Index

Sr. No	Practical Name	Date	Signature
1	Create Data Model using Cassandra.	09/10/2019	
2	Conversion from different formats to HORUS format.	14/10/2019	
3	Auditing through Logging.	14/10/2019	
4	Retrieving Data.	16/10/2019	
5	Assessing Data.	23/10/2019	
6	Processing Data.	04/11/2019	
7	Transforming Data.	06/11/2019	
8	Organizing Data.	13/11/2019	
9	Generating Reports.	13/11/2019	
10	Data Visualization with Power BI.	14/10/2019	

Practical No. 1

1. Create data model using Cassandra.

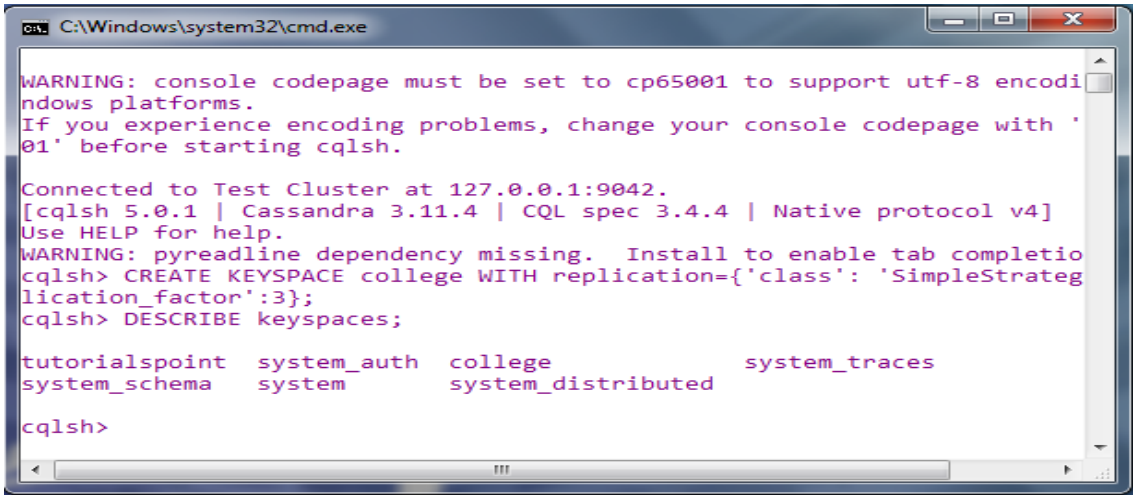
Required Software:

Java v1.8, Python v2.7, Cassandra File

ANS:

Create Key space:

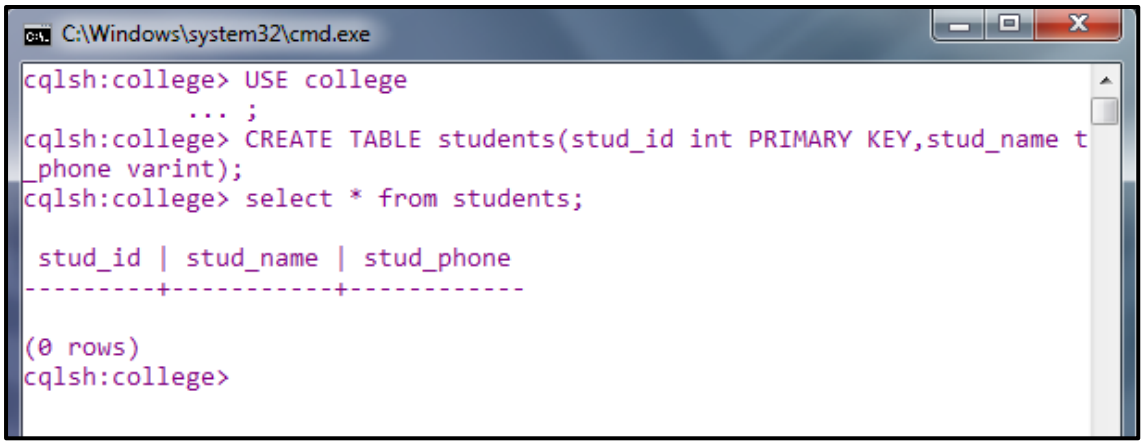
cqlsh> CREATE KEYSPACE college WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};



Create Table

cqlsh:college> CREATE TABLE students(stud_id int PRIMARY KEY,stud_name text,stud_phone varint);

cqlsh:college> select * from students;



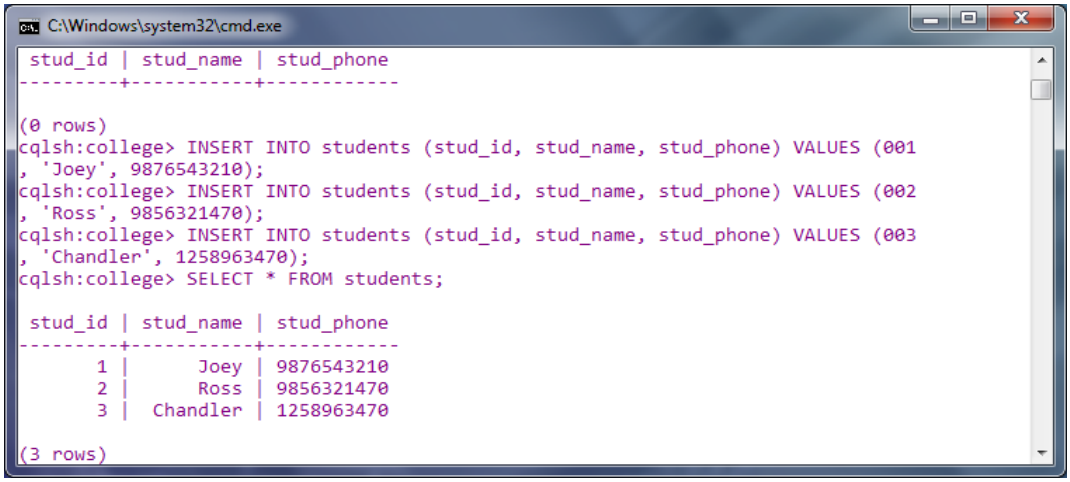
Creating Data in a Table

cqlsh:college> INSERT INTO students (stud_id, stud_name, stud_phone) VALUES (001, 'Joey', 9876543210);

cqlsh:college> INSERT INTO students (stud_id, stud_name, stud_phone) VALUES (002, 'Ross', 9856321470);

cqlsh:college> INSERT INTO students (stud_id, stud_name, stud_phone) VALUES (003, 'Chandler', 1258963470);

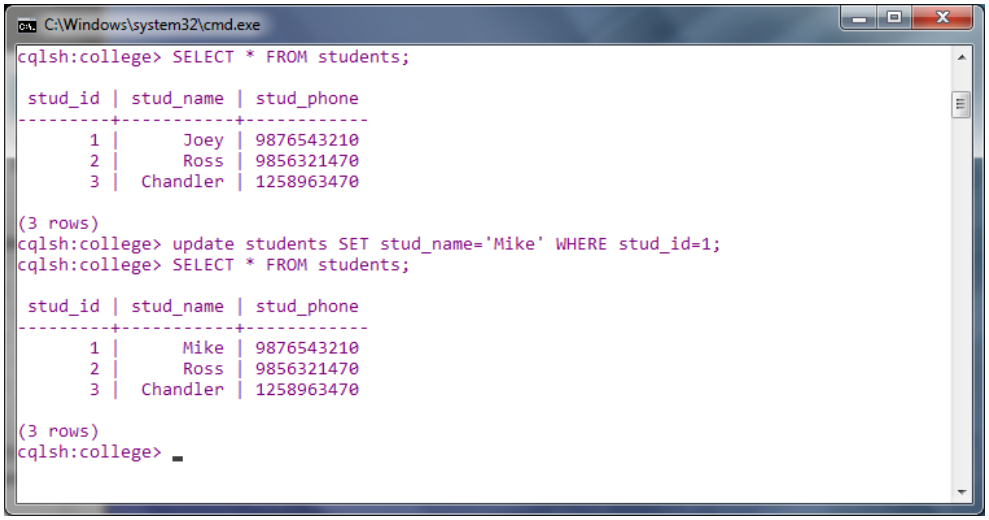
cqlsh:college> SELECT * FROM students;



Update data in table

cqlsh:college> update students SET stud_name='Mike' WHERE stud_id=1;

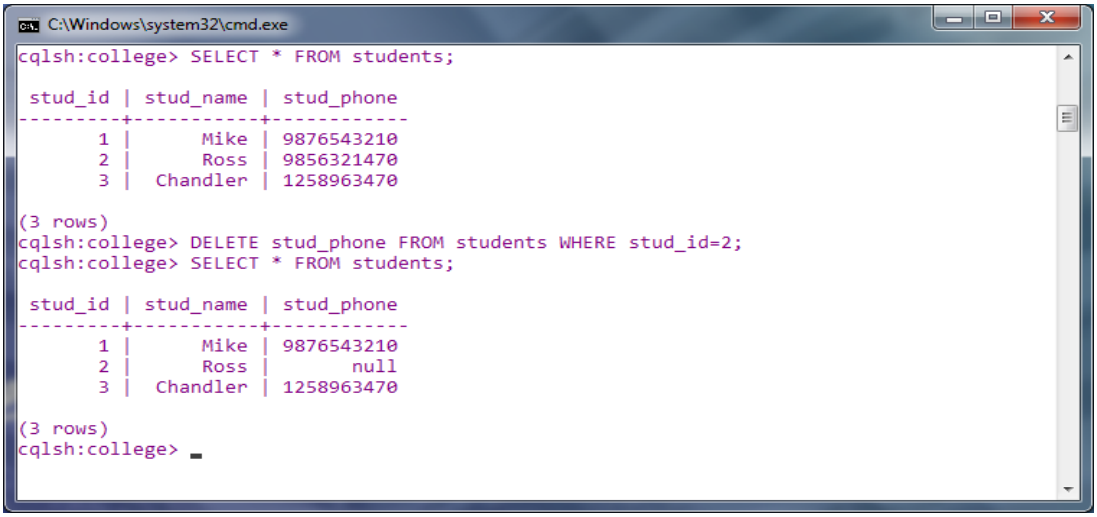
cqlsh:college> SELECT * FROM students;



Delete data from table

cqlsh:college> DELETE stud_phone FROM students WHERE stud_id=2;

cqlsh:college> SELECT * FROM students;



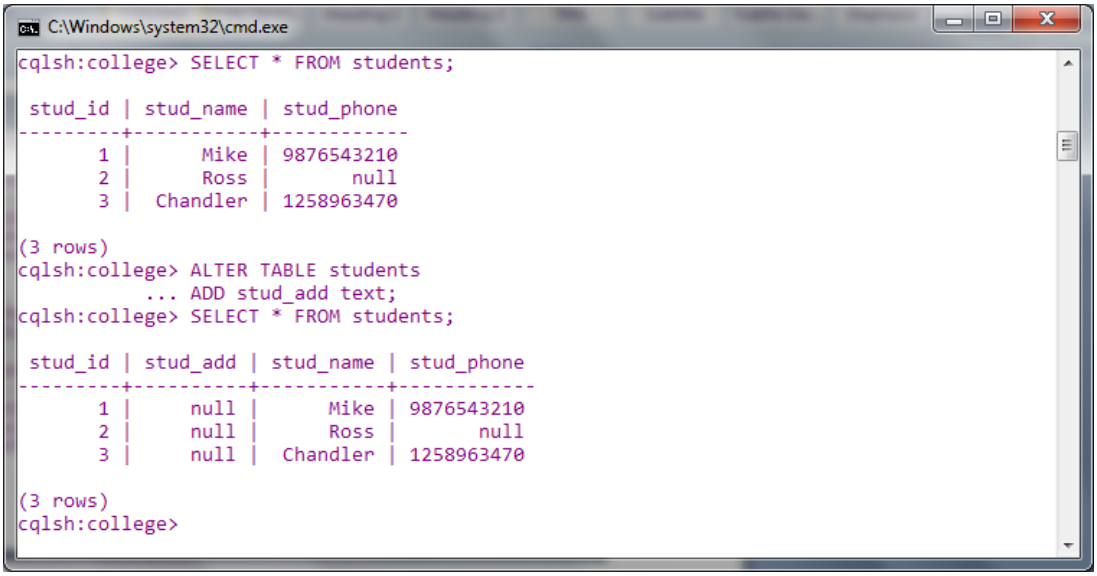
Alter Table

1. **Adding a column**

cqlsh:college> ALTER TABLE students

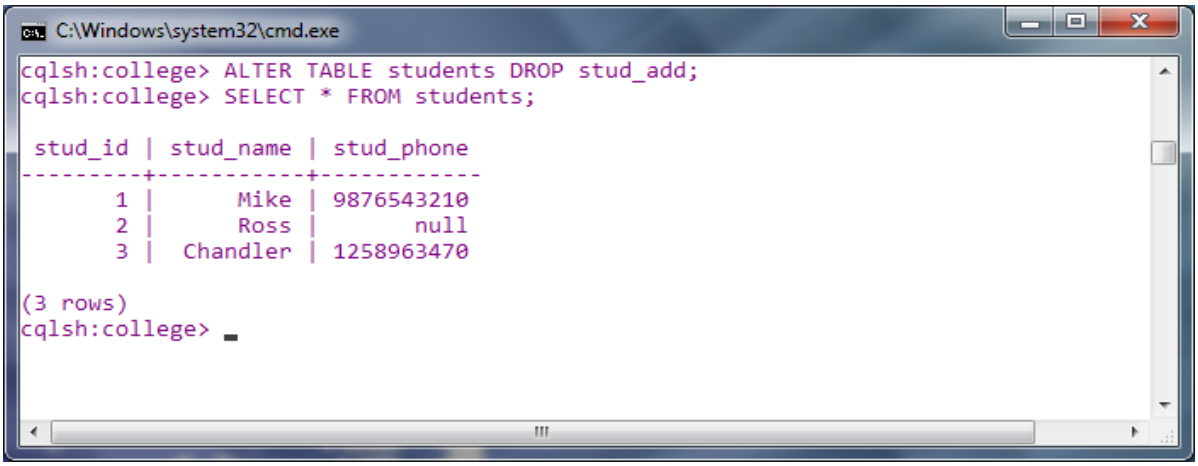
... ADD stud_add text;

cqlsh:college> SELECT * FROM students;



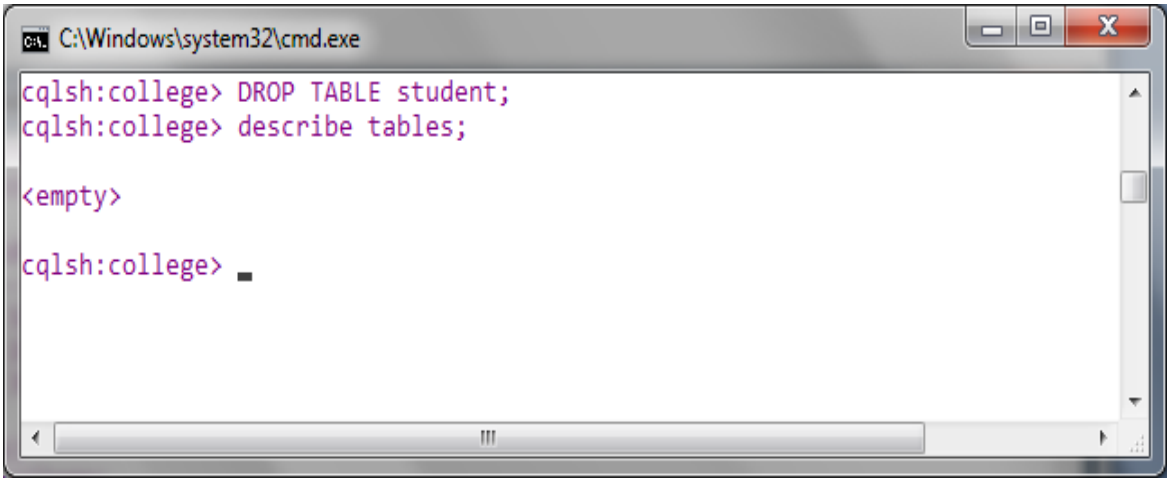
2. **Dropping a column**

cqlsh:college> ALTER TABLE students DROP stud_add;
cqlsh:college> SELECT * FROM students;



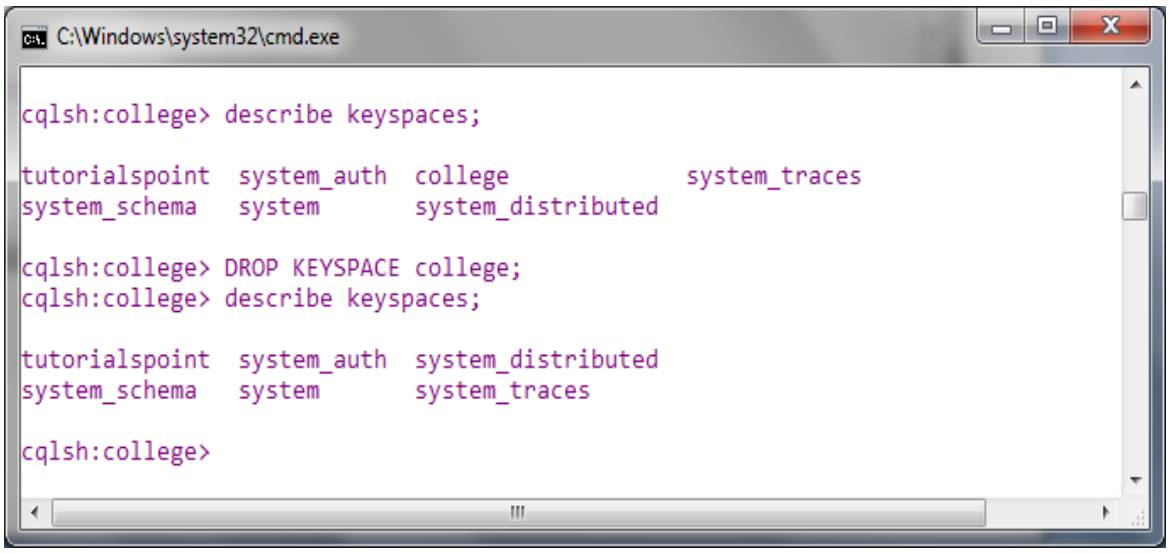
Drop Table

cqlsh:college> DROP TABLE student;
cqlsh:college> describe tables;



Drop Keyspace

cqlsh:college> DROP KEYSPACE college;
cqlsh:college> describe keyspaces;



Practical No. 2

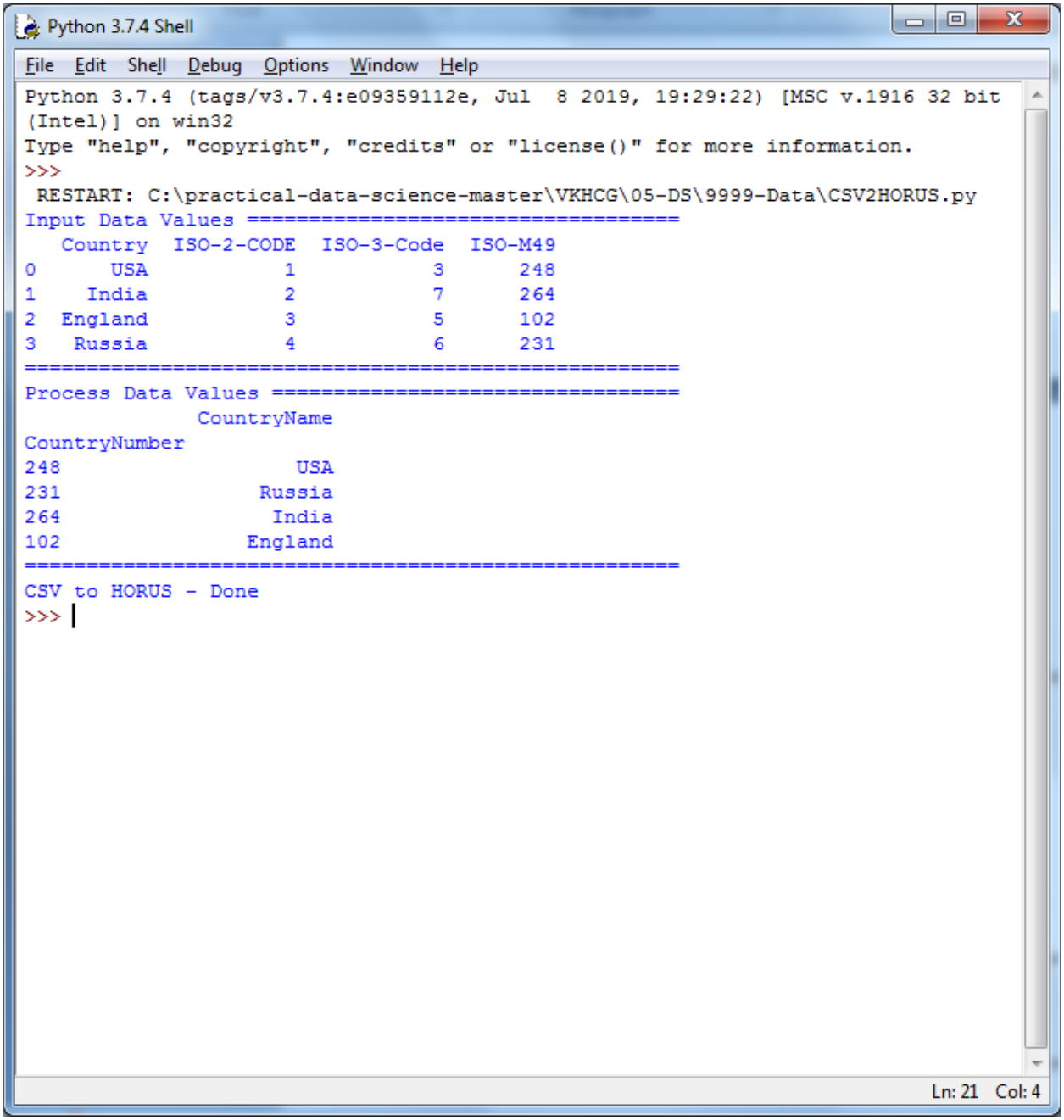
Conversion from different formats to HORUS format.

