

Submitted in partial fulfillment of the
Requirements for the award of the Degree of

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)

By

SIDDIQUE IMRAN AHMED

Seat Number: (3254636)

**SUBJECT NAME:
DATA SCIENCE AND RESERCH IN COMPUTING**



DEPARTMENT OF INFORMATION TECHNOLOGY

N. G. ACHARYA & D. K. MARATHE COLLEGE
(Affiliated to University of Mumbai)
MUMBAI , 400071
MAHARASHTRA
2022-23

DEPARTMENT OF INFORMATION TECHNOLOGY
N. G. ACHARYA & D. K. MARATHE COLLEGE
(Affiliated to University of Mumbai)
MUMBAI – MAHARASHTRA - 400071

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that **Siddiqui Imran Ahmed** bearing Seat No: 3254636 submitted journal of DATA SCIENCE AND RESEARCH IN COMPUTING in partial fulfillment of the requirements for the award of Degree of **MASTER OF SCIENCE in INFORMATION TECHNOLOGY** from University of Mumbai.

Internal Guide

Coordinator

External Examiner

Date:

College Seal

INDEX

Sr. No.	Practicals	Signature
1.	A. Write Python program to convert CSV to HORUS format. B. Write python program to convert XML to HORUS format. C. Write python program to convert picture JPEG to HOURS format.	
2.	Utilities and Auditing a. Fixer utility b. Data Binning or Bucketing c. Averaging of Data d. Outlier Detection e. Logging	
3.	A. Vermeulen PLC B. Building a diagram for scheduling of jobs. C. Krenwaller AG Picking content for Bill boards. D. XML Processing	
4.	A. Write python program to create the Network routing diagram form the give data on routers. B. Keep only the rows that contains a maximum of two missing values. C. Write python program to build directed acyclic graph. D. Write a python program to create a delivery route using the give data.	

5.	A. Build the Time, Hub and Satellite. B. Golden Nominal	
6	Transform Superstep	
7	Organize Superstep	

Practical No. 1

A.Text delimited CSV to HORUS format

Write Python / R Program to convert from the following formats to 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/9999Data/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      ...
12      American Samoa
8      Algeria
248      Albania
4      Aland Islands
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/9999Data/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
=====
>>>

```

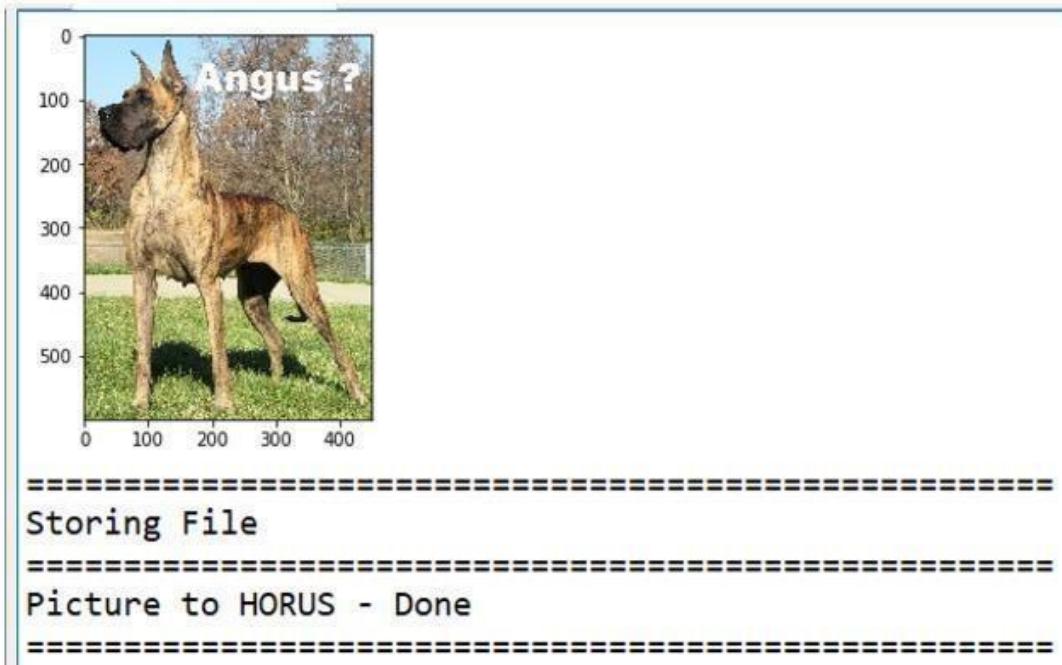
C. Picture (JPEG) to HORUS format

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 :



Practical No 2

Basic Utility

1. Load data as per input agreement.
2. Apply processing rules of utility.
- 3. Save data as per output agreement. There are three types of utilities**

- Data processing utilities
- Maintenance utilities
- Processing utilities

A. Fixer Utilities Code

:

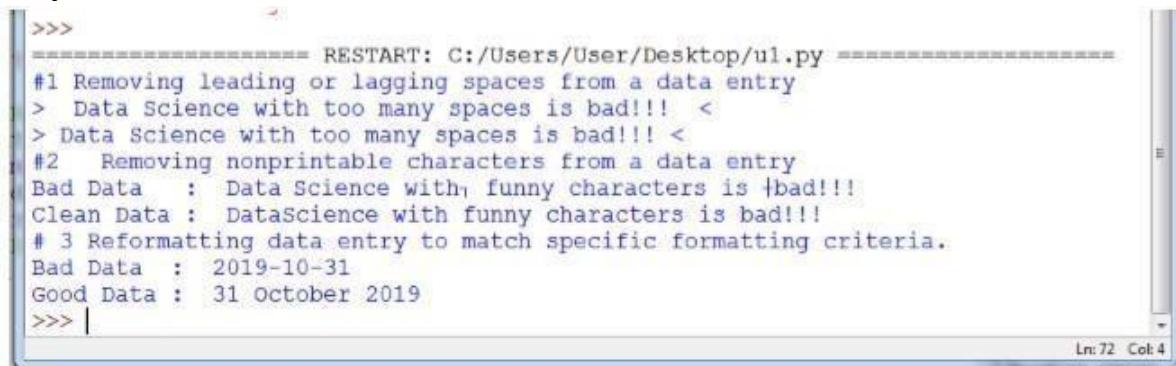
```
import string import  
datetime as dt  
  
# 1 Removing leading or lagging spaces from a data entry print('#1  
Removing leading or lagging spaces from a data entry'); baddata =  
" Data Science with too many spaces is bad!!! "  
print('>',baddata,'<') cleandata=baddata.strip()  
print('>',cleandata,'<')  
  
# 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:



The screenshot shows a terminal window with the following content:

```

>>>
=====
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
>>> |

```

The terminal window has a status bar at the bottom right indicating "Ln: 72 Col: 4".

B. Data binning or Bucketing 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')

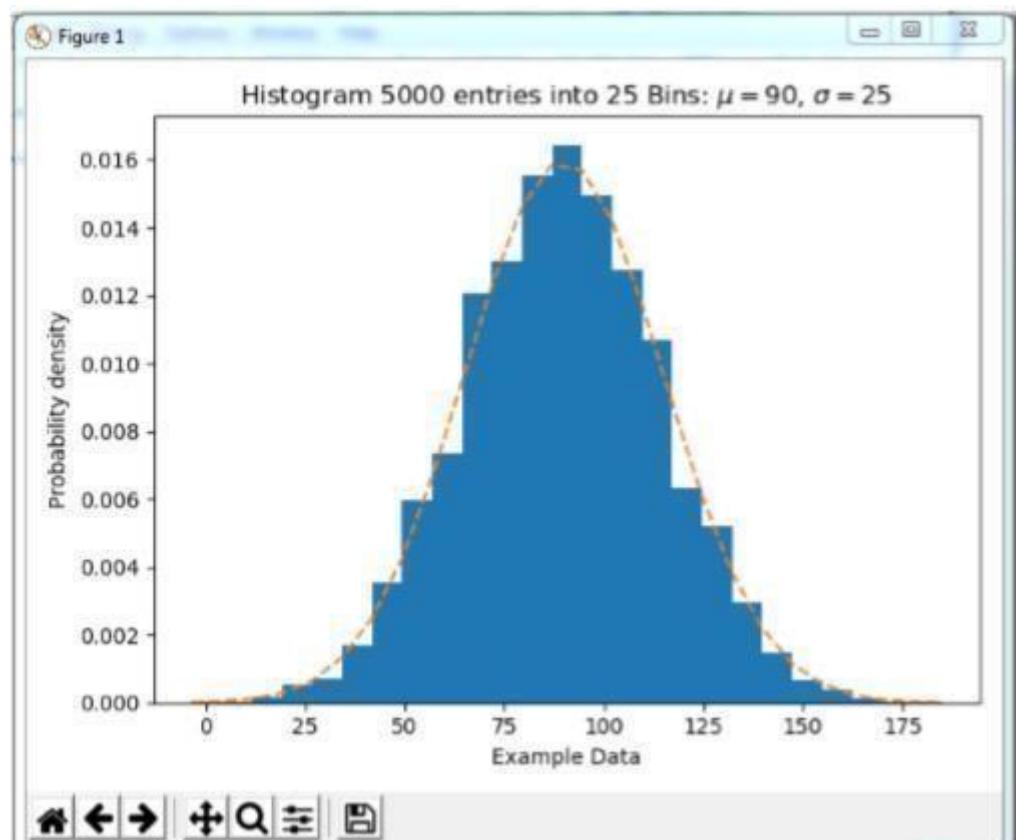
ax.set_ylabel('Probability density') sTitle=r'Histogram ' + str(len(x)) + '
entries into ' + str(num_bins) + ' Bins:
$\mu=' + str(mu) + '$, $\sigma=' +str(sigma) + '$' ax.set_title(sTitle)

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
[3542 rows x 3 columns]
   Country Place_Name    Latitude
DE      Munich     48.143223
GB      London      51.509406
US      New York     40.747044
Name: Latitude, dtype: float64
```

D. 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  
... ... ... ...  
3434 GB London 51.5092  
3435 GB London 51.5092  
3436 GB London 51.5163  
3437 GB London 51.5085  
3438 GB London 51.5136
```

[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
1925	GB	London	51.5237
1926	GB	London	51.5237
1927	GB	London	51.5232
3436	GB	London	51.5163
3438	GB	London	51.5136
Lower than 51.50617687562166			
Country Place_Name Latitude			
1915	GB	London	51.4739
Not Outliers			
Country Place_Name Latitude			
1917	GB	London	51.5085
1918	GB	London	51.5085
1922	GB	London	51.5085
1928	GB	London	51.5085
1929	GB	London	51.5085
...
3432	GB	London	51.5092
3433	GB	London	51.5092
3434	GB	London	51.5092
3435	GB	London	51.5092
3437	GB	London	51.5085

[1485 rows x 3 columns]

E. 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','03Process','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 hdrl in log.handlers[:]:  
  
# remove all old handlers    log.removeHandler(hdrl)  
  
#-----  
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)  
  
30 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='Aplication-' + 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:\V р KHCG\77-Yoke\Yoke_Logging.py =====
Set up: C:/V р KHCG/01-Vermeulen/01-Retrieve/Logging/Logging_61705603-bb6e-47f0-b5a
9-23d42e267311.log
root      : INFO    Practical Data Science is fun!.
Aplication-01-Vermeulen-01-Retrieve-info: INFO    Practical Data Science logge
d information message.
Aplication-01-Vermeulen-01-Retrieve-warning: WARNING  Practical Data Science lo
gged a warning message.
Aplication-01-Vermeulen-01-Retrieve-error: ERROR    Practical Data Science logg
ed an error message.
Set up: C:/V р KHCG/01-Vermeulen/02-Assess/Logging/Logging_a7fecb9b-4d40-474e-bc2d-
994958d85194.log
```

Practical No. 3

A. Vermeulen PLC :

The company has two main jobs on which to focus your attention:

- Designing a routing diagram for company
- Planning a schedule of jobs to be performed for the router network

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

Code :

```
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	-73.988	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 another router in New York (40.7214, -74.0052) is 4.448 kilometers, or 2.762 miles.

B .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.

Code :

```

#####
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!! #####
```

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

C. 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. **code :**

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

Output :

```

>>>
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

D . XML processing.

Code :

```

#####
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=N
one, keep='first', inplace=False)
print('Deduplicated XML Data Frame') print('Rows
:',unxmldata.shape[0]) print('Columns
:',unxmldata.shape[1]) print(unxmldata)
#####
#print('### Done!! #####')
#####
```

OUTPUT :

```

Working Base : C:/VKHCG using win32
*****REDACTED*****
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]
Duplicated 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]

```

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



The screenshot shows a Windows File Explorer window with the following details:

- Path:** Computer > Local Disk (C:) > VKHCG > 02-Krennwallner > 01-Retrieve > 01-EDS > 02-Python
- Files:**
 - skt0p: Retrieve_Online_Visitor.csv
 - wnloads: **Retrieve_Online_Visitor** (highlighted)
 - cent Places: Retrieve_Online_Visitor_values.json
 - Backup_Planes_Mobile_tables.json
- Details View:**

Name	Date modified	Type	Size
Retrieve_Online_Visitor.csv	10/21/2019 6:55 PM	Text File	22 KB
Retrieve_Online_Visitor	10/21/2019 7:07 PM	XML Document	1,713 KB
Retrieve_Online_Visitor_values.json	10/21/2019 6:55 PM	WordStar archive	34 KB
Backup_Planes_Mobile_tables.json	10/21/2019 6:55 PM	WordStar archive	408 KB

Content Preview:

```

<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>40.7520</Latitude>
    <Latitude>40.7520</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>

```

Practical No. 4

Assessing Data

Assess Superstep

Data quality refers to the condition of a set of qualitative or quantitative variables. Data quality is a multidimensional measurement of the acceptability of specific data sets. In business, data quality is measured to determine whether data can be used as a basis for reliable intelligence extraction for supporting organizational decisions. Data profiling involves observing in your data sources all the viewpoints that the information offers. The main goal is to determine if individual viewpoints are accurate and complete. The Assess superstep determines what additional processing to apply to the 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 : Write Python program to create the network routing diagram from the given data on routers.

Code :

```
#####
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 :

Go to C:\VKHCG\01-Vermeulen\02-Assess\01-EDS\02-Python folder and open AssessNetworkRouting-Company.csv

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

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

Code :

```

# -*- 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!!')
#####

```

Output :

	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	256	14	
16	10	Good	Better	Best			15	
17	10	Good		Best	64	164	16	
18	10	Good	Better		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

	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	256	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	

After

Row with more than two missing values got deleted.

C : Write a Python program to build directed acyclic graph.

