C: A program to implement simple calculator operations like addition,subtraction, multiplication and division.

Code:

1. **RPCServer.java**

```
importjava.util.*;
import java.net.*;
classRPCServer
{
	DatagramSocket ds;
	DatagramPacketdp;
	String str,methodName,result;
	int val1,val2;
	RPCServer()
	{
		try
		{
			ds=new DatagramSocket(1200);
			byte b[]=new byte[4096];
			while(true)
			{
				dp=new DatagramPacket(b,b.length);
				ds.receive(dp);
				str=new String(dp.getData(),0,dp.getLength());
				if(str.equalsIgnoreCase("q"))
				{
					System.exit(1);
				}
				else
				{
					StringTokenizerst = new StringTokenizer(str," ");
					inti=0;
					while(st.hasMoreTokens())
					{
						String token=st.nextToken();
						methodName=token;
						val1 = Integer.parseInt(st.nextToken());
						val2 = Integer.parseInt(st.nextToken());
					}
				}
				System.out.println(str);
				InetAddressia = InetAddress.getLocalHost();
				if(methodName.equalsIgnoreCase("add"))
				{
					result= "" + add(val1,val2);
				}
				else if(methodName.equalsIgnoreCase("sub"))
				{
					result= "" + sub(val1,val2);
				}
```

```
                                else if(methodName.equalsIgnoreCase("mul"))
                                {
                                        result= "" + mul(val1,val2);

                                }
                                else if(methodName.equalsIgnoreCase("div"))
                                {
                                        result= "" + div(val1,val2);
                                }
                                byte b1[]=result.getBytes();
                                DatagramSocket ds1 = new DatagramSocket();
                                DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
                                System.out.println("result :
                                "+result+"\n"); ds1.send(dp1);
                        }
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
        publicint add(int val1, int val2)
        {
                return val1+val2;
        }
        publicint sub(int val3, int val4)
        {
                return val3-val4;
        }

        publicintmul(int val3, int val4)
        {
                return val3*val4;
        }
        publicint div(int val3, int val4)
        {
                return val3/val4;
        }

        public static void main(String[] args)
        {
                newRPCServer();
        }
}
```

2. **RPCClient.java**

```
import java.io.*;
import java.net.*;
classRPCClient
```

```
{
        RPCClient()

        {
                try
                {
                        InetAddressia = InetAddress.getLocalHost();
                        DatagramSocket ds = new DatagramSocket();
                        DatagramSocket ds1 = new DatagramSocket(1300);
                        System.out.println("\nRPC Client\n");
                        System.out.println("Enter method name and parameter like add 3
4\n");
                        while (true)
                        {

                                BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));

                                String str = br.readLine();
                                byte b[] = str.getBytes();
                                DatagramPacketdp = new
DatagramPacket(b,b.length,ia,1200);
                                ds.send(dp);
                                dp = new DatagramPacket(b,b.length);
                                ds1.receive(dp);
                                String s = new String(dp.getData(),0,dp.getLength());
                                System.out.println("\nResult = " + s + "\n");
                        }
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args)
        {
                newRPCClient();
        }
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe - java RPCClient

E:\Ds_Yugi>javac RPCClient.java

E:\Ds_Yugi>java RPCClient

RPC Client

Enter method name and parameter like add 34

sub 10 8

Result = 2

mul 27 2

Result = 54

add 20 7

Result = 27

div 10 2
```

```
C:\WINDOWS\system32\cmd.exe - java RPCServer

E:\Ds_Yugi>javac RPCServer.java

E:\Ds_Yugi>java RPCServer
sub 10 8
result : 2

mul 27 2
result : 54

add 20 7
result : 27

div 10 2
result : 5
```

Practical 2D: A program that finds the square, square root, cube and cube root of the enterednumber.

Code:

1. **RPCNumServer.java**

```
importjava.util.*;
import java.net.*;
import java.io.*;
classRPCNumServer
 {
        DatagramSocket ds;
        DatagramPacketdp;
        String str,methodName,result;
        intval;
        RPCNumServer()
        {
                try
                {
                        ds=new DatagramSocket(1200);
                        byte b[]=new byte[4096];
                        while(true)
                        {
                                dp=new DatagramPacket(b,b.length);
                                ds.receive(dp);
                                str=new
                                String(dp.getData(),0,dp.getLength());
                                if(str.equalsIgnoreCase("q")) {
                                        System.exit(1);
                                }
                                else
                                {
                                        StringTokenizerst = new StringTokenizer(str," ");
                                        inti=0;
                                        while(st.hasMoreTokens())
                                        {
                                                String token=st.nextToken();
                                                methodName=token;
                                                val = Integer.parseInt(st.nextToken());
                                        }
                                }
                                System.out.println(str);
                                InetAddressia = InetAddress.getLocalHost();
                                if(methodName.equalsIgnoreCase("square"))
                                {
                                        result= "" + square(val);
                                }
                                else if(methodName.equalsIgnoreCase("squareroot"))
                                {
                                        result= "" + squareroot(val);
                                }
```

```
                                else if(methodName.equalsIgnoreCase("cube"))
                                {
                                        result= "" + cube(val);

                                }
                                else if(methodName.equalsIgnoreCase("cuberoot"))
                                {
                                        result= "" + cuberoot(val);
                                }
                                byte b1[]=result.getBytes();
                                DatagramSocket ds1 = new DatagramSocket();
                                DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
                                System.out.println("result :
                                "+result+"\n"); ds1.send(dp1);
                        }
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
        public double square(int a) throws Exception
        {
                doubleans;
                ans = a*a;
                returnans;
        }
        public double squareroot(int a) throws Exception
        {
                doubleans;
                ans = Math.sqrt(a);
                returnans;
        }
        public double cube(int a) throws Exception
        {
                doubleans;
                ans = a*a*a;
                returnans;
        }
        public double cuberoot(int a) throws Exception
        {
                doubleans;
                ans = Math.cbrt(a);
                returnans;
        }
        public static void main(String[] args)
        {
                newRPCNumServer();
```

```
        }
}
```

2. **RPCNumClient.java**

```
import java.io.*;
import java.net.*;
classRPCNumClient
{
        RPCNumClient()
        {
                try
                {
                        InetAddressia = InetAddress.getLocalHost();
                        DatagramSocket ds = new DatagramSocket();
                        DatagramSocket ds1 = new DatagramSocket(1300);
                        System.out.println("\nRPC Client\n");
                        System.out.println("1. Square of the number - square\n2. Square root
of the number - squareroot\n3. Cube of the number - cube\n4. Cube root of the number -
cuberoot");
                        System.out.println("Enter method name and the number\n");
                        while (true)
                        {
                                BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
                                String str = br.readLine();
                                byte b[] = str.getBytes();
                                DatagramPacketdp = new
DatagramPacket(b,b.length,ia,1200);
                                ds.send(dp);
                                dp = new DatagramPacket(b,b.length);
                                ds1.receive(dp);
                                String s = new String(dp.getData(),0,dp.getLength());
                                System.out.println("\nResult = " + s + "\n");
                        }
                }
                catch (Exception e)
                {
                        e.printStackTrace();
                }
        }
        public static void main(String[] args)
        {
                newRPCNumClient();
        }
}
```

Output:

C:\WINDOWS\system32\cmd.exe - java RPCNumServer

```
E:\Ds_Yugi>javac RPCNumServer.java

E:\Ds_Yugi>java RPCNumServer
square 7
result : 49.0

squareroot 25
result : 5.0

cube 3
result : 27.0

cuberoot 27
result : 3.0
```

C:\WINDOWS\system32\cmd.exe - java RPCNumClient

```
1. Square of the number - square
2. Square root of the number - squareroot
3. Cube of the number - cube
4. Cube root of the number - cuberoot
Enter method name and the number

square 7

Result = 49.0

squareroot 25

Result = 5.0

cube 3

Result = 27.0

cuberoot 27

Result = 3.0
```

Practical No: 03

Aim: A multicast Socket example.
Code:

1. **BroadcastServer.java**

```
import java.net.*;
import java.io.*;
importjava.util.*;
public class BroadcastServer
{
	public static final int PORT = 1234;
	public static void main(String args[])throws
	Exception {
		MulticastSocket socket;
		DatagramPacket packet;
		InetAddress address;
		// set the multicast address to your local subnet
		address = InetAddress.getByName("239.1.2.3");
		socket = new MulticastSocket();
		// join a Multicast group and send the group
		messages socket.joinGroup(address);
		byte[] data =
		null; for(;;)
		{
			Thread.sleep(10000);
			System.out.println("Sending "); String
			str = ("This is Neha Calling...."); data
			= str.getBytes();
			packet = new DatagramPacket(data, str.length(),address,PORT);
			// Sends the packet
			socket.send(packet);
		} // end for
	} // end main
 } // end class BroadcastServer
```

2. **BroadcastClient.java**

```
import java.net.*;
import java.io.*;
public class BroadcastClient
{
	public static final int PORT = 1234;
	public static void main(String args[])throws
	Exception {
		MulticastSocket socket;
		DatagramPacket packet;
		InetAddress address;
```

```
                // set the mulitcast address to your local subnet
                address = InetAddress.getByName("239.1.2.3");
                socket = new MulticastSocket(PORT);
                //join a Multicast group and wait for a message
                socket.joinGroup(address); byte[] data = new
                byte[100];
                packet = new DatagramPacket(data,data.length);
                for(;;)
                {
                        // receive the packets
                        socket.receive(packet);
                        String str = new String(packet.getData()); System.out.println("Message
                        received from "+ packet.getAddress() + "
Message is : "+str);
                } // for
        } // main
} // end BroadcastClient
```

Output:

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>javac BroadcastServer.java

E:\Ds_Yugi>java BroadcastServer
Sending
Sending
Sending
Sending
```

```
E:\Ds_Yugi>javac BroadcastClient.java

E:\Ds_Yugi>java BroadcastClient

Message received from /10.29.26.232 Message is: This is Neha Calling....

Message received from /10.29.26.232 Message is: This is Neha Calling....

Message received from /10.29.26.232 Message is: This is Neha Calling....