CSV to HORUS

INPUT:

```
import pandas as pd
from datetime import datetime
sInputFileName='C:/practical-data-science-master/VKHCG/05-DS/9999-
Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
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)

now=datetime.now()
print("now = ",now)
dt_string=now.strftime("%d/%m/%y %H:%M:%S")
print("Date and Time= ",dt_string)
f=open('C:/practical-data-science-master/VKHCG/05-DS/9999-
Data/Country_Code_Log.txt',"a")
f.write("Delete column activity recorded at ")
f.write(dt_string)
f.close()
# 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)
OutputData=ProcessData
sOutputFileName='C:/practical-data-science-master/VKHCG/05-DS/9999-Data/HORUS-
CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('CSV to HORUS - 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:\practical-data-science-master\VKHCG\05-DS\9999-Data\CSV2HORUS.py
Input Data Values ==================================
Country ISO-2-CODE ISO-3-Code ISO-M49
0 USA 1 3 248
1 India 2 7 264
2 England 3 5 102
3 Russia 4 6 231
==========
Process Data Values ==================================
CountryName
CountryNumber
248 USA
231 Russia
264 India
102 England
==========
CSV to HORUS - Done
>>>
Ln: 21 Col: 4

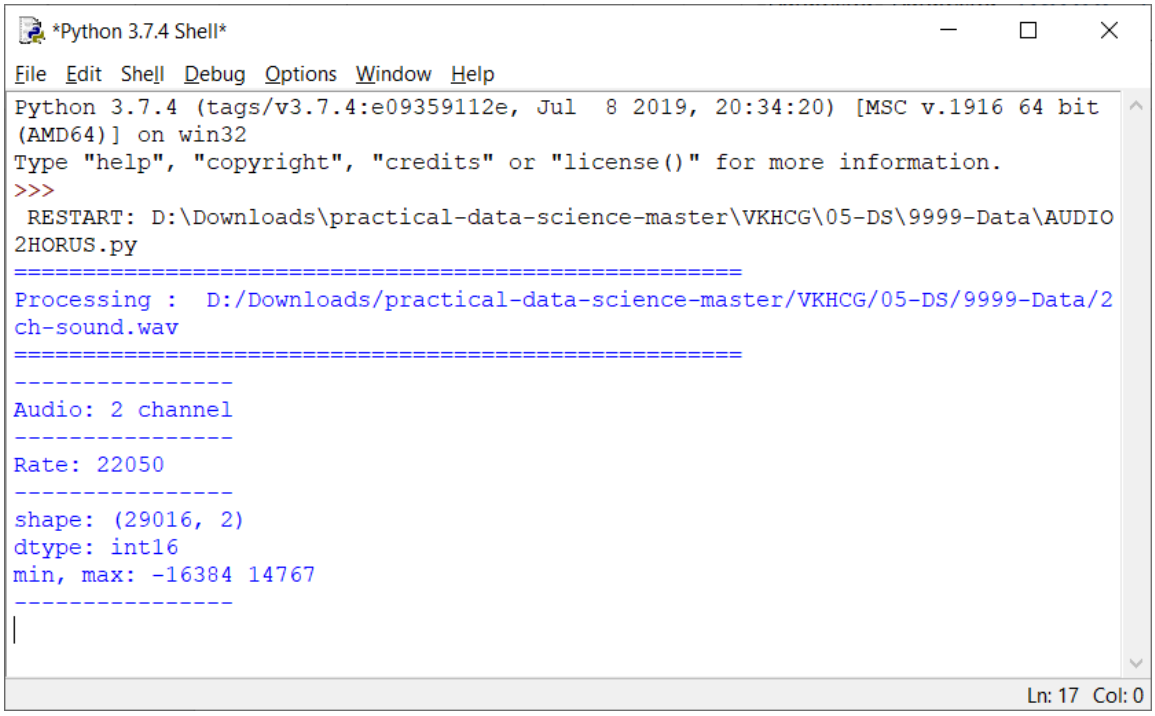
CSV TO AUDIO

INPUT:

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

```
sOutputFileName='D:/Downloads/practical-data-science-master/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='D:/Downloads/practical-data-science-master/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('Processing : ', sInputFileName)
InputRate, InputData = wavfile.read(sInputFileName)
show_info("6 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='D:/Downloads/practical-data-science-master/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
sInputFileName='D:/Downloads/practical-data-science-master/VKHCG/05-DS/9999-Data/8ch-sound.wav'
print('Processing : ', sInputFileName)
InputRate, InputData = wavfile.read(sInputFileName)
show_info("8 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='D:/Downloads/practical-data-science-master/VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('Audio to HORUS - 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, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: D:\Downloads\practical-data-science-master\VKHCG\05-DS\9999-Data\AUDIO2HORUS.py
Processing : D:/Downloads/practical-data-science-master/VKHCG/05-DS/9999-Data/2ch-sound.wav
Audio: 2 channel
Rate: 22050
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
Ln: 17 Col: 0

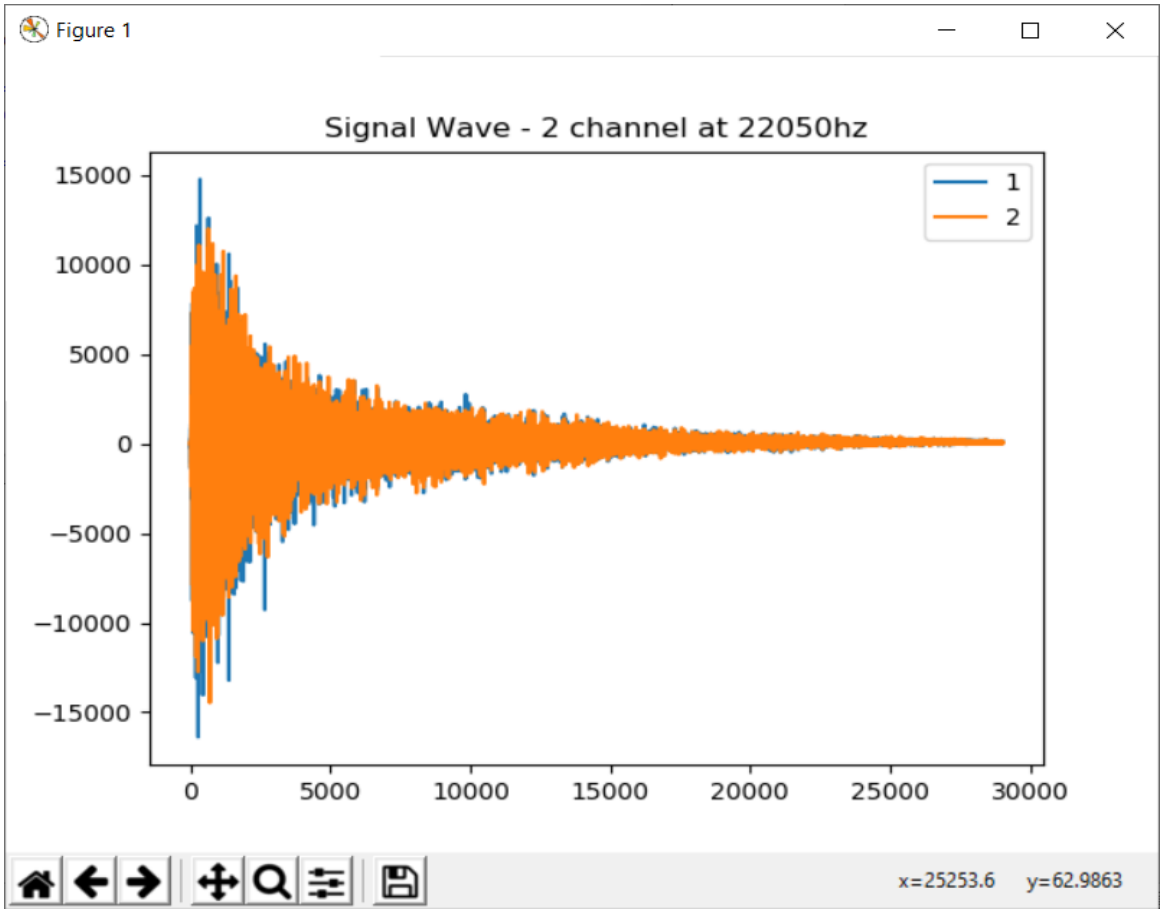


Figure 1
Signal Wave - 2 channel at 22050hz
1
2
15000
10000
5000
0
−5000
−10000
−15000
0
5000
10000
15000
20000
25000
30000
x=25253.6 y=62.9863

Practical No. 3

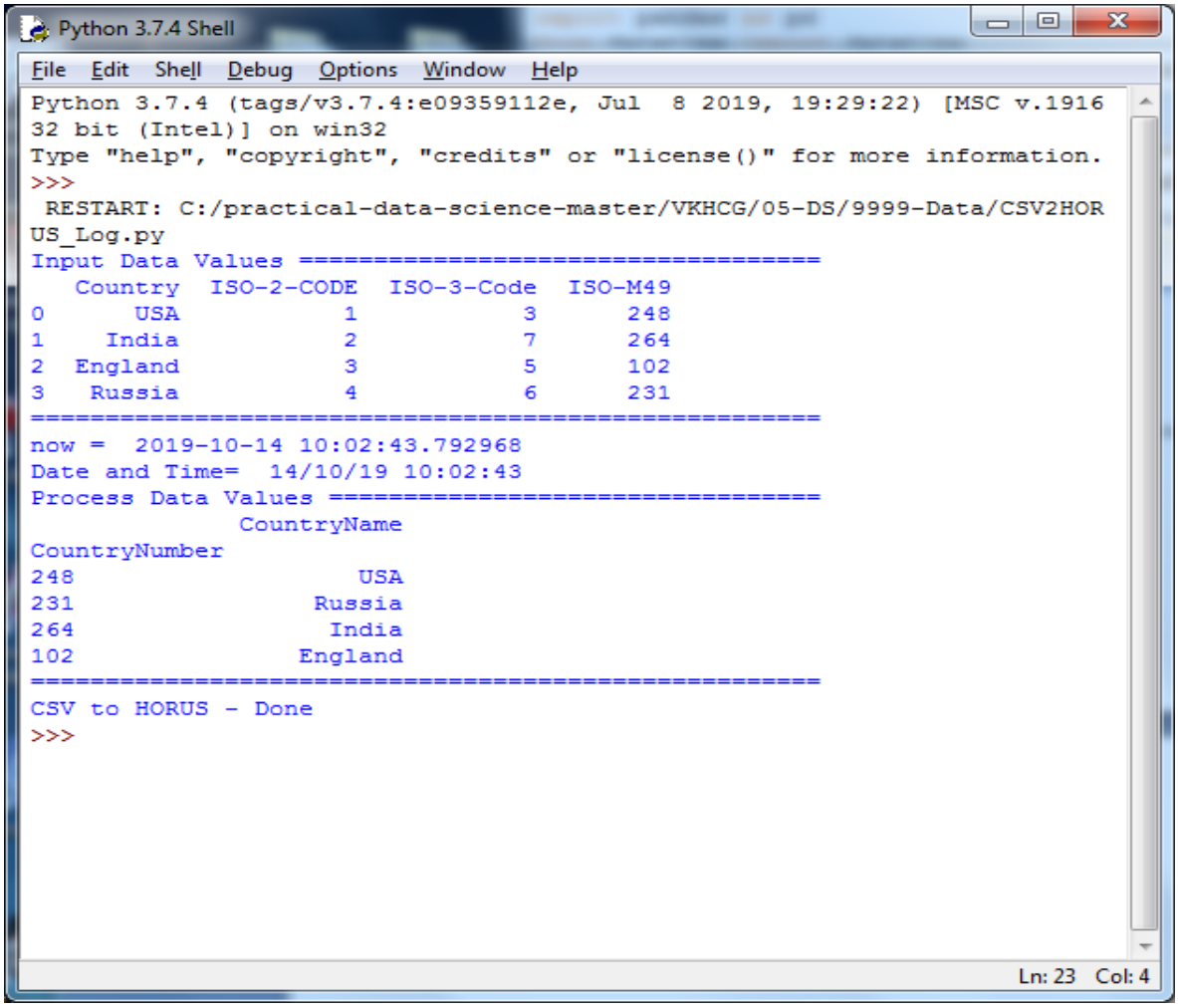
Auditing through Logging

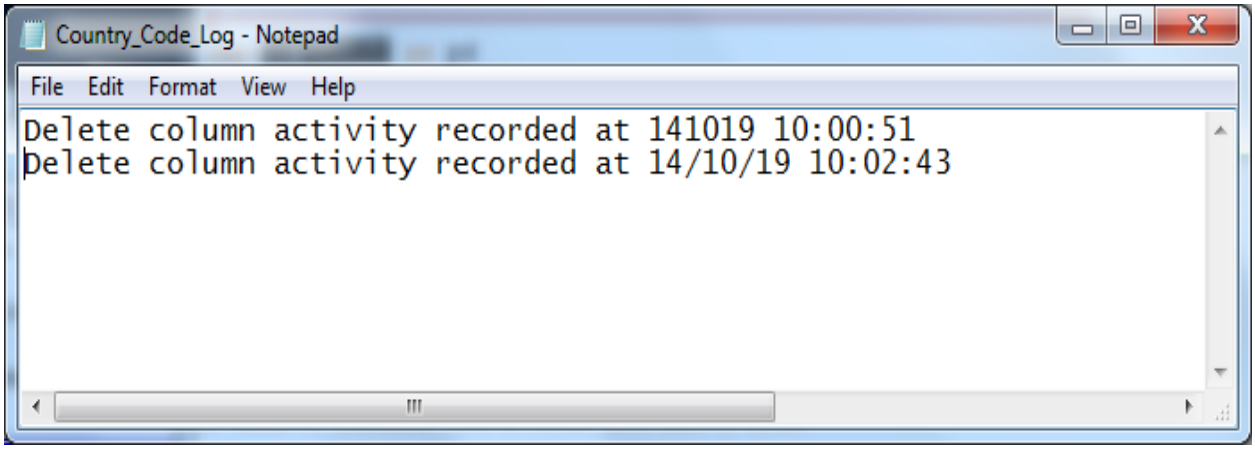
CSV to HORUS log file

INPUT:

```
import pandas as pd
from datetime import datetime
sInputFileName='C:/practical-data-science-master/VKHCG/05-DS/9999-
Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
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)
now=datetime.now()
print("now = ",now)
dt_string=now.strftime("%d/%m/%y %H:%M:%S")
print("Date and Time= ",dt_string)
f=open('C:/practical-data-science-master/VKHCG/05-DS/9999-
Data/Country_Code_Log.txt',"a")
f.write("Delete column activity recorded at ")
f.write(dt_string)
f.close()
# 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)
OutputData=ProcessData
sOutputFileName='C:/practical-data-science-master/VKHCG/05-DS/9999-Data/HORUS-
CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - 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:/practical-data-science-master/VKHCG/05-DS/9999-Data/CSV2HOR
US_Log.py
Input Data Values ===================================
Country ISO-2-CODE ISO-3-Code ISO-M49
0 USA 1 3 248
1 India 2 7 264
2 England 3 5 102
3 Russia 4 6 231
===
now = 2019-10-14 10:02:43.792968
Date and Time= 14/10/19 10:02:43
Process Data Values ================================
CountryName
CountryNumber
248 USA
231 Russia
264 India
102 England
===
CSV to HORUS - Done
>>>
Ln: 23 Col: 4


Country_Code_Log - Notepad
File Edit Format View Help
Delete column activity recorded at 141019 10:00:51
Delete column activity recorded at 14/10/19 10:02:43

Practical No. 4

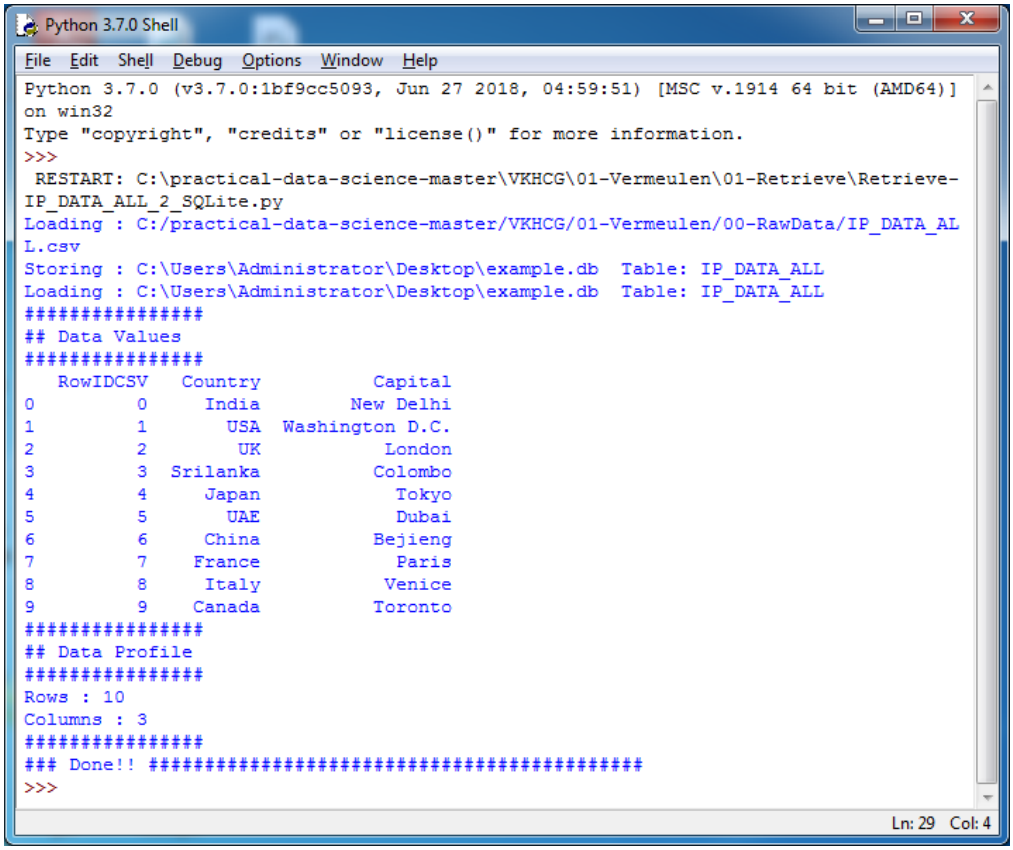
Retrieving the data.

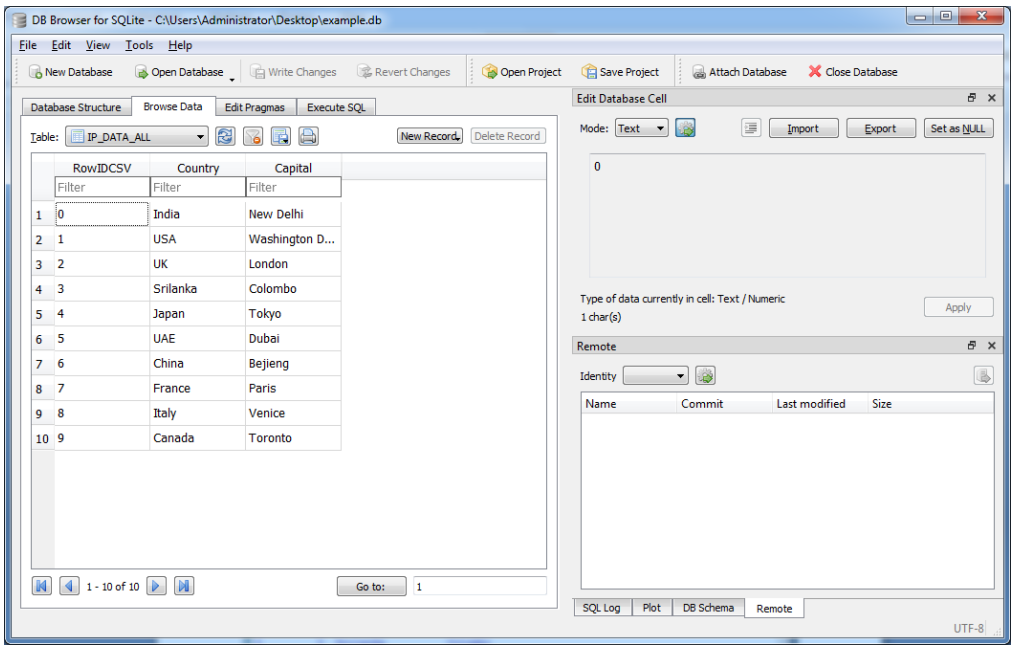
CSV to SQL

INPUT:

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

sDatabaseName="C:\\Users\\Administrator\\Desktop\\example.db"
conn = sq.connect(sDatabaseName)
sFileName='C:/practical-data-science-master/VKHCG/01-Vermeulen/00-
RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")

IP_DATA_ALL.index.names = ['RowIDCSV']
sTable='IP_DATA_ALL'

print('Storing :',sDatabaseName,' Table:',sTable)
IP_DATA_ALL.to_sql(sTable, conn, if_exists="replace")

print('Loading :',sDatabaseName,' Table:',sTable)
TestData=pd.read_sql_query("select * from IP_DATA_ALL;", conn)

print('## Data Values')
print(TestData)
print('## Data Profile')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('### Done!! ‘)
```

OUTPUT:


Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\practical-data-science-master\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-IP_DATA_ALL_2_SQLite.py
Loading : C:/practical-data-science-master/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Storing : C:\Users\Administrator\Desktop\example.db Table: IP_DATA_ALL
Loading : C:\Users\Administrator\Desktop\example.db Table: IP_DATA_ALL
################
Data Values
################
RowIDCSV Country Capital
0 0 India New Delhi
1 1 USA Washington D.C.
2 2 UK London
3 3 Srilanka Colombo
4 4 Japan Tokyo
5 5 UAE Dubai
6 6 China Bejieng
7 7 France Paris
8 8 Italy Venice
9 9 Canada Toronto
################
Data Profile
################
Rows : 10
Columns : 3
################
Done!!
>>>
Ln: 29 Col: 4


DB Browser for SQLite - C:\Users\Administrator\Desktop\example.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: IP_DATA_ALL New Record Delete Record
RowIDCSV Country Capital
1 0 India New Delhi
2 1 USA Washington D...
3 2 UK London
4 3 Srilanka Colombo
5 4 Japan Tokyo
6 5 UAE Dubai
7 6 China Bejieng
8 7 France Paris
9 8 Italy Venice
10 9 Canada Toronto
1 - 10 of 10 Go to: 1
Edit Database Cell
Mode: Text Import Export Set as NULL
0
Type of data currently in cell: Text / Numeric
1 char(s)
Apply
Remote
Identity
Name Commit Last modified Size
SQL Log Plot DB Schema Remote
UTF-8

Practical No. 5

Assessing the Data.