Code :

```

#####
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() #
```

Output :

Rows : 150

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

Link : US to DE

Link : US to GB

Link : DE to US

Link : DE to GB Link

: GB to US

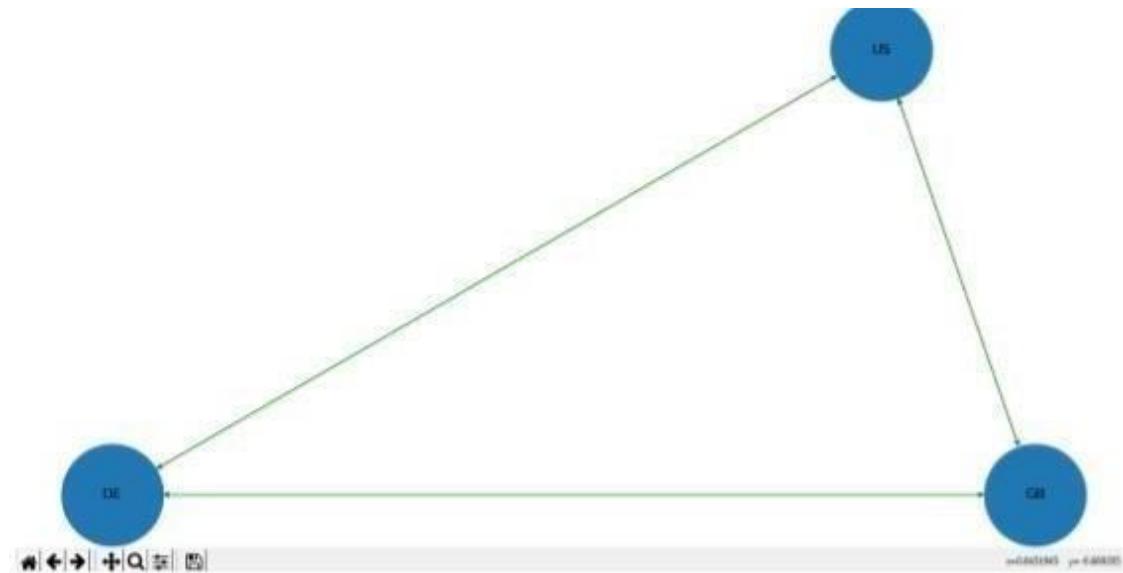
Link : GB to DE

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

Nodes of graph: ['US', 'DE', 'GB']

Edges of graph: [('US', 'DE'), ('US', 'GB'), ('DE', 'US'), ('DE', 'GB'), ('GB', 'US'), ('GB', 'DE')]

```
#####
```



D : 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\03Hillman\02-Assess.

Code :

```
#####
# -*- coding: utf-8 -*-
#####
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=='KA
13']) if
nSet==True:

WarehouseData=WarehouseData.append(WarehouseRawData[WarehouseRawData.postcode=='SW
1W']])
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-HH:',sNode0,'

to ', sNode1, \

Distance:',distancemeters,'met

ers',\

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-SS:',sNode0,'\
to ', sNode1, \
Distance:',distancemeters,'met
ers',\
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!!
#####')

```

Output :

```
[Python 3.7.4 Shell]
File Edit Shell Debug Options Window Help
===== RESTART: C:/VKHCG/03-Hillman/02-Assess/Assess-Shipping-Routes.py =====
#####
Working Base : C:/VKHCG using win32
#####
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcode_Warehouse.csv
Loaded Warehouses : {'id' 'postcode' 'latitude' 'longitude'}
#####
Storing : C:/VKHCG/03-Hillman/02-Assess/SQLite/hillman.db Table: Assess_Warehouse_UK
#####
      id postcode latitude longitude
IDNumber
0       2    AB10  57.13514 -2.11731
1       3    AB11  57.13875 -2.09089
2       4    AB12  57.10100 -2.11060
3000   3003    PA80  0.00000  0.00000
3001   3004    L80   0.00000  0.00000
#####
Rows : 7
#####
Loading : C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve_GB_Postcodes_Shops.csv
Loaded Shops : {'version https://git-lfs.github.com/spec/v1'}
#####
Storing : C:/VKHCG/03-Hillman/02-Assess/SQLite/hillman.db Table: Assess_Shop_UK
```

Practical No. 5

A : Build the Time, Hub and Satellites.

Code :

```
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=tzzone('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]),
              ('UTCDDateTimeValue', [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)

```

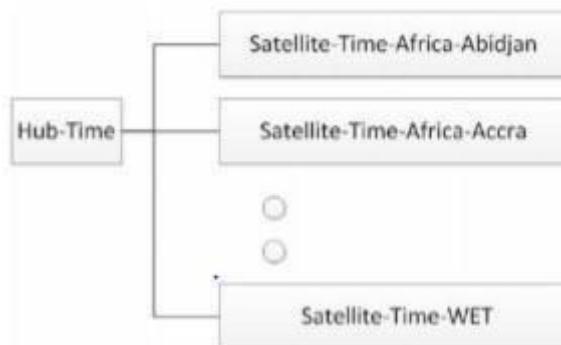
```

TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")
#####
# print('#####') print('Vacuum Databases') sSQL="VACUUM;"
sql.execute(sSQL,conn1) sql.execute(sSQL,conn2) print('#####')
#####
print('#####') 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

Output :



B : 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 program 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.

Code :

```

#####
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-PersonGender'
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('#####') print('###')
Done!! #####

```

Output:

```

===== 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!! #####

```

Practical No 6

Transform Superstep

The Transform superstep allows you, as a data scientist, to take data from the data vault and formulate answers to questions raised by your investigations. The transformation step is the data science process that converts results into insights. It takes standard data science techniques and methods to attain insight and knowledge about the data that then can be transformed into actionable decisions, which, through storytelling, you can explain to non-data scientists what you have discovered in the data lake.

To illustrate the consolidation process, the example show a person being borne. Open a new file in the Python editor and save it as Transform-Gunnarsson_is_Born.py in directory C:\VKHCG\01-Vermeulen\04-Transform.

Code:

```
#####
# 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('#####')

```

Output:

Guðmundur Gunnarsson was born on December 20, 1960, at 9:15 in Landspítali, Hringbraut 101, 101 Reykjavík, Iceland

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

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

Practical No 7

Organize Superstep

The Organize superstep takes the complete data warehouse you built at the end of the Transform superstep and subsections it into business-specific data marts. A data mart is the access layer of the data warehouse environment built to expose data to the users. The data mart is a subset of the data warehouse and is generally oriented to a specific business group.

Horizontal Style

Performing horizontal-style slicing or subsetting of the data warehouse is achieved by applying a filter technique that forces the data warehouse to show only the data for a specific preselected set of filtered outcomes against the data population. The horizontal-style slicing selects the subset of rows from the population while preserving the columns. That is, the data science tool can see the complete record for the records in the subset of records.

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

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

```

Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
RESTART: C:/Users/User/AppData/Local/Programs/Python/Python37-32/Organize01.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

#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
>>> |
Ln: 2301 Col: 4

```

The horizontal-style slicing selects the 194 subset of rows from the 1080 rows while preserving the columns.

Vertical Style

Performing vertical-style slicing or subsetting of the data warehouse is achieved by applying a filter technique that forces the data warehouse to show only the data for specific preselected filtered outcomes against the data population. The vertical-style slicing selects the subset of columns from the population, while preserving the rows. That is, the data science tool can see only the preselected columns from a record for all the records in the population.

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

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

```

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

```
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-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
#####
>>> |
```

Ln: 28 Col: 4

The vertical-style slicing selects 3 of 5 from the population, while preserving the rows [1080].

Island Style

Performing island-style slicing or subsetting of the data warehouse is achieved by applying a combination of horizontal- and vertical-style slicing. This generates a subset of specific rows and specific columns reduced at the same time.

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

```
#####
# -*- coding: utf-8 -*-
#####
##### 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(sDataWareho
useDir):
    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:

```

```

Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
>>>
===== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Island.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): 771
Horizontal Data Set (Columns): 3
#####
>>> |
Ln: 53 Col: 4

```

This generates a subset of 771 rows out of 1080 rows and 3 columns out of 5.

Secure Vault Style

The secure vault is a version of one of the horizontal, vertical, or island slicing techniques, but the outcome is also attached to the person who performs the query. This is common in multi-security environments, where different users are allowed to see different data sets.

This process works well, if you use a role-based access control (RBAC) approach to restricting system access to authorized users. The security is applied against the “role,” and a person can then, by the security system, simply be added or removed from the role, to enable or disable access.

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

```

#####
# -*- coding: utf-8 -*-
#####
##### 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(sDataWareho
useDir):
    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-Secure
#####
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

Association rule learning is a rule-based machine-learning method for discovering interesting relations between variables in large databases, similar to the data you will find in a data lake. The technique enables you to investigate the interaction between data within the same population. Lift

is simply estimated by the ratio of the joint probability of two items x and y, divided by the product of their individual probabilities:

$$Lift = \frac{P(x,y)}{P(x)P(y)}$$

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

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

```

retu
rn 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!! #####
>>> |

```

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_N
ame'][i]
    Node1=CompanyData['Company_Country_N
ame'][j] if Node0 != Node1:
        G.add_edge(Node0,Node1)

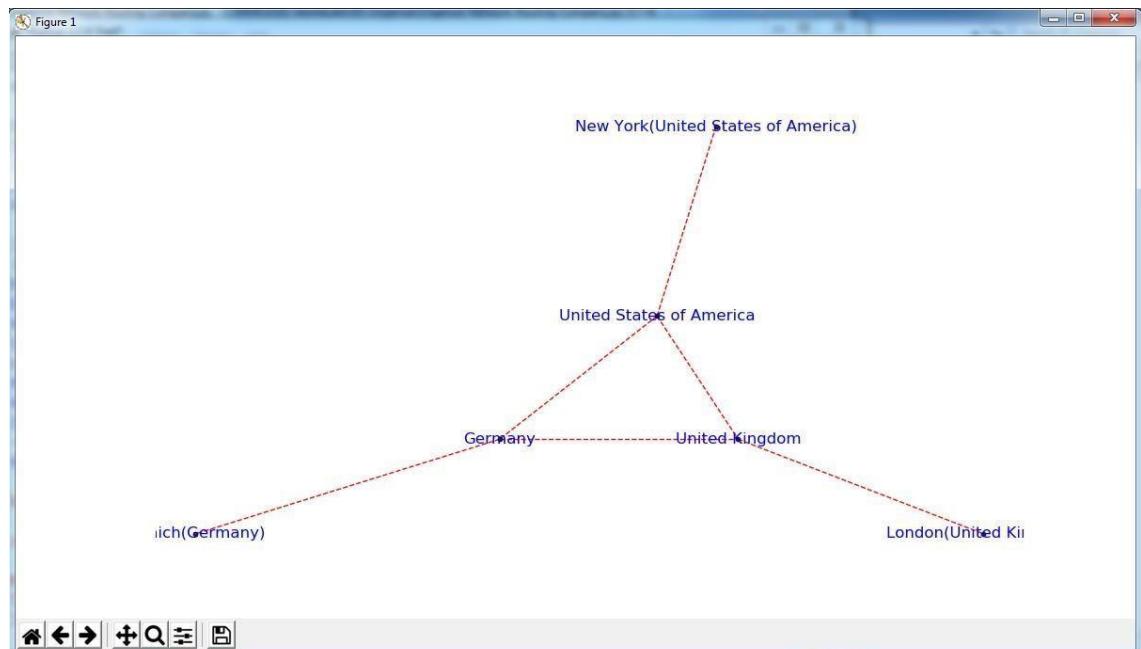
for i in range(CompanyData.shape[0]):
    Node0=CompanyData['Company_Country_Name'][i]
```

```

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

To enable the marketing salespeople to sell billboard content, they will require a diagram to show what billboards connect to which office content publisher. Each of Krennwallner's billboards has a proximity sensor that enables the content managers to record when a registered visitor points his/her smartphone at the billboard content or touches the near-field pad with a mobile phone.

Program will assist you in building an organized graph of the billboards' locations data to help you to gain insights into the billboard locations and content picking process.

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=n
x,Gr
aph(
) for
i in
sSa
mpl
e:
```

```

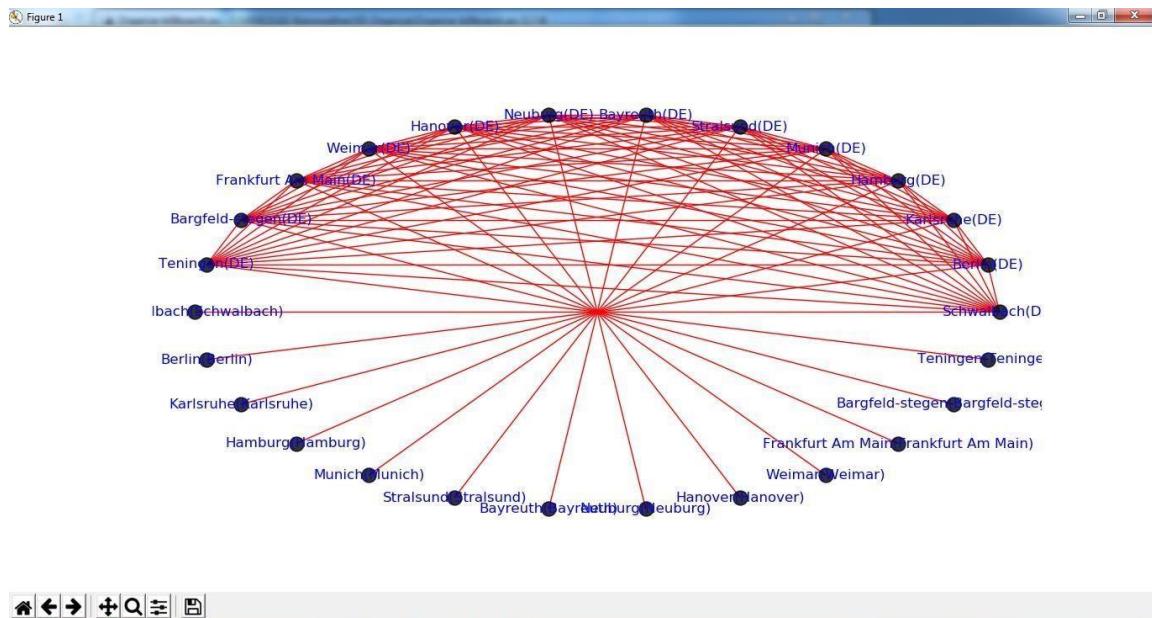
for j
in
sSa
mpl
e:
    Node0=BillboardData['BillboardPlaceName'][i] + '('+
BillboardData['BillboardCountry'][i] + ')'
Node1=BillboardData['BillboardPlaceName'][j] + '('+
BillboardData['BillboardCountry'][i] + ')'      if Node0 != Node1:
    G.add_edge(Node0,Node1)

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

Hillman requires a new delivery route plan from HQ-KA13's delivery region. The managing director has to know the following:

- What his most expensive route is, if the cost is £1.50 per mile and two trips are planned per day
- What the average travel distance in miles is for the region per 30-day month

With your newfound knowledge in building the technology stack for turning data lakes into business assets, can you convert the graph stored in the Assess step called "Assess_Best_Logistics" into the shortest path between the two points?

C:\VKHCG\03-Hillman\05-Organise\Organise-Routes.py

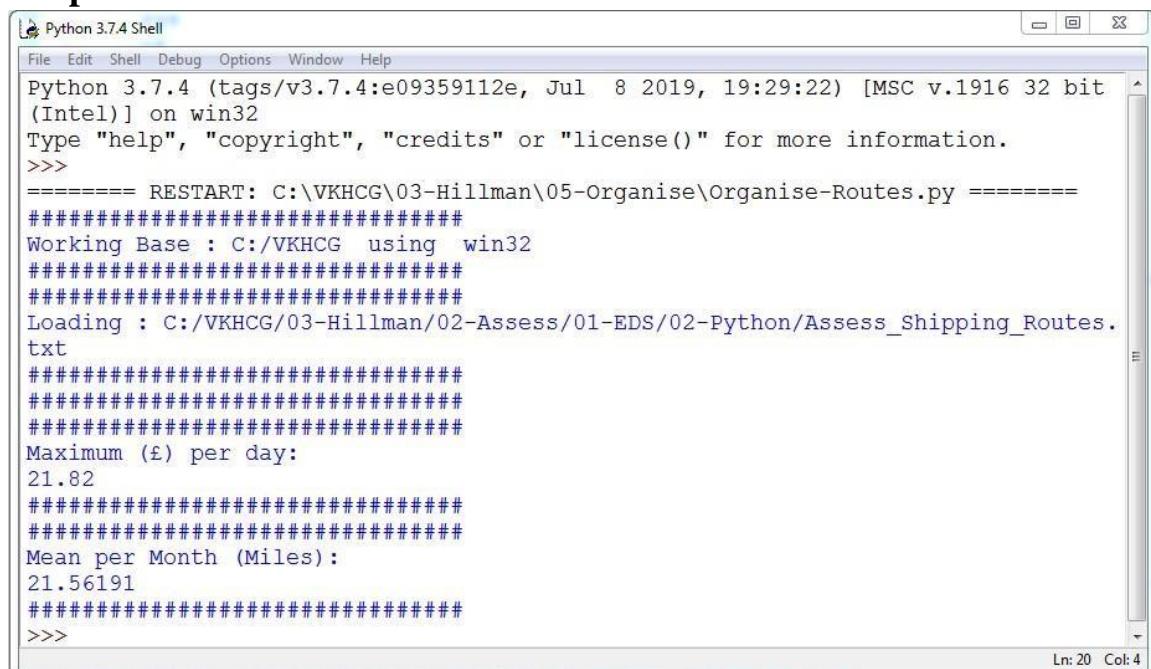
```
# -*- coding: utf-8 -*-
#####
##### import sys import os
import pandas as pd
#####
Base='C:/VKHCG'
#####
print('#####')
print('Working Base :',Base, ' using ',
sys.platform)
print('#####')
#####
sInputFileName='02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.txt'
#####
sOutputFileName='05-Organise/01-EDS/02-Python/Organise-Routes.csv'
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:



The screenshot shows the Python 3.7.4 Shell window. The title bar says "Python 3.7.4 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```

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\03-Hillman\05-Organise\Organise-Routes.py ======
#####
Working Base : C:/VKHCG using win32
#####
Loading : C:/VKHCG/03-Hillman/02-Assess/01-EDS/02-Python/Assess_Shipping_Routes.
txt
#####
Maximum (£) per day:
21.82
#####
Mean per Month (Miles):
21.56191
#####
>>>

```

The status bar at the bottom right indicates "Ln: 20 Col: 4".

[Clark Ltd](#)

Our financial services company has been tasked to investigate the options to convert 1 million pounds sterling into extra income. Mr. Clark Junior suggests using the simple variance in the daily rate between the British pound sterling and the US dollar, to generate extra income from trading. Your chief financial officer wants to know if this is feasible?

[Simple Forex Trading Planner](#)

Your challenge is to take 1 million US dollars or just over six hundred thousand 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**

```
# -*- coding: utf-8 -*-
#####
##### 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(1000000.000000 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=ForexDa
y['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['Mone
y'][j],2))
sCodeIn=ForexDayRate['PCurrency'][j]
sCodeOut=ForexDayRate['Currency'][j]

sSQL='UPDATE MoneyData SET Date= "'+sDay + '", '
sSQL+= 'Money = '+ nMoney + ', Currency="'+ sCodeOut
+'"'    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.

INDEX

Sr. No.	Practicals	Signature
1.	Import Data In Power BI From Microsoft Excel And CSV.	
2.	Import Data In Power BI From Microsoft Excel And CSV	
3.	Collect the primary data in excel and apply the scientific method.	
4.	Collect the secondary data in excel and apply scientific method	
5.	Design the survey from and collect the suitable data and perform analysis	
6.	Implement the different scales of measurement on data: Nominal & Ordinal	
7.	Implement the different scales of measurement on data: Ratio and Interval	
8.	Perform t-test on the data	

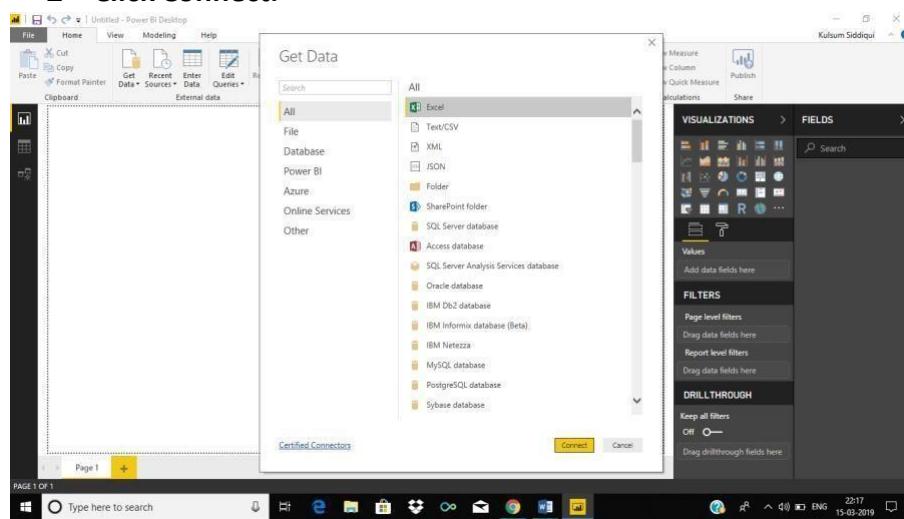
PRACTICAL 1

Aim:- Import Data In Power BI From Microsoft Excel And CSV.

- For Excel:

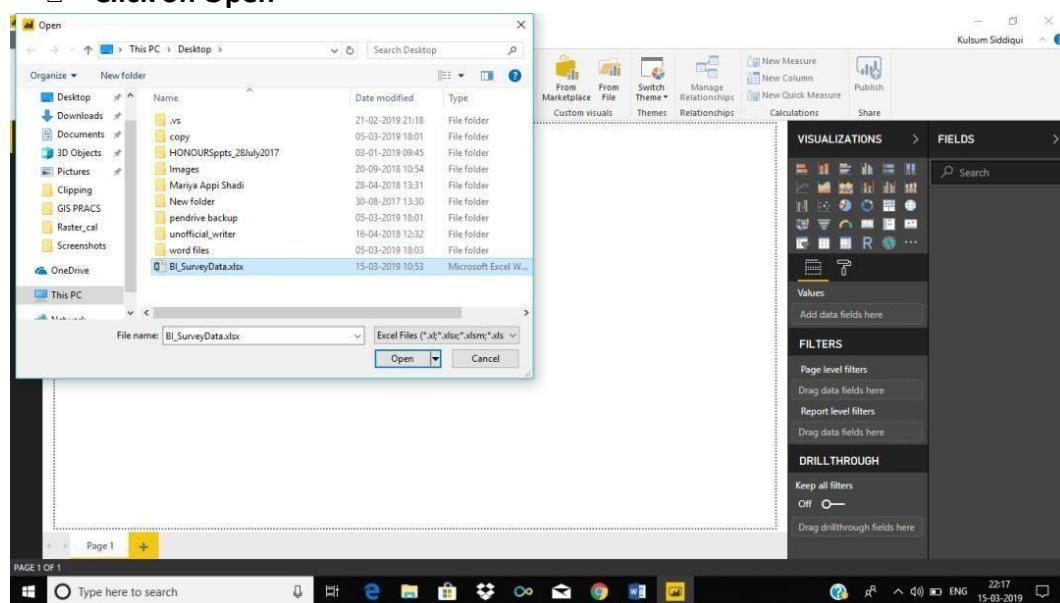
Step: 01

- Open Power BI desktop app, Click on Get data option
- Select Excel from the options that appear.
- Click Connect.



Step No: 02

- Select the Excel file from the storage.
- Click on Open



Step No: 03

- Select(check the check box) of the sheet
- Click on Load to load the excel data onto the Power BI workspace to operation operations onto it.

The screenshot shows the Power BI Desktop application window. On the left, the Navigator pane displays a list of files: 'BI.SurveyData.xlsx [1]' and 'Sheet1'. The main area shows a preview of 'Sheet1' data with columns: Serial No., Your Age Group, Gender, and State the best Quality/ Talent. The Fields pane on the right lists various data fields. At the bottom, there are 'Load', 'Edit', and 'Cancel' buttons.

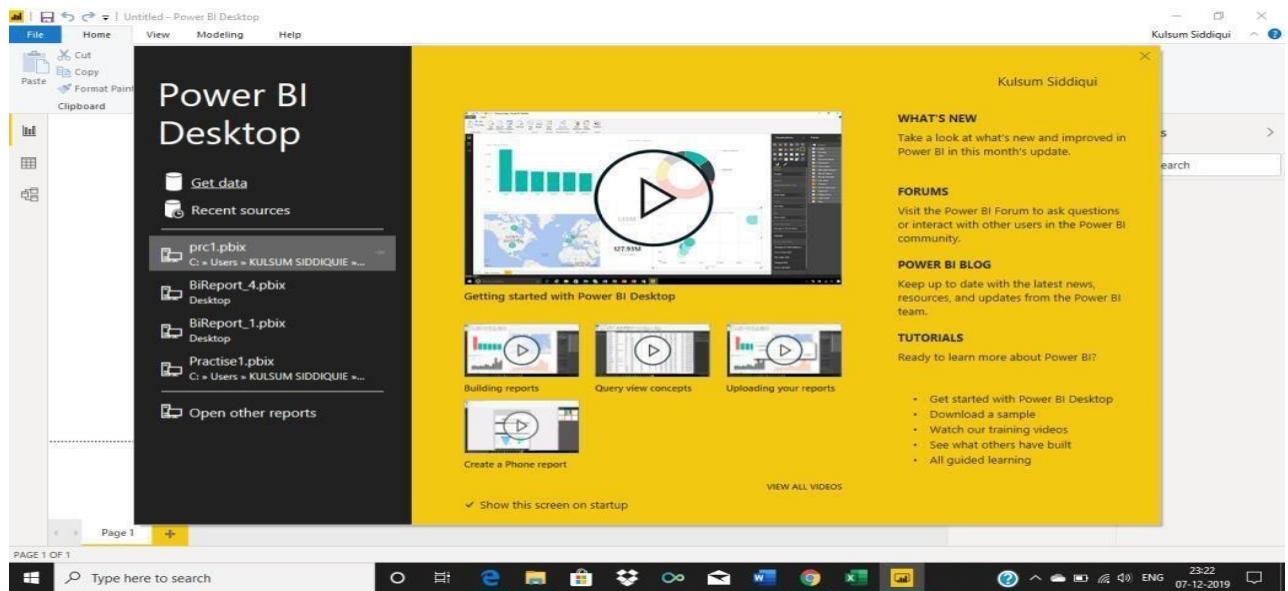
Step No:04

- The Data fields appears onto the right side of the screen, Next to it is the Visualizations panel.
- Here user can drag and drop fields, Also can make graphs of all sort.

The screenshot shows the Power Query Editor window. The left pane displays a list of transformation steps for a query named 'Date'. The right pane shows a preview of the data, which consists of several columns: MonthID, Month, Quarter, and Year. The top ribbon shows various options like File, Home, Transform, Add Column, View, and Help. The bottom status bar indicates '7 COLUMNS, 999+ ROWS' and 'PREVIEW DOWNLOADED ON WEDNESDAY'.

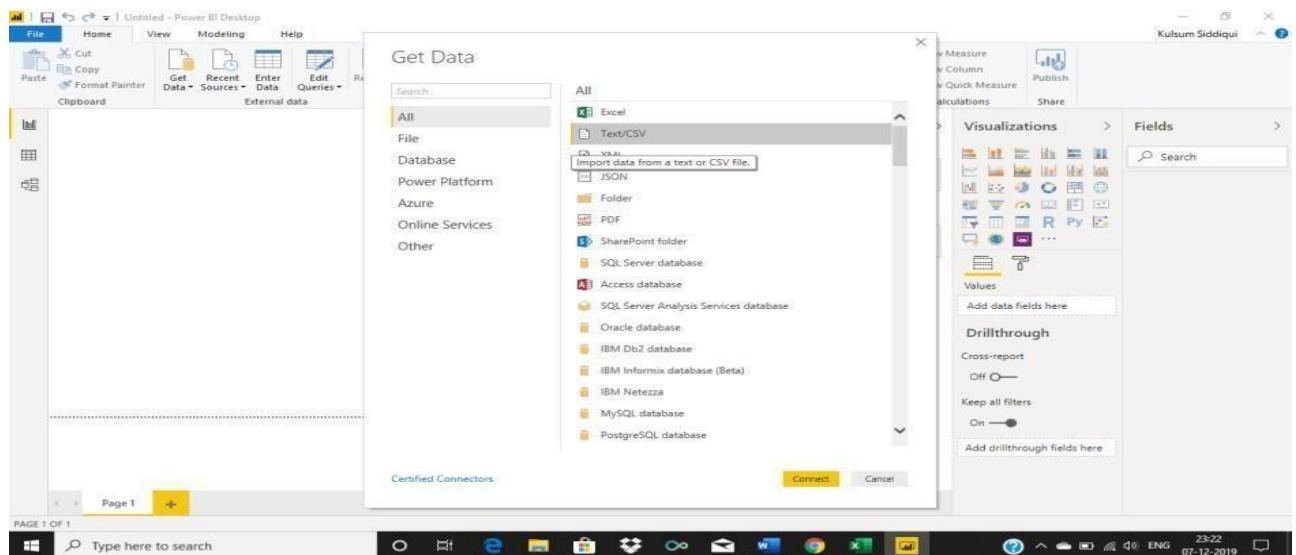
□ For CSV file:

Step 01: Open Power BI. Click On GET DATA.