Message received from /10.29.26.232 Message is: This is Neha Calling....
```

Practical No: 04

Aim: Write a program to show the object communication using RMI.

Practical 4A: A RMI based application program to display current date and time.
Code:

1. **InterDate.java**

```
importjava.rmi.*;
public interface InterDate extends Remote
{
	public String display() throws Exception;

}
```

2. **ServerDate.java**

```
importjava.rmi.*;
importjava.rmi.server.*;
importjava.util.*;
public class ServerDate extends UnicastRemoteObject implements
InterDate {
	publicServerDate() throws Exception
	{
	}
	public String display() throws Exception
	{
		String str = "";
		Date d = new Date();
		str = d.toString();
		returnstr;
	}
	public static void main(String args[]) throws
	Exception {
		ServerDate s1 = new ServerDate();
		Naming.bind("DS",s1);
		System.out.println("Object registered.....");
	}
}
```

3. **ClientDate.java**

```
importjava.rmi.*;
import java.io.*;
public class ClientDate
{
	public static void main(String args[]) throws
	Exception {
		String s1;
		InterDate h1 = (InterDate)Naming.lookup("DS");
		s1 = h1.display();
```

```
                System.out.println(s1);
        }
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe - rmiregistry
E:\Ds_Yugi>javac ServerDate.java

E:\Ds_Yugi>javac ClientDate.java

E:\Ds_Yugi>rmic ServerDate
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\Ds_Yugi>rmiregistry
```

```
C:\WINDOWS\system32\cmd.exe - java ServerDate

E:\Ds_Yugi>java ServerDate
Object registered.....
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>java ClientDate
Thu Jan 04 17:38:00 IST 2018
```

Practical 4B: A RMI based application program that converts digits to words, e.g. 123 willbe converted to one two three.

Code:

1. **InterConvert.java**

```
importjava.rmi.*;
public interface InterConvert extends Remote
{
        public String convertDigit(String no) throws Exception;
}
```

2. **ServerConvert.java**

```
importjava.rmi.*;
importjava.rmi.server.*;
public class ServerConvert extends UnicastRemoteObject implements
InterConvert {
        publicServerConvert() throws Exception
        {
        }
```

```
public String convertDigit(String no) throws Exception
{
        String str = "";
        for(inti = 0; i<no.length(); i++)
        {
                int p = no.charAt(i);
                if( p == 48)
                {
                        str += "zero ";
                }
                if( p == 49)
                {
                        str += "one ";
                }
                if( p == 50)
                {
                        str += "two ";
                }
                if( p == 51)
                {
                        str += "three ";
                }
                if( p == 52)
                {
                        str += "four ";
                }
                if( p == 53)
                {
                        str += "five ";
                }
                if( p == 54)
                {
                        str += "six ";
                }
                if( p == 55)
                {
                        str += "seven ";
                }
                if( p == 56)
                {
                        str += "eight ";
                }
                if( p == 57)
                {
                        str += "nine ";
                }
        }
        returnstr;
}
```

```
        public static void main(String args[]) throws
        Exception {
                ServerConvert s1 = new ServerConvert();
                Naming.bind("Wrd",s1);
                System.out.println("Object registered....");
        }
}
```

3. **ClientConvert.java**

```
importjava.rmi.*;
import java.io.*;
public class ClientConvert
{
        public static void main(String args[]) throws
        Exception {
                InterConvert h1 = (InterConvert)Naming.lookup("Wrd");
                BufferedReaderbr = new BufferedReader(new
InputStreamReader(System.in));
                System.out.println("Enter a number : \t");
                String no = br.readLine();
                String ans = h1.convertDigit(no);
                System.out.println("The word representation of the entered digit is : " +ans);
        }
}
```

Output:

```
C:\WINDOWS\system32\cmd.exe - rmiregistry
E:\Ds_Yugi>javac ServerConvert.java

E:\Ds_Yugi>javac ClientConvert.java

E:\Ds_Yugi>rmic ServerConvert
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\Ds_Yugi>rmiregistry
```

```
C:\WINDOWS\system32\cmd.exe - java ServerConvert

E:\Ds_Yugi>java ServerConvert
Object registered....
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>java ClientConvert
Enter a number :
123
The word representation of the entered digit is : one two three

E:\Ds_Yugi>java ClientConvert
Enter a number :
456
The word representation of the entered digit is : four five six
```

Practical No: 05

Aim: Show the implementation of web services.

What Are Web Services?

Web services are client and server applications that communicate over the World WideWeb's (WWW) HyperText Transfer Protocol (HTTP). As described by the World Wide Web Consortium (W3C), web services provide a standard means of interoperating between software applications running on a variety of platforms and frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions, thanks to the use of XML. Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticated added-value services.

Types of Web Services:

On the conceptual level, a service is a software component provided through a network-accessible endpoint. The service consumer and provider use messages to exchange invocation request and response information in the form of self-containing documents that make very few assumptions about the technological capabilities of the receiver.

On a technical level, web services can be implemented in various ways. The two types of web services can be distinguished as "big" web services and "RESTful" web services.

1) **"Big" Web Services:**

In Java EE 6, JAX-WS provides the functionality for "big" web services. Big web services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

The SOAP message format and the WSDL interface definition language have gained widespread adoption. Many development tools, such as NetBeans IDE, can reduce the complexity of developing web service applications.

A SOAP-based design must include the following elements.

- A formal contract must be established to describe the interface that the web service offers. WSDL can be used to describe the details of the contract, which may include messages, operations, bindings, and the location of the web service. You may also process SOAP messages in a JAX-WS service without publishing a WSDL.
- The architecture must address complex nonfunctional requirements. Many web service specifications address such requirements and establish a common vocabulary for them. Examples include transactions, security, addressing, trust, coordination, and so on.
- The architecture needs to handle asynchronous processing and invocation. In such cases, the infrastructure provided by standards, such as Web Services Reliable Messaging (WSRM), and APIs, such as JAX-WS, with their client-side asynchronous invocation support, can be leveraged out of the box.

2) **RESTful Web Services:**

In Java EE 6, JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service–API definitions.

Project Jersey is the production-ready reference implementation for the JAX-RS specification. Jersey implements support for the annotations defined in the JAX-RS specification, making it easy for developers to build RESTful web services with Java and the Java Virtual Machine (JVM).

Because RESTful web services use existing well-known W3C and Internet Engineering Task Force (IETF) standards (HTTP, XML, URI, MIME) and have a lightweight infrastructure that allows services to be built with minimal tooling, developing RESTful web services is inexpensive and thus has a very low barrier for adoption. You can use a development tool such as NetBeans IDE to further reduce the complexity of developing RESTful web services.

A RESTful design may be appropriate when the following conditions are met.

- The web services are completely stateless. A good test is to consider whether the interaction can survive a restart of the server.

- A caching infrastructure can be leveraged for performance. If the data that the web service returns is not dynamically generated and can be cached, the caching infrastructure that web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.
- The service producer and service consumer have a mutual understanding of the context and content being passed along. Because there is no formal way to describe the web services interface, both parties must agree out of band on the schemas that describe the data being exchanged and on ways to process it meaningfully. In the real world, most commercial applications that expose services as RESTful implementations also distribute so-called value-added toolkits that describe the interfaces to developers in popular programming languages.
- Bandwidth is particularly important and needs to be limited. REST is particularly useful for limited-profile devices, such as PDAs and mobile phones, for which the overhead of headers and additional layers of SOAP elements on the XML payload must be restricted.
- Web service delivery or aggregation into existing web sites can be enabled easily with a RESTful style. Developers can use such technologies as JAX-RS and Asynchronous JavaScript with XML (AJAX) and such toolkits as Direct Web Remoting (DWR) to consume the services in their web applications. Rather than starting from scratch, services can be exposed with XML and consumed by HTML pages without significantly refactoring the existing web site architecture. Existing developers will be more productive because they are adding to something they are already familiar with rather than having to start from scratch with new technology.

Deciding Which Type of Web Service to Use:

Basically, you would want to use RESTful web services for integration over the web and use big web services in enterprise application integration scenarios that have advanced quality of service (QoS) requirements.

- **JAX-WS**: addresses advanced QoS requirements commonly occurring in enterprisecomputing. When compared to JAX-RS, JAX-WS makes it easier to support the WS-* set of protocols, which provide standards for security and reliability, among other things, and interoperate with other WS-* conforming clients and servers.
- **JAX-RS**: makes it easier to write web applications that apply some or all of theconstraints of the REST style to induce desirable properties in the application, such as loose coupling (evolving the server is easier without breaking existing clients), scalability (start small and grow), and architectural simplicity (use off-the-shelf components, such as proxies or HTTP routers). You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale. Clients can choose to consume some or all aspects of the service and mash it up with other web-based services.

Practical 5A: Implementing "Big" Web Service.

1) **Creating a Web Service**

A. **Choosing a Container:**

1. oose File > New Project. Select Web Application from the Java Web.

2. Name the project CalculatorWSApplication. Select a location for the project. Click Next.

3. Select your server and Java EE version and click Finish.

B. **Creating a Web Service from a Java Class**

1. Right-click the CalculatorWSApplication node and choose New > Web Service.

2. Name the web service CalculatorWS and type org.me.calculator in Package. Leave Create Web Service from Scratch selected. If you are creating a Java EE 6 project on GlassFish or WebLogic, select Implement Web Service as a Stateless Session Bean.

3. Click Finish. The Projects window displays the structure of the new web service and the source code is shown in the editor area.

2) **Adding an Operation to the Web Service**

The goal of this exercise is to add to the web service an operation that adds two numbers received from a client. The NetBeans IDE provides a dialog for adding an operation to a web service. You can open this dialog either in the web service visual designer or in the web service context menu.

A. **<u>To add an operation to the web service</u>:**

1. Change to the Design view in the editor.

2. Click Add Operation in either the visual designer or the context menu. The Add Operation dialog opens.
3. In the upper part of the Add Operation dialog box, type add in Name and type int in the Return Type drop-down list.
4. In the lower part of the Add Operation dialog box, click Add and create a parameter of type int named i.
5. Click Add again and create a parameter of type int called j. You now see the following:

6. Click OK at the bottom of the Add Operation dialog box. You return to the editor.
7. The visual designer now displays the following:

8. Click Source. And code the following.