1. **Drop a column - parameter = ‘‘how’’**

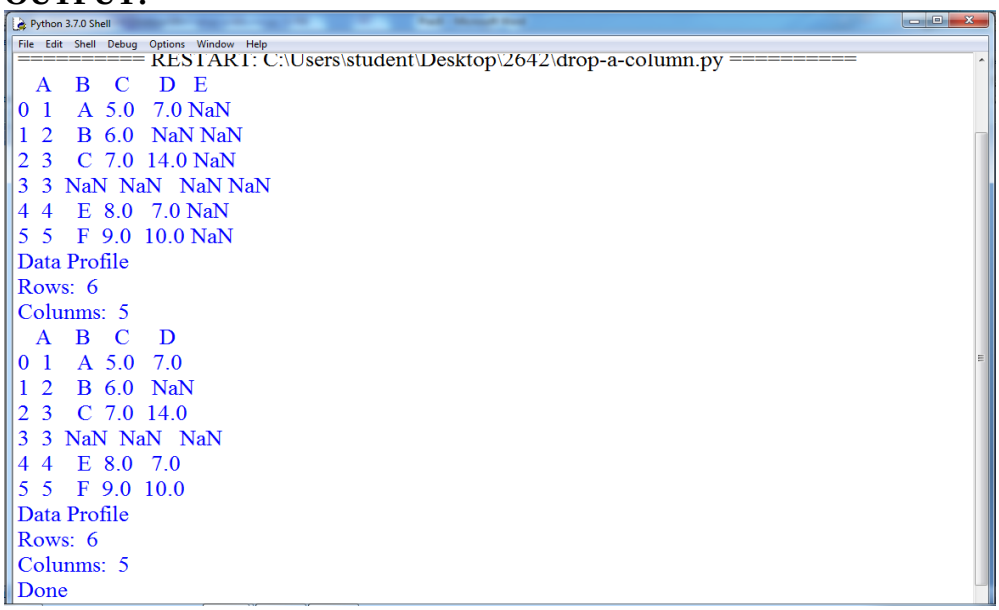
INPUT:

```
import sys
import os
import pandas as pd
sInputFile = "Good-or-Bad.csv"
sOutputFile="Good-orBaad-01.csv"
RawData=pd.read_csv("c:/Users/student/Desktop/2642/Good-or-Bad.csv",header=0)
print(RawData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData=RawData.dropna(axis=1, how=”all”)
print(TestData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData.to_csv("C:/Users/student/Desktop/2642/Good-or-Baad-01.csv",index="False")
print("Done")
```

Good-or-Bad.csv-

A	B	C	D	E
1	A	5	7	
2	B	6		
3	C	7	14	
3				
4	E	8	7	
5	F	9	10	

OUTPUT:

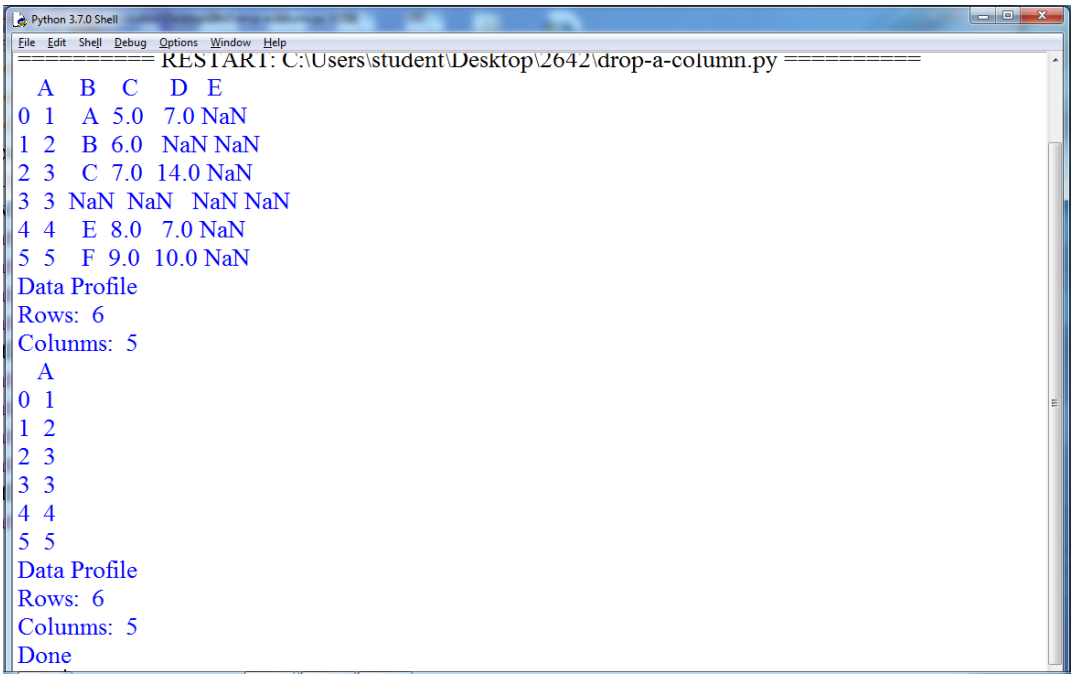


2. **Drop a column – parameter= “any”**

INPUT:

```
import sys
import os
import pandas as pd
sInputFile = "Good-or-Bad.csv"
sOutputFile="Good-orBaad-01.csv"
RawData=pd.read_csv("c:/Users/student/Desktop/2642/Good-or-Bad.csv",header=0)
print(RawData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData=RawData.dropna(axis=1,how="any")
print(TestData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData.to_csv("C:/Users/student/Desktop/2642/Good-or-Baad-01.csv",index="False")
print("Done")
```

OUTPUT:



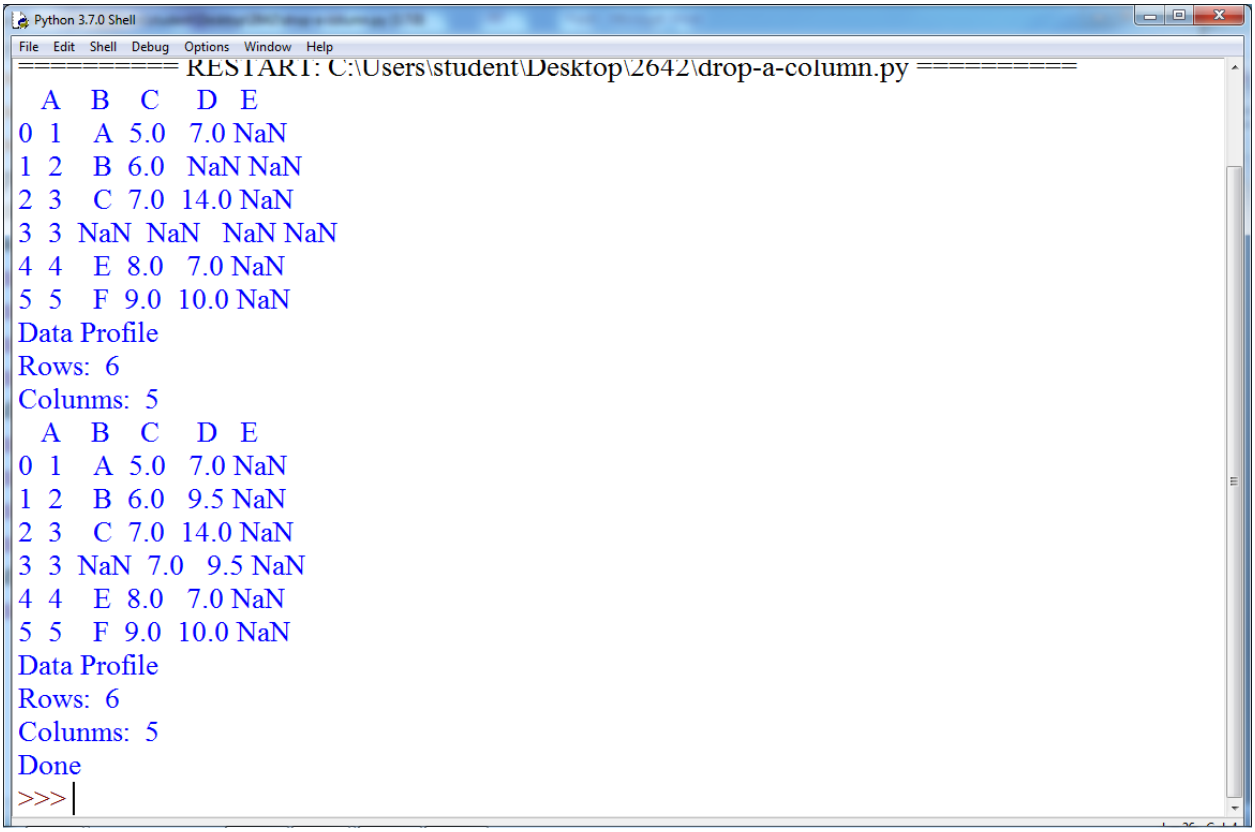
3. **Filing the missing using mean(), min(), max() method:**

i. **Using mean():**

INPUT:

```
import sys
import os
import pandas as pd
sInputFile = "Good-or-Bad.csv"
sOutputFile="Good-orBaad-01.csv"
RawData=pd.read_csv("c:/Users/student/Desktop/2642/Good-or-Bad.csv",header=0)
print(RawData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData=RawData.fillna(RawData.mean())
print(TestData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData.to_csv("C:/Users/student/Desktop/2642/Good-or-Baad-01.csv",index="False")
print("Done")
```

OUTPUT:

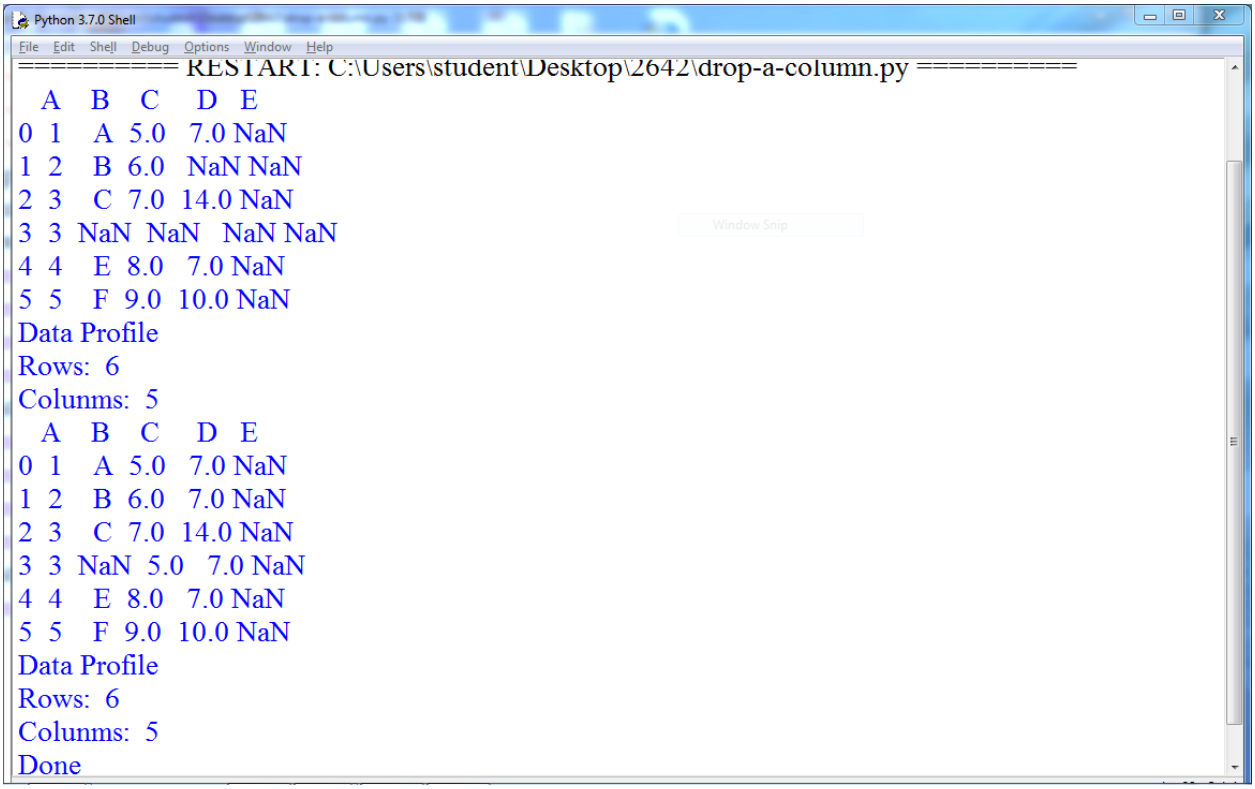


ii. Using min():

INPUT:

```
import sys
import os
import pandas as pd
sInputFile = "Good-or-Bad.csv"
sOutputFile="Good-orBaad-01.csv"
RawData=pd.read_csv("c:/Users/student/Desktop/2642/Good-or-Bad.csv",header=0)
print(RawData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData=RawData.fillna(RawData.min())
print(TestData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Columns: ",RawData.shape[1])
TestData.to_csv("C:/Users/student/Desktop/2642/Good-or-Baad-01.csv",index="False")
print("Done")
```

OUTPUT:

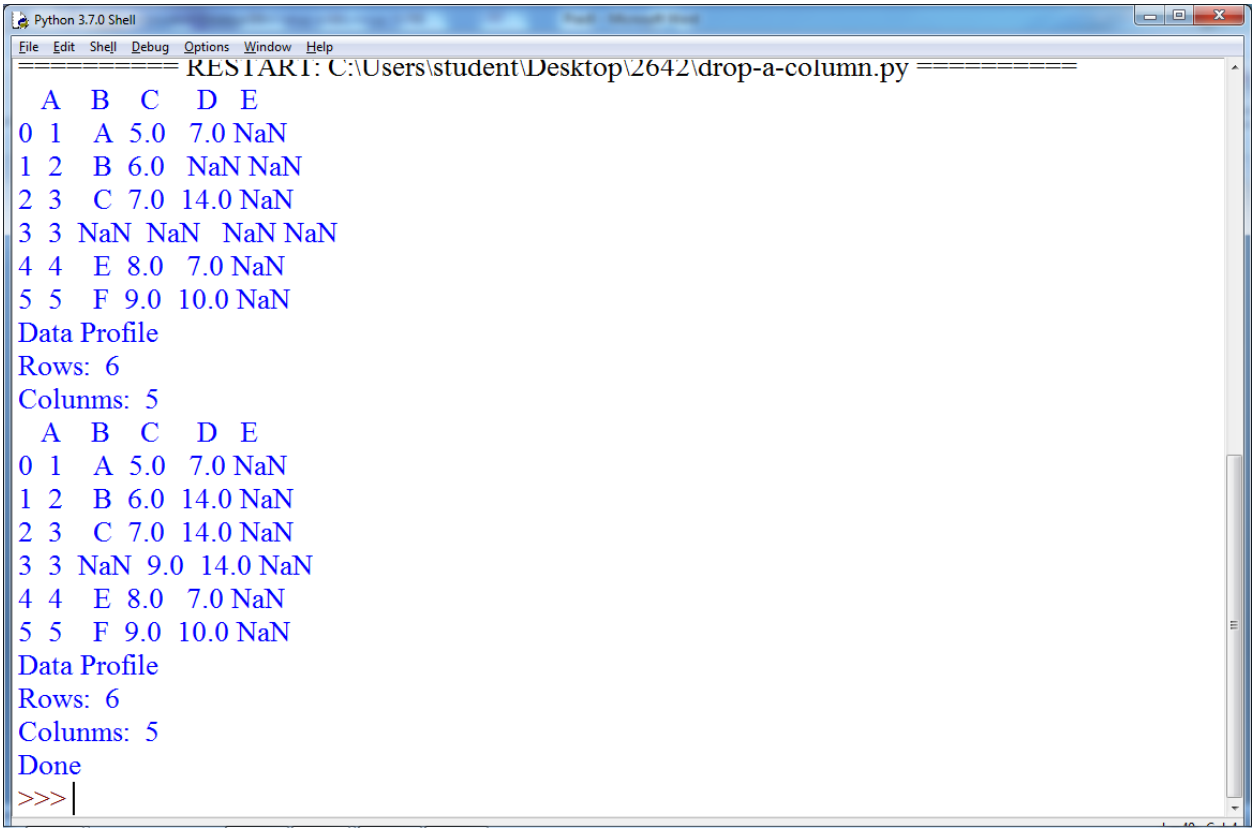


iii. Using max():

INPUT:

```
import sys
import os
import pandas as pd
sInputFile = "Good-or-Bad.csv"
sOutputFile="Good-orBaad-01.csv"
RawData=pd.read_csv("c:/Users/student/Desktop/2642/Good-or-Bad.csv",header=0)
print(RawData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Colunms: ",RawData.shape[1])
TestData=RawData.fillna(RawData.max())
print(TestData)
print("Data Profile")
print("Rows: ",RawData.shape[0])
print("Colunms: ",RawData.shape[1])
TestData.to_csv("C:/Users/student/Desktop/2642/Good-or-Baad-01.csv",index="False")
print("Done")
```

OUTPUT:



Practical No. 6

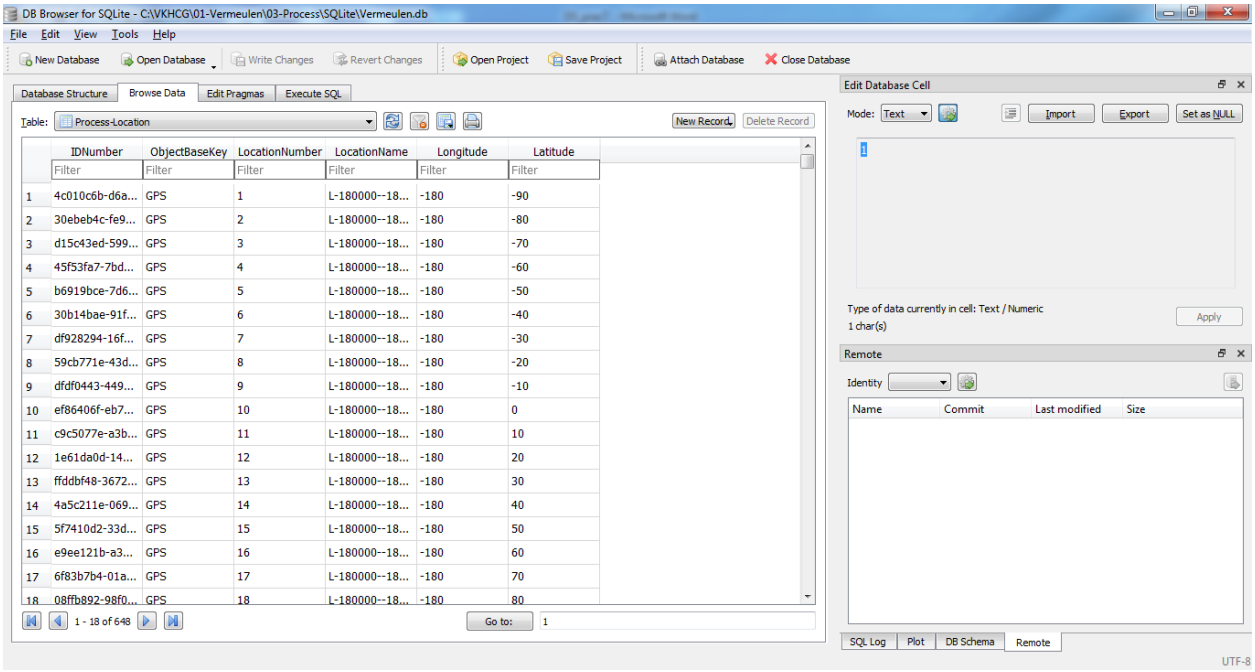
Processing Data

Process Location

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

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

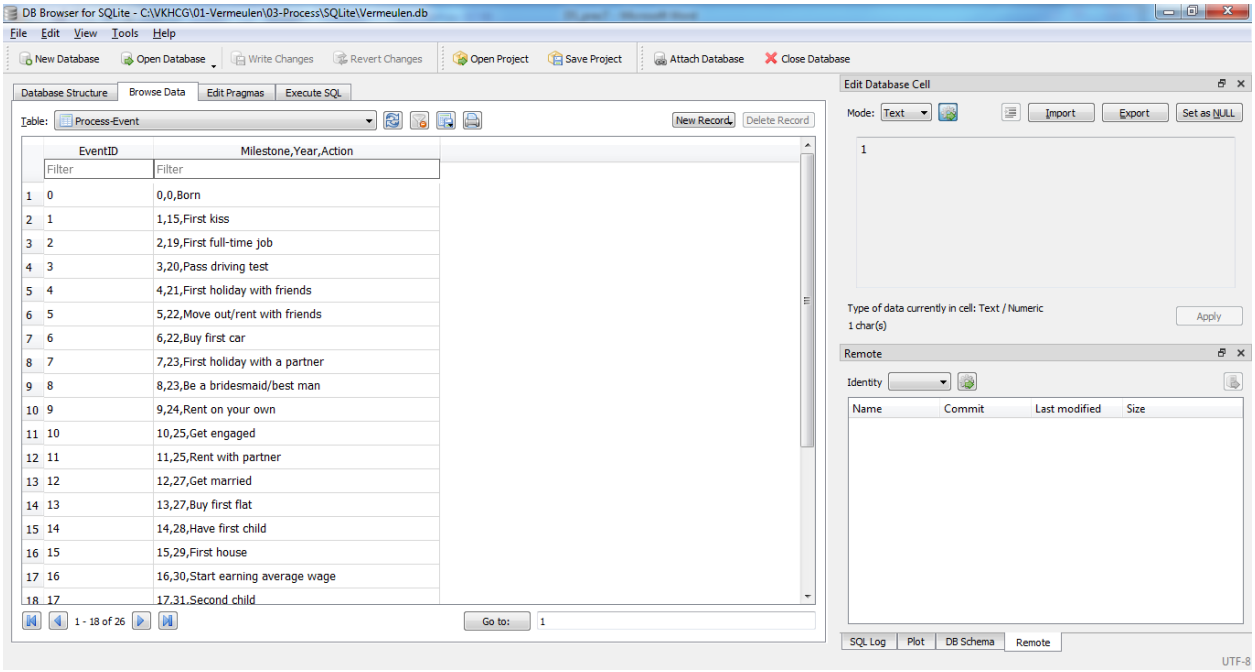
OUTPUT:



Process Event

```
import sys
import os
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
Base='C:/VKHCG'
print('Working Base :',Base, ' using ', sys.platform)
Company='01-Vermeulen'
InputFileName='Action_Plan.csv'
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
sFileName=Base + '/' + Company + '/00-RawData/' + InputFileName
print('Loading :',sFileName)
EventRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
EventRawData.index.names=['EventID']
EventHubIndex=EventRawData
sTable = 'Process-Event'
print('Storing :',sDatabaseName,' Table:',sTable)
EventHubIndex.to_sql(sTable, conn1, if_exists="replace")
sTable = 'Hub-Event'
print('Storing :',sDatabaseName,' Table:',sTable)
#EventHubIndex.to_sql(sTable, conn2, if_exists="replace")
print('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
#sql.execute(sSQL,conn2)
print('### Done!! ############################################')
```

OUTPUT:


DB Browser for SQLite - C:\VKHCG\01-Vermeulen\03-Process\SQLite\Vermeulen.db
File Edit View Tools Help
New Database
Open Database
Write Changes
Revert Changes
Open Project
Save Project
Attach Database
Close Database
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: Process-Event
New Record
Delete Record
EventID
Milestone,Year,Action
Filter
1 0 0,0,Born
2 1 1,15,First kiss
3 2 2,19,First full-time job
4 3 3,20,Pass driving test
5 4 4,21,First holiday with friends
6 5 5,22,Move out/rent with friends
7 6 6,22,Buy first car
8 7 7,23,First holiday with a partner
9 8 8,23,Be a bridesmaid/best man
10 9 9,24,Rent on your own
11 10 10,25,Get engaged
12 11 11,25,Rent with partner
13 12 12,27,Get married
14 13 13,27,Buy first flat
15 14 14,28,Have first child
16 15 15,29,First house
17 16 16,30,Start earning average wage
18 17 17,31,Second child
1 - 18 of 26
Go to:
Edit Database Cell
Mode: Text
Import
Export
Set as NULL
Type of data currently in cell: Text / Numeric
1 char(s)
Apply
Remote
Identity
Name
Commit
Last modified
Size
SQL Log
Plot
DB Schema
Remote
UTF-8

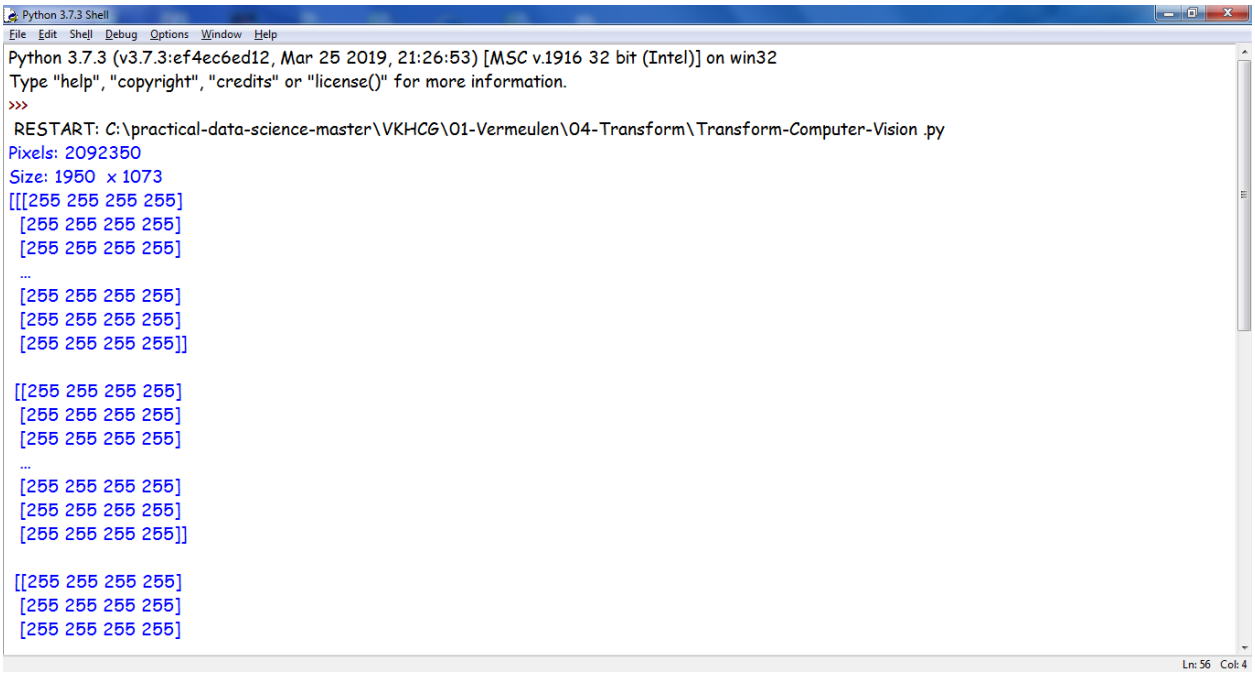
Practical No. 7

Transforming Data.

INPUT:

```
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
sPicNameIn='C:/practical-data-science-master/VKHCG/01-Vermeulen/00-
RawData/AudiR8.png'
imageIn = Image.open(sPicNameIn)
fig1=plt.figure(figsize=(10, 10))
fig1.suptitle('Audi R8', fontsize=20)
imgplot = plt.imshow(imageIn)
plt.show()

imagewidth, imageheight = imageIn.size
imageMatrix=np.asarray(imageIn)
pixelscnt = (imagewidth * imageheight)
print('Pixels:', pixelscnt)
print('Size:', imagewidth, ' x', imageheight,)
print(imageMatrix)
```

OUTPUT:



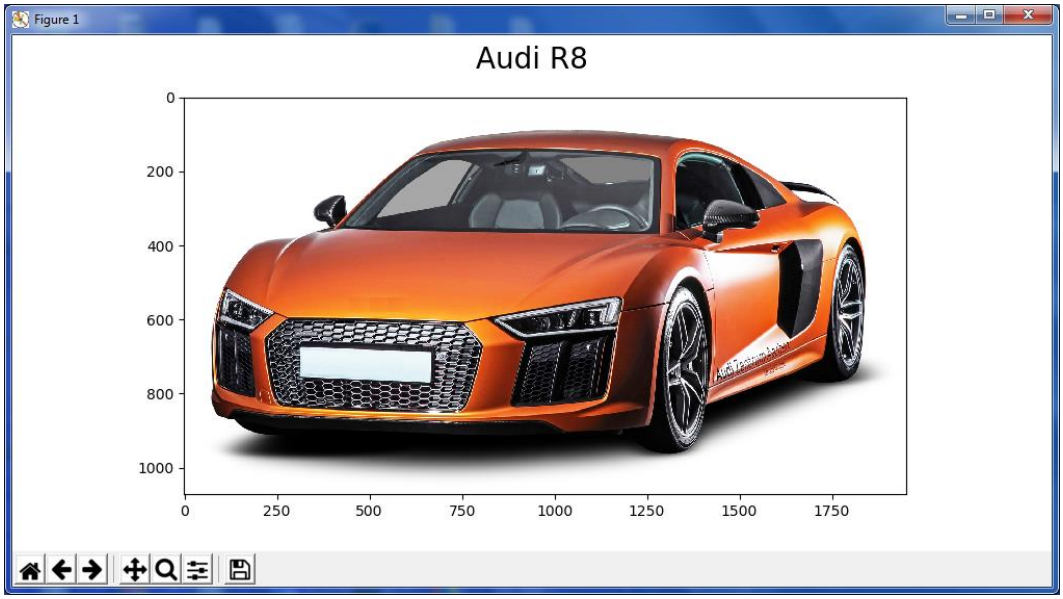


Figure 1
Audi R8
0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750

Transform.py

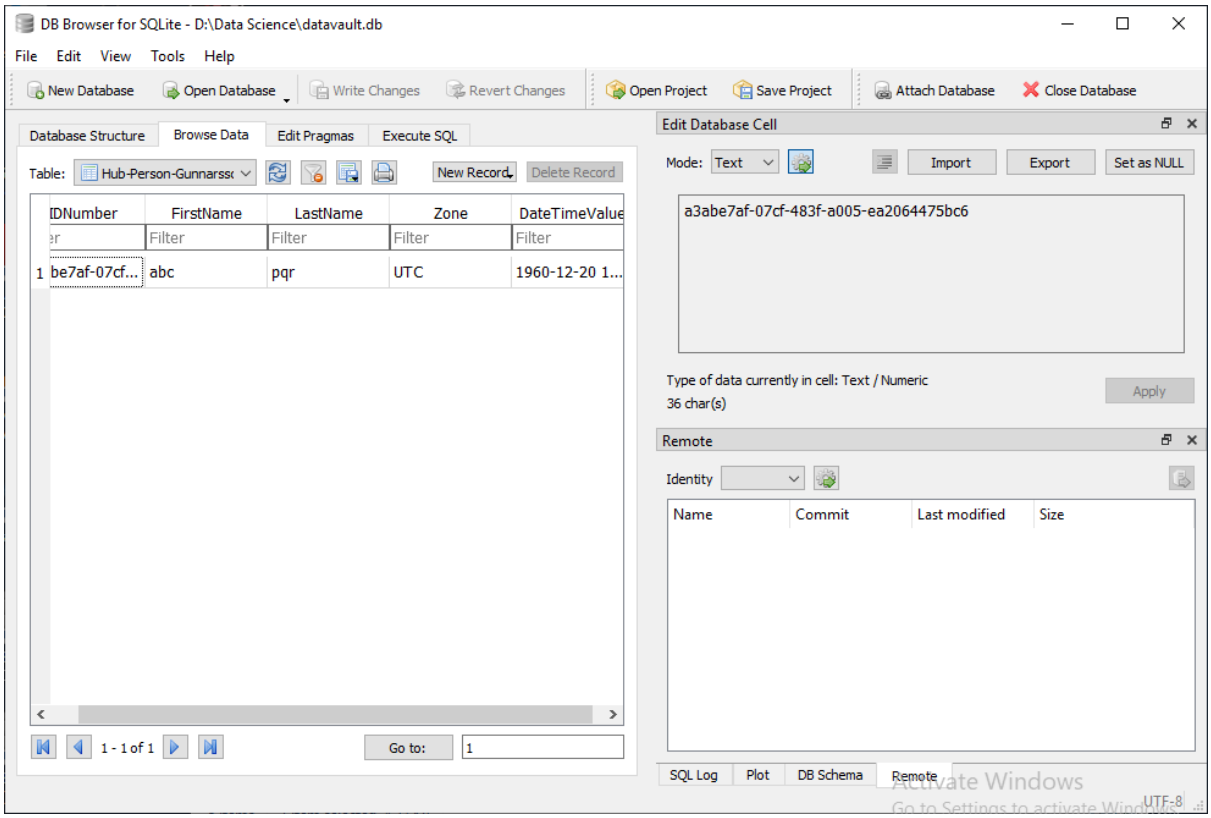
INPUT:

```
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

sDatabaseName='D:/Data Science/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
sDatabaseName='D:/Data Science/datavault.db'
conn2 = sq.connect(sDatabaseName)
sDatabaseName='D:/Data Science/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
print('\n#################################')
print('Time Category')
print('UTC Time')
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
print(BirthDateZoneUTCStr)
print('#################################')

print('Birth Date in Reykjavik :')
BirthZone = 'Atlantic/Reykjavik'
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
print(BirthDateStr)
print('#################################')
################################################################
IDZoneNumber=str(uuid.uuid4())
sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-')
TimeLine=[('ZoneBaseKey', ['UTC']),
          ('IDNumber', [IDZoneNumber]),
          ('DateTimeKey', [sDateTimeKey]),
          ('UTCDateTimeValue', [BirthDateZoneUTC]),
          ('Zone', [BirthZone]),
          ('DateTimeValue', [BirthDateStr])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
```

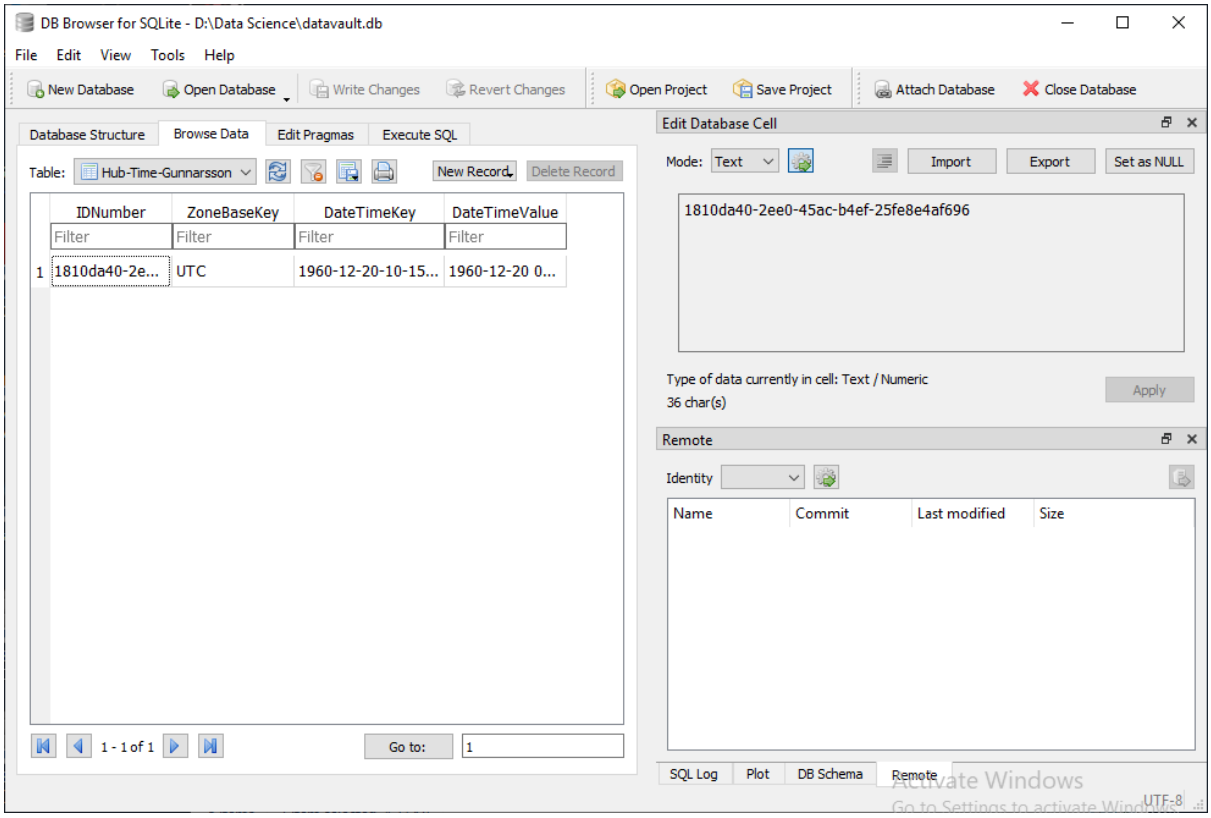
```
#################################################################
sTable = 'Hub-Time-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Time-Gunnarsson'
TimeHubIndex.to_sql(sTable, conn3, if_exists="replace")
#################################################################
TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]
TimeSatelliteIndex=TimeSatellite.set_index(['IDNumber'],inplace=False)
#################################################################
BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')
sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
TimeSatelliteIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Time-' + BirthZoneFix + '-Gunnarsson'
TimeSatelliteIndex.to_sql(sTable, conn3, if_exists="replace")
#################################################################
print('\n#################################')
print('Person Category')
FirstName = 'abc'
LastName = 'pqr'
print('Name:',FirstName,LastName)
print('Birth Date:',BirthDateLocal)
print('Birth Zone:',BirthZone)
print('UTC Birth Date:',BirthDateZoneStr)
print('#################################')
#################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('IDNumber', [IDPersonNumber]),
            ('FirstName', [FirstName]),
            ('LastName', [LastName]),
            ('Zone', ['UTC']),
            ('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
#################################################################
PersonHub=PersonFrame
PersonHubIndex=PersonHub.set_index(['IDNumber'],inplace=False)
#################################################################
sTable = 'Hub-Person-Gunnarsson'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
PersonHubIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-Person-Gunnarsson'
PersonHubIndex.to_sql(sTable, conn3, if_exists="replace")
```

OUTPUT:

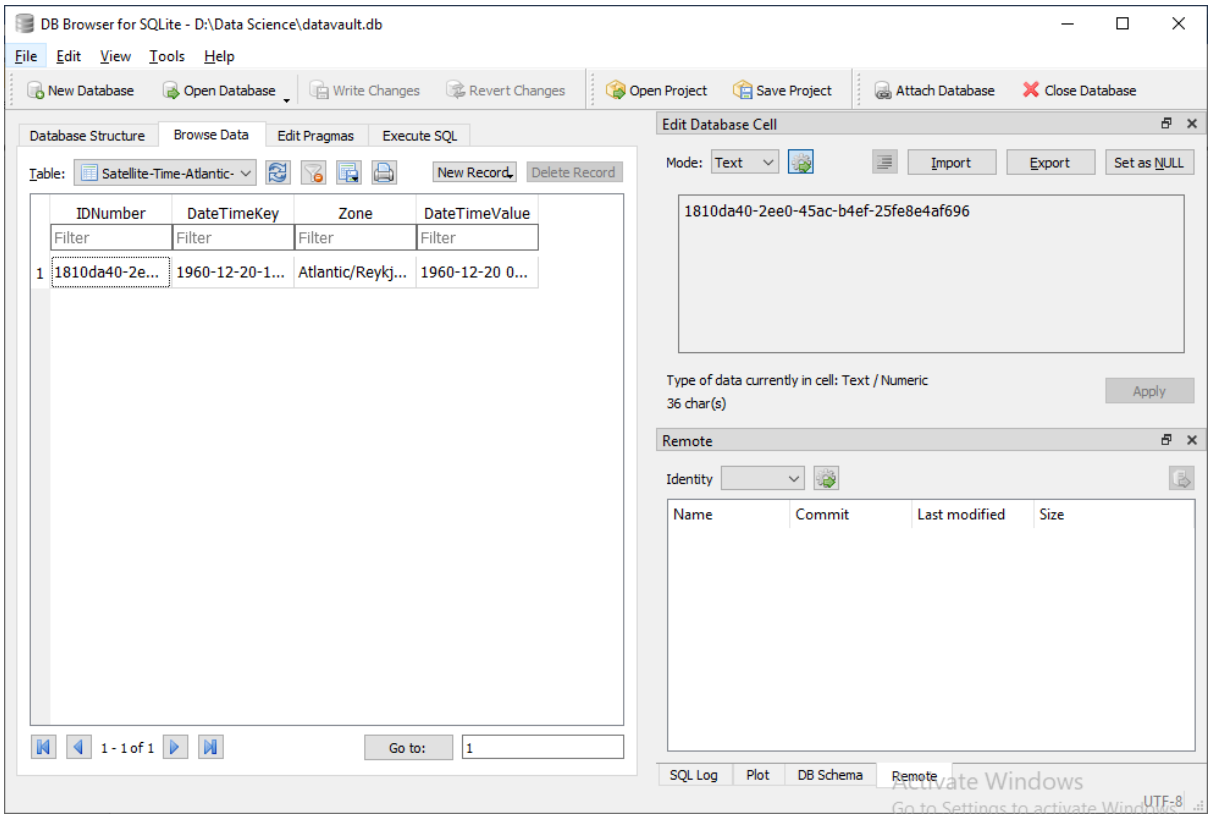
Transform Hub-Person-Gunnarsson Vermulan.db (1st)



Transform Hub-Time-Gunnarsson



Transform Hub-Person-Gunnarsson



Sun_model.py (Data Warehouse)

INPUT:

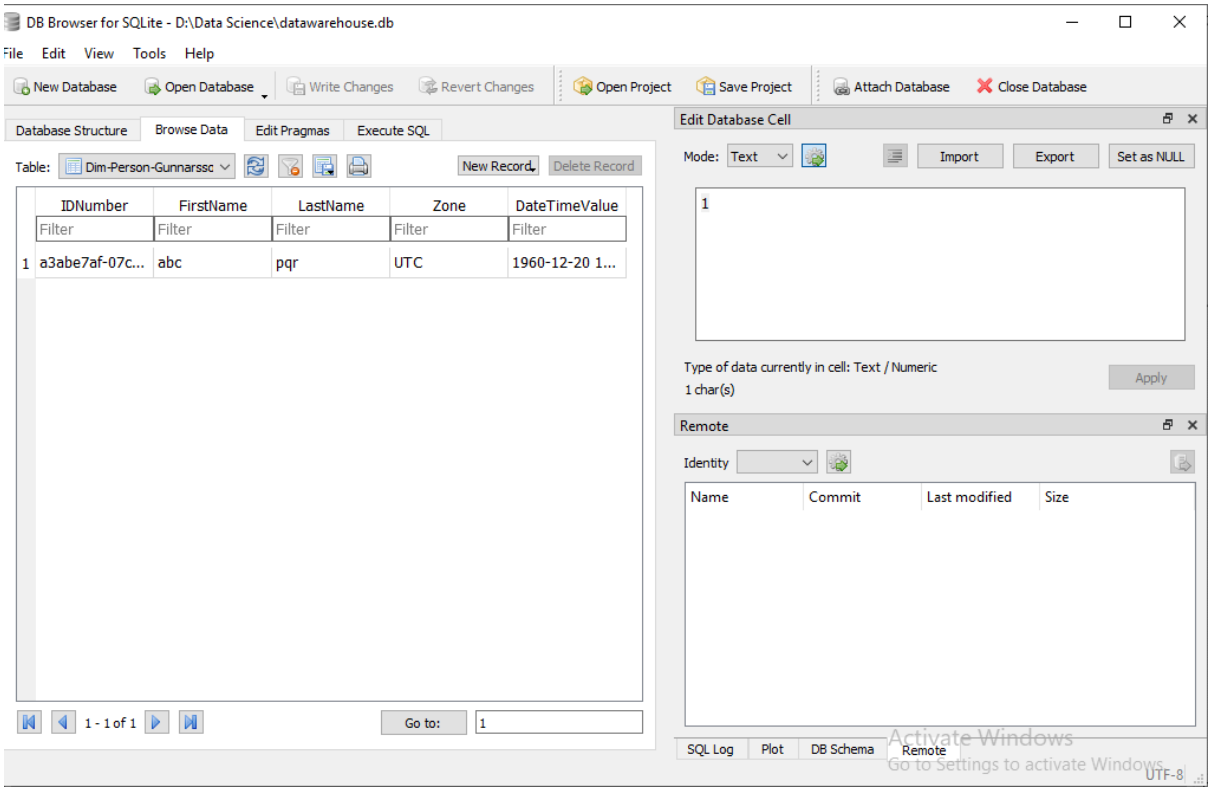
```
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid

pd.options.mode.chained_assignment = None
sDatabaseName='D:/Data Science/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
sDatabaseName='D:/Data Science/datavault.db'
conn2 = sq.connect(sDatabaseName)
sDatabaseName='D:/Data Science/datawarehouse.db'
conn3 = sq.connect(sDatabaseName)
print('\n#################################')
sSQL=" SELECT DateTimeValue FROM [Hub-Time-Gunnarsson];"
DateDataRaw=pd.read_sql_query(sSQL, conn2)
DateData=DateDataRaw.head(1000)
print(DateData)
```

```
print('Time Dimension')
print('\n###################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
    BirthZone = ('Atlantic/Reykjavik','Europe/London','UCT')
    for j in range(len(BirthZone)):
        t+=1
        print(t,mt*3)
        BirthDateZoneStr=DateData[DateTimeKey]
        BirthDateLocal=DateData[DateTimeValue]
        BirthZone='UCT'

        IDTimeNumber=str(uuid.uuid4())
        TimeLine=[('TimeID', [str(IDTimeNumber)]),
                ('UTCDate', [str(BirthDateZoneStr)]),
                ('LocalTime', [str(BirthDateLocal)]),
                ('TimeZone', [str(BirthZone)])]
        if t==1:
            TimeFrame = pd.DataFrame.from_items(TimeLine)
        else:
            TimeRow = pd.DataFrame.from_items(TimeLine)
            TimeFrame=TimeFrame.append(TimeRow)
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
sTable = 'Dim-Time'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn3, if_exists="replace")
sSQL=" SELECT " + \
     " FirstName," + \
     " SecondName," + \
     " LastName," + \
     " BirthDateKey " + \
     " FROM [Hub-Person];"
PersonDataRaw=pd.read_sql_query(sSQL, conn2)
PersonData=PersonDataRaw.head(1000)
print('\n###################################')
print('Dimension Person')
print('\n###################################')
t=0
mt=DateData.shape[0]
for i in range(mt):
    t+=1
    print(t,mt)
    FirstName = str(PersonData["FirstName"])
    SecondName = str(PersonData["SecondName"])
    if len(SecondName) > 0:
```

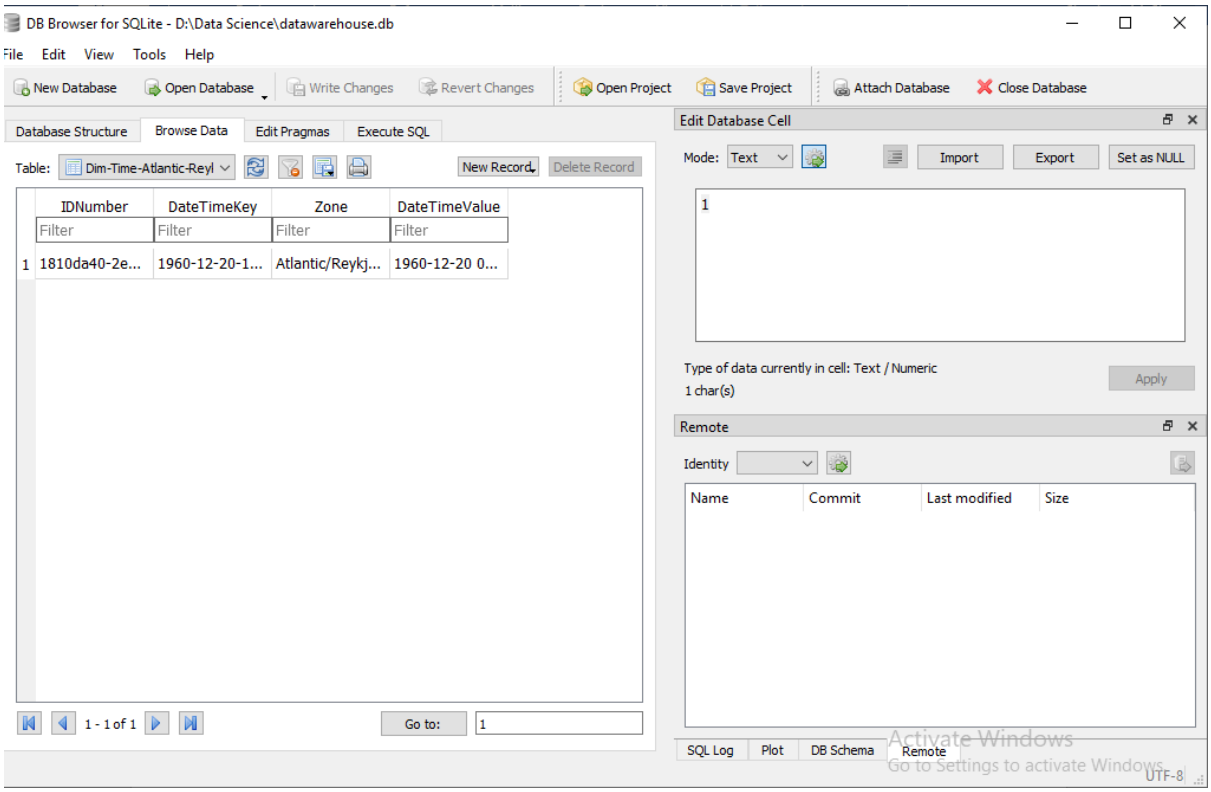
```
        SecondName=""
    LastName = str(PersonData["LastName"])
    BirthDateKey = str(PersonData["BirthDateKey"])
    IDPersonNumber=str(uuid.uuid4())
    PersonLine=[('PersonID', [str(IDPersonNumber)]),
                ('FirstName', [FirstName]),
                ('SecondName', [SecondName]),
                ('LastName', [LastName]),
                ('Zone', [str('UTC')]),
                ('BirthDate', [BirthDateKey])]
    if t==1:
        PersonFrame = pd.DataFrame.from_items(PersonLine)
    else:
        PersonRow = pd.DataFrame.from_items(PersonLine)
        PersonFrame = PersonFrame.append(PersonRow)
DimPerson=PersonFrame
print(DimPerson)
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
sTable = 'Dim-Person'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn3, if_exists="replace")
```

OUTPUT:

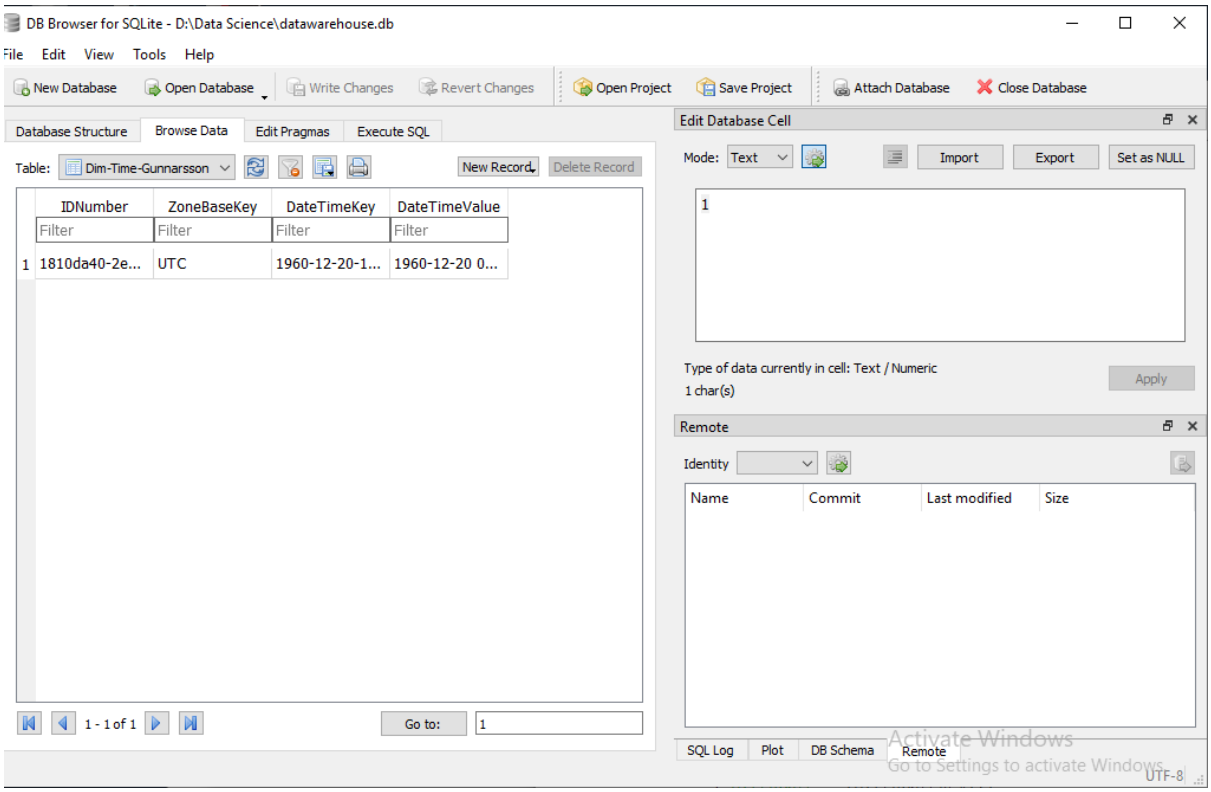
Dim-Hub-Person-Gunnarsson



Dim- Atlantic-reyl



Dim- Hub-Time-Gunnarsson



Practical No. 8

Organizing Data

ORGANISE

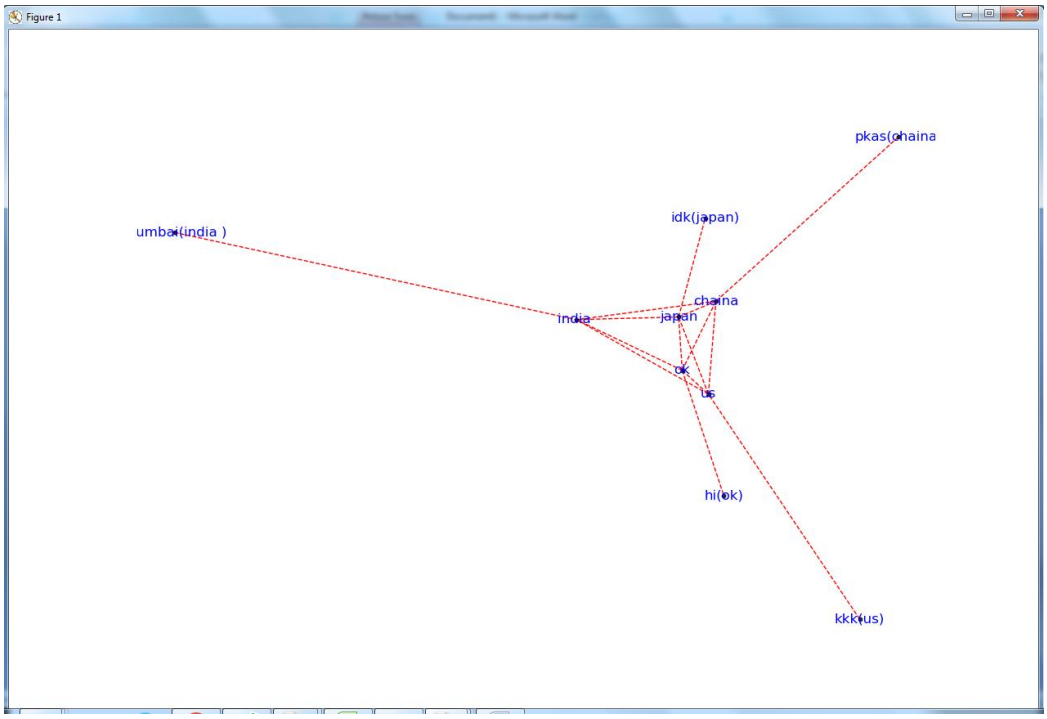
Organise-Network-Routing-Company

```
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
pd.options.mode.chained_assignment = None
sFileName='C:\\Users\\Administrator\\Downloads\\practical-data-science-master\\practical-
data-science-master\\VKHCG\\01-Vermeulen\\02-Assess\\01-EDS\\02-Python\\Assess-
Network-Routing-Company.csv'
print('Loading :',sFileName)
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print(CompanyData.head())
print(CompanyData.shape)
G=nx.Graph()
for i in range(CompanyData.shape[0]):
    for j in range(CompanyData.shape[0]):
        Node0=CompanyData['Company_Country_Name'][i]
        Node1=CompanyData['Company_Country_Name'][j]
        if Node0 != Node1:
            G.add_edge(Node0,Node1)
for i in range(CompanyData.shape[0]):
    Node0=CompanyData['Company_Country_Name'][i]
    Node1=CompanyData['Company_Place_Name'][i] + '('+
CompanyData['Company_Country_Name'][i] + ')'
    if Node0 != Node1:
        G.add_edge(Node0,Node1)
print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
sFileName='C:\\Users\\Administrator\\Downloads\\practical-data-science-master\\practical-
data-science-master\\VKHCG\\01-Vermeulen\\02-Assess\\01-EDS\\02-Python\\Assess-
Network-Routing-Company.csv'
print('Storing :',sFileName)
#nx.write_gml(G, sFileName)
sFileName='C:\\Users\\Administrator\\Downloads\\practical-data-science-master\\practical-
data-science-master\\VKHCG\\01-Vermeulen\\02-Assess\\01-EDS\\02-Python\\Assess-
Network-Routing-Company.csv'
print('Storing Graph Image:',sFileName)
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('### Done!! ######################')
```

OUTPUT

```
################################
('Loading :', 'C:\\Users\\Administrator\\Downloa
ork-Routing-Company.csv')
################################
################################
  Company_Country_Name Company_Place_Name
0               india              mumbai
1               japan                 idk
2                  us                 kkk
3              chaina                pkas
4                  ok                  hi
(5, 2)
('Nodes:', 10)
('Edges:', 15)
################################
('Storing :', 'C:\\Users\\Administrator\\Downloa
ork-Routing-Company.csv')
################################
################################
('Storing Graph Image:', 'C:\\Users\\Administrat
Assess-Network-Routing-Company.csv')
################################
```

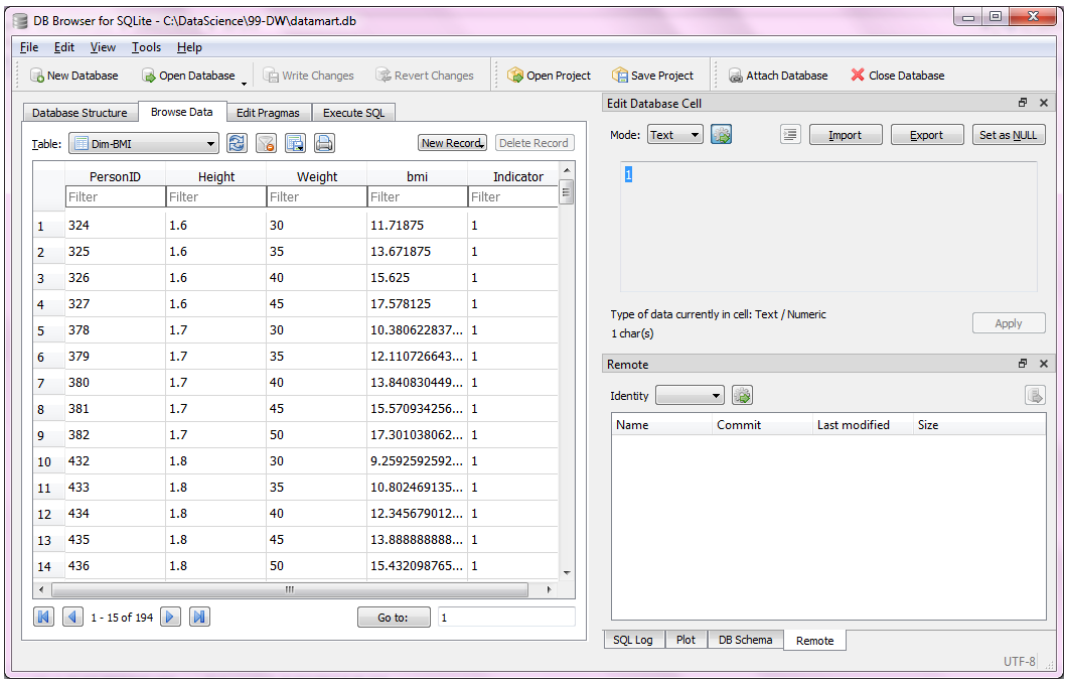


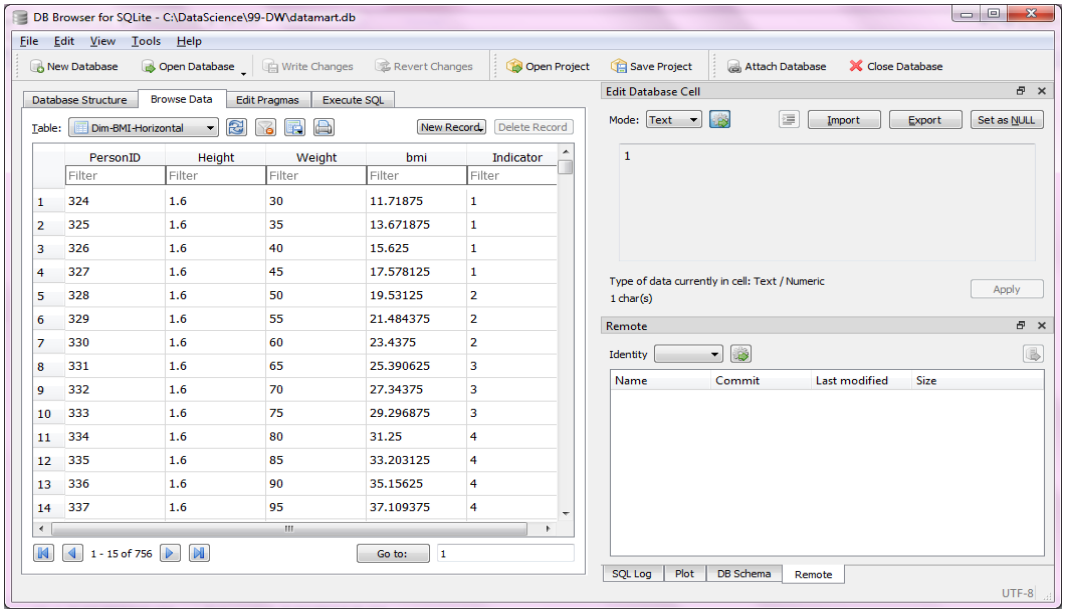
Organize horizontal

```
import sys
import os
import pandas as pd
import sqlite3 as sq
sDatabaseName='C:/Users/student/Downloads/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
sDatabaseName= 'C:/Users/student/Downloads/datamart.db'
conn2 = sq.connect(sDatabaseName)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
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-Horizontal'
print('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI-Horizontal'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Horizontal];"
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:

```
#################
('Loading :', 'C:/Users/student/Downloads/datamart.db', ' Table:', 'Dim-BMI')
#################################
('Loading :', 'C:/Users/student/Downloads/datamart.db', ' Table:', 'Dim-BMI')
#################################