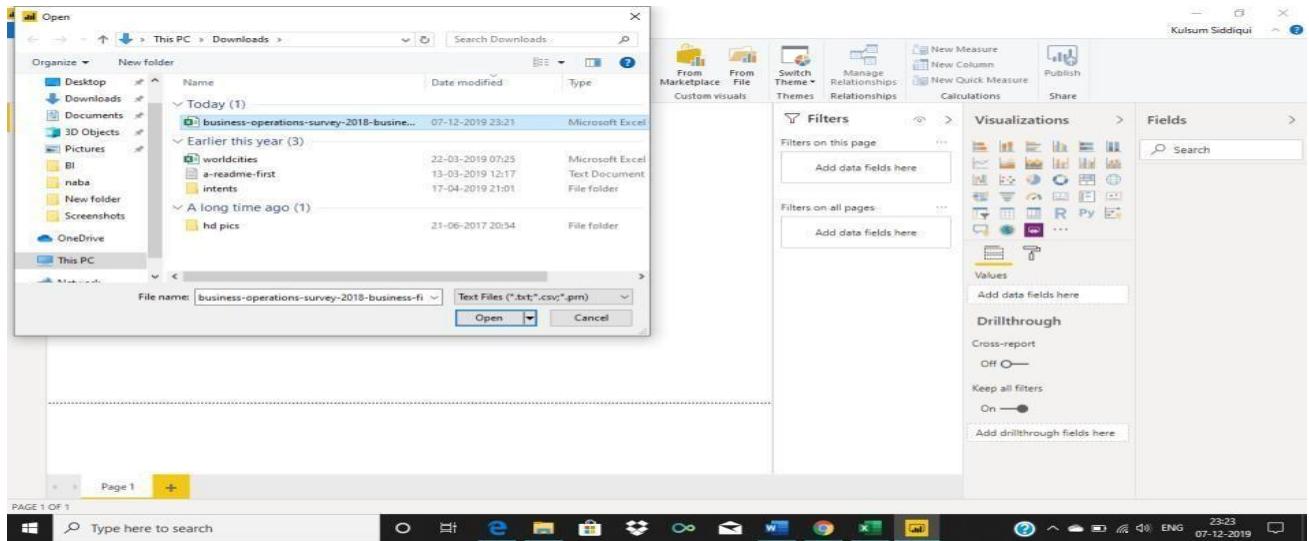


□ Step 02:

Select Text/CSV from the set of options.



Step:03:
Browse to the file location from the system and select the file required to be imported.



Step:04
Select the file origin, delimiter as comma, Data type detection as first 200 rows
Then click on load.

File Home Insert View Models Power BI Desktop

business-operations-survey-2018-business-finance-csv.csv

File Origin Delimiter Data Type Detection

1252: Western European (Windows) Comma Based on first 200 rows

description

Colon

At the end of the last financial year, which of the following...
Equal sign
Semicolon
Space
Tab
Colon & space
Custom
Fixed width

level size line_code value

0 6-19 employees D0201 14604
0 20-49 employees D0201 3792
0 50-99 employees D0201 1073
0 100+ employees D0201 777
1 total D0201 2360
2 total D0201 24
2 total D0201 23
2 total D0201 575
2 total D0201 30
2 total D0201 2694
2 total D0201 525
2 total D0201 362
2 total D0201 276
2 total D0201 226
2 total D0201 216
2 total D0201 84
2 total D0201 452
2 total D0201 331
2 total D0201 89

Fields Search

Py

Load Transform Data Cancel

File here

File Home View Modeler

Cut Get Recent Sources

Format Painter

Paste

Clipboard

business-operations-survey-2018-business-finance-csv.csv

File Origin Delimited

1232: Western European (Windows) Comma

Data Type Detection

Based on first 200 rows Based on first 200 rows Based on entire dataset Do not detect data types

description	Industry	item_code	value
At the end of the last financial year, which of the following?	total	D001001	14604
At the end of the last financial year, which of the following?	total	D001001	3790
At the end of the last financial year, which of the following?	total	D001001	2075
At the end of the last financial year, which of the following?	total	D001001	777
At the end of the last financial year, which of the following?	Agriculture, Forestry, & Fishing	1 total	2303
At the end of the last financial year, which of the following?	Agriculture	2 total	2001
At the end of the last financial year, which of the following?	Commercial fishing	2 total	24
At the end of the last financial year, which of the following?	Forestry & logging	2 total	125
At the end of the last financial year, which of the following?	Agriculture, forestry, & fishing support services	2 total	529
At the end of the last financial year, which of the following?	Mining	1 total	30
At the end of the last financial year, which of the following?	Manufacturing	1 total	2694
At the end of the last financial year, which of the following?	Food, beverage, & tobacco	2 total	525
At the end of the last financial year, which of the following?	Textile, clothing, footwear, & leather	2 total	162
At the end of the last financial year, which of the following?	Wood & paper products	2 total	276
At the end of the last financial year, which of the following?	Printing, publishing, & recorded media	2 total	120
At the end of the last financial year, which of the following?	Petroleum, coal, chemical, & associated product	2 total	226
At the end of the last financial year, which of the following?	Non-metallic mineral product	2 total	84
At the end of the last financial year, which of the following?	Metal product	2 total	492
At the end of the last financial year, which of the following?	Transport and industrial machinery & equipment	2 total	531
At the end of the last financial year, which of the following?	Other machinery & equipment	2 total	99

Load Transform Data Cancel

Fields Search

Adds here

Page 1

Type here to search

The screenshot shows the Microsoft Power BI Desktop application interface. At the top, there's a ribbon with tabs: File, Home, View, Modeling, and Help. The Home tab is selected. Below the ribbon, there's a toolbar with various icons for file operations like Paste, Cut, Copy, Format Painter, Get Data, Refresh, and Insert. To the right of the toolbar is a message bar stating "There are pending changes in your queries that haven't been applied." with a "Apply changes" button. The main workspace is titled "Load" and contains a single item: "business-operations-survey-2019-business-finance.csv (2)". A progress bar indicates "Creating connection in model...". On the right side of the screen, there's a "Visualizations" ribbon with a search bar at the top. It includes sections for Filters on this page, Add data fields here, Values, Drillthrough, Cross-report, Off, Keep all filters, and On. At the bottom left, there are navigation buttons for Page 1 and a yellow "Next" button. The bottom of the screen shows the Windows taskbar with the Start button, a search bar, and pinned icons for Edge, File Explorer, Mail, Photos, OneDrive, and others.

Step: 05

Click on table on the left side panel, the data appears in form of data, further analysis can be performed on this data.

The screenshot shows the Power BI Desktop interface. On the left, there is a table with columns: description, industry, level, size, line_code, and value. The table contains approximately 6,580 rows of data. On the right, there is a 'Fields' pane with a search bar and a list of fields: description, industry, level, size, and value. The 'description' field is currently selected. The top menu bar includes File, Home, Modeling, Help, and various ribbon tabs like Get Data, Insert, and Publish. The status bar at the bottom shows the date (07-12-2019) and time (23:25). A taskbar at the bottom of the screen shows icons for various Windows applications.

description	industry	level	size	line_code	value
At the end of the last financial year, which of the following types of business did this business have any of?	Insurance	2 total	D0201	0	
At the end of the last financial year, which of the following types of business did this business have any of?	Insurance	2 total	D0204	0	
At the end of the last financial year, did this business use any of the following types of business?	Insurance	2 total	D0301.01	0	
At the end of the last financial year, did this business use any of the following types of business?	Insurance	2 total	D0302.01	0	
Over the last financial year, how have this business's existing credit been used?	Insurance	2 total	D0401.01	0	
Over the last financial year, how have this business's existing credit been used?	Insurance	2 total	D0401.03	0	
Over the last financial year, how have this business's existing credit been used?	Insurance	2 total	D0402.01	0	
Over the last financial year, how have this business's existing credit been used?	Insurance	2 total	D0403.03	0	
Over the last financial year, how have this business's existing credit been used?	Insurance	2 total	D0404.01	0	
Over the last financial year, how have this business's existing credit been used?	Insurance	2 total	D0404.03	0	
Why has this business not requested equity finance in the last financial year?	Insurance	2 total	D0602	0	
Why has this business not requested equity finance in the last financial year?	Insurance	2 total	D0603	0	
Why has this business not requested equity finance in the last financial year?	Insurance	2 total	D0605	0	
Why has this business not requested equity finance in the last financial year?	Insurance	2 total	D0606	0	
Why has this business not requested equity finance in the last financial year?	Insurance	2 total	D0607	0	
How easy or hard was it to raise the equity finance you requested?	Insurance	2 total	D0700.01	0	
How easy or hard was it to raise the equity finance you requested?	Insurance	2 total	D0700.02	0	
How easy or hard was it to raise the equity finance you requested?	Insurance	2 total	D0700.04	0	
How easy or hard was it to raise the equity finance you requested?	Insurance	2 total	D0700.06	0	
For your most recent equity request, how did you intend to use the money?	Insurance	2 total	D0801	0	
For your most recent equity request, how did you intend to use the money?	Insurance	2 total	D0802	0	
For your most recent equity request, how did you intend to use the money?	Insurance	2 total	D0803	0	
For your most recent equity request, how did you intend to use the money?	Insurance	2 total	D0804	0	
For your most recent equity request, how did you intend to use the money?	Insurance	2 total	D0805	0	

Practical 2

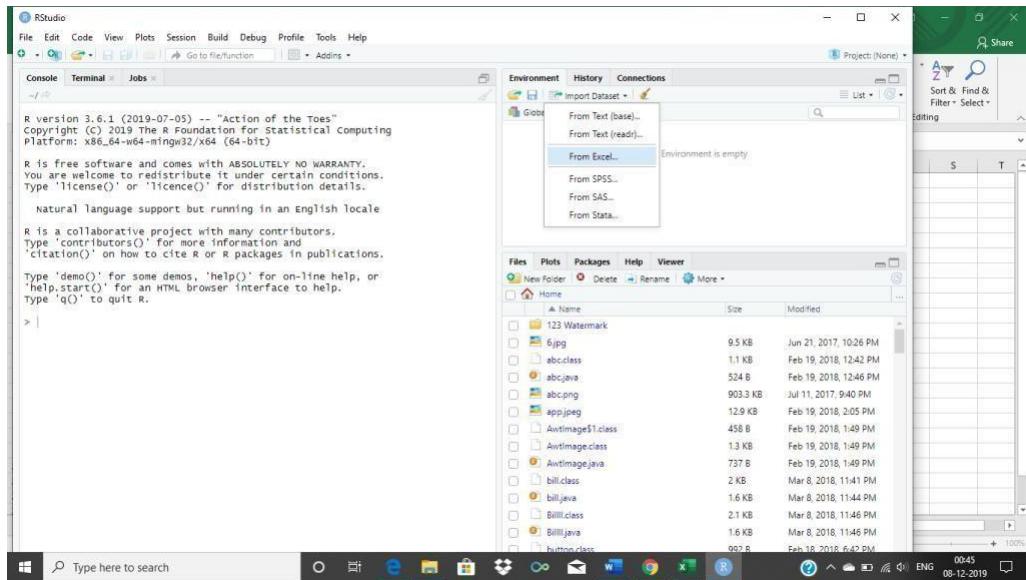
Aim: Import Data In Power BI From Microsoft Excel And CSV.

For Excel

Steps:

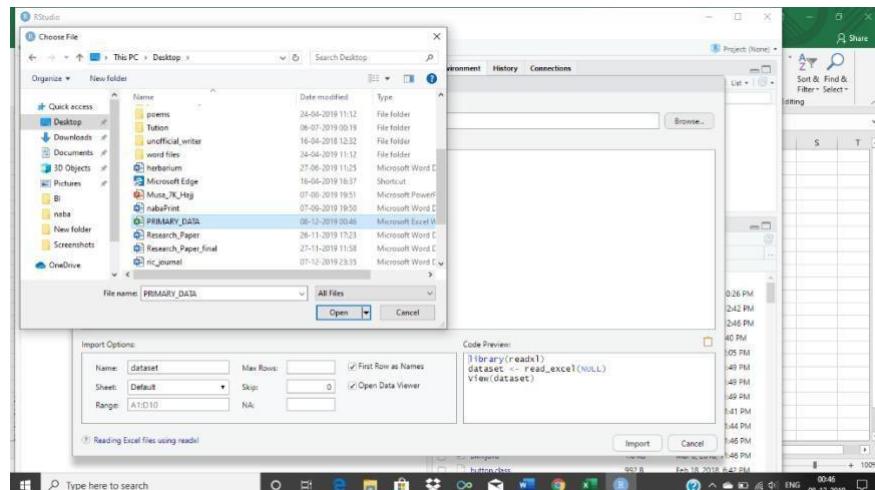
1. Open R Studio, Click on import dataset.

2. Select from excel.



3. Browse the file location and select the desired file.

4. Click on open.



5.

6. Select the data type and make necessary settings.

7. Click on import.

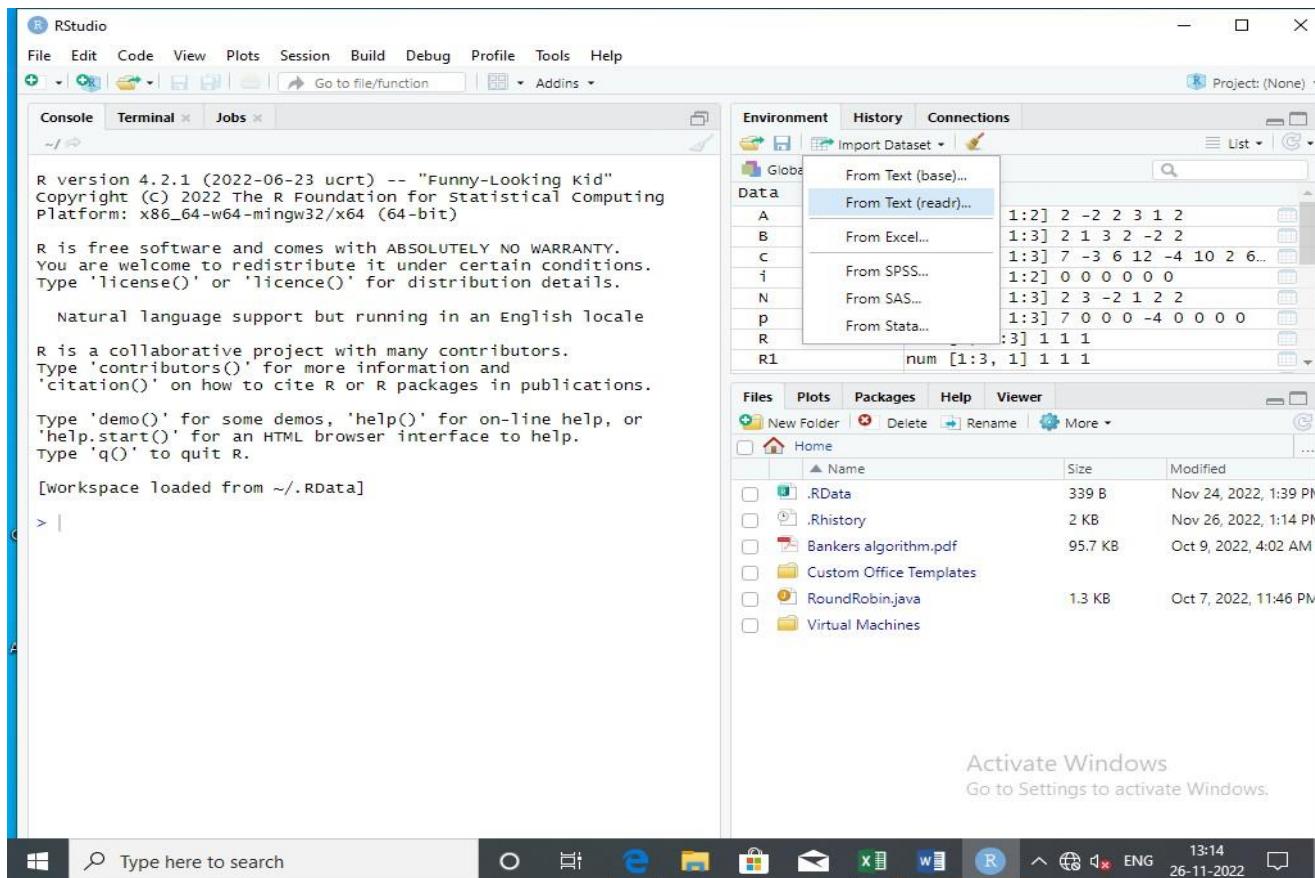
R version
Copyright: Platform:
R is free software; you are welcome to redistribute it under certain conditions.
Type 'tic' to start collecting function execution times.
Type 'con' to get information about the connection to R.
Type 'citation()' for more information and cite R or R packages in publications.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> library(readxl)
> PRIMARY_DATA <- read_excel("c:/users/kulsum siddiqi/Desktop/PRIMARY_DATA.xlsx")
> View(PRIMARY_DATA)

PRIMARY_DATA <- read_excel("c:/users/kulsum siddiqi/Desktop/PRIMARY_DATA.xlsx")

□ For CSV

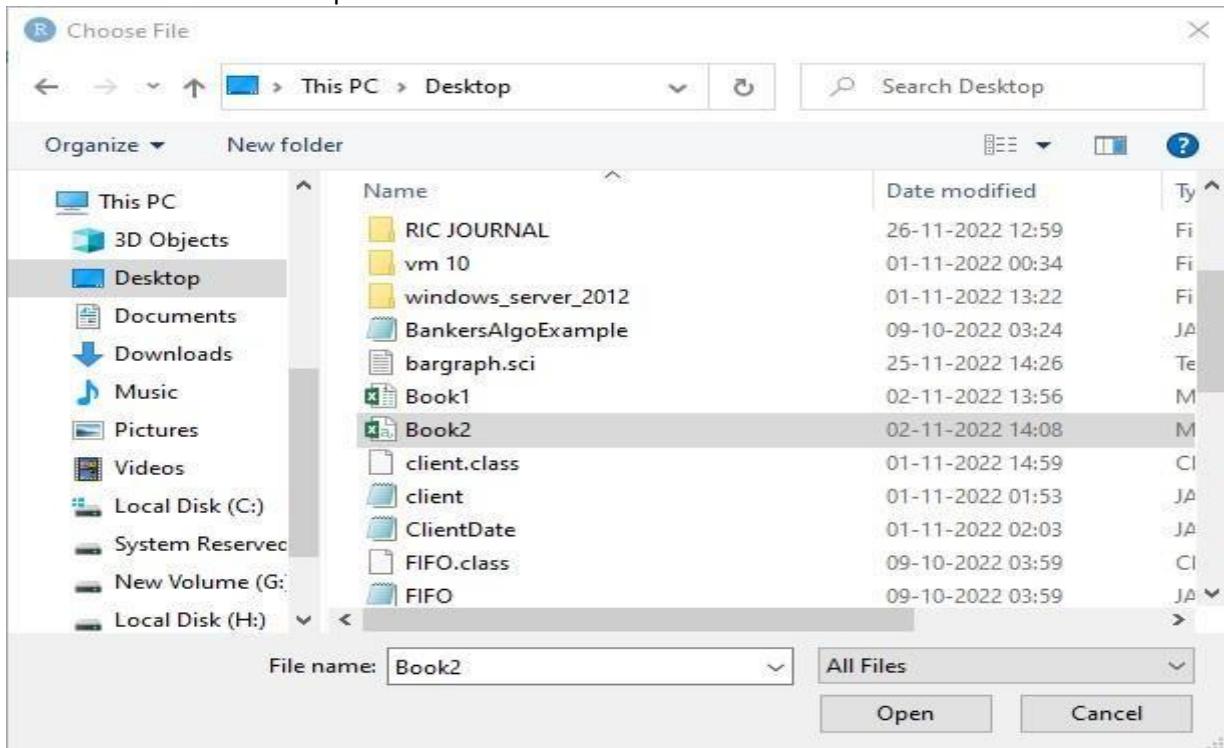
Steps:

1. Open R Studio, click on import data set.
2. Select from Text(readr)



3. Browse the file location and select the desired file(.csv).

4. Click on open.



5. Select the data type and make necessary settings.

6. Click on Import.

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Project: (None)

Import Text Data

File/URL: C:/Users/MSC_IT/Desktop/Book2.csv

Data Preview:

	name (character)	course (character)	roll no (double)
1	prathmesh	it	32
2	mane	it	33
3	suyash	it	45
4	shubham	it	67
5	yusuf	it	78

Showing 1 to 5

Console Te

Type 'lic'

Natural

R is a co Type 'com' 'citation'

Type 'dem' 'help.sta' Type 'q()

[workspace] loaded from history

Import Options:

Name: Book2 First Row as Names Delimiter: Comma Escape: None

Skip: 0 Trim Spaces Quotes: Default Comment: Default

Open Data Viewer Locale: Configure... NA: Default

Code Preview:

```
library(readr)
Book2 <- read_csv(
  "C:/Users/MSC_IT/Desktop/Book2.csv",
  check.names = TRUE)
```

Activate Windows Go to Settings to activate Windows.

12:55 26-11-2022

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Project: (None)

Import Dataset

Global Environment

Data

	A	B	C	i	N	P	R
1	num [1:3, 1:2] 2 -2 2 3 1 2	num [1:2, 1:3] 2 1 3 2 -2 2	num [1:3, 1:3] 7 -3 6 12 -4 10 2 6...	num [1:3, 1:2] 0 0 0 0 0 0	num [1:2, 1:3] 2 3 -2 1 2 2	num [1:3, 1:3] 7 0 0 0 -4 0 0 0 0	num [1, 1:3] 1 1 1

Files Plots Packages Help Viewer

New Folder Delete Rename More

Home

Name	Size	Modified
.RData	339 B	Nov 24, 2022, 1:39 PM
.Rhistory	1.9 KB	Nov 26, 2022, 12:54 PM
Bankers algorithm.pdf	95.7 KB	Oct 9, 2022, 4:02 AM
Custom Office Templates		
RoundRobin.java	1.3 KB	Oct 7, 2022, 11:46 PM
Virtual Machines		

Activate Windows Go to Settings to activate Windows.

12:56 26-11-2022

Practical 3

Aim: Collect the primary data in excel and apply the scientific method.

Collect data in Excel.

1. Concatenation operation performed.

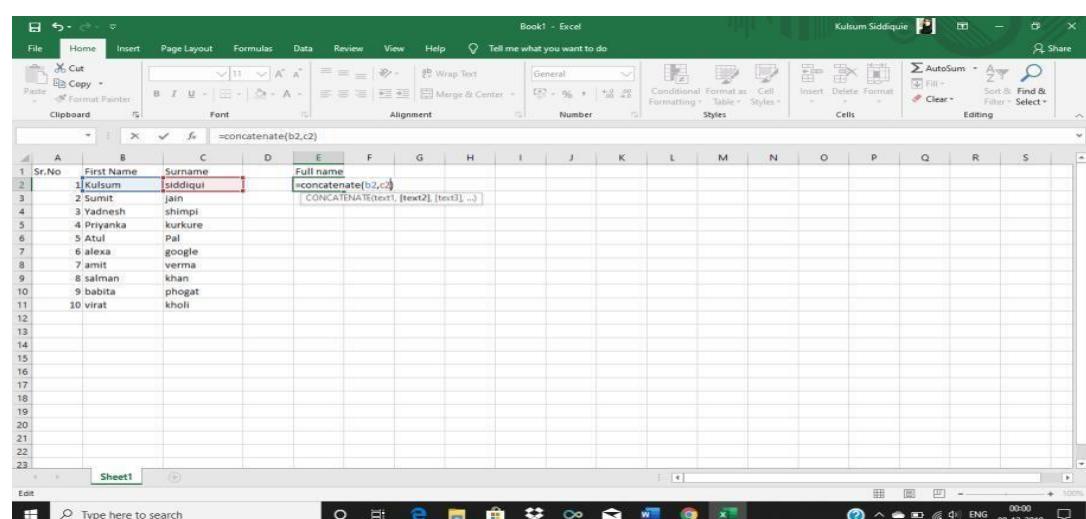
Steps:

Create a separate column for full name.

Select a cell, enter the formula: **Concatenate (string1, string2)**

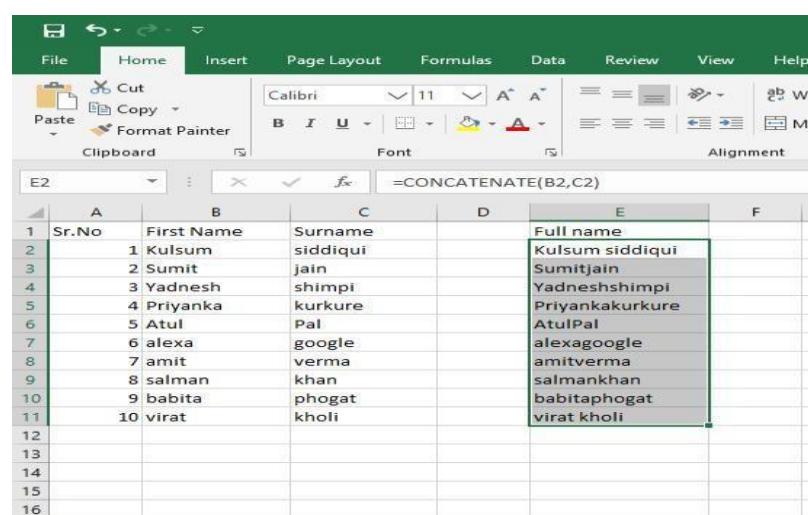
Concatenate (B2, C2)

Enter the cell location where the strings are placed, press enter.



The screenshot shows a Microsoft Excel spreadsheet titled "Book1 - Excel". The data is organized into columns A, B, and C. Column A contains "Sr.No", column B contains "First Name", and column C contains "Surname". A new column E is added with the header "Full name". In cell E2, the formula "=concatenate(b2,c2)" is typed into the formula bar. The formula is also visible in the cell itself. The formula bar also shows the expanded formula as "CONCATENATE(text1, [text2], [text3], ...)". The rest of the spreadsheet is empty, with rows 3 through 11 containing sample data.

drag the cell for all columns.



The screenshot shows the same Microsoft Excel spreadsheet after the formula has been copied down. The "Full name" column (E) now contains the concatenated names for all rows. The formula in the first cell, E2, is still visible in the formula bar as "=CONCATENATE(B2,C2)". The data in columns A, B, and C remains the same as in the previous screenshot.

Sr.No	First Name	Surname	Full name
1	Kulsum	siddiqui	Kulsum siddiqui
2	Sumit	jain	Sumitjain
3	Yadnesh	shimpi	Yadneshshimpi
4	Priyanka	kurkure	Priyankakurkure
5	Atul	Pal	AtulPal
6	alexa	google	alexagoogle
7	amit	verma	amitverma
8	salman	khan	salmankhan
9	babita	phogat	babitaphogat
10	virat	kholi	virat kholi

Operation:02 Statistical

Steps:

1. Store the data into the excel sheet.

roll no	name	subject1	subject2	subject3	subject4	percentage	grade
1	1 A	10	10	10	10	12	C
2	2 B	15	18	10	14	14	C
3	3 C	14	20	10	17	18	B
4	4 D	17	14	17	20	18	B
5	5 E	2	4	7	15	17	C
6	6 F	14	17	12	15	11	C
7	7 G	14	12	12	13	12	C
8	8 H	15	15	15	15	15	B
9	9 I	0	5	1	14	12	C
10	10 J	3	18	6	16	12	C
11	11 K	17	20	20	15	13	C
12	12 L	16	12	13	15	17	B
13	13 M	18	14	17	15	16	B
14	14 N	19	20	20	16	12	C
15	15 O	4	10	9	8	8	C
16	16 P	6	8	5	7	10	C
17	17 Q	7	7	7	7	7	C
18	18 R	10	12	10	12	10	C
19	19 S	18	16	14	15	20	B
20	20 T	13	12	14	16	12	C
21							

2. Select Statistical operations from the function selection panel.

roll no	name	subject1	subject2	subject3	subject4	percentage	grade
5	4 D	17	14	17	20	18	C
6	5 E	2	4	7	15	17	B
7	6 F	14	17	12	15	11	C
8	7 G	14	12	12	13	12	C
9	8 H	15	15	15	15	15	B
10	9 I	0	5	1	14	12	C
11	10 J	3	18	6	16	12	C
12	11 K	17	20	20	15	13	C
13	12 L	16	12	13	15	17	B
14	13 M	18	14	17	15	16	B
15	14 N	19	20	20	16	12	C
16	15 O	4	10	9	8	8	C
17	16 P	6	8	5	7	10	C
18	17 Q	7	7	7	7	7	C
19	18 R	10	12	10	12	10	C
20	19 S	18	16	14	15	20	B
21	20 T	13	12	14	16	12	C
22	min	0	4				
23	max	19	20				
24	average	11.40909	13.05091				
25							
26							
27							

3. Finding Minimum value from the columns

Select the cell, enter function: MIN(c2:c21)

Press enters.

4. Finding the maximum value from the columns

Select the cell, enter the function: MAX(c2:c21)

Press enters.

The screenshot shows an Excel spreadsheet titled "PRIMARY_DATA - Excel". The formula bar at the top displays =MAX(C2:C21). The data in columns C and D is as follows:

	C	D
3	2 B	15
4	3 C	18
5	4 D	17
6	5 E	14
7	6 F	12
8	7 G	12
9	8 H	13
10	9 I	15
11	10 J	16
12	11 K	20
13	12 L	13
14	13 M	17
15	14 N	20
16	15 O	10
17	16 P	8
18	17 Q	5
19	18 R	7
20	19 S	12
21	20 T	14
22	min	16
23	max	20
24	average	AVERAGE(C2:C21)
25		AVERAGE(number1, [number2], ...)

5. Finding the average value from the columns

Select the cell, enter the function: AVERAGE(c2:c21)

Press enters

The screenshot shows an Excel spreadsheet titled "PRIMARY_DATA - Excel". The formula bar at the top displays =AVERAGE(C2:C21). The data in columns C and D is as follows:

	C	D
3	2 B	15
4	3 C	18
5	4 D	17
6	5 E	14
7	6 F	12
8	7 G	12
9	8 H	13
10	9 I	15
11	10 J	16
12	11 K	20
13	12 L	13
14	13 M	17
15	14 N	20
16	15 O	10
17	16 P	8
18	17 Q	5
19	18 R	7
20	19 S	12
21	20 T	14
22	min	16
23	max	20
24	average	AVERAGE(C2:C21)
25		AVERAGE(number1, [number2], ...)

6. Finding the percentage of marks of 5 columns.

Select the cell, enter=((c2+d2+e2+f2+g2)*100)/100

Press enters.

PRIMARY_DATA - Excel

roll no	name	subject1	subject2	subject3	subject4	subject5	percentage	grade
1	A	10	10	10	15	12	=C2*D2+E2+F2+G2)*100/100	57
2	B	15	18	17	14	14		78
3	C	14	20	10	17	18		79
4	D	17	14	17	20	18		86
5	E	2	4	7	15	17		45
6	F	14	17	12	15	11		69
7	G	14	12	12	13	12		63
8	H	15	15	15	15	15		75
9	I	0	5	1	2	14		22
10	J	3	18	6	16	12		55
11	K	17	20	20	15	13		85
12	L	16	12	13	15	17		73
13	M	18	14	17	15	16		80
14	N	19	20	20	16	12		87
15	O	4	10	9	8	8		39
16	P	6	8	5	7	10		36
17	Q	7	7	7	7	7		35
18	R	10	12	10	12	10		54
19	S	18	16	14	15	20		83
20	T	13	12	14	16	12		67
22	min	0	4	1	2	7		
23	max	19	20	20	20	20		

Drag the cell to find the percentages for the whole column.

PRIMARY_DATA - Excel

roll no	name	subject1	subject2	subject3	subject4	subject5	percentage	grade
1	A	10	10	10	15	12	=((C2+D2+E2+F2+G2)*100/100)	57
2	B	15	18	17	14	14		78
3	C	14	20	10	17	18		79
4	D	17	14	17	20	18		86
5	E	2	4	7	15	17		45
6	F	14	17	12	15	11		69
7	G	14	12	12	13	12		63
8	H	15	15	15	15	15		75
9	I	0	5	1	2	14		22
10	J	3	18	6	16	12		55
11	K	17	20	20	15	13		85
12	L	16	12	13	15	17		73
13	M	18	14	17	15	16		80
14	N	19	20	20	16	12		87
15	O	4	10	9	8	8		39
16	P	6	8	5	7	10		36
17	Q	7	7	7	7	7		35
18	R	10	12	10	12	10		54
19	S	18	16	14	15	20		83
20	T	13	12	14	16	12		67
22	min	0	4	1	2	7		
23	max	19	20	20	20	20		

7. Finding median from the column data.

Select the cell, enter the formula:

=median(c2:c21)Press enters.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
9.	8 H		15	15	15	15	15	78	pass											
10.	9 I		0	5	1	2	14	22	fail											
11.	10 J		3	18	6	16	12	55	pass											
12.	11 K		17	20	20	15	13	85	pass											
13.	12 L		16	12	13	15	17	73	pass											
14.	13 M		18	14	17	15	16	80	pass											
15.	14 N		19	20	20	16	12	87	pass											
16.	15 O		4	10	9	8	8	39	pass											
17.	16 P		6	8	5	7	10	36	pass											
18.	17 Q		7	7	7	7	7	35	fail											
19.	18 R		10	12	10	12	10	54	pass											
20.	19 S		18	16	14	15	20	83	pass											
21.	20 T		13	12	14	16	12	67	pass											
22.	min		0	4	1	2	7	22												
23.	max		19	20	20	20	20	87												
24.	average		11.40909	13.09091	11.68182	13.18182	13.40909	62.59090909												
25.	median		=median(C2:C21)																	
26.																				
27.																				
28.																				
29.																				
30.																				
31.																				

Drag the same for all the column.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
7.	6 F		14	17	12	15	11	69	pass											
8.	7 G		14	12	12	13	12	63	pass											
9.	8 H		15	15	15	15	15	75	pass											
10.	9 I		0	5	1	2	14	22	fail											
11.	10 J		3	18	6	16	12	55	pass											
12.	11 K		17	20	20	15	13	85	pass											
13.	12 L		16	12	13	15	17	73	pass											
14.	13 M		18	14	17	15	16	80	pass											
15.	14 N		19	20	20	16	12	67	pass											
16.	15 O		4	10	9	8	8	39	pass											
17.	16 P		6	8	5	7	10	36	pass											
18.	17 Q		7	7	7	7	7	35	fail											
19.	18 R		10	12	10	12	10	54	pass											
20.	19 S		18	16	14	15	20	83	pass											
21.	20 T		13	12	14	16	12	67	pass											
22.	min		0	4	1	2	7	22												
23.	max		19	20	20	20	20	87												
24.	average		11.40909	13.09091	11.68182	13.18182	13.40909	62.59090909												
25.	median		14	13	12	15	12.5	68												
26.																				
27.																				
28.																				
29.																				
30.																				
31.																				

Operation:03 Logical

1. Select Logical operations from the set of given lists.
2. Select “IF” from the list of logical operations.
3. Select the cells, write the formula for if

condition:IF(H2>35," Pass"," Fail")

4. Press enter.

5. Drag the same for column.

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do PRIMARY_DATA - Excel Kulum Siddique

Function Arguments

Logical test: $H2 > 35$ **Value_if_true**: "12=pass" **Value_if_false**: "12=fail"

Checks whether a condition is met, and returns one value if TRUE, and another value if FALSE.

Value_if_false is the value that is returned if Logical_test is FALSE. If omitted, FALSE is returned.

Formula result: "12=pass"

Help on this function

OK Cancel

Sheet1

Type here to search

01:00 06-12-2019

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do PRIMARY_DATA - Excel Kulum Siddique

IF: $=IF(H2>35,"12=pass","12=fail")$

Sheet1

Type here to search

08:44 06-12-2019

Operation: Maths and Trig.

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do PRIMARY_DATA - Excel Kulum Siddique

Insert Function

Search for a function: Type a brief description of what you want to do and then click Go Or select a category: Math & Trig

Selected function: AGGREGATE

Description: Returns an aggregate in a list or database.

Help on this function

OK Cancel

Sheet1

Type here to search

01:02 06-12-2019

Book1 - Excel (Product Activation Failed)																				
FILE	HOME	INSERT	PAGE LAYOUT	FORMULAS	DATA	REVIEW	VIEW													
fx Insert Function	AutoSum Recently Financial Logical Text Date & Time Lookup & Reference Math & Trig More Functions										Name Manager Define Name Use in Formula Trace Precedents Show Formulas Create from Selection Trace Dependents Error Checking Remove Arrows Evaluate Formula									
	Used Time Reference Functions										Watch Window Calculation Options Calculation									
J2	:	X	✓	fx	=AGGREGATE(1,4,C2:G2)															
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Rollno	Name	Subject1	Subject2	Subject3	Subject4	Subject5	Percentage	Grade	Aggregate										
2		1 A	10	10	10	15	12	57	Pass	11.4										
3		2 B		15	18	17	14	14	Pass	15.6										
4		3 C		14	20	12	17	18	Pass	16.2										
5		4 D		17	14	17	20	18	Pass	17.2										
6		5 E		2	4	7	15	17	Pass	9										
7		6 F		14	17	12	15	11	Pass	13.8										
8		7 G		14	12	12	13	12	Pass	12.6										
9		8 H		15	15	15	15	15	Pass	15										
10		9 I		0	5	1	2	14	Fail	4.4										
11		10 J		3	18	6	16	12	Pass	11										
12		11 K		17	20	20	15	13	Pass	17										
13		12 L		16	12	13	15	17	Pass	14.6										
14		13 M		18	14	17	15	16	Pass	16										
15		14 N		19	20	20	16	12	Pass	17.4										
16		15 O		4	10	9	8	8	Pass	7.8										
17		16 P		6	8	5	7	10	Pass	7.2										
18		17 Q		7	7	7	7	7	Fail	7										
19		18 R		10	12	10	12	10	Pass	10.8										
20		19 S		18	16	14	15	20	Pass	16.6										
21		20 T		13	12	14	16	12	Pass	13.4										
22																				
23	Minimum			0	4	1	2	7												

Sheet1

READY

No new notifications

Practical 4

Aim: Collect the secondary data in excel and apply scientific method

1. Store the secondary data in Excel.

The data stored below in the excel is sheet the data downloaded from the source:

<https://www.fbbva.es/microsite/multivariate-statistics/data.html>

For performing scientific operations.

The screenshot shows a Microsoft Excel spreadsheet titled "countries [Compatibility Mode] - Excel". The data is organized into two main sections: a header row and a data table. The header row contains columns labeled I through Z, with specific column headers like "I", "E", "HR", "BR", "RU", "D", "TR", "MA", "PE", "NG", "F", "MX", and "ZA" placed above the corresponding columns. The data table begins at row 2, listing 14 countries: Italy, Spain, Croatia, Brazil, Russia, Germany, Turkey, Morocco, Peru, Nigeria, France, Mexico, and South Africa. Each country has a row of 14 numerical values corresponding to the columns in the header. The Excel interface includes a ribbon bar with various tabs like Home, Insert, Page Layout, Formulas, Data, Review, View, and Help. The status bar at the bottom right shows the date and time as 08-12-2019 and 12:59.

	I	E	HR	BR	RU	D	TR	MA	PE	NG	F	MX	ZA
1	0	2	5	5	8	7	3	5	6	8	4	5	7
2	Italy	2	0	5	4	9	7	4	7	3	8	4	4
3	Spain	5	5	0	7	4	3	6	7	7	8	4	6
4	Croatia	5	4	7	0	9	8	3	6	2	7	4	3
5	Brazil	8	9	4	9	0	4	7	7	8	7	7	5
6	Russia	7	7	3	8	4	0	7	8	8	7	7	7
7	Germany	3	4	6	3	7	7	0	5	4	5	6	4
8	Turkey	5	7	7	6	7	8	5	0	7	4	6	4
9	Morocco	6	3	7	2	8	8	4	7	0	6	7	2
10	Peru	8	8	8	7	8	8	5	4	6	0	6	3
11	Nigeria	4	4	4	4	7	4	6	6	7	6	0	8
12	France	5	4	6	3	7	8	4	6	2	3	8	0
13	Mexico	7	6	7	5	7	8	5	4	4	3	7	4
14	SouthAfrica	7	6	7	5	7	8	5	4	4	3	7	0

2. Select the cell.

The screenshot shows a Microsoft Excel spreadsheet titled "countries [Compatibility Mode] - Excel". The data is organized into columns labeled A through Z. Row 1 contains category names: I, E, HR, BR, RU, D, TR, MA, P, F, NG, M, X, Z, A. Rows 2 through 14 contain data for 13 countries: Italy, Spain, Croatia, Brazil, Russia, Germany, Turkey, Morocco, Peru, Nigeria, France, Mexico, and South Africa. Each country has a row of 14 numerical values corresponding to the categories in row 1. The cells are highlighted in green, indicating they are selected.

3. From the list of functions, select Statistical, from the statistical functional select:Quartile.

The screenshot shows the Microsoft Excel Function Library dialog box. The "Statistical" category is selected under the "Function Library" tab. The "QUARTILE.EXC" function is highlighted in the list. The background shows the same data table as the previous screenshot, with the "countries" sheet selected. The status bar at the bottom right indicates the date as 08-12-2019 and the time as 13:04.

4. Enter the number of cells as input array, select quartile from 0

to 4, Select 1st quartile that gives 25th quartile results:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		I	E	HR	BR	RU	D	TR	MA	PE	NG	F	MX	ZA											
2	Italy	0	2	5	5	8	7	3	5	6	9	0	4	5	7	=QUARTILE.INC(b2:n2)									
3	Spain	2	0	5	4	9	7	4	7	3	8	4	4	6											
4	Croatia	5	5	0	7	4	3	6	7	7	8	4	4	6	7										
5	Brazil	5	4	7	0	9	8	3	6	2	7	4	3	5											
6	Russia	8	9	4	9	0	4	7	7	8	8	7	7	7											
7	Germany	7	7	3	8	4	0	7	8	8	8	4	8	8											
8	Turkey	3	4																						
9	Morocco	6	3																						
10	Peru	6	3																						
11	Nigeria	8	8																						
12	France	4	4																						
13	Mexico	5	4																						
14	SouthAfrica	7	6																						

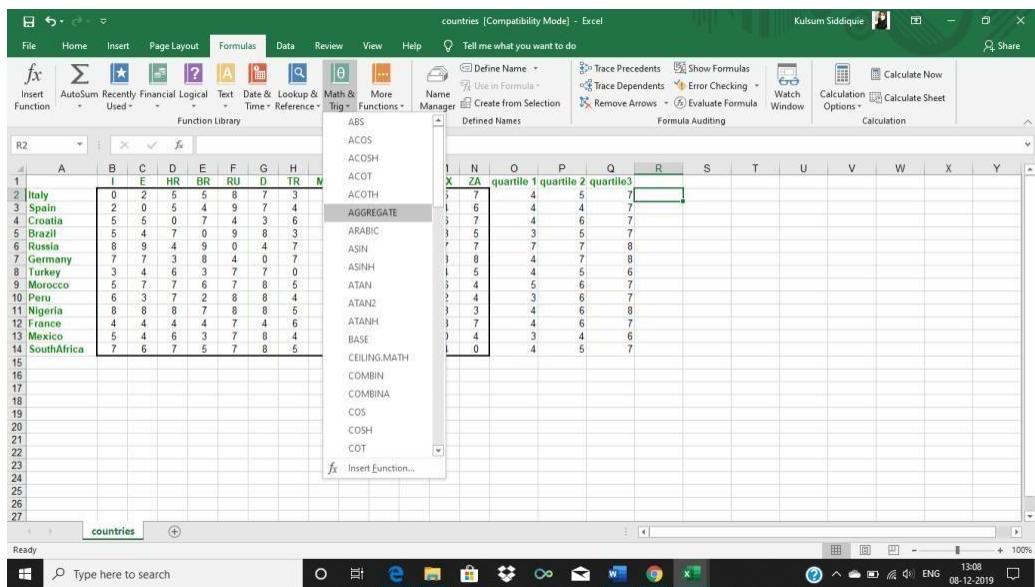
Select the 2nd quartile 3rd quartile, that gives result for 50th and 75th sample.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		I	E	HR	BR	RU	D	TR	MA	PE	NG	F	MX	ZA	quartile1	quartile3									
2	Italy	0	2	5	5	8	7	3	5	6	9	0	4	5	7	4	5	quartile1	quartile3						
3	Spain	2	0	5	4	9	7	4	7	3	8	4	4	6	7	4	4	quartile1	quartile3						
4	Croatia	5	5	0	7	4	3	6	7	7	8	4	6	7	4	6	6	quartile1	quartile3						
5	Brazil	5	4	7	0	9	8	3	6	2	7	4	3	5	3	5	5	quartile1	quartile3						
6	Russia	8	9	4	9	0	4	7	7	8	8	7	7	7	7	7	7	quartile1	quartile3						
7	Germany	7	7	3	8	4	0	7	8	8	8	4	8	8	4	7	7	quartile1	quartile3						
8	Turkey	3	4	6	3	7	7	0	5	4	5	6	4	5	4	5	5	quartile1	quartile3						
9	Morocco	5	7	7	6	7	8	5	0	7	4	6	6	4	5	6	6	quartile1	quartile3						
10	Peru	6	3	7	2	8	8	4	7	0	6	7	2	4	3	6	6	quartile1	quartile3						
11	Nigeria	8	8	7	8	8	5	4	6	0	6	3	3	3	4	6	6	quartile1	quartile3						
12	France	4	4	4	4	7	4	6	6	7	6	0	8	7	4	6	6	quartile1	quartile3						
13	Mexico	5	4	6	3	7	8	4	6	2	3	8	0	4	3	4	4	quartile1	quartile3						
14	SouthAfrica	7	6	7	5	7	8	5	4	4	3	7	4	0	4	5	5	quartile1	quartile3						

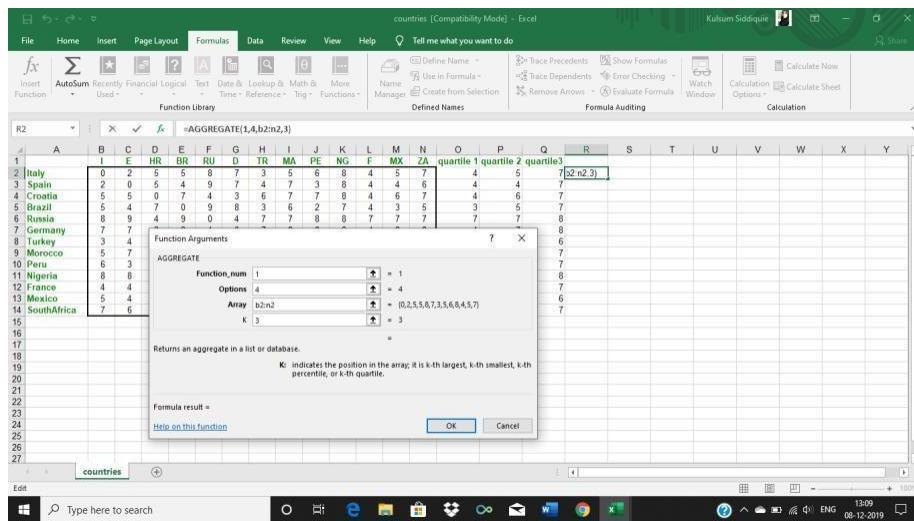
Operation: Aggregate function.

Steps:

1. select the cell
2. Select math and trig. From the list of functions.
3. Select the aggregate function from the list.
4. Click ok. Then enter



- 5 .fill the necessary details in the dialog box of the function that appears.



Operation: Multinomial function.

Steps:

1. select the cell
2. Select math and trig. From the list of functions.
3. Multinomial function from the list.
4. Fill the list of numbers in the array
5. Click ok. Then enter
6. Drag the cell for all columns.

Operation: Standard Variation function.

Steps:

1. select the cell
2. Select statistical. From the list of functions.
3. Select the standard deviation function from the list.
4. Click ok. Then enter
5. Select the cells for calculation, press enter.
6. Drag for whole column.

The screenshot shows an Excel spreadsheet titled "countries [Compatibility Mode] - Excel". The data consists of 15 rows of countries and their corresponding values across columns B through P. Row 1 contains column headers like I, E, HR, BR, RU, D, TR, MA, PE, NG, F, MX, and ZA. Rows 2 through 15 list countries such as Italy, Spain, Croatia, Brazil, Russia, Germany, Morocco, Peru, Nigeria, France, Mexico, and South Africa, each with a set of numerical values. The formula bar at the top has the formula =STDEV.S(b2:n2) entered. A function dialog box is open over the spreadsheet, specifically the "Function Arguments" dialog for the STDEV.S function. It shows "Number1" as b2:n2 and "Number2" as empty. Below the dialog, the formula result is displayed as 2.34520788. The status bar at the bottom right indicates the date and time as 08-12-2019 and 13:40.

Operation: Variance function.

Steps:

1. select the cell
2. Select statistical. From the list of functions.
3. Select the variance function from the list.
4. Click ok. Then enter
5. Select the cells for calculation, press enter.

Drag for whole column

Screenshot of Microsoft Excel showing the Formula Arguments dialog box for the VAR.P function.

The dialog box shows the formula `=VAR.P(b2:n2)` and the following arguments:

- Number1:** `b2:n2` (Range selected: B2:N2)
- Number2:** (Range selected: B2:N2)

The calculated result is `= 5.076923077`.

The formula result is `= 5.076923077`.

Help on this function

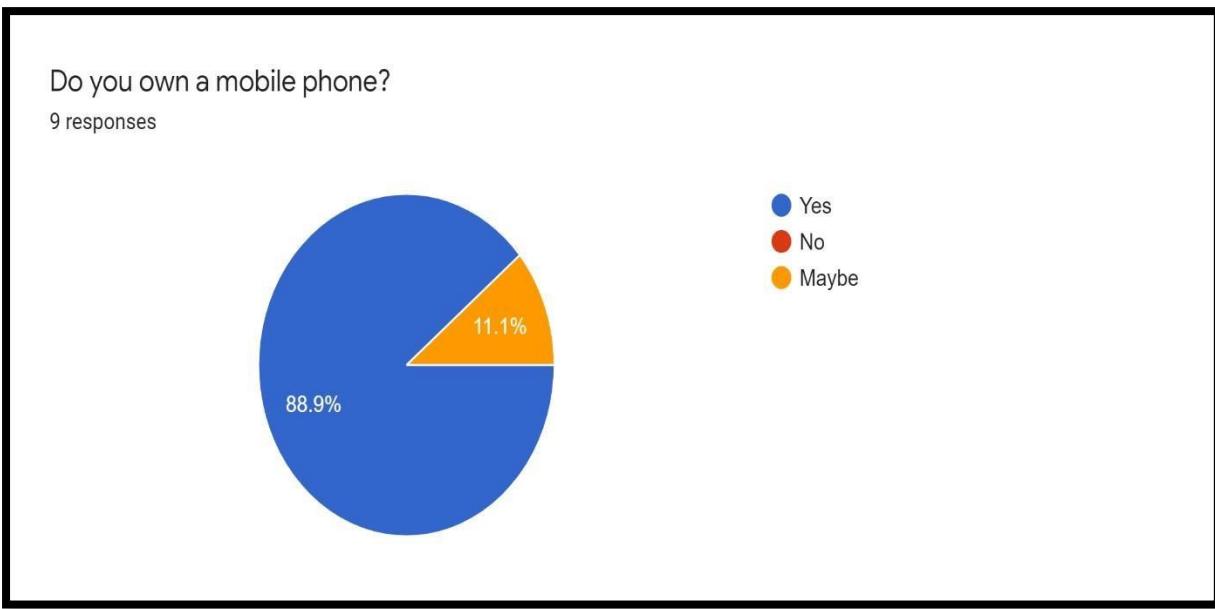
OK Cancel

The main Excel window shows a table of countries and their population data. The columns include: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y. The rows include: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27. The data starts from row 2, column B, with values such as Italy (0, 2, 5, 6, 8, 7, 3, 5, 6, 8, 4, 5, 6, 7, 4, 5), Spain (2, 0, 5, 4, 9, 7, 4, 7, 3, 8, 4, 4, 6, 7, 4, 4), and so on. The last row shown is South Africa (7, 6).

Practical 5

Aim: Design the survey from and collect the suitable data and perform analysis

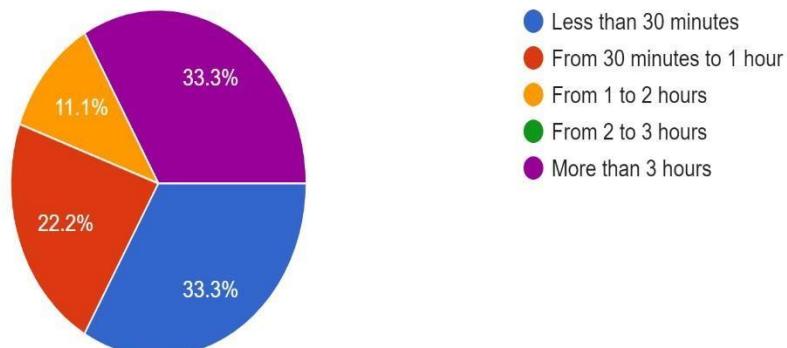
Mobile Phone Usage



According to the survey, in todays world everyone carries with their own mobile phones. Usage of mobiles goes at the huge extend. Everyone is dealing the new technology application and also aware of latest trends. The usage of mobile is consistently increases at huge extend. Everyone carries with the self owned mobile phones. But the truth is todays world is unaware of proper usage of mobile phones. And according to the survey 88.9% population having their own mobile phones and 11.1 using their colleagues mobile phones.

How much time do you spend on your mobile phone on average in a day (calls only)?

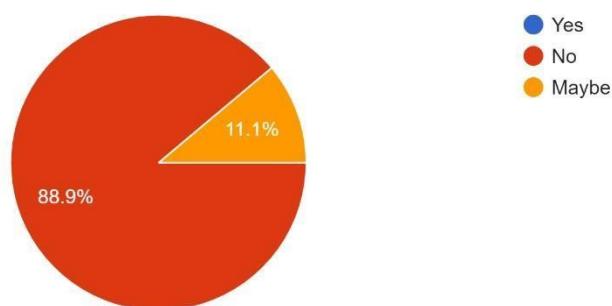
9 responses



Large Number of people are busy on call more than 3hours. 33.3% population are busy on mobile phones at huge extend. The survey tells that 22.2% people are busy on calls from 1-2 hours respectively. But very few 22.2% population are using 30min to 1 hour only on daily usage. Then 11.1% population are busy on calls for 1-2 hours maximum time respectively.

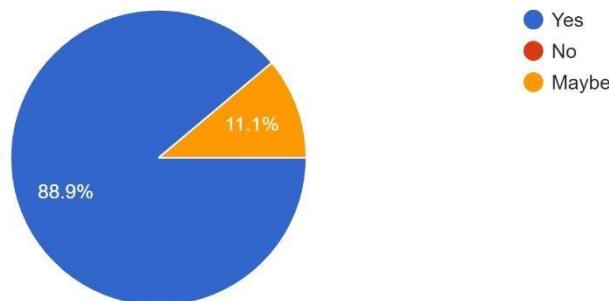
Do you own a second mobile phone?

9 responses



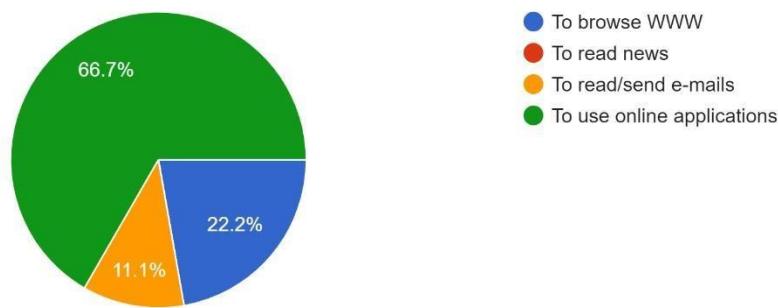
In todays world everyone having the 2 mobile phones. From the survey we get know that 88.9% population having 2 cell phones for their daily purpose need. But only 11.1% are not dealing with the usage of 2 mobile phones.

Would you classify Internet as an important part of a mobile phone?
9 responses



In todays world everyone having the 2 mobile phones. From the survey we get know that 88.9% population having 2 cell phones for their daily purpose need. But only 11.1% are not dealing with the usage of 2 mobile phones.

What is your primary purpose for using internet on your mobile phone?
9 responses

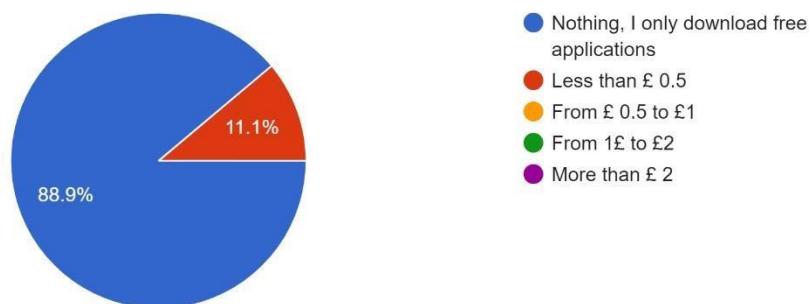


According to the survey, 66.7% using the online application for using internet on mobile phones. But from that 11.1% population use it only for reading and sending the emails.

And 22.2% population are busy in dealing with browser to browse www on the internet through the mobile phone usage.

How much would you be ready to spend for an online application that you want to obtain?

9 responses



According to survey of this questionnaire, we came to an conclusion that 88.9% using the internet for downloading the free application only. But 11.1% are ready for spending less than 0.5 total cost for the download of the paid applications.

We came to conclusion that more 10% population are reading some cost to the application which are paid to use, and apart from that 90% population are using the free download applications only according to their need and use also.

Practical 6

Aim: Implement the different scales of measurement on data: Nominal & Ordinal

Nominal data, also called categorical data, does not have a natural sequence. Instead, the data is typically in named categories or labels without numeric significance.

For example, is a survey question below that asks for a favourite beverage, with choices of coffee, tea, water, or milk will generate nominal data in 4 categories as in the bar chart.

Steps:

1. Store the data into excel format consisting of customer serial no. and preferred beverage.

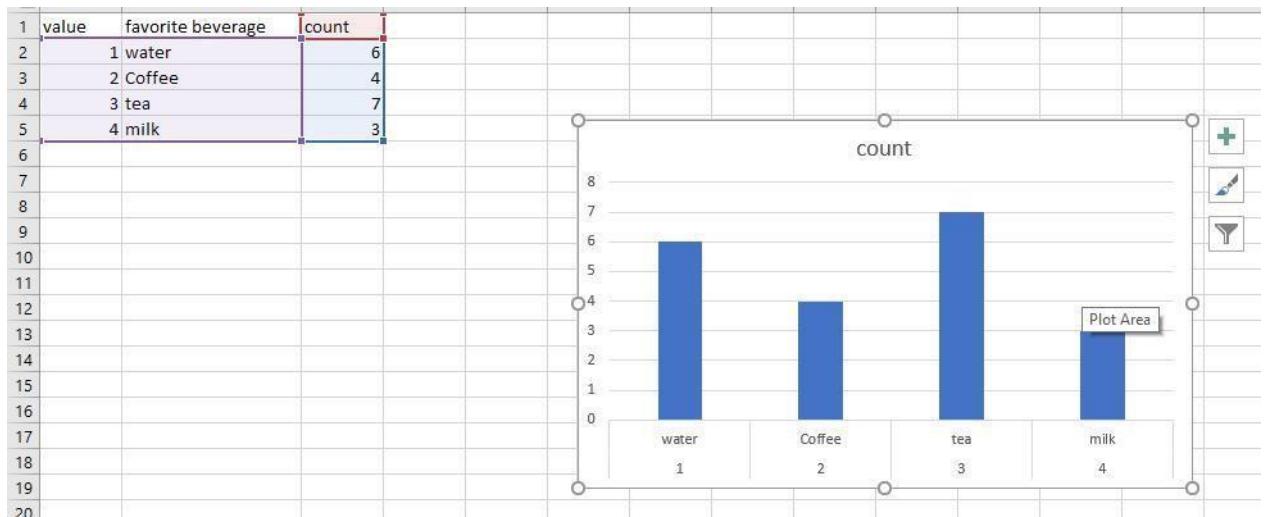
2. Sort the data by giving label to the selection by using if conditions.

The screenshot shows a Microsoft Excel spreadsheet titled "Book1 - Excel". The table consists of three columns: "value", "favorite beverage", and "count". The data is as follows:

value	favorite beverage	count
1	water	6
2	Coffee	4
3	tea	7
4	milk	3

The "count" column is highlighted with a blue selection bar. The status bar at the bottom right indicates the date and time as "08-12-2019 17:25".

3. Represent the data through the help of bar graph presented:



Ordinal Data:

Ordinal data has a distinct order or natural sequence. An example is survey data collected in response to the question like this:

How old are you?

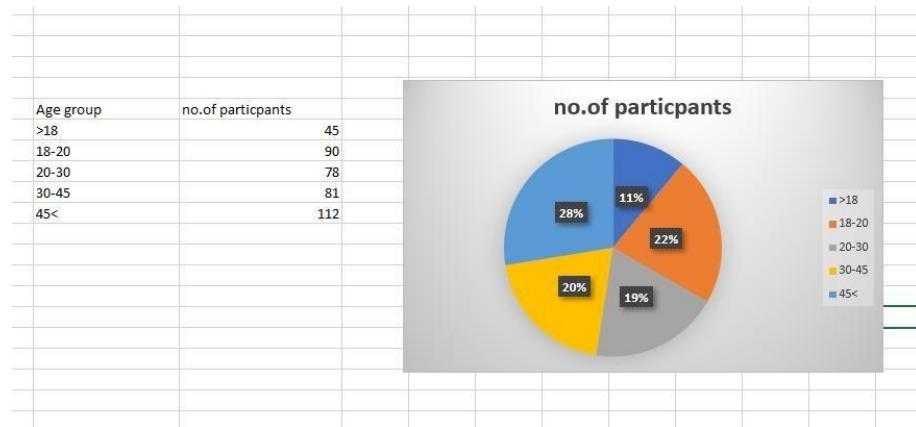
- Under 18**
- 18-20**
- 21-30**
- 31-45**
- 45+**

The chart above shows how the data can be plotted in an Excel column chart that follows the natural order.

4. For Ordinal data, enter the data:

Age group	no.of participants
>18	45
18-20	90
20-30	78
30-45	81
45<	112

5. Plot the information into percentage pie diagram and bar graph in increasing to decreasing order of the ages.



Practical 7

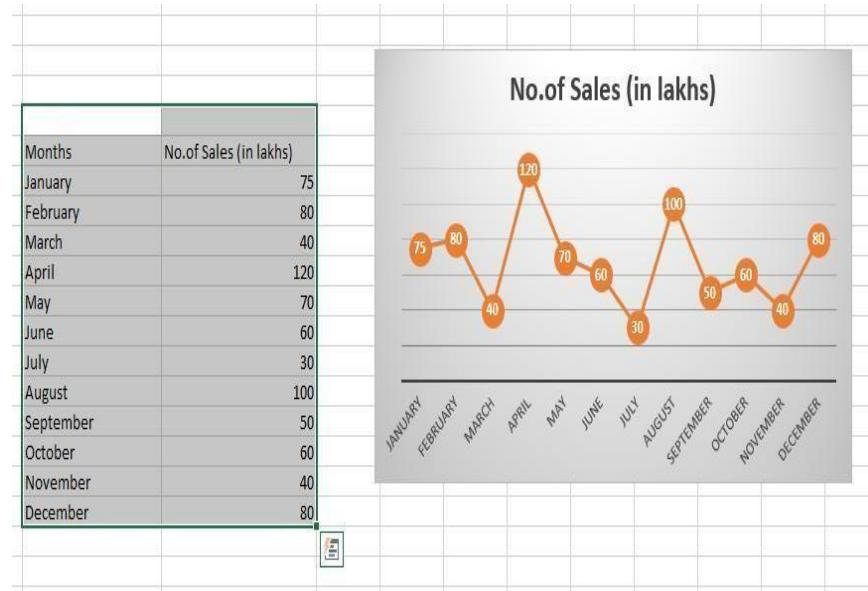
Aim: Implement the different scales of measurement on data: Ratio and Interval

Interval data is like ordinal except we can say the intervals between each value are equally split. The most common example is temperature in degrees Fahrenheit. The difference between 29 and 30 degrees is the same magnitude as the difference between 78 and 79 (although I know I prefer the latter). With attitudinal scales and the Likert questions you usually see on a survey, these are rarely interval, although many points on the scale likely are of equal intervals.

Ratio data is interval data with a natural zero point. For example, time is ratio since 0 time is meaningful. Degrees Kelvin has a 0 point (absolute 0) and the steps in both these scales have the same degree of magnitude.

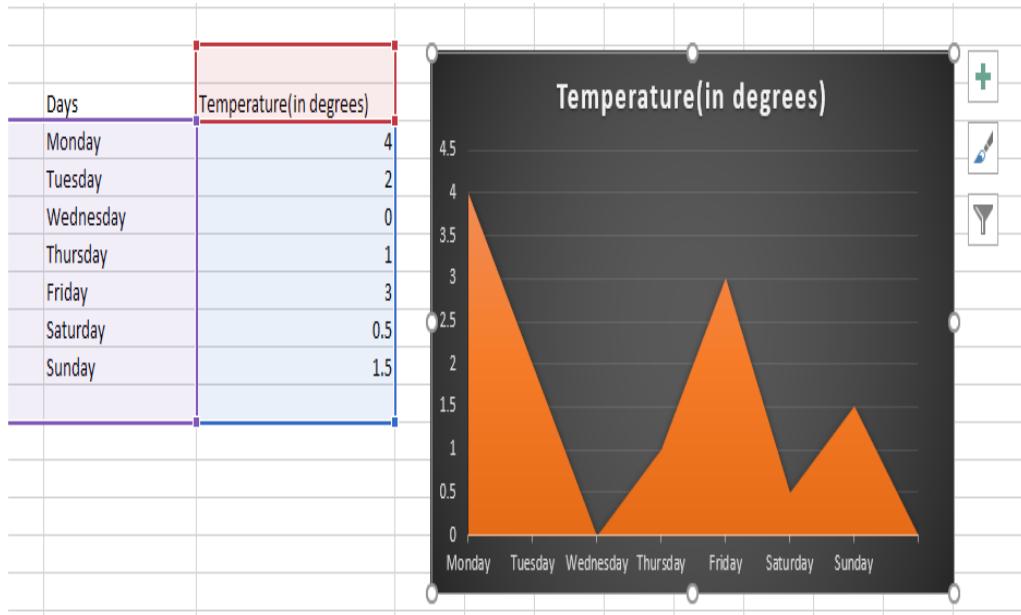
Steps:

1. Enter the data of sales months of the year.
2. Represent the data with respect to months, where the difference between each month is equally 1 month on the X axis.



Steps:

- 1. Enter the data of temperature of the week in Artic circle.**
- 2. Represent the data with respect to week days, where the difference between each day is 24hrs and also the temperature can be absolute zero i.e. 0 Celsius.**



Practical 8

Aim: Perform t-test on the data.

Steps:

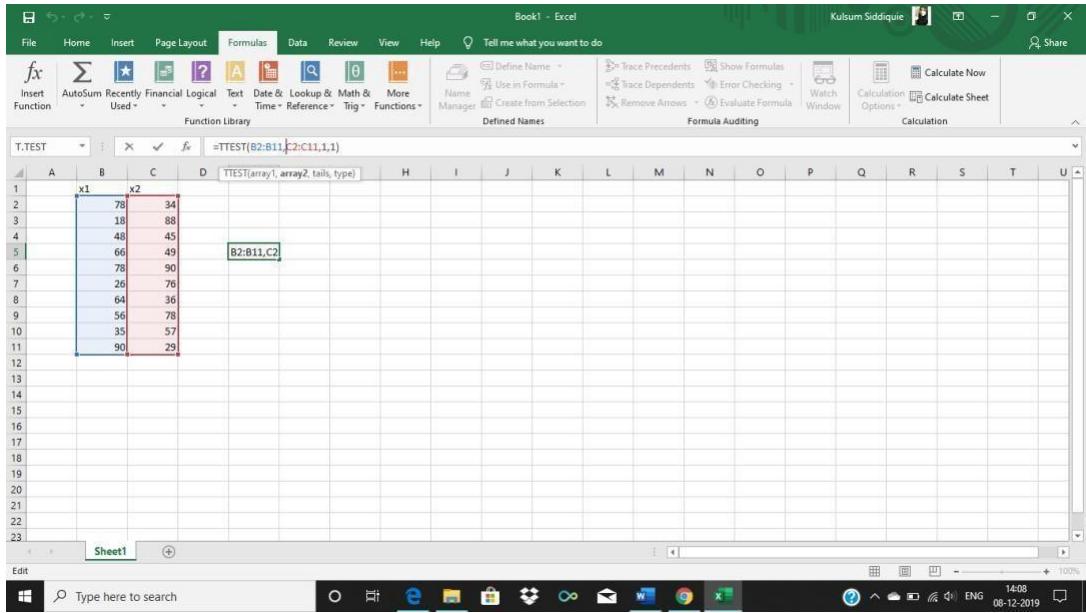
1. Store the two arrays of data. In excel.

The screenshot shows a Microsoft Excel spreadsheet titled "Book1 - Excel". The data is organized into two columns: "x1" and "x2". Column "x1" contains values 78, 18, 48, 66, 78, 26, 64, 56, 35, and 90. Column "x2" contains values 34, 88, 45, 49, 90, 76, 36, 78, 57, and 29. The formula bar at the top shows the address "B12". The status bar at the bottom indicates the date and time as "08-12-2019 13:59".

2. Select the cell, enter the function: test (array1, array2, tails, type)

The screenshot shows the Microsoft Excel interface with the "T.TEST" function dialog box open. The formula bar displays the function as "=T.TEST()". The dialog box is titled "Function Arguments" and shows four parameters: "Array1" (selected), "Array2" (not selected), "Tails" (set to 1), and "Type" (set to 2). The "OK" button is visible at the bottom right of the dialog box. The background shows the same data as the previous screenshot, with the formula bar showing "B12" and the status bar indicating the date and time.

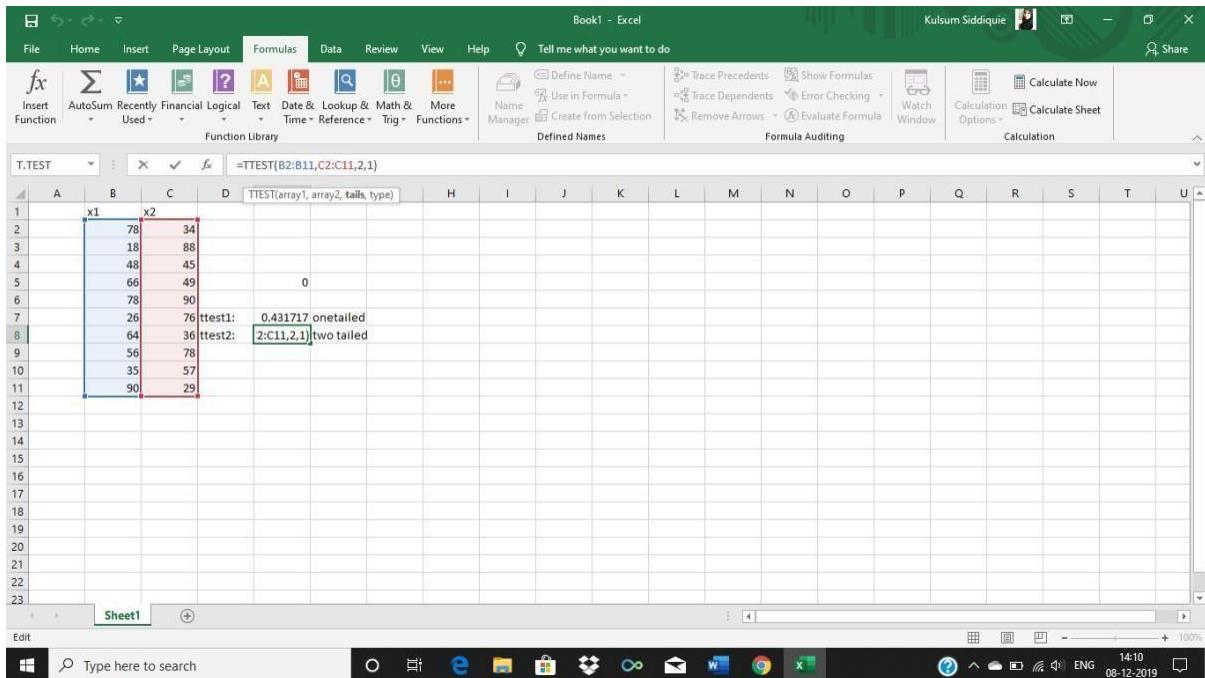
- Enter for one tailed test:
 $=ttest(b2: b11, c2:c11,1,1)$
 Press enter.



The screenshot shows an Excel spreadsheet titled "Book1 - Excel". The formula bar displays the formula $=TTEST(B2:B11,C2:C11,1,1)$. The data range selected is B2:B11, C2:C11. The result of the formula, 0.431717, is displayed in cell D1. The status bar at the bottom right shows the date and time as 08-12-2019 14:08.

	x1	x2
2	78	34
3	18	88
4	48	45
5	66	49
6	78	90
7	26	76
8	64	36
9	56	78
10	35	57
11	90	29

- Enter for two tailed test:



The screenshot shows an Excel spreadsheet titled "Book1 - Excel". The formula bar displays the formula $=TTEST(B2:B11,C2:C11,2,1)$. The data range selected is B2:B11, C2:C11. The result of the formula, 0.431717, is displayed in cell D1. A note "onetailed" is displayed above the result, and another note "twotailed" is displayed below it. The status bar at the bottom right shows the date and time as 08-12-2019 14:10.

	x1	x2
2	78	34
3	18	88
4	48	45
5	66	49
6	78	90
7	26	76
8	64	36
9	56	78
10	35	57
11	90	29

=ttest (b2: b11, c2:c11,1)

Press enter.