```
@WebMethod(operationName = "add")
   publicint add(@WebParam(name = "i") inti, @WebParam(name = "j") int j)
{
        int k = i + j;
```

```
        return k;
    }
```

3) **Deploying and Testing the Web Service**

After you deploy a web service to a server, you can use the IDE to open the server's test client, if the server has a test client. The GlassFish and WebLogic servers provide test clients.

A. **To test successful deployment to a GlassFish or WebLogic server:**

1. Right-click the project and choose Deploy. The IDE starts the application server, builds the application, and deploys the application to the server

2. In the IDE's Projects tab, expand the Web Services node of the CalculatorWSApplication project. Right-click the CalculatorWS node, and choose Test Web Service.

3. The IDE opens the tester page in your browser, if you deployed a web application to the GlassFish server.
4. If you deployed to the GlassFish server, type two numbers in the tester page, as shown below:

5. The sum of the two numbers is displayed:

4) **Consuming the Web Service**

Now that you have deployed the web service, you need to create a client to make use of the web service's add method.

1. **Client: Java Class in Java SE Application**

1. Choose File > New Project. Select Java Application from the Java category. Name the project CalculatorWS_Client_Application. Leave Create Main Class selected and accept all other default settings. Click Finish.

2. Right-click the CalculatorWS_Client_Application node and choose New > Web Service Client. The New Web Service Client wizard opens.

3. Select Project as the WSDL source. Click Browse. Browse to the CalculatorWS web service in the CalculatorWSApplication project. When you have selected the web service, click OK.

4. Do not select a package name. Leave this field empty.

5. Leave the other settings at default and click Finish. The Projects window displays the new web service client, with a node for the add method that you created:

6. Double-click your main class so that it opens in the Source Editor. Drag the add node below the main() method.

You now see the following:

```
public static void main(String[] args)
{
   // TODO code application logic here
}
private static int add(inti, int j)
{
        org.me.calculator.CalculatorWS_Service service = new
org.me.calculator.CalculatorWS_Service();
        org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();
        return port.add(i, j);
}
```

7. In the main() method body, replace the TODO comment with code that initializes values for i and j, calls add(), and prints the result.

```
public static void main(String[] args)
{
   inti = 3;
   int j = 4;
   int result = add(i, j);
   System.out.println("Result = " + result);
}
```

8. Surround the main() method code with a try/catch block that prints an exception.

```
public static void main(String[] args)
{
   try
  {
      inti = 3;

      int j = 4;
      int result = add(i, j);
      System.out.println("Result = " + result);
   } catch (Exception ex) {
      System.out.println("Exception: " + ex);
   }
}
```

9. Right-click the project node and choose Run.

The Output window now shows the sum:

```
compile:
run:
Result = 7
BUILD SUCCESSFUL (total time: 1 second)
```

Practical 5B: Implementing Web Service that connects to MySQL database.

Building Web Service:-

- JAX-WS is an important part of the Java EE platform.
- JAX-WS simplifies the task of developing Web services using Java technology.
- It provides support for multiple protocols such as SOAP, XML and by providing a facility for supporting additional protocols along with HTTP.
- With its support for annotations, JAX-WS simplifies Web service development and reduces the size of runtime files.
- Here basic demonstration of using IDE to develop a JAX-WS Web Service is given.
- After creating the web service, create web service clients that use the Web service over a network which is called consuming a web service.

- The client is a servlet in a web application.
- Let's build a Web Service that returns the book name along with its cost for a particular ISBN.
- To begin building this service, create the data store. The server will access the data stored in a MySQL table to serve the client.

1)

2) **Creating MySQL DB Table**

create database bookshop;
use bookshop;

- **Create a table named Books that will store valid books information**

create table books(isbn varchar(20) primary key, bookname varchar(100), bookprice varchar(10));

- **Insert valid records in the Books table**

insert into books values("111-222-333","Learn My SQL","250");
insert into books values("111-222-444","Java EE 6 for Beginners","850");
insert into books values("111-222-555","Programming with Android","500");
insert into books values("111-222-666","Oracle Database for you","400");
insert into books values("111-222-777","Asp.Net for advanced programmers","1250");

2) **Creating a web service**

i. **Choosing a container**
- Web service can be either deployed in a Web container or in an EJB container.
- If a Java EE 6 application is created, use a Web container because EJBs can be placed directly in a Web application.

ii. **Creating a web application**
- To create a Web application, select File - New Project.
- New Project dialog box appears. Select Java Web available under the Categories section and Web Application available under the Projects section. Click Next.
- New Web Application dialog box appears. Enter BookWS as the project name in the Project Name textbox and select the option Use Dedicated Folder for Storing Libraries.
- Click Next. Server and Settings section of the New Web Application dialog box appears. Choose the default i.e. GlassFish v3 Domain as the Web server, the Java EE 6 Web as the Java EE version and the Context Path.
- Click –Finish
- The Web application named BookWS is created.

iii. **Creating a web service**
- Right-click the BookWS project and select New -> Web Service as shown in diagram.

- New Web Service dialog box appears. Enter the name BookWS in the Web Service Name textbox, webservice in the Package textbox, select the option Create Web Service from scratch and also select the option implement web service as a stateless session bean as shown in the diagram.

- Click Finish.
- The web service in the form of java class is ready.

3) **<u>Designing the web service</u>**

Now add an operation which will accept the ISBN number from the client to the web service.

i. **Adding an operation to the web service**

- Change the source view of the BookWS.java to design view by clicking Design available just below the name of the BookWS.java tab.
- The window changes as shown in the diagram.
- Click Add Operation available in the design view of the web service.
- Add Operation dialog appears. Enter the name getBookDetails in the Name textbox and java.lang.String in the Return Type textbox as shown in the diagram.
- In Add Operation dialog box, click Add and create a parameter of the type String named isbn as shown in the diagram.

- Click Ok. The design view displays the operation added as shown in the diagram.

- Click Source. The code spec expands due to the operation added to the web service as shown in the diagram.

- Modify the code spec of the web service BookWS.java.

Code Spec

```
packagewebservice;
importjava.sql.*;
importjavax.jws.WebMethod;
importjavax.jws.WebParam;
```

```
importjavax.jws.WebService;
importjavax.ejb.Stateless;
@WebService()
@Stateless()
public class BookWS {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "getBookDetails") public
    String getBookDetails(@WebParam(name = "isbn") String
    isbn) {
        //TODO write your implementation code here
        Connection dbcon = null;
        Statement stmt = null;
        ResultSetrs = null;
        String query = null;
        try
        {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            dbcon =
DriverManager.getConnection("jdbc:mysql://localhost/bookshop","root","123");
            stmt = dbcon.createStatement();
            query = "select * from books where isbn = '" +isbn+
            "'"; rs = stmt.executeQuery(query); rs.next();
            String bookDetails = "<h1>The name of the book is <b>"
+rs.getString("bookname") + "</b> and its cost is <b>" +rs.getString("bookprice") +
"</b></h1>.";
            returnbookDetails;
        }
        catch(Exception e)
        {
            System.out.println("Sorry failed to connect to the database.." + e.getMessage());
        }
        return null;
    }
```

4) **Adding the MySQL connector**

- We need to add a reference of MySQL connector to our web service. It is via this connector that our web service will be able to communicate with the database.
- Right click on the libraries and select Add JAR/Folder as shown in the diagram.

- Choose the location where mysql-coonector-java-5.1.10-bin is located, select it and click on open as shown.

5) **<u>Deploying and testing the web service</u>**

- When a web service is deployed to a web container, the IDE allows testing the web service to see if it functions as expected.
- The tester application provided by GlassFish, is integrated into the IDE for this purpose as it allows the developer to enter values and test them.
- No facility for testing whether an EJB module is deployed successfully is currently available.
- To test the BookWS application, right click the BookWS project and select Deploy as shown in the diagram.

- The IDE starts the server, builds the application and deploys the application to the server.
- Follow the progress of these operations in the BookWS (run-deploy) and GlassFish v3 Domain tabs in the Output view.
- Now expand the web services directory of the BookWS project, right-click the BookWS Web service and select Test web service as shown in the diagram.
- The IDE opens the tester page in the web browser, if the web application is deployed using GlassFish server as shown in the figure.

- Enter the ISBN number as shown in the diagram.
- Click getBookDetails. The book name and its cost are displayed as shown in the diagram.

6) **Consuming the web service**

- Once the web service is deployed, the next most logical step is to create a client to make use of the web service's getBookDetails() method.

i. **Creating a web application**

- To create a web application, select File -> New Project.
- New project dialog box appears, select java web available under the categories section and web application available under the projects section. Click Finish.
- New web application dialog box appears. Enter BookWSServletClient as the project name in the Project Name textbox and select the option Use Dedicated Folder for Storing Libraries.

- Click Next. Server and settings section of the new web application, dialog box appears. Choose the default i.e. GlassFish v3 Domain as the web serevr, the Java EE 6 web as the Java EE version and the context path.
- Click Finish.
- The web application named BookWSServletClient is created.

ii. **Adding the web service to the client application**

- Right-click the BookWSServletClient project and select New -> Web Service Client as shown in the diagram.

- New Web Service Client dialog box appears. In the Project section, click Browse and browse through the web service which needs to be consumed. Click ok. The name of the web service appears in the New Web Service Client as shown in the diagram.

- Leave the other settings as it is. Click Finish
- The Web Service Reference directory is added to the BookWSServletClient application as shown in the diagram. It displays the structure of the newly created client including the getBookDetails() method created earlier.

iii. **Creating a servlet**

- Create retreiveBookDetails.java using NetBeans IDE.
- Right click source package directory, select New -> Servlet.
- New Servlet dialog box appears. Enter retreiveBookDetails in the Class Name textbox and enter servlet in the package textbox.
- Click Next. Configure Servlet Deployment section of the New Servlet dialog box appears. Keep the defaults.
- Click Finish.
- This creates the servlet named retreiveBookDetails.java in the servlet package.
- retreiveBookDetails.java is available with the default skeleton created by the NetBeans IDE which needs to be modified to hold the application logic.
- In the retreieveBookDetails.java source code, remove the following comments available in the body of the processRequest() method.

  /*TODO output your page here*/
- Replace the following code spec:
  out.println("<h1>Servlet retreiveBookDetails at " + request.getContextPath () + "</h1>");
  With the code spec of the getBookDetails() operation of the web service by dragging and dropping the getBookDetails operation as shown in the diagram.

- The Servlet code spec changes as shown in the diagram

- The web service is instantiated by the @WebServiceRef annotation.
- Now change the following code spec:
  java.lang.Stringisbn = “”;
  to
  java.lang.Stringisbn = request.getParameter(“isbn”);

iv. **Creating an HTML form**

- Once the web service is added and the servlet is created, the form to accept ISBN from the user needs to be coded.
- Since NetBeans IDE by default [as a part of Web Application creation] makes available index.jsp file. Modify it to hold the following code spec.

```
<%@page contentType="text/html" pageEncoding="UTF-
8"%><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
    8"><title>SOAP Cleint - Get Book Details</title>
  </head>
  <body bgcolor="pink">
    <form name="frmgetBookDetails" method="post"
      action="retreiveBookDetails"><h1>
        ISBN : <input type="text"
      name="isbn"/><br><br></h1>

      <input type="submit"
    value="Submit"/></form>
  </body>
</html>
```

v. **Building the Web Application**

- Build the web application.

- Right click BookWSServletClient project and select Build.
- Once the Build menu item is clicked the details about the compilation and building of the BookWSServletClient Web application appears in the output – BookWSServletClient (dist) window.

vi. **Running the Application**

- Once the compilation and building of the web application is done run the application.
- Right click the BookWSServerCleint project and select run.
- Once the run processing completes in NetBeans IDE a web browser is automatically launched and the BookWSServletCleint application is executed as shown in the diagram.
- Enter the ISBN as shown in the diagram

- Click Submit. The book name and its cost are displayed as shown in the diagram.

Practical:06

Aim: Implement Xen virtualization and manage with Xen Center

- Install **XenServer** in VMware Workstation and select Guest operating system as Linux.

Note IP Address – " 192.168.124.137" ping it from command prompt.

- Now Install Citrix App if not installed
- Now Open Citrix XenCenter – and Click and **Add Server.**

- Fill IP address copied from Installation and User name as "root" and Password as "root123" which we had given during installation and Click on Add.
- Then click on Ok
- Now Click on New Storage
- Select Window File Sharing (CIFS) and click on next
- Uncheck Auto generate option Click on Next.
- Provide the path of shared windows XP image and enter local pc credential , click on Finish
- Click on New VM – and Windows XP SP3

- Select ISO file and click on next –
- Click on Next –
- Uncheck – Start the new VM and click on create now
- Now Right click on Windows XP and Start -

Installation is successful and virtual node has been created if we get below Welcome screen of Windows XP machine.

Practical: 07

1. **Aim:**Implement virtualization using VMWare ESXi Server and managing with vCenter

Steps:

Install **ESXiiso**in VMWare workstation.

InstallVMware vSphere Client

In vSphere create new **Virtual Machine.** Install Windows XP iso file and open it.

Practical: 08

Aim: Implement Windows Hyper V virtualization

First we have to uninstall vmware software if already installed on computer becausethe VMware Workstation installer does not support running on a Hyper-V virtual machine. after uninstalling vmware we can proceed to next step go to controlpanel and click on uninstall a program.

Click on Turn windows features on or off.
Now in windows features check on Hyper-V option.

After Restart **Search for hyper-v manager in search box and click on that.**

for creating virtual machine first we have to create virtual switch click on virtual switch manager option.

Select External as a connection type and then click on create virtual switch.
Create new Virtual switch and install windows XP .iso

and virtual machine will start.

Practical: 09

Aim: Develop application for Microsoft Azure.

Step 1:

To develop an application for Windows Azure on Visual Studio install the **"Microsoft Azure SDK for .NET (VS 2010) – 2.8.2.1"**

Step2:

Turn windows Features ON or OFF:

Go to Control panel and click on programs.
Turn Windows features on or off.

Step3:

Now, Start the visual studio 2010 and Go To File->New->Project

Expand Visual C#-> Select Cloud

Right Click on WebRole1>>ADD>>New Item

Add a New web Form. Give it a name. Click Add

Deploy the project:

Run Project

Welcome TO Windows Azure.....!!!!

Practical: 10

Aim: Develop application for Google App Engine

- Open Eclipse Luna. Go to **Help Menu Install New Software…**
- In **Install** window Click on the **"Add"** button besides the **Work with** textbox. **Add Repository** window appears. Enter the **Location** as **"https://dl.google.com/eclipse/plugin/4.4"** and click on "**OK**" button.
- From the available softwares select the required softwares and tools as shown in the below image for the **GAE**. Then click on the **"Next"** button.
- In the **Install Details** window click on **"Next"** button.
- In the Next Window "Review the Items to be Installed" then click on **"Next"**
- In the next window for Review Licenses select the option **"I accept……"** and click on **"Finish"**
- button.
- After Installation you will get option to **"Restart Eclipse",** click on **Yes.** So that the software you selected gets updated...
- Now, go to **File Menu_New_Other.**

In the New window select **Google_Web Application Project** and click on **"Next"** button.

Enter the details for the new Web application project. Deselect the **Use Google Web Toolkit** option under the section **Google SDKs**. Click on the **"Finish"** button.

From the **Package Explorer** open the **.java** file (Here it is **"Google_App_EngineServlet.java"**).

Edit the file as required (Unedited file too can be used. Here the editing is done to "what should be displayed" on the browser). **Save** the file. Click on the **Run** option available on the Tools bar.

In the browser (Here, Google Chrome) type the address as **"localhost:8888"which is "Default"**.

In **localhost:8888** the link to the **Google_App_EngineServlet.java** file as **Google_App_Engine**is displayed. Click on this link. It will direct you to **"localhost:8888/Google_App_Engine"**.

The **output text entered** in the **java** program is **displayed as the output** when clicked the link
"Google_App_Engine"

SOFT COMPUTING TECHNIQUES

TABLE OF CONTENTS

No	Details	Signature
1	Implement the following:	
a	Design a simple linear neural network model.	
b	Calculate the output of neural net using both binary and bipolar sigmoidal function.	
2	Implement the following:	
a	Generate AND/NOT function using McCulloch-Pitts neural net.	
b	Generate XOR function using McCulloch-Pitts neural net.	
3	Implement the Following	
a	Write a program to implement Hebb's rule.	
b	Write a program to implement of delta rule.	
4	Implement the Following	
a	Write a program for Back Propagation Algorithm	
b	Write a program for error Backpropagation algorithm.	
5.	Implement the Following	
a	Write a program for Hopfield Network.	
b	Write a program for Radial Basis function	
6.	Implement the Following	
a	Kohonen Self organizing map	
b	Adaptive resonance theory	

7.	Implement the Following	
a	Write a program for Linear separation.	
b	Write a program for Hopfield network model for associative memory	
8.	Implement the Following	
a	Membership and Identity Operators | in, not in,	
b.	Membership and Identity Operatorsis, is not	
9.	Implement the Following	
a	Find ratios using fuzzy logic	
b	Solve Tipping problem using fuzzy logic	
10.	Implement the Following	
a	Implementation of Simple genetic algorithm	
b	Create two classes: City and Fitness using Genetic algorithm	

Practical 1

Practical 1 a: Design a simple linear neural network model.

```
x=float(input("Enter value of x:"))
w=float(input("Enter value of weight w:"))
b=float(input("Enter value of bias b:"))
net = int(w*x+b)
if(net<0):
   out=0
elif((net>=0)&(net<=1)):
   out =net
else:
   out=1
print("net=",net)
print("output=",out)
```

1b: Calculate the output of neural net using both binary and bipolar sigmoidal function.

For the network shown in the figure 1, calculate the net input to output neuron.

Figure 1 Neural net.

Code :

```
# number of elements as input
n = int(input("Enter number of elements : "))

# In[2]:
print("Enter the inputs")
inputs = [] # creating an empty list for inputs

# iterating till the range
for i in range(0, n):
        ele = float(input())
        inputs.append(ele) # adding the element

print(inputs)

# In[3]:
print("Enter the weights")
```

```
# creating an empty list for weights
weights = []

# iterating till the range
for i in range(0, n):
        ele = float(input())
        weights.append(ele) # adding the element

print(weights)

# In[4]:
print("The net input can be calculated as   Yin = x1w1 + x2w2 + x3w3")

# In[5]:
Yin = []
for i in range(0, n):
        Yin.append(inputs[i]*weights[i])
print(round(sum(Yin),3))
```

Output :

```
Enter number of elements : 3
Enter the inputs
0.3
0.5
0.6
[0.3, 0.5, 0.6]
Enter the weights
0.2
0.1
-0.3
[0.2, 0.1, -0.3]
The net input can be calculated as   Yin = x1w1 + x2w2 + x3w3
-0.07
```

1. **Problem statement :**
1. Calculate the net input for the network shown in Figure 2 with bias included in the network.

Figure 2 Simple neural net.

Code :

```
n = int(input("Enter number of elements : "))
print("Enter the inputs:")
inputs = [] # creating an empty list for inputs
for i in range(0, n):
        ele = float(input())
        inputs.append(ele) # adding the element
print(inputs)
print("Enter the weights:")
weights = []
for i in range(0, n):
        ele = float(input())
        weights.append(ele) # adding the element
print(weights)
b=float(input("Enter bias value:"))
print("The net input can be calculated as   Yin = b + x1w1 + x2w2:")
Yin = []
for i in range(0, n):
        Yin.append(inputs[i]*weights[i])
print(round((sum(Yin)+b),3))
```

```
Enter number of elements : 2
Enter the inputs:
0.2
0.6
[0.2, 0.6]
Enter the weights:
0.3
0.7
[0.3, 0.7]
Enter bias value:0.45
The net input can be calculated as   Yin = b + x1w1 + x2w2:
0.93
```

Practical 2

Practical 2a: Implement AND/NOT function using McCulloch-Pits neuron (use binary data representation).

Code:

```
# enter the no of inputs
num_ip = int(input("Enter the number of inputs : "))
#Set the weights with value 1
w1 = 1
w2 = 1
print("For the ", num_ip , " inputs calculate the net input using yin = x1w1 + x2w2 ")
x1 = []
x2 = []
for j in range(0, num_ip):
   ele1 = int(input("x1 = "))
   ele2 = int(input("x2 = "))
   x1.append(ele1)
   x2.append(ele2)
print("x1 = ",x1)
print("x2 = ",x2)
n = x1 * w1
m = x2 * w2
Yin = []
for i in range(0, num_ip):
        Yin.append(n[i] + m[i])
print("Yin = ",Yin)
#Assume one weight as excitatory and the other as inhibitory, i.e.,
Yin = []
for i in range(0, num_ip):
        Yin.append(n[i] - m[i])
print("After assuming one weight as excitatory and the other as inhibitory Yin = ",Yin)
#From the calculated net inputs, now it is possible to fire the neuron for input (1, 0)
#only by fixing a threshold of 1, i.e., θ ≥ 1 for Y unit.
#Thus, w1 = 1, w2 = -1; θ ≥ 1
Y=[]
for i in range(0, num_ip):
   if(Yin[i]>=1):
        ele=  1
        Y.append(ele)
   if(Yin[i]<1):
        ele=  0
        Y.append(ele)
print("Y = ",Y)
```

Output:

```
Enter the number of inputs : 4
For the  4  inputs calculate the net input using yin = x1w1 + x2w2

x1 = 0

x2 = 0

x1 = 0

x2 = 1

x1 = 1

x2 = 0

x1 = 1

x2 = 1
x1 =  [0, 0, 1, 1]
x2 =  [0, 1, 0, 1]
Yin =  [0, 1, 1, 2]
 After assuming one weight as excitatory and the other as inhibitory Yin =  [0, 1, -1, 0]
Y =  [0, 1, 0, 0]

In [14]: |
```

Practical 2b: Generate XOR function using McCulloch-Pitts neural net

The XOR (exclusive or) function is defined by the following truth table:

Input1	Input2	XOR Output
0	0	0
0	1	1
1	0	1
1	1	0

```
#Getting weights and threshold value
import numpy as np
print('Enter weights')
w11=int(input('Weight w11='))
w12=int(input('weight w12='))
w21=int(input('Weight w21='))
w22=int(input('weight w22='))
v1=int(input('weight v1='))
```

```
v2=int(input('weight v2='))
print('Enter Threshold Value')
theta=int(input('theta='))
x1=np.array([0, 0, 1, 1])
x2=np.array([0, 1, 0, 1])
z=np.array([0, 1, 1, 0])
con=1
y1=np.zeros((4,))
y2=np.zeros((4,))
y=np.zeros((4,))
while con==1:
    zin1=np.zeros((4,))
    zin2=np.zeros((4,))
    zin1=x1*w11+x2*w21
    zin2=x1*w21+x2*w22

    print("z1",zin1)
    print("z2",zin2)
    for i in range(0,4):
        if zin1[i]>=theta:
            y1[i]=1
        else:
            y1[i]=0

        if zin2[i]>=theta:
            y2[i]=1
        else:
            y2[i]=0

    yin=np.array([])
    yin=y1*v1+y2*v2
    for i in range(0,4):
        if yin[i]>=theta:
            y[i]=1
        else:
            y[i]=0

    print("yin",yin)
    print('Output of Net')
    y=y.astype(int)
    print("y",y)
    print("z",z)

    if np.array_equal(y,z):
        con=0
    else:
        print("Net is not learning enter another set of weights and Threshold value")
        w11=input("Weight w11=")
        w12=input("weight w12=")
```

```
    w21=input("Weight w21=")
    w22=input("weight w22=")
    v1=input("weight v1=")
    v2=input("weight v2=")
    theta=input("theta=")

print("McCulloch-Pitts Net for XOR function")
print("Weights of Neuron Z1")
print(w11)
print(w21)
print("weights of Neuron Z2")
print(w12)
print(w22)
print("weights of Neuron Y")
print(v1)
print(v2)
print("Threshold value")
print(theta)
```

Output:

```
Enter weights
Weight w11=1
weight w12=-1
Weight w21=-1
weight w22=1
weight v1=1
weight v2=1
Enter Threshold Value
theta=1
z1 [ 0 -1  1  0]
z2 [ 0  1 -1  0]
yin [0. 1. 1. 0.]
Output of Net
y [0 1 1 0]
z [0 1 1 0]
McCulloch-Pitts Net for XOR function
Weights of Neuron Z1
1
-1
weights of Neuron Z2
-1
1
weights of Neuron Y
1
1
Threshold value
1
```

Practical 3

3a. Write a program to implement Hebb's rule.

Using the Hebb rule, find the weights required to perform the following classifications of the given input patterns shown in Figure 16. The pattern is shown as 3 × 3 matrix form in the squares. The "+" symbols represent the value "1" and empty squares indicate "−1." Consider "I" belongs to the members of class (so has target value 1) and "O" does not belong to the members of class (so has target value −1).

```
import numpy as np
#first pattern
x1=np.array([1,1,1,-1,1,-1,1,1,1])
#second pattern
x2=np.array([1,1,1,1,-1,1,1,1,1])
#initialize bais value
b=0
#define target
y=np.array([1,-1])

wtold=np.zeros((9,))
wtnew=np.zeros((9,))

wtnew=wtnew.astype(int)
wtold=wtold.astype(int)

bais=0
print("First input with target =1")
for i in range(0,9):
    wtold[i]=wtold[i]+x1[i]*y[0]
wtnew=wtold
b=b+y[0]
```

```
print("new wt =", wtnew)
print("Bias value",b)

print("Second input with target =-1")
for i in range(0,9):
   wtnew[i]=wtold[i]+x2[i]*y[1]
b=b+y[1]
print("new wt =", wtnew)
print("Bias value",b)
```

```
First input with target =1
new wt = [ 1  1  1 -1  1 -1  1  1  1]
Bias value 1
Second input with target =-1
new wt = [ 0  0  0 -2  2 -2  0  0  0]
Bias value 0
```

Practical 3b: Write a program to implement of delta rule.

```
#supervised learning
import numpy as np
import time
np.set_printoptions(precision=2)
x=np.zeros((3,))
weights=np.zeros((3,))
desired=np.zeros((3,))
actual=np.zeros((3,))

for i in range(0,3):
    x[i]=float(input("Initial inputs:"))

for i in range(0,3):
    weights[i]=float(input("Initial weights:"))

for i in range(0,3):
    desired[i]=float(input("Desired output:"))

a=float(input("Enter learning rate:"))

actual=x*weights
print("actual",actual)
print("desired",desired)

while True:
    if np.array_equal(desired,actual):
        break    #no change
    else:
        for i in range(0,3):
            weights[i]=weights[i]+a*(desired[i]-actual[i])

    actual=x*weights
    print("weights",weights)
    print("actual",actual)
    print("desired",desired)
print("*"*30)
print("Final output")
print("Corrected weights",weights)
print("actual",actual)
print("desired",desired)
```

```
Initial inputs:1
Initial inputs:1
Initial inputs:1
Initial weights:1
Initial weights:1
Initial weights:1
Desired output:2
Desired output:3
Desired output:4
Enter learning rate:1
actual [1. 1. 1.]
desired [2. 3. 4.]
weights [2. 3. 4.]
actual [2. 3. 4.]
desired [2. 3. 4.]
******************************
Final output
corrected weights [2. 3. 4.]
actual [2. 3. 4.]
desired [2. 3. 4.]
```

Practical 4

Practical 4a: Write a program for Back Propagation Algorithm

```
import numpy as np
import decimal
import math
np.set_printoptions(precision=2)
v1=np.array([0.6, 0.3])
v2=np.array([-0.1, 0.4])
w=np.array([-0.2,0.4,0.1])
b1=0.3
b2=0.5
x1=0
x2=1
alpha=0.25
print("calculate net input to z1 layer")
zin1=round(b1+ x1*v1[0]+x2*v2[0],4)
print("z1=",round(zin1,3))

print("calculate net input to z2 layer")
zin2=round(b2+ x1*v1[1]+x2*v2[1],4)
print("z2=",round(zin2,4))
print("Apply activation function to calculate output")
z1=1/(1+math.exp(-zin1))
z1=round(z1,4)
z2=1/(1+math.exp(-zin2))
z2=round(z2,4)
print("z1=",z1)
print("z2=",z2)
```

```
print("calculate net input to output layer")
yin=w[0]+z1*w[1]+z2*w[2]
print("yin=",yin)

print("calculate net output")
y=1/(1+math.exp(-yin))
print("y=",y)

fyin=y *(1- y)
dk=(1-y)*fyin
print("dk",dk)

dw1= alpha * dk * z1
dw2= alpha * dk * z2
dw0= alpha * dk

print("compute error portion in delta")

din1=dk* w[1]
din2=dk* w[2]
print("din1=",din1)
print("din2=",din2)

print("error in delta")
fzin1= z1 *(1-z1)
print("fzin1",fzin1)
d1=din1* fzin1
fzin2= z2 *(1-z2)
print("fzin2",fzin2)
d2=din2* fzin2

print("d1=",d1)
print("d2=",d2)

print("Changes in weights between input and hidden layer")
dv11=alpha * d1 * x1
print("dv11=",dv11)
dv21=alpha * d1 * x2
print("dv21=",dv21)
dv01=alpha * d1
print("dv01=",dv01)
dv12=alpha * d2 * x1
print("dv12=",dv12)
dv22=alpha * d2 * x2
print("dv22=",dv22)
dv02=alpha * d2
print("dv02=",dv02)
print("Final weights of network")
v1[0]=v1[0]+dv11
```

```
v1[1]=v1[1]+dv12
print("v=",v1)
v2[0]=v2[0]+dv21
v2[1]=v2[1]+dv22
print("v2",v2)
w[1]=w[1]+dw1
w[2]=w[2]+dw2
b1=b1+dv01
b2=b2+dv02
w[0]=w[0]+dw0
print("w=",w)
print("bias b1=",b1, " b2=",b2)
```

Output:

```
z1= 0.2
calculate net input to z2 layer
z2= 0.9
Apply activation function to calculate output
z1= 0.5498
z2= 0.7109
calculate net input to output layer
yin= 0.09101
calculate net output
y= 0.5227368084248941
dk 0.11906907074145694
compute error portion in delta
din1= 0.04762762829658278
din2= 0.011906907074145694
error in delta
fzin1 0.24751996
fzin2 0.20552119000000002
d1= 0.011788788650865037
d2= 0.0024471217110978417
Changes in weights between input and hidden layer
dv11= 0.0
dv21= 0.0029471971627162592
dv01= 0.0029471971627162592
dv12= 0.0
dv22= 0.0006117804277744604
dv02= 0.0006117804277744604
Final weights of network
v= [0.6 0.3]
v2 [-0.1  0.4]
w= [-0.17  0.42  0.12]
bias b1= 0.30294719716271623  b2= 0.5006117804277744
```

Practical 4b: Write a Program For Error Back Propagation Algorithm (Ebpa) Learning

```
import math
a0=-1
t=-1
w10=float(input("Enter weight first network"))
b10=float(input("Enter base first network:"))
w20=float(input("Enter weight second network:"))
b20=float(input("Enter base second network:"))
c=float(input("Enter learning coefficient:"))
n1=float(w10*c+b10)
a1=math.tanh(n1)
n2=float(w20*a1+b20)
a2=math.tanh(float(n2))
e=t-a2
s2=-2*(1-a2*a2)*e
s1=(1-a1*a1)*w20*s2

w21=w20-(c*s2*a1)
w11=w10-(c*s1*a0)
b21=b20-(c*s2)
b11=b10-(c*s1)
print("The updated weight of first n/w w11=",w11)
print("The uploaded weight of second n/w w21= ",w21)
print("The updated base of first n/w b10=",b10)
print("The updated base of second n/w b20= ",b20)
```

```
Enter weight first network:12
Enter base first network:35
Enter weight second network:23
Enter base second network:45
Enter learning coefficient:11
The updated weight of first n/w w11= 12.0
The uploaded weight of second n/w w21=  23.0
The updated base of first n/w b10= 35.0
The updated base of second n/w b20=  45.0
```

Practical 5

Practical 5a: Write a program for Hopfield Network.

```
#include "hop.h"
neuron::neuron(int *j)
{
inti;
for(i=0;i<4;i++)
    {
    weightv[i]= *(j+i);
    }
}
int neuron::act(int m, int *x)
{
inti;
int a=0;
for(i=0;i<m;i++)
    {
    a += x[i]*weightv[i];
    }
return a;
}
int network::threshld(int k)
{
if(k>=0)
    return (1);
else
    return (0);
}
network::network(int a[4],int b[4],int c[4],int d[4])
{
nrn[0] = neuron(a) ;
nrn[1] = neuron(b) ;
nrn[2] = neuron(c) ;
nrn[3] = neuron(d) ;
}
void network::activation(int *patrn)
{
inti,j;
for(i=0;i<4;i++)
    {
    for(j=0;j<4;j++)
        {
        cout<<"\n nrn["<<i<<"].weightv["<<j<<"] is "
            <<nrn[i].weightv[j];
        }
    nrn[i].activation = nrn[i].act(4,patrn);
    cout<<"\nactivation is "<<nrn[i].activation;
```

```
    output[i]=threshld(nrn[i].activation);
    cout<<"\noutput value is  "<<output[i]<<"\n";
    }
}
void main ()
{
int patrn1[]= {1,0,1,0},i;
int wt1[]= {0,-3,3,-3};
int wt2[]= {-3,0,-3,3};
int wt3[]= {3,-3,0,-3};
int wt4[]= {-3,3,-3,0};
cout<<"\nTHIS PROGRAM IS FOR A HOPFIELD NETWORK WITH A SINGLE LAYER
OF";
cout<<"\n4 FULLY INTERCONNECTED NEURONS. THE NETWORK SHOULD
RECALLTHE";
cout<<"\nPATTERNS 1010 AND 0101 CORRECTLY.\n";
//create the network by calling its constructor.
// the constructor calls neuron constructor as many times as thenumber of
// neurons in the network.
network h1(wt1,wt2,wt3,wt4);
//present a pattern to the network and get the activations of theneurons
h1.activation(patrn1);
//check if the pattern given is correctly recalled and give message
for(i=0;i<4;i++)
    {
    if (h1.output[i] == patrn1[i])
        cout<<"\n pattern= "<<patrn1[i]<<
        "  output = "<<h1.output[i]<<"  component matches";
    else
        cout<<"\n pattern= "<<patrn1[i]<<
        "  output = "<<h1.output[i]<<
        "  discrepancy occurred";
    }
cout<<"\n\n";
int patrn2[]= {0,1,0,1};
h1.activation(patrn2);
for(i=0;i<4;i++)
    {
    if (h1.output[i] == patrn2[i])
        cout<<"\n pattern= "<<patrn2[i]<<
        "  output = "<<h1.output[i]<<"  component matches";
    else
        cout<<"\n pattern= "<<patrn2[i]<<
        "  output = "<<h1.output[i]<<
        "  discrepancy occurred";
     }
}
======== End code of main program==============
```

```
//Hop.h
//Single layer Hopfield Network with 4 neurons
#include <stdio.h>
#include <iostream.h>
#include <math.h>
class neuron
{
protected:
    int activation;
    friend class network;
public:
    intweightv[4];
    neuron() {};
    neuron(int *j) ;
    int act(int, int*);
};
class network
{
public:
    neuron   nrn[4];
    int output[4];
    intthreshld(int) ;
    void activation(int j[4]);
    network(int*,int*,int*,int*);
};
```

Practical 5b: Write a program for Radial Basis function

```
from scipy import *
from scipy.linalg import norm, pinv
from matplotlib import pyplot as plt
class RBF:
    def __init__(self, indim, numCenters, outdim):
        self.indim =indim
        self.outdim =outdim
        self.numCenters =numCenters
        self.centers =[random.uniform(-1, 1, indim) for i in range(numCenters)]
        self.beta =8
        self.W =random.random((self.numCenters, self.outdim))
    def _basisfunc(self, c, d):
        assert len(d) ==self.indim
        return exp(-self.beta *norm(c-d)**2)
    def _calcAct(self, X):
        # calculate activations of RBFs
        G =zeros((X.shape[0], self.numCenters), float)
        for ci, c in enumerate(self.centers):
            for xi, x in enumerate(X):
                G[xi,ci] =self._basisfunc(c, x)
        return G
    def train(self, X, Y):
        """ X: matrix of dimensions n x indim
            y: column vector of dimension n x 1 """
        # choose random center vectors from training set
        rnd_idx =random.permutation(X.shape[0])[:self.numCenters]
        self.centers =[X[i,:] for i in rnd_idx]
        print("center", self.centers)
        # calculate activations of RBFs
        G =self._calcAct(X)
        print (G)
        # calculate output weights (pseudoinverse)
        self.W =dot(pinv(G), Y)
    def test(self, X):
        """ X: matrix of dimensions n x indim """
        G =self._calcAct(X)
        Y =dot(G, self.W)
        return Y
if __name__ =='__main__':
    # ----- 1D Example ------------------------------------------------
    n =100

    x =mgrid[-1:1:complex(0,n)].reshape(n, 1)
    # set y and add random noise
    y =sin(3*(x+0.5)**3-1)
    # y += random.normal(0, 0.1, y.shape)
    # rbf regression
```

```
rbf =RBF(1, 10, 1)
rbf.train(x, y)
z =rbf.test(x)
# plot original data
plt.figure(figsize=(12, 8))
plt.plot(x, y, 'k-')
# plot learned model
plt.plot(x, z, 'r-', linewidth=2)
# plot rbfs
plt.plot(rbf.centers, zeros(rbf.numCenters), 'gs')
for c in rbf.centers:
    # RF prediction lines
    cx =arange(c-0.7, c+0.7, 0.01)
    cy =[rbf._basisfunc(array([cx_]), array([c])) for cx_ in cx]
    plt.plot(cx, cy, '-', color='gray', linewidth=0.2)
plt.xlim(-1.2, 1.2)
plt.show()
```

Practical 6

Practical 6a: Self-Organizing Maps

The SOM algorithm is used to compress the information to produce a similarity graph while preserving the topologic relationship of the input data space.

The basic SOM model construction algorithm can be interpreted as follows:

1) Create and initialize a matrix (weight vector) randomly to hold the neurons. If the matrix can be initialized with order and roughly compiles with the input density function, the map will converge quickly

2) Read the input data space. For each observation (instance), use the optimum fit approach, which is based on the Euclidean distance

$c = arg_i \min \| x - m_i \|$

to find the neuron which best matches this observation. Let x denote the training vector from the observation and mi denote a single neuron in the matrix. Update that neuron to resemble that observation using the following equation:

$mi\ (t + 1) = mi\ (t) + h(t)[x(t) - mi\ (t)]$ (4) $mi(t)$:

the weight vector before the neuron is updated.

$(t + 1)$: the weight vector after the neuron is updated.

(t): the training vector from the observation.

$h(t)$: the neighborhood function (a smoothing kernel defined over the lattice points), defined though the following equation:

$h(t) = \{\ \alpha(t), i \in Nc\ 0, i \in Nc$ (5)

: the neighborhood set, which decreases with time.

(t): the learning-rate factor which can be linear, exponential or inversely proportional.

It is a monotonically decreasing function of time (t)

In general, SOMs might be useful for visualizing high-dimensional data in terms of its similarity structure. Especially large SOMs (i.e. with large number of Kohonen units) are known to perform mappings that preserve the topology of the original data, i.e. neighboring data points in input space will also be represented in adjacent locations on the SOM.

The following code shows the 'classic' color mapping example, i.e. the SOM will map a number of colors into a rectangular area.

```
frommvpa2.suiteimport*
```

First, we define some colors as RGB values from the interval (0,1), i.e. with white being (1, 1, 1) and black being (0, 0, 0). Please note, that a substantial proportion of the defined colors represent variations of 'blue', which are supposed to be represented in more detail in the SOM.

```
colors=np.array(
[[0.,0.,0.],
[0.,0.,1.],
```

```
[0.,0.,0.5],
[0.125,0.529,1.0],
[0.33,0.4,0.67],
[0.6,0.5,1.0],
[0.,1.,0.],
[1.,0.,0.],
[0.,1.,1.],
[1.,0.,1.],
[1.,1.,0.],
[1.,1.,1.],
[.33,.33,.33],
[.5,.5,.5],
[.66,.66,.66]])

# store the names of the colors for visualization later on
color_names= \
['black','blue','darkblue','skyblue',
'greyblue','lilac','green','red',
'cyan','violet','yellow','white',
'darkgrey','mediumgrey','lightgrey']
```

Now we can instantiate the mapper. It will internally use a so-called Kohonen layer to map the data onto. We tell the mapper to use a rectangular layer with 20 x 30 units. This will be the output space of the mapper. Additionally, we tell it to train the network using 400 iterations and to use custom learning rate.

```
som=SimpleSOMMapper((20,30),400,learning_rate=0.05)
```

Finally, we train the mapper with the previously defined ‘color’ dataset.

```
som.train(colors)
```

Each unit in the Kohonen layer can be treated as a pointer into the high-dimensional input space, that can be queried to inspect which input subspaces the SOM maps onto certain sections of its 2D output space. The color-mapping generated by this example’s SOM can be shown with a single matplotlib call:

```
pl.imshow(som.K,origin='lower')
```

And now, let’s take a look onto which coordinates the initial training prototypes were mapped to. The get those coordinates we can simply feed the training data to the mapper and plot the output.

```
mapped=som(colors)

pl.title('Color SOM')
# SOM's kshape is (rows x columns), while matplotlib wants (X x Y)
fori,minenumerate(mapped):
pl.text(m[1],m[0],color_names[i],ha='center',va='center',
bbox=dict(facecolor='white',alpha=0.5,lw=0))
```

Color SOM
yellow
white
lightgrey
mediumgrey
violet
red
darkgrey
greyblue
lilac
black
skyblue
green
darkblue
blue
cyan
0
5
10
15
20
25
5
10
15

PRACTICAL 6B: ADAPTIVE RESONANCE THEORY

```
from__future__import
division


importnumpyas np
fromneupy.utilsimportformat_data
fromneupy.core.propertiesimport (ProperFractionProperty,
IntProperty)
fromneupy.algorithms.baseimportBaseNetwork
__all__= ('ART1',)
classART1(BaseNetwork):
"""
    Adaptive Resonance Theory (ART1) Network for binary
    data clustering.
    Notes
    -----
    - Weights are not random, so the result will be
      always reproduceble.
    Parameters
    ----------
    rho : float
        Control reset action in training process. Value must be
        between ``0`` and ``1``, defaults to ``0.5``.
n_clusters : int
        Number of clusters, defaults to ``2``. Min value is also
``2``.
    {BaseNetwork.Parameters}
    Methods
    -------
    train(X)
        ART trains until all clusters are found.
    predict(X)
        Each prediction trains a new network. It's an alias to
        the ``train`` method.
    {BaseSkeleton.fit}
    Examples
    --------
>>>import numpy as np
>>>from neupy import algorithms
>>>
>>>data = np.array([
...     [0, 1, 0],
...     [1, 0, 0],
...     [1, 1, 0],
... ])
>>>>
```

```
>>>artnet = algorithms.ART1(
...     step=2,
...     rho=0.7,
... n_clusters=2,
...     verbose=False
... )
>>>artnet.predict(data)
   array([ 0.,  1.,  1.])
   """
   rho =ProperFractionProperty(default=0.5)
n_clusters=IntProperty(default=2, minval=2)
deftrain(self, X):
        X =format_data(X)
ifX.ndim!=2:
raiseValueError("Input value must be 2 dimensional, got "
"{}".format(X.ndim))
nsamples, n_features=X.shape
n_clusters=self.n_clusters
        step =self.step
        rho =self.rho
ifnp.any((X !=0) & (X !=1)):
raiseValueError("ART1 Network works only with binary
matrices")
ifnothasattr(self, 'weight_21'):
self.weight_21 =np.ones((n_features, n_clusters))
ifnothasattr(self, 'weight_12'):
            scaler = step / (step +n_clusters-1)
self.weight_12 = scaler *self.weight_21.T
        weight_21 =self.weight_21
        weight_12 =self.weight_12
ifn_features!= weight_21.shape[0]:
raiseValueError("Input data has invalid number of features. "
"Got {} instead of {}"
"".format(n_features, weight_21.shape[0]))
        classes =np.zeros(n_samples)
# Train network
fori, p inenumerate(X):
disabled_neurons= []
reseted_values= []
            reset =True
while reset:
                output1 = p
                input2 = np.dot(weight_12, output1.T)
                output2 =np.zeros(input2.size)
                input2[disabled_neurons] =-np.inf
winner_index= input2.argmax()
                output2[winner_index] =1
                expectation = np.dot(weight_21, output2)
                output1 =np.logical_and(p, expectation).astype(int)
```

```
reset_value= np.dot(output1.T, output1) / np.dot(p.T, p)
            reset =reset_value< rho
if reset:
disabled_neurons.append(winner_index)
reseted_values.append((reset_value, winner_index))
iflen(disabled_neurons) >=n_clusters:
# Got this case only if we test all possible clusters
                reset =False
winner_index=None
ifnot reset:
ifwinner_indexisnotNone:
                    weight_12[winner_index, :] = (step * output1) / (
                        step + np.dot(output1.T, output1) -1
                    )
                    weight_21[:, winner_index] = output1
else:
# Get result with the best `rho`
winner_index=max(reseted_values)[1]
                classes[i] =winner_index
return classes
defpredict(self, X):
returnself.train(X)
```

Practical 7

Practical 7a: Line Separation

```
import numpy as np
import matplotlib.pyplot as plt
def create_distance_function(a, b, c):
    """ 0 = ax + by + c """
    def distance(x, y):
        """ returns tuple (d, pos)
            d is the distance
            If pos == -1 point is below the line,
            0 on the line and +1 if above the line
        """
        nom = a * x + b * y + c
        if nom == 0:
            pos = 0
        elif (nom<0 and b<0) or (nom>0 and b>0):
            pos = -1
        else:
            pos = 1
        return (np.absolute(nom) / np.sqrt( a ** 2 + b ** 2), pos)
    return distance

points = [ (3.5, 1.8), (1.1, 3.9) ]
fig, ax = plt.subplots()
ax.set_xlabel("sweetness")
ax.set_ylabel("sourness")
ax.set_xlim([-1, 6])
ax.set_ylim([-1, 8])
X = np.arange(-0.5, 5, 0.1)
colors = ["r", ""] # for the samples
size = 10
for (index, (x, y)) in enumerate(points):
    if index== 0:
        ax.plot(x, y, "o", color="darkorange", markersize=size)
    else:
        ax.plot(x, y, "oy", markersize=size)
        step = 0.05
for x in np.arange(0, 1+step, step):
    slope = np.tan(np.arccos(x))
    dist4line1 = create_distance_function(slope, -1, 0)
    #print("x: ", x, "slope: ", slope)
    Y = slope * X

    results = []
    for point in points:
        results.append(dist4line1(*point))
    #print(slope, results)
    if (results[0][1] != results[1][1]):
```

```
        ax.plot(X, Y, "g-")
    else:
        ax.plot(X, Y, "r-")

plt.show()
```

Practical 8

Practical 8a: Membership and Identity operators in, not in.

```
# Python program to illustrate
# Finding common member in list
# without using 'in' operator

# Define a function() that takes two lists
def overlapping(list1,list2):
	c=0
	d=0
	for i in list1:
		c+=1
	for i in list2:
		d+=1
	for i in range(0,c):
		for j in range(0,d):
			if(list1[i]==list2[j]):
				return 1
	return 0
list1=[1,2,3,4,5]
list2=[6,7,8,9]
if(overlapping(list1,list2)):
	print("overlapping")
else:
	print("not overlapping")
```

```
# Python program to illustrate
# Finding common member in list
# without using 'in' operator

# Define a function() that takes two lists
def overlapping(list1,list2):

	c=0
	d=0
	for i in list1:
		c+=1
	for i in list2:
		d+=1
	for i in range(0,c):
		for j in range(0,d):
			if(list1[i]==list2[j]):
				return 1
	return 0
list1=[1,2,3,4,5]
list2=[6,7,8,9]
```

```
if(overlapping(list1,list2)):
        print("overlapping")
else:
        print("not overlapping")
```

Practical 8b: Membership and Identity Operators is, is not

```
# Python program to illustrate the use
# of 'is' identity operator
x = 5
if (type(x) is int):
        print ("true")
else:
        print ("false")
```

```
# Python program to illustrate the
# use of 'is not' identity operator
x = 5.2
if (type(x) is not int):
        print ("true")
else:
        print ("false")
```

Practical 9

Practical 9a: Find the ratios using fuzzy logic

pip install fuzzywuzzy
pip install python-Levenshtein

```
# Python code showing all the ratios together,
# make sure you have installed fuzzywuzzy module

from fuzzywuzzy import fuzz
from fuzzywuzzy import process
s1 = "I love fuzzysforfuzzys"
s2 = "I am loving fuzzysforfuzzys"
print ("FuzzyWuzzy Ratio:", fuzz.ratio(s1, s2))
print ("FuzzyWuzzyPartialRatio: ", fuzz.partial_ratio(s1, s2))
print ("FuzzyWuzzyTokenSortRatio: ", fuzz.token_sort_ratio(s1, s2))
print ("FuzzyWuzzyTokenSetRatio: ", fuzz.token_set_ratio(s1, s2))
print ("FuzzyWuzzyWRatio: ", fuzz.WRatio(s1, s2),'\n\n')
# for process library,
query = 'fuzzys for fuzzys'
choices = ['fuzzy for fuzzy', 'fuzzy fuzzy', 'g. for fuzzys']
print ("List of ratios: ")
print (process.extract(query, choices), '\n')
print ("Best among the above list: ",process.extractOne(query, choices))
```

Practical 9b: Solve Tipping Problem using fuzzy logic

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# New Antecedent/Consequent objects hold universe variables and membership
# functions
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

# Auto-membership function population is possible with .automf(3, 5, or 7)
quality.automf(3)
service.automf(3)

# Custom membership functions can be built interactively with a familiar,
# Pythonic API
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

"""
To help understand what the membership looks like, use the ``view`` methods.
"""

# You can see how these look with .view()
quality['average'].view()
"""
.. image:: PLOT2RST.current_figure
"""
service.view()
"""
.. image:: PLOT2RST.current_figure
"""
tip.view()
"""
.. image:: PLOT2RST.current_figure
```

Practical 10

Practical 10: Implementation of simple genetic algorithm

```
import random
# Number of individuals in each generation
POPULATION_SIZE =100
# Valid genes
GENES ='''abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOP
QRSTUVWXYZ 1234567890, .-;:_!"#%&/()=?@${[]}'''
# Target string to be generated
TARGET ="I love GeeksforGeeks"
class Individual(object):
    '''
    Class representing individual in population
    '''
    def __init__(self, chromosome):
        self.chromosome =chromosome
        self.fitness =self.cal_fitness()
    @classmethod
    def mutated_genes(self):
        '''
        create random genes for mutation
        '''
        global GENES
        gene =random.choice(GENES)
        return gene
    @classmethod
    def create_gnome(self):
        '''
        create chromosome or string of genes
        '''
        global TARGET
        gnome_len =len(TARGET)
        return[self.mutated_genes() for _ in range(gnome_len)]
    def mate(self, par2):
        '''
        Perform mating and produce new offspring
        '''
        # chromosome for offspring
        child_chromosome =[]
        for gp1, gp2 in zip(self.chromosome, par2.chromosome):

            # random probability
            prob =random.random()

            # if prob is less than 0.45, insert gene
            # from parent 1
            if prob< 0.45:
```

```
            child_chromosome.append(gp1)

        # if prob is between 0.45 and 0.90, insert
        # gene from parent 2
        elif prob< 0.90:
            child_chromosome.append(gp2)

        # otherwise insert random gene(mutate),
        # for maintaining diversity
        else:
            child_chromosome.append(self.mutated_genes())

    # create new Individual(offspring) using
    # generated chromosome for offspring
    return Individual(child_chromosome)

  def cal_fitness(self):
    '''
    Calculate fittness score, it is the number of
    characters in string which differ from target
    string.
    '''
    global TARGET
    fitness =0
    for gs, gt in zip(self.chromosome, TARGET):
        if gs !=gt: fitness+=1
    return fitness

# Driver code
def main():
  global POPULATION_SIZE

  #current generation
  generation =1

  found =False
  population =[]

  # create initial population
  for _ in range(POPULATION_SIZE):
            gnome =Individual.create_gnome()
            population.append(Individual(gnome))

  while not found:

    # sort the population in increasing order of fitness score
    population =sorted(population, key =lambda x:x.fitness)

    # if the individual having lowest fitness score ie.
```

```
        # 0 then we know that we have reached to the target
        # and break the loop
        if population[0].fitness <=0:
            found =True
            break

        # Otherwise generate new offsprings for new generation
        new_generation =[]

        # Perform Elitism, that mean 10% of fittest population
        # goes to the next generation
        s =int((10*POPULATION_SIZE)/100)
        new_generation.extend(population[:s])

        # From 50% of fittest population, Individuals
        # will mate to produce offspring
        s =int((90*POPULATION_SIZE)/100)
        for _ in range(s):
            parent1 =random.choice(population[:50])
            parent2 =random.choice(population[:50])
            child =parent1.mate(parent2)
            new_generation.append(child)

        population =new_generation

        print("Generation: {}\tString: {}\tFitness:
{}".format(generation,"".join(population[0].chromosome),population[0].fitness))
        generation +=1


    print("Generation: {}\tString: {}\tFitness: {}".format(generation,
"".join(population[0].chromosome),
          population[0].fitness))

if __name__ =='__main__':
    main()
```

Output:

```
Generation: 1    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18
Generation: 2    String: tO{"-?=jH[k8=B4]Oe@}    Fitness: 18
Generation: 3    String: .#lRWf9k_Ifslw #O$k_    Fitness: 17
Generation: 4    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 5    String: .-1Rq?9mHqk3Wo]3rek_    Fitness: 16
Generation: 6    String: A#ldW) #lIkslwcVek)    Fitness: 14
Generation: 7    String: A#ldW) #lIkslwcVek)    Fitness: 14
Generation: 8    String: (, o x _x%Rs=, 6Peek3   Fitness: 13
                 .
                 .
```

.

```
Generation: 29    String: I lope Geeks#o, Geeks   Fitness: 3
Generation: 30    String: I loMeGeeksfoBGeeks    Fitness: 2
Generation: 31    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 32    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 33    String: I love Geeksfo0Geeks    Fitness: 1
Generation: 34    String: I love GeeksforGeeks    Fitness: 0
```

Practical 10 b: Create two classes: City and Fitness using Genetic algorithm

first create a City class that will allow us to create and handle our cities.

Create Population

```
import numpy as np, random, operator, pandas as pd, matplotlib.pyplot as plt
from tkinter import Tk, Canvas, Frame, BOTH, Text
import math
class City:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def distance(self, city):
        xDis = abs(self.x - city.x)
        yDis = abs(self.y - city.y)
        distance = np.sqrt((xDis ** 2) + (yDis ** 2))
        return distance
    def __repr__(self):
        return "(" + str(self.x) + "," + str(self.y) + ")"
class Fitness:
    def __init__(self, route):
        self.route = route
        self.distance = 0
        self.fitness= 0.0
    def routeDistance(self):
        if self.distance ==0:
            pathDistance = 0
            for i in range(0, len(self.route)):
                fromCity = self.route[i]
                toCity = None
                if i + 1 < len(self.route):
                    toCity = self.route[i + 1]
                else:
                    toCity = self.route[0]
                pathDistance += fromCity.distance(toCity)
            self.distance = pathDistance
        return self.distance
    def routeFitness(self):
        if self.fitness == 0:
            self.fitness = 1 / float(self.routeDistance())
        return self.fitness
def createRoute(cityList):
```

```
    route = random.sample(cityList, len(cityList))
    return route
def initialPopulation(popSize, cityList):
    population = []
    for i in range(0, popSize):
        population.append(createRoute(cityList))
    return population
def rankRoutes(population):
    fitnessResults = {}
    for i in range(0,len(population)):
        fitnessResults[i] = Fitness(population[i]).routeFitness()
    return sorted(fitnessResults.items(), key = operator.itemgetter(1), reverse = True)
def selection(popRanked, eliteSize):
    selectionResults = []
    df = pd.DataFrame(np.array(popRanked), columns=["Index","Fitness"])
    df['cum_sum'] = df.Fitness.cumsum()
    df['cum_perc'] = 100*df.cum_sum/df.Fitness.sum()
    for i in range(0, eliteSize):
        selectionResults.append(popRanked[i][0])
    for i in range(0, len(popRanked) - eliteSize):
        pick = 100*random.random()
        for i in range(0, len(popRanked)):
            if pick <= df.iat[i,3]:
                selectionResults.append(popRanked[i][0])
                break
    return selectionResults
def matingPool(population, selectionResults):
    matingpool = []
    for i in range(0, len(selectionResults)):
        index = selectionResults[i]
        matingpool.append(population[index])
    return matingpool
def breed(parent1, parent2):
    child = []
    childP1 = []
    childP2 = []
    geneA = int(random.random() * len(parent1))
    geneB = int(random.random() * len(parent1))
    startGene = min(geneA, geneB)
    endGene = max(geneA, geneB)
    for i in range(startGene, endGene):
        childP1.append(parent1[i])
    childP2 = [item for item in parent2 if item not in childP1]
    child = childP1 + childP2
    return child
def breedPopulation(matingpool, eliteSize):
    children = []
    length = len(matingpool) - eliteSize
    pool = random.sample(matingpool, len(matingpool))
```

```
    for i in range(0,eliteSize):
        children.append(matingpool[i])
    for i in range(0, length):
        child = breed(pool[i], pool[len(matingpool)-i-1])
        children.append(child)
    return children
def mutate(individual, mutationRate):
    for swapped in range(len(individual)):
        if(random.random() < mutationRate):
            swapWith = int(random.random() * len(individual))
            city1 = individual[swapped]
            city2 = individual[swapWith]
            individual[swapped] = city2
            individual[swapWith] = city1
    return individual
def mutatePopulation(population, mutationRate):
    mutatedPop = []
    for ind in range(0, len(population)):
        mutatedInd = mutate(population[ind], mutationRate)
        mutatedPop.append(mutatedInd)
    return mutatedPop
def nextGeneration(currentGen, eliteSize, mutationRate):
    popRanked = rankRoutes(currentGen)
    selectionResults = selection(popRanked, eliteSize)
    matingpool = matingPool(currentGen, selectionResults)
    children = breedPopulation(matingpool, eliteSize)
    nextGeneration = mutatePopulation(children, mutationRate)
    return nextGeneration
def geneticAlgorithm(population, popSize, eliteSize, mutationRate, generations):
    pop = initialPopulation(popSize, population)
    print("Initial distance: " + str(1 / rankRoutes(pop)[0][1]))
    for i in range(0, generations):
        pop = nextGeneration(pop, eliteSize, mutationRate)
    print("Final distance: " + str(1 / rankRoutes(pop)[0][1]))
    bestRouteIndex = rankRoutes(pop)[0][0]
    bestRoute = pop[bestRouteIndex]
    return bestRoute
def geneticAlgorithmPlot(population, popSize, eliteSize, mutationRate, generations):
    pop = initialPopulation(popSize, population)
    progress = []
    progress.append(1 / rankRoutes(pop)[0][1])
    for i in range(0, generations):
        pop = nextGeneration(pop, eliteSize, mutationRate)
        progress.append(1 / rankRoutes(pop)[0][1])
    plt.plot(progress)
    plt.ylabel('Distance')
    plt.xlabel('Generation')
    plt.show()
def main():
```

```
    cityList = []
    for i in range(0,25):
        cityList.append(City(x=int(random.random() * 200), y=int(random.random() * 200)))
    geneticAlgorithmPlot(population=cityList, popSize=100, eliteSize=20,
mutationRate=0.01, generations=500)
if __name__ == '__main__':
    main()
```