#################################
('Storing :', 'C:/Users/student/Downloads/datamart.db', '\n Table:', 'Dim-BMI-Horizontal')

#################################
#################################
('Loading :', 'C:/Users/student/Downloads/datamart.db', ' Table:', 'Dim-BMI-Horizontal')
#################################
#################################
('Full Data Set (Rows):', 1080)
('Full Data Set (Columns):', 5)
#################################
('Horizontal Data Set (Rows):', 194)
('Horizontal Data Set (Columns):', 5)
#################################
```



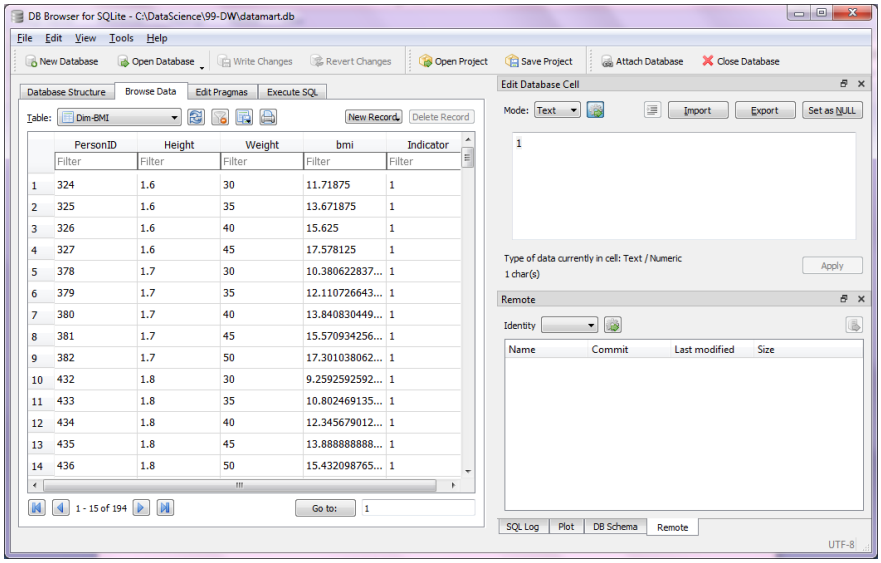


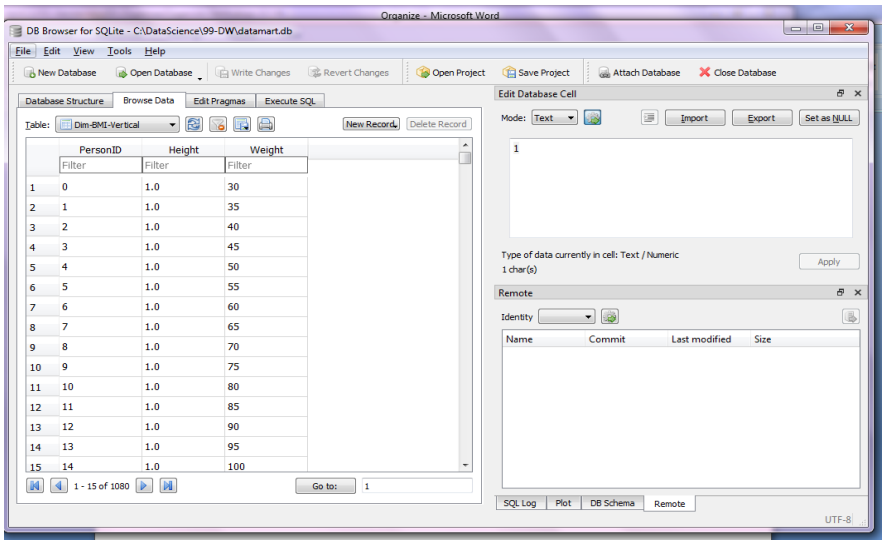
Organize vertical

```
import sys
import os
import pandas as pd
import sqlite3 as sq
sDatabaseName='C:/Users/student/Downloads/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
sDatabaseName='C:/Users/student/Downloads/datamart.db'
conn2 = sq.connect(sDatabaseName)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
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('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
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:

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Vertical.py ======
('Loading :', 'C:/Users/student/Downloads/datamart.db', ' Table:', 'Dim-BMI')
################################
('Loading :', 'C:/Users/student/Downloads/datamart.db', ' Table:', 'Dim-BMI')
################################

################################
('Storing :', 'C:/Users/student/Downloads/datamart.db', '\n Table:', 'Dim-BMI-Vertical')

################################
################
('Loading :', 'C:/Users/student/Downloads/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)
################################
>>>
```





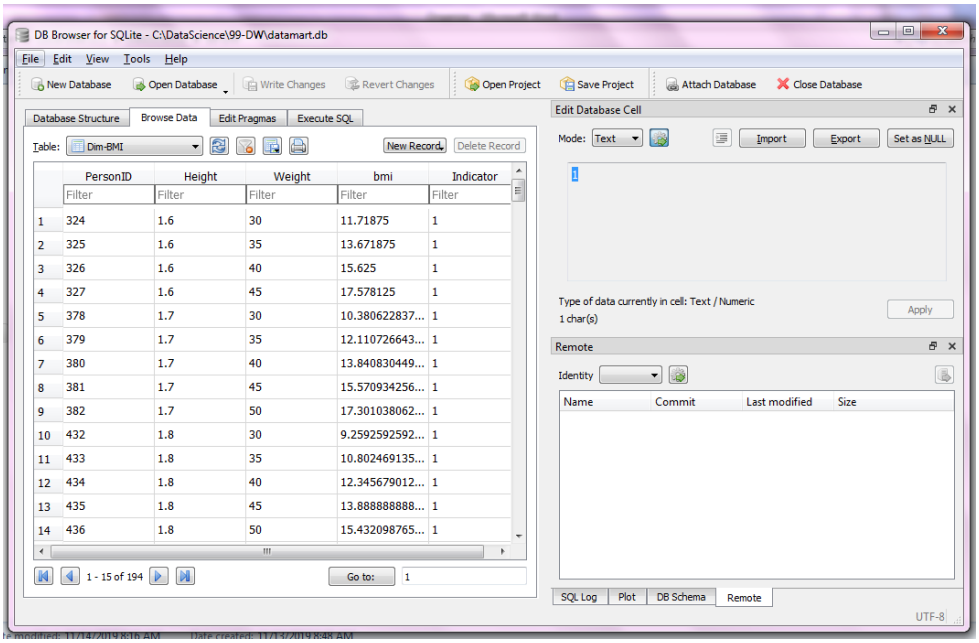
Organize Island

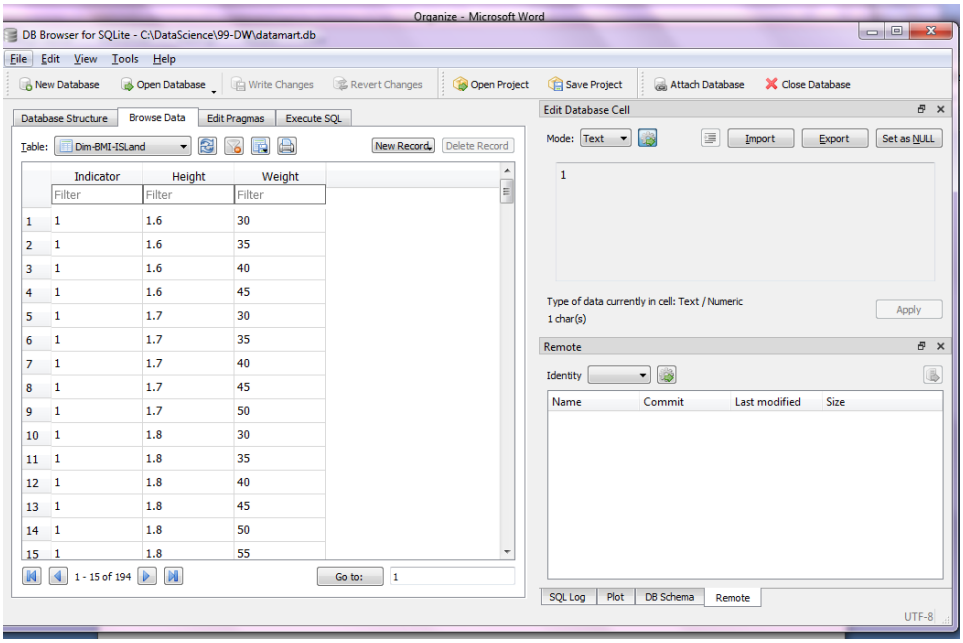
```
import sys
import os
import pandas as pd
import sqlite3 as sq
sDatabaseName='C:/Users/student/Downloads/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####################################################################
sDatabaseName='C:/Users/student/Downloads/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####################################################################
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])
```

OUTPUT:

```
>>>
======= RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Island.py =======
################
('Loading :', 'C:/Users/student/Downloads/datamart.db', ' Table:', 'Dim-BMI')
################
('Loading :', 'C:/Users/student/Downloads/datamart.db', ' Table:', 'Dim-BMI')

################################
('Storing :', 'C:/Users/student/Downloads/datamart.db', '\n Table:', 'Dim-BMI-Vertical')

################################
################################
('Loading :', 'C:/Users/student/Downloads/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)
################################
>>>
```



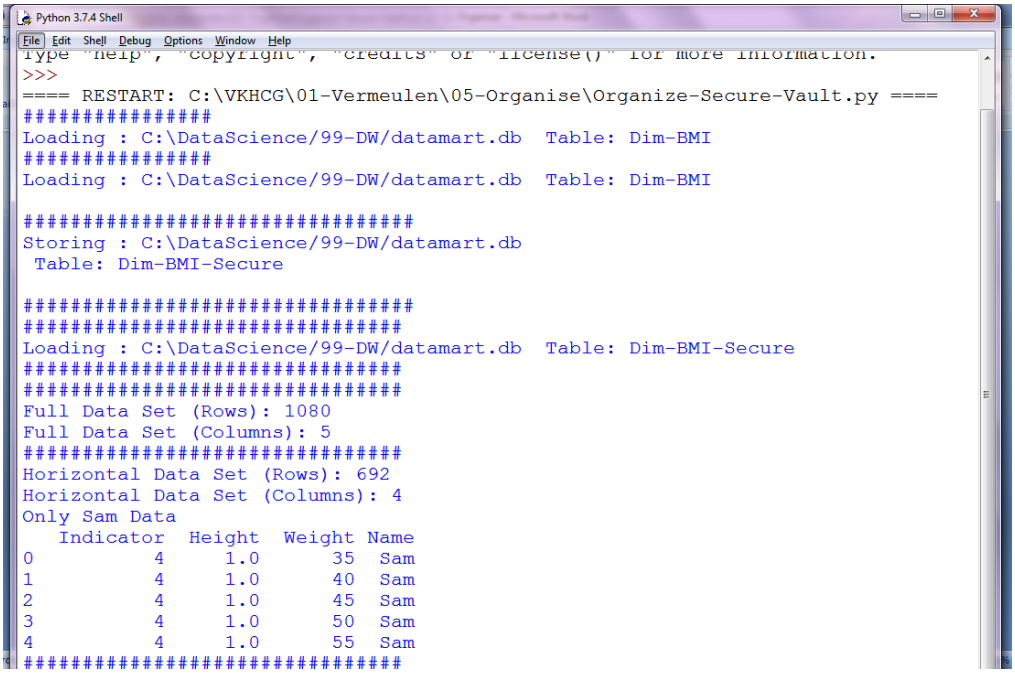


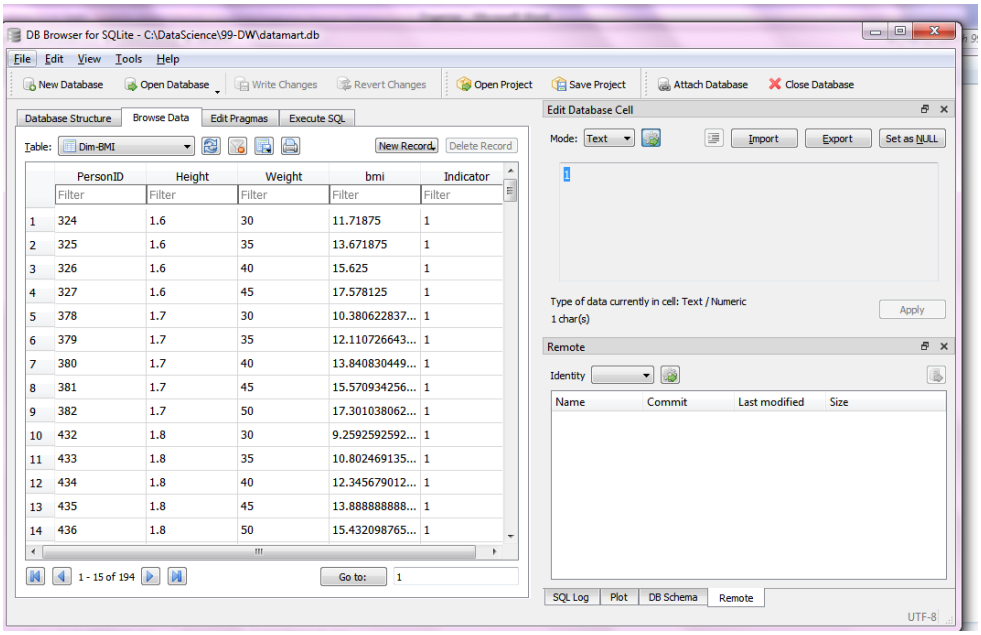
Secure-Vault:-

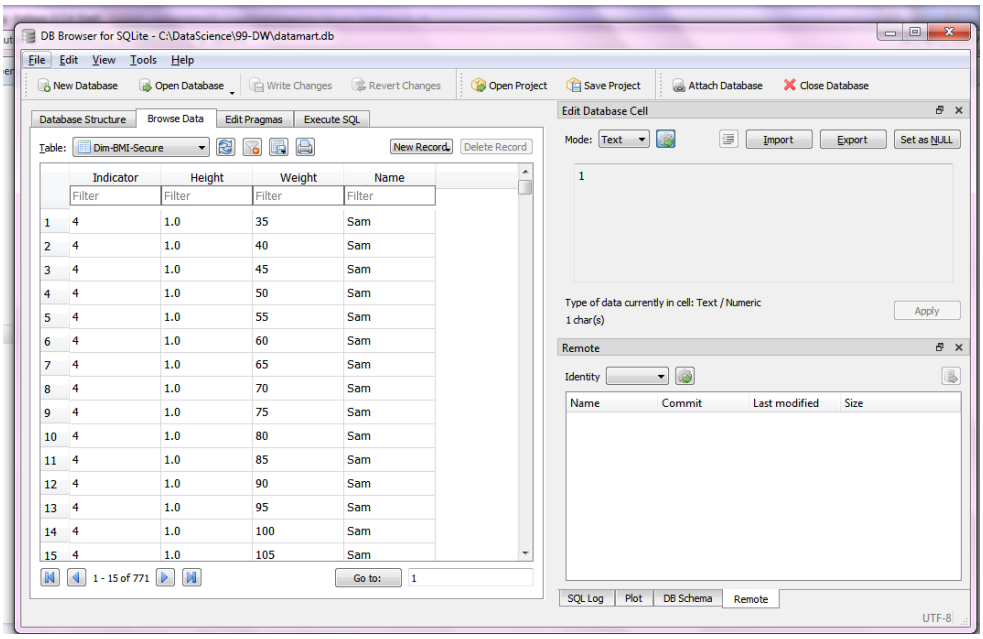
```
# -*- coding: utf-8 -*-
import sys
import os
import pandas as pd
import sqlite3 as sq
sDataWarehouseDir='C:\DataScience/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
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('Storing :',sDatabaseName,'\n Table:',sTable)
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
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])
print('Only Sam Data')
print(PersonFrame2.head())
```

OUTPUT:




DB Browser for SQLite - C:\DataScience\99-DW\datamart.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: Dim-BMI
New Record Delete Record
PersonID Height Weight bmi Indicator
1 324 1.6 30 11.71875 1
2 325 1.6 35 13.671875 1
3 326 1.6 40 15.625 1
4 327 1.6 45 17.578125 1
5 378 1.7 30 10.380622837... 1
6 379 1.7 35 12.110726643... 1
7 380 1.7 40 13.840830449... 1
8 381 1.7 45 15.570934256... 1
9 382 1.7 50 17.301038062... 1
10 432 1.8 30 9.2592592592... 1
11 433 1.8 35 10.802469135... 1
12 434 1.8 40 12.345679012... 1
13 435 1.8 45 13.888888888... 1
14 436 1.8 50 15.432098765... 1
1 - 15 of 194
Go to: 1
Edit Database Cell
Mode: Text Import Export Set as NULL
1
Type of data currently in cell: Text / Numeric
1 char(s)
Apply
Remote
Identity
Name Commit Last modified Size
SQL Log Plot DB Schema Remote
UTF-8


DB Browser for SQLite - C:\DataScience\99-DW\datamart.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: Dim-BMI-Secure
New Record Delete Record
Indicator Height Weight Name
1 4 1.0 35 Sam
2 4 1.0 40 Sam
3 4 1.0 45 Sam
4 4 1.0 50 Sam
5 4 1.0 55 Sam
6 4 1.0 60 Sam
7 4 1.0 65 Sam
8 4 1.0 70 Sam
9 4 1.0 75 Sam
10 4 1.0 80 Sam
11 4 1.0 85 Sam
12 4 1.0 90 Sam
13 4 1.0 95 Sam
14 4 1.0 100 Sam
15 4 1.0 105 Sam
1 - 15 of 771
Go to: 1
Edit Database Cell
Mode: Text Import Export Set as NULL
1
Type of data currently in cell: Text / Numeric
1 char(s)
Apply
Remote
Identity
Name Commit Last modified Size
SQL Log Plot DB Schema Remote
UTF-8

Practical No. 9

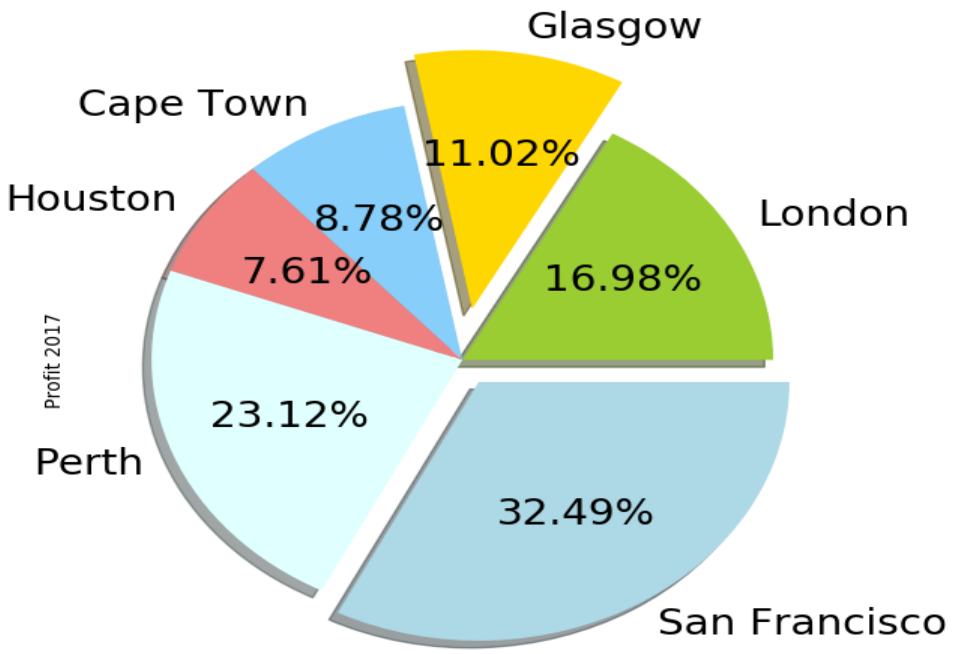
Generating Reports

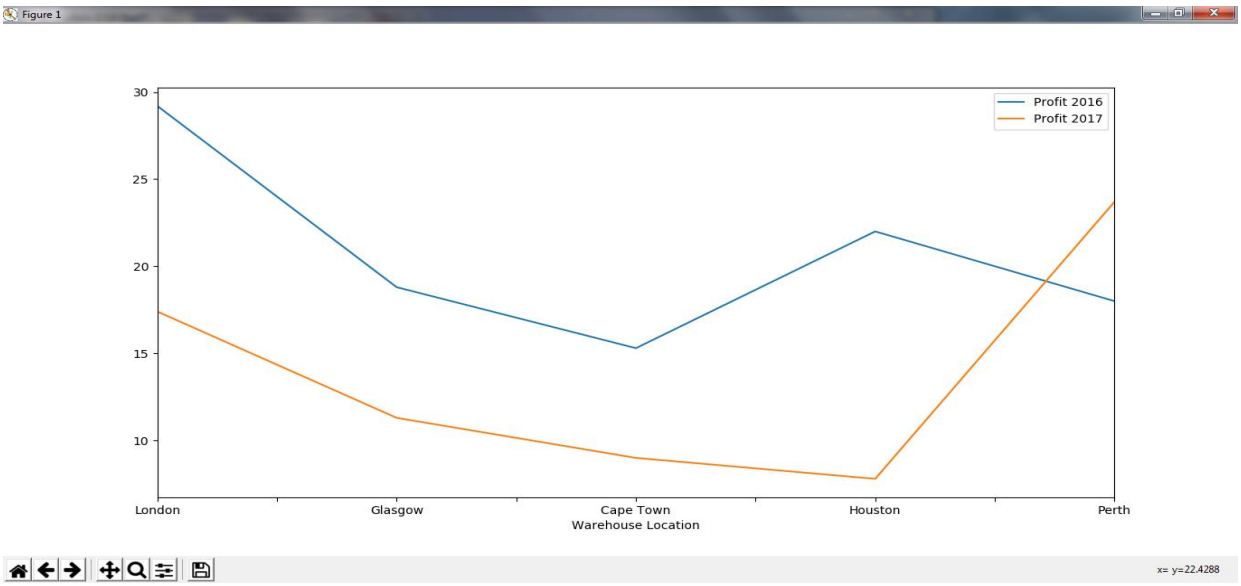
INPUT:

```
import sys
import os
import pandas as pd
import matplotlib as ml
from matplotlib import pyplot as plt
data=[
['London',       29.2, 17.4],
['Glasgow',      18.8, 11.3],
['Cape Town',  15.3, 9.0],
['Houston',      22.0, 7.8],
['Perth',          18.0, 23.7],
['San Francisco',         11.4, 33.3]
]
os_new=pd.DataFrame(data)
pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'Conversion Percent',
       'URL', 'Stats URL'],
      dtype='object')
os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)
os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)
os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

explode = (0, 0.2, 0, 0, 0, 0.1)
labels=os_new['Warehouse Location']
colors_mine = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral', 'lightcyan','lightblue']
os_new.plot(figsize=(10, 10),kind="pie", y="Profit 2017",autopct='%.2f%%', \
           shadow=True, explode=explode, legend = False, colors = colors_mine,\
           labels=labels, fontsize=20)
sPicNameOut1='D:/DataScience/pie_explode.png'
plt.savefig(sPicNameOut1,dpi=600)
plt.show()
os_new.iloc[:5].plot(figsize=(10, 10),kind='line',x='Warehouse Location',\
          y=['Profit 2016','Profit 2017']);
plt.show()
os_new.iloc[:5].plot(figsize=(10, 10),kind='bar',x='Warehouse Location',\
          y=['Profit 2016','Profit 2017']);
plt.show()
os_new.iloc[:5].plot(figsize=(10, 10),kind='barh',x='Warehouse Location',\
          y=['Profit 2016','Profit 2017']);
plt.show()
os_new.iloc[:5].plot(figsize=(10, 10),kind='area',x='Warehouse Location',\
          y=['Profit 2016','Profit 2017'],stacked=False);
plt.show()
os_new.iloc[:5].plot(figsize=(10, 10),kind='scatter',x='Profit 2016',\
          y='Profit 2017',color='DarkBlue',marker='D');
```

```
plt.show()
os_new.iloc[:5].plot(figsize=(13, 10),kind='hexbin',x='Profit 2016',\
        y='Profit 2017', gridsize=25);
plt.show()
```

OUTPUT:





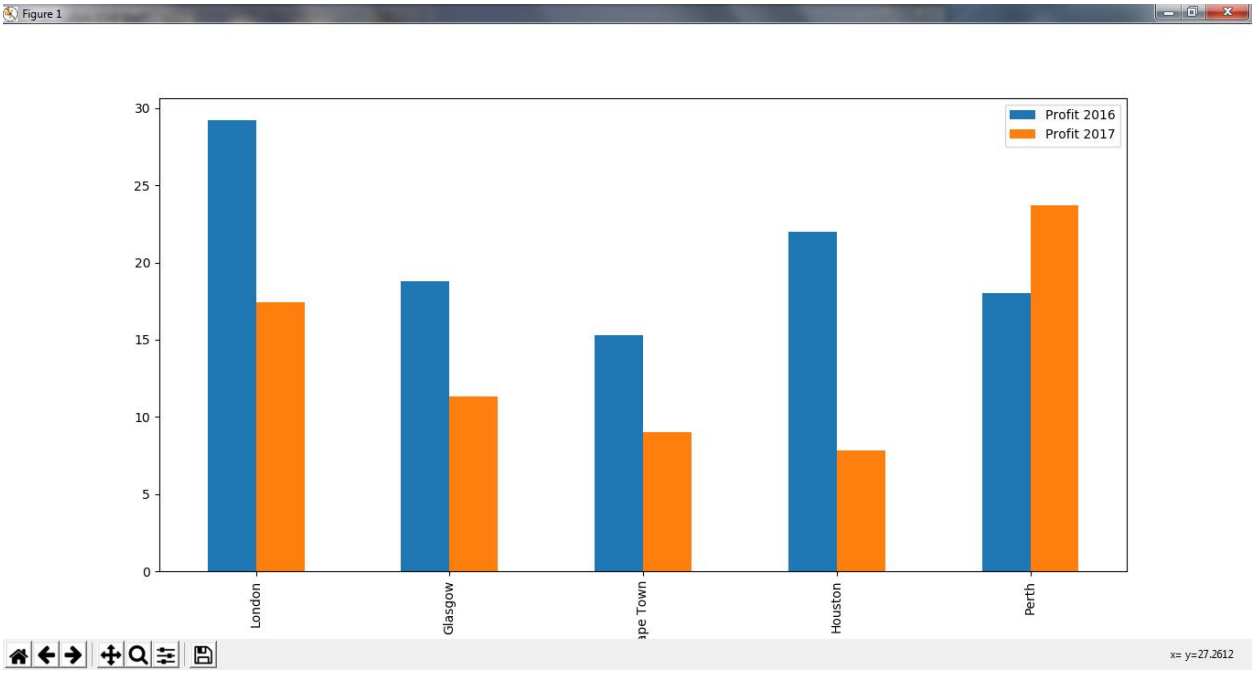


Figure 1
Profit 2016
Profit 2017
30
25
20
15
10
5
0
London
Glasgow
Houston
Perth
x= y=27.2612

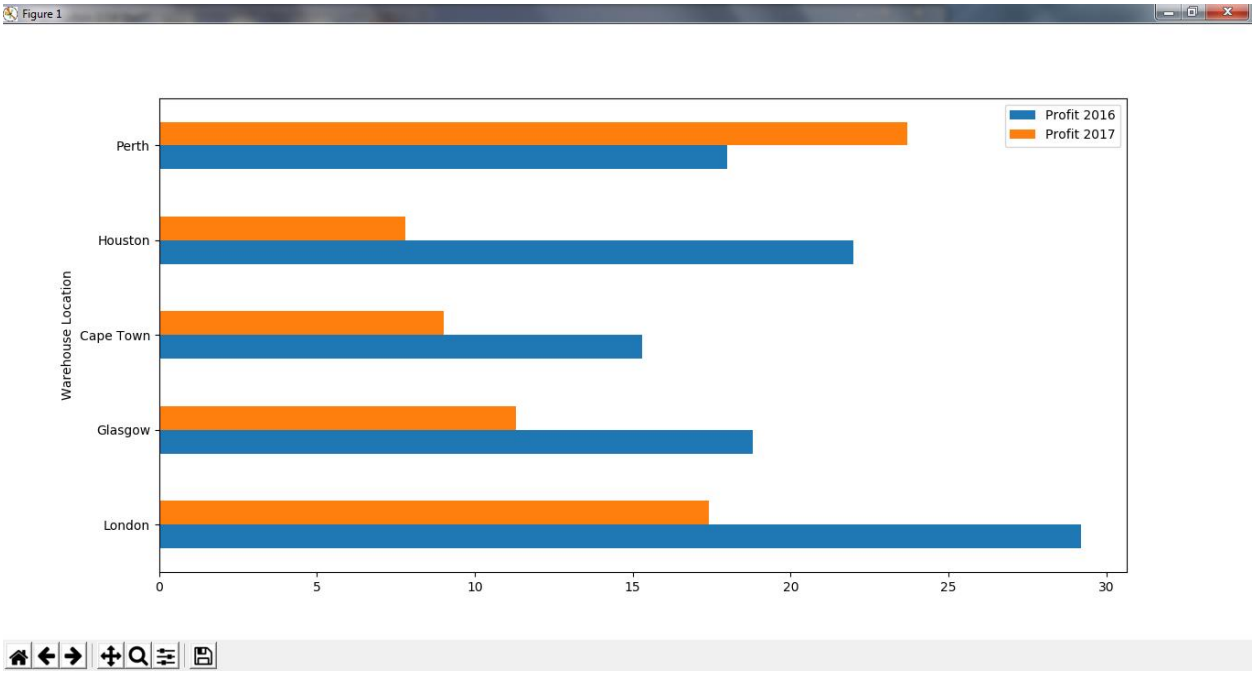


Figure 1
Profit 2016
Profit 2017
Perth
Houston
Cape Town
Glasgow
London
Warehouse Location
0
5
10
15
20
25
30

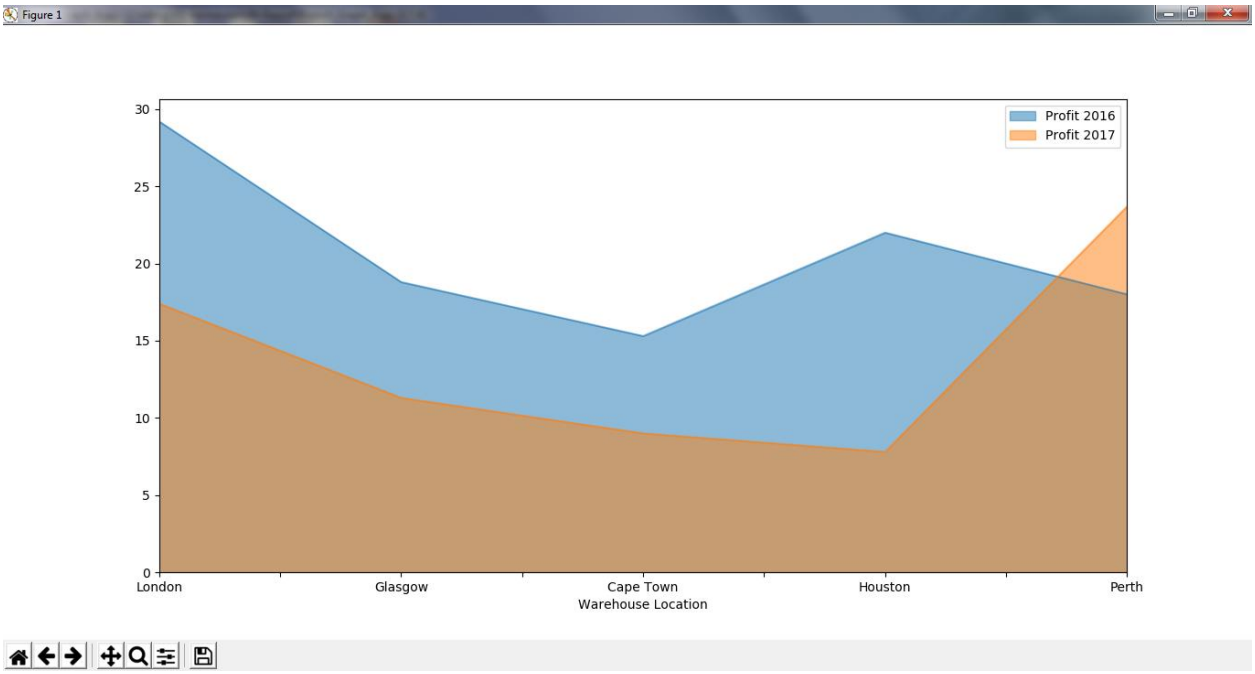


Figure 1
Profit 2016
Profit 2017
30
25
20
15
10
5
0
London
Glasgow
Cape Town
Houston
Perth
Warehouse Location

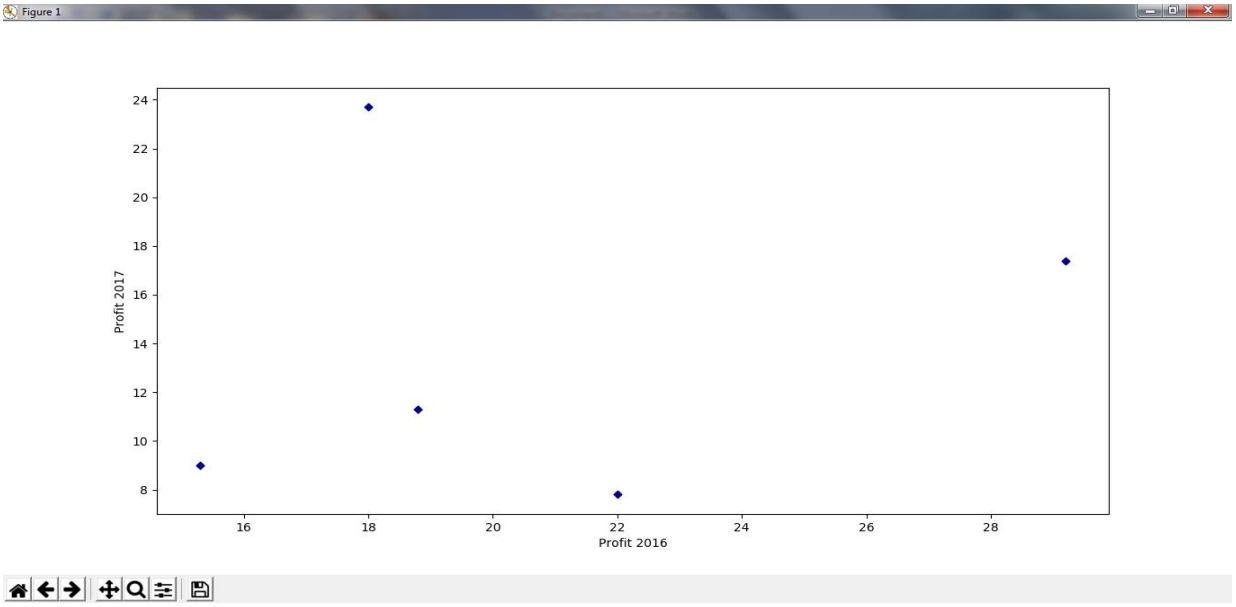


Figure 1
24
22
20
18
16
14
12
10
8
Profit 2017
16
18
20
22
24
26
28
Profit 2016

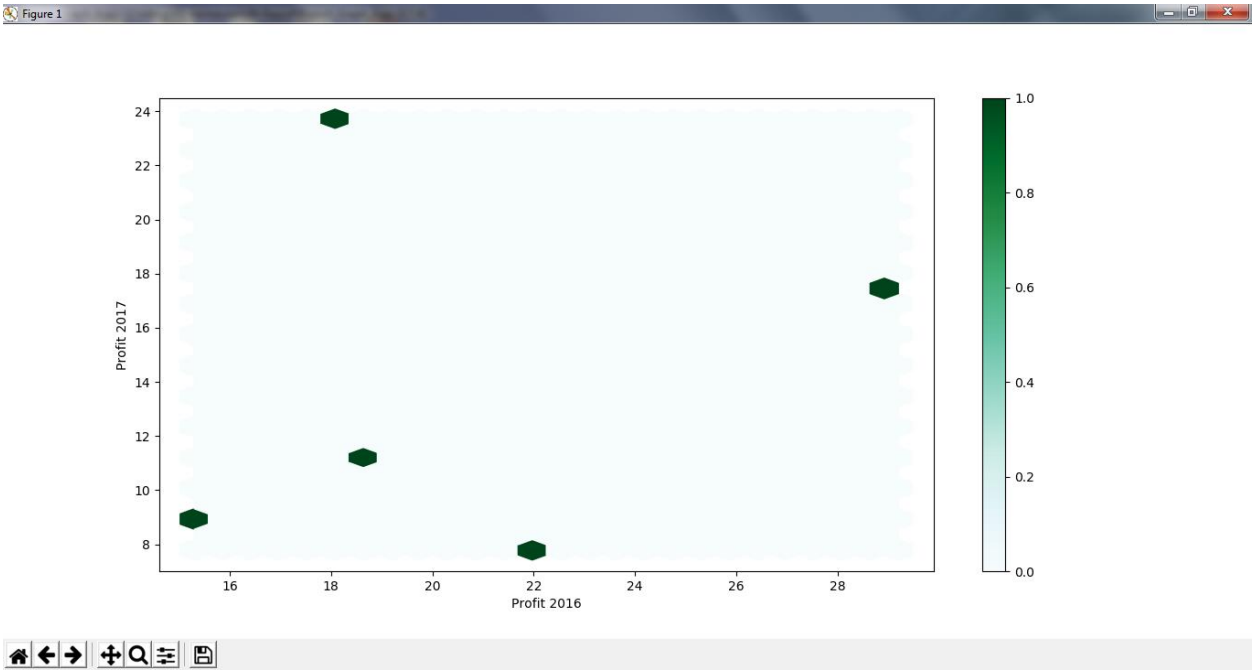


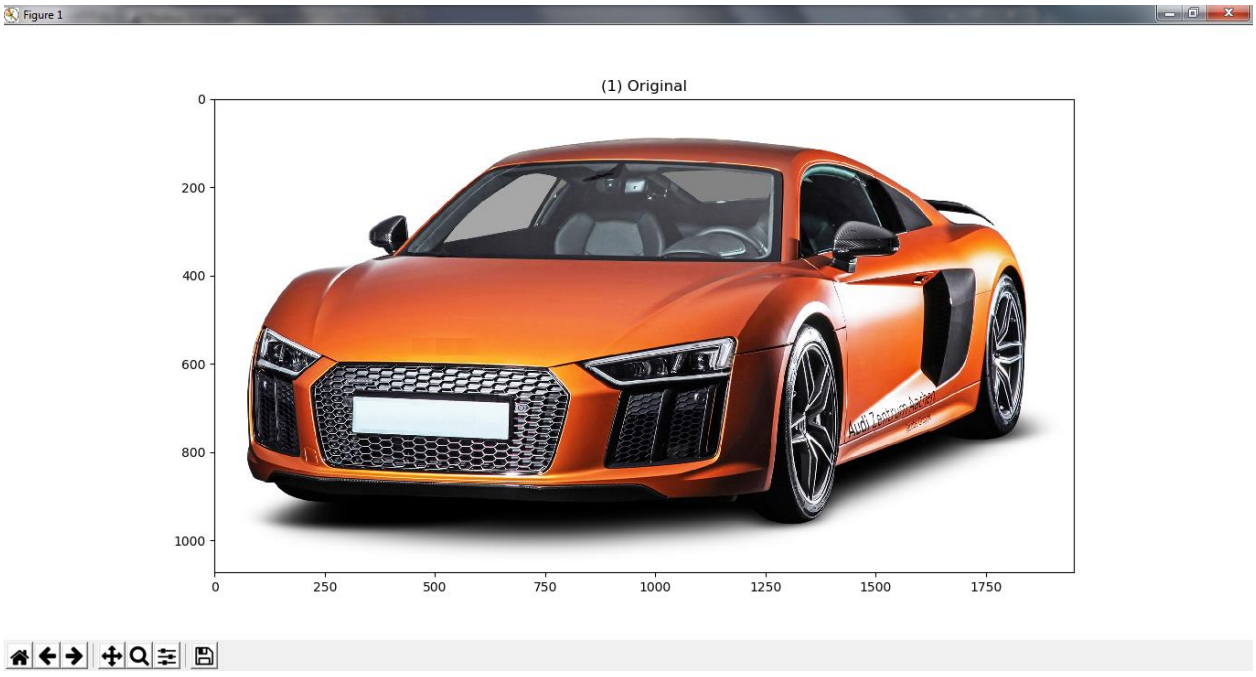
Figure 1
Profit 2017
Profit 2016
24
22
20
18
16
14
12
10
8
16
18
20
22
24
26
28
1.0
0.8
0.6
0.4
0.2
0.0

INPUT:

```
import sys
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
sPicName='C:/vkhcg/01-Vermeulen/00-RawData/AudiR8.png'
t=0
img=mpimg.imread(sPicName)
print('Size:', img.shape)
plt.figure(figsize=(10, 10))
t+=1
sTitle= '(' + str(t) + ') Original'
plt.title(sTitle)
plt.imshow(img)
plt.show()
for c in range(img.shape[2]):
    t+=1
        plt.figure(figsize=(10, 10))
    sTitle= '(' + str(t) + ') Channel: ' + str(c)
    plt.title(sTitle)
    lum_img = img[:,:,c]
    plt.imshow(lum_img)
    plt.show()
```

OUTPUT:

Size: (1073, 1950, 4)



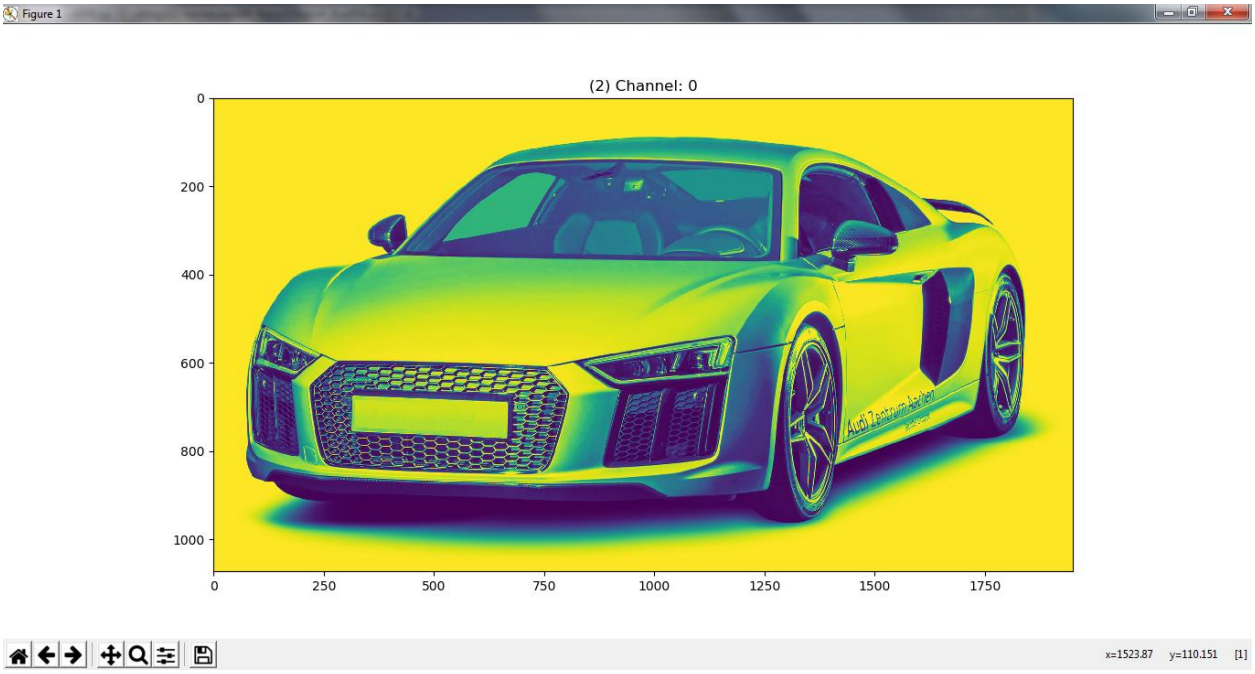


Figure 1
(2) Channel: 0
0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750
x=1523.87 y=110.151 [1]

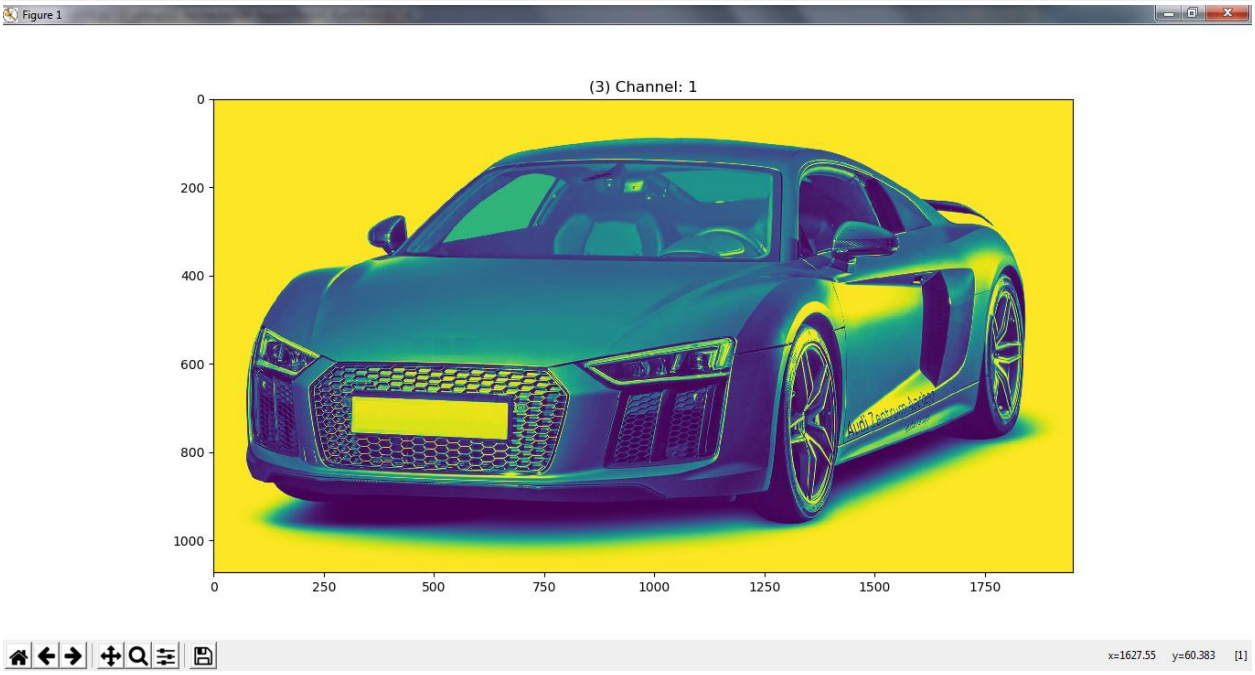


Figure 1
(3) Channel: 1
0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750
x=1627.55 y=60.383 [1]

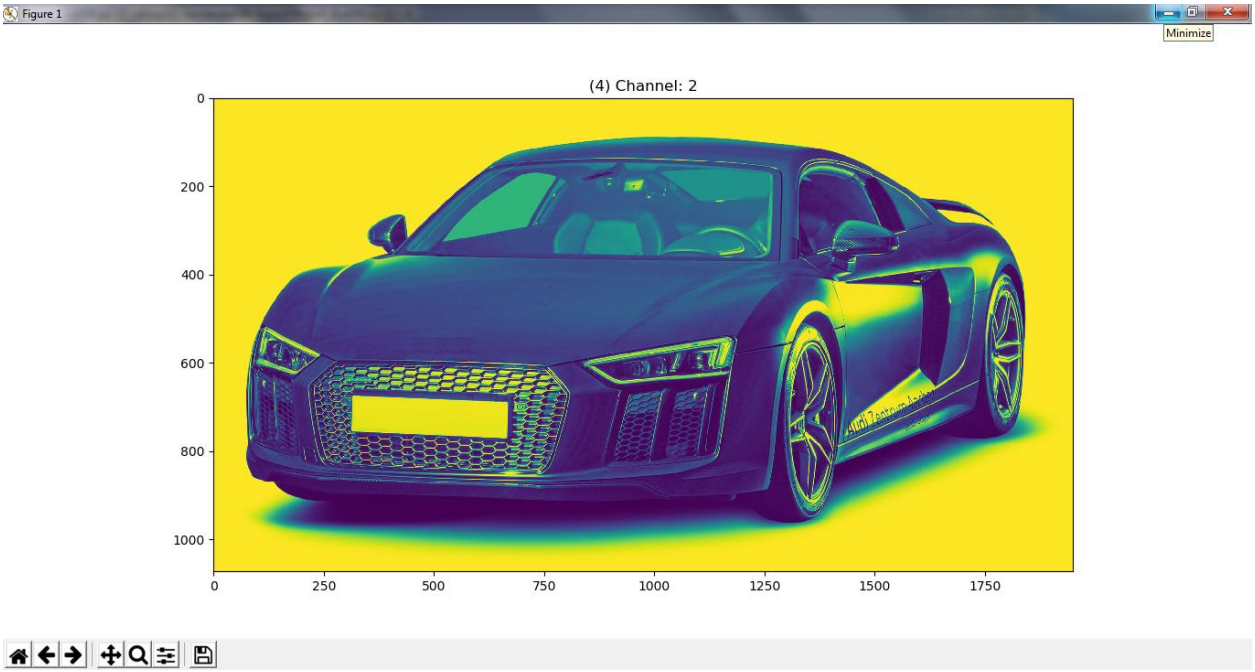


Figure 1
Minimize
(4) Channel: 2
0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750

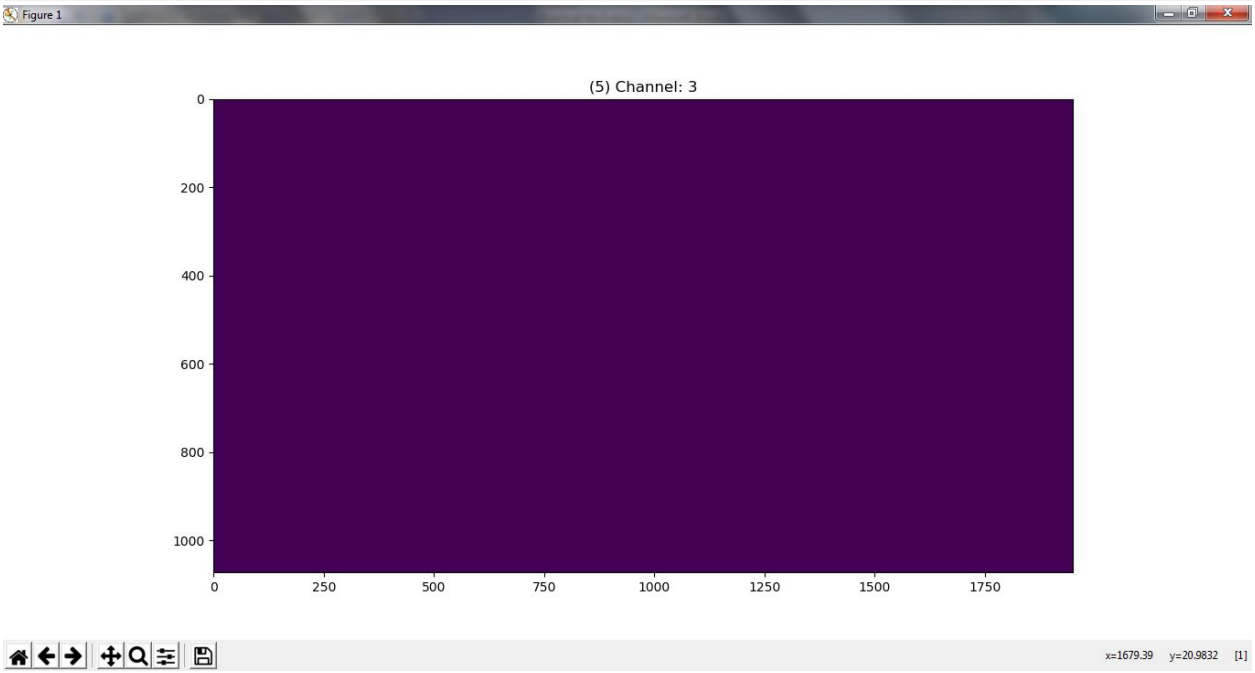
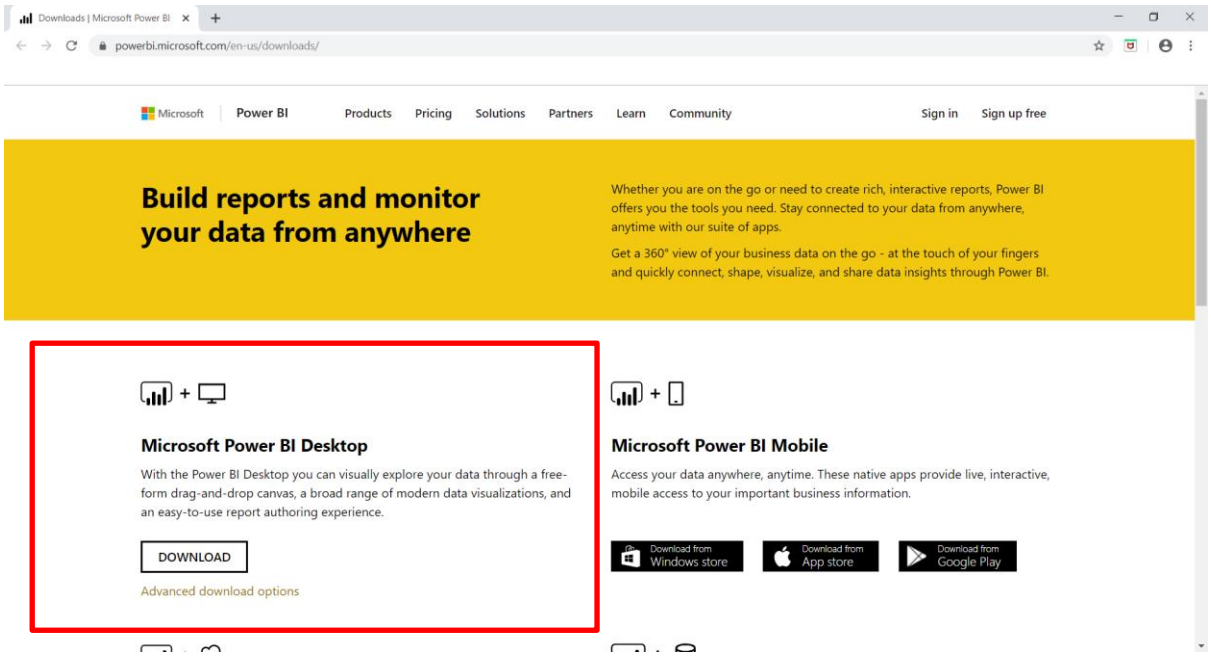


Figure 1
(5) Channel: 3
0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750
x=1679.39
y=20.9832
[1]

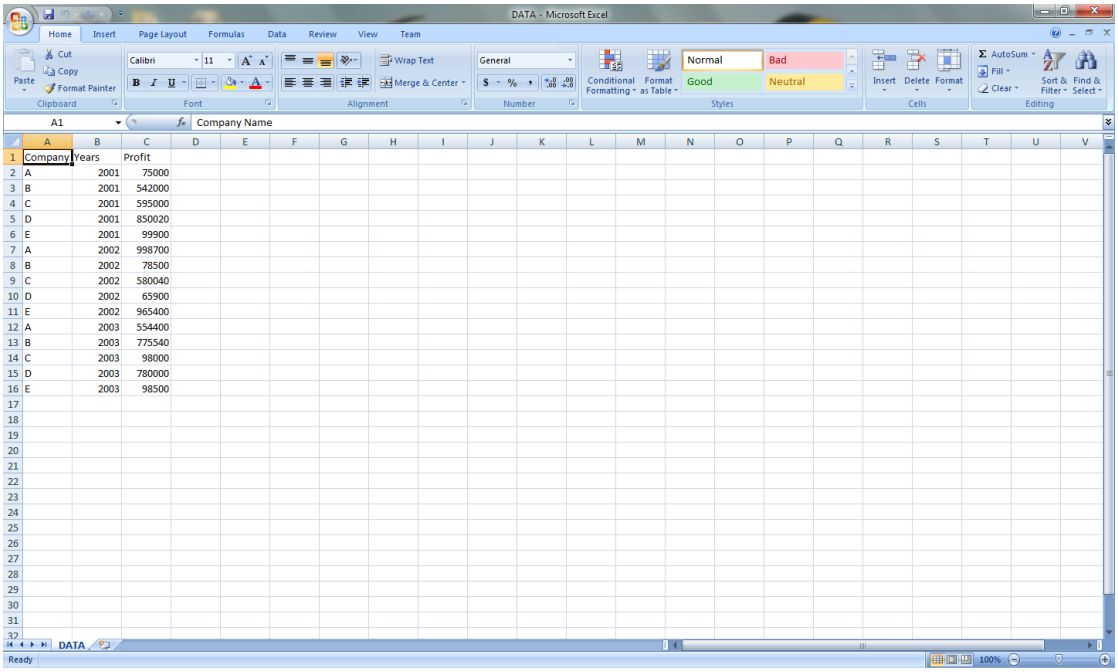
Practical No. 10

Data Visualization with Power BI

Step 1: Install and Open Power BI.

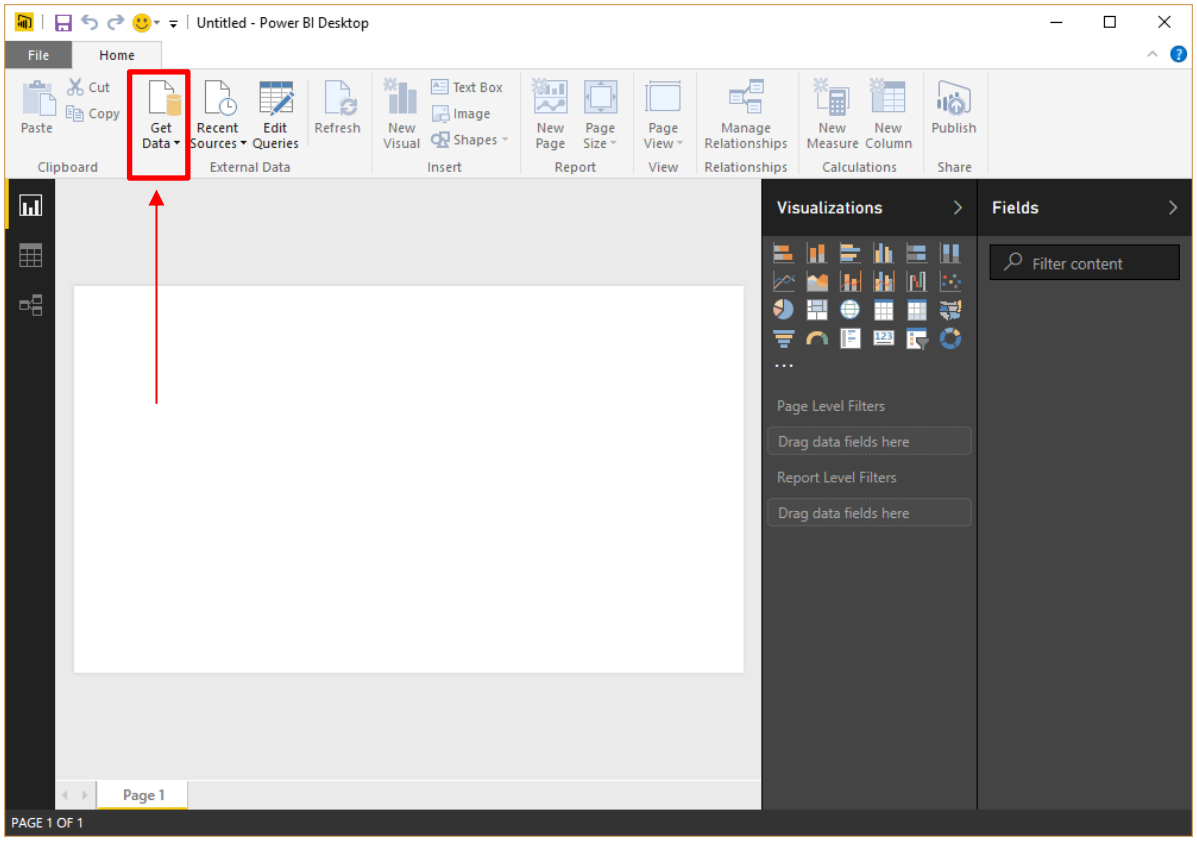


Step 2:Create .xlsx/.csv file like following

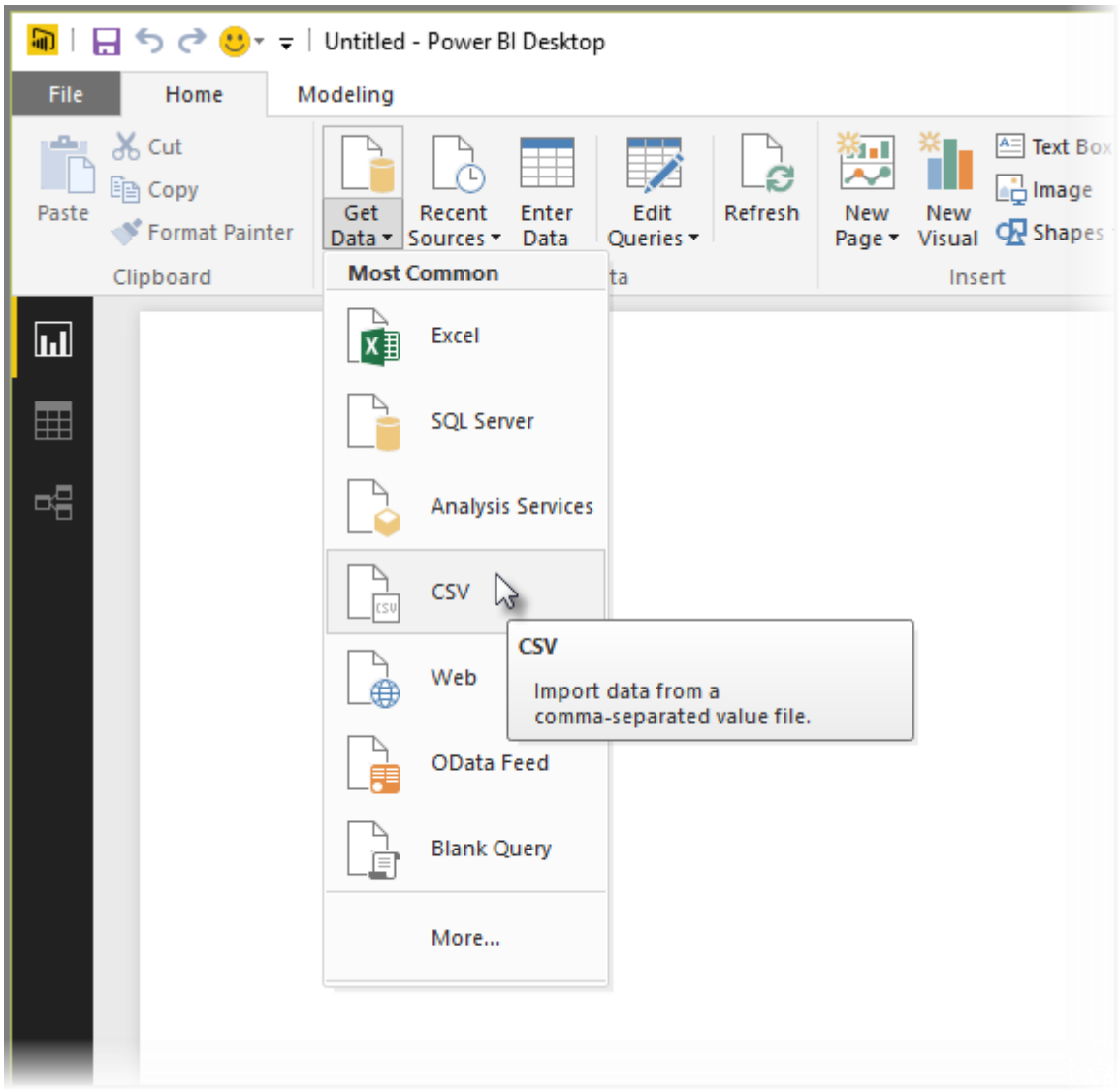


Company	Years	Profit
A	2001	75000
B	2001	542000
C	2001	595000
D	2001	850020
E	2001	99900
A	2002	998700
B	2002	78500
C	2002	580040
D	2002	65900
E	2002	965400
A	2003	554400
B	2003	775540
C	2003	98000
D	2003	780000
E	2003	98500

Step 3: Click on Get data in Power BI application



Step 4: And select the xlsx/csv file which was created then load it



Step5: On Axis select Year and Company name, On legend select company name, On value select profit.

OUTPUT: