INDEX

Sr. No.	Practical name	Date	Sign
1	Creating and using database in Cassandra		
2A	Text Delimited CSV to HORUS format		
2B	XML to HORUS format		
2C	JSON to HORUS format		
2D	MySQL database to HORUS format		
2E	Image to HORUS format		
2F	Video to HORUS format		
2G	Audio to HORUS format		
3A	Fixers Utilities		
3B	Data Binning or Bucketing		
3C	Averaging of data		
3D	Outlier Detection		
3E	Logging		
4A	Data processing using R		
4B	Program retrieve different attributes of data		
4C	Data pattern		
4D	Loading IP_DATA_ALL		
5A	Perform error management on the given data using pandas package		
5B	Write python/R program to create the network routing diagram from the given data on routers		
7	Transforming data		
8	Organizing data		
9	Generating data		
10	Power BI		

Environment Variables
User variables for Mscit11
Variable
Value
OneDrive
C:\Users\Mscit11\OneDrive
Path
C:\Users\Mscit11\AppData\Local\Programs\Python\Python311\Scr...
QT_DEVICE_PIXEL_RATIO
auto
TEMP
C:\Users\Mscit11\AppData\Local\Temp
TMP
C:\Users\Mscit11\AppData\Local\Temp
UIPATH_LANGUAGE
en
New...
Edit...
Delete
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
JAVA_HOME
C:\Program Files\Java\jdk1.8.0_211
NUMBER_OF_PROCESSORS
4
OS
Windows_NT
Path
C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Pro...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
New...
Edit...
Delete
OK
Cancel

Edit environment variable
C:\Users\Mscit11\AppData\Local\Programs\Python\Python311\Scripts\
C:\Users\Mscit11\AppData\Local\Programs\Python\Python311\
C:\Users\Mscit11\AppData\Local\Programs\Python\Python310\Scripts\
C:\Users\Mscit11\AppData\Local\Programs\Python\Python310\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
%USERPROFILE%\.dotnet\tools
C:\Program Files\Java\jdk1.8.0_211\bin
C:\apache-cassandra-3.11.4\bin
C:\Python27
New
Edit
Browse...
Delete
Move Up
Move Down
Edit text...
OK
Cancel

bin
This PC › Local Disk (C:) › apache-cassandra-3.11.4 › bin
Name
Date modified
Type
Size
cassandra
cassandra
cassandra.in
cassandra.in.sh
cassandra
cqlsh
cqlsh
cqlsh
debug-cql
debug-cql
nodetool
nodetool
source-conf
sstableloader
sstableloader
sstablescrub
sstablescrub
sstableupgrade
sstableupgrade
sstableutil
sstableutil
sstableverify
sstableverify
26 items
1 item selected 6.94 KB

cqlsh.py - C:\apache-cassandra-3.11.4\bin\cqlsh.py (2.7.18)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5
#!/bin/sh
-*- mode: Py
Licensed to
Foundation (ASF)
or more contributor license agreements. See the
distributed with this work for additional informa
regarding copyright ownership. The ASF licenses
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in
with the License. You may obtain a copy of the
#
http://www.apache.org/licenses/LICENSE-2.0
#

Python 2.7.18 Shell
File Edit Shell Debug Options Window Help
Python 2.7.18 (v2.7.18:8d21aa21f2, Apr 20 2020, 13:19:08) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\apache-cassandra-3.11.4\bin\cqlsh.py ==========
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh> CREATE KEYSPACE DS WITH replication = {'class':'SimpleStrategy',
'replication factor' : 3};

Python 2.7.18 Shell

File Edit Shell Debug Options Window Help

```
WARNING: pyreadline dependency missing.  Install to enable tab completion.
cqlsh> USE DS;
cqlsh:ds> CREATE TABLE emp(
      ... emp_id int PRIMARY KEY,
      ... emp_name text,
      ... emp_city text,
      ... emp_sal varint,
      ... emp_phone varint
      ... );
cqlsh:ds> insert into emp(emp_id,emp_name,emp_city,emp_sal,emp_phone)values
      ... (1, 'Nikhil', 'Thane',10000,12345);
cqlsh:ds> insert into emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) values
      ...  (2, 'Jayesh', 'Titwala',15000,22345);
cqlsh:ds> insert into emp (emp_id, emp_name, emp_city, emp_sal, emp_phone) values
      ... (3, 'Shraddha', 'Juhu',18000,33345);
```

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

 emp_id | emp_city | emp_name | emp_phone | emp_sal
--------+----------+----------+-----------+---------
      1 |    Thane |   Nikhil |     12345 |   10000
      2 |  Titwala |   Jayesh |     22345 |   15000
      3 |     Juhu | Shraddha |     33345 |   18000

(3 rows)
cqlsh:ds> select * from emp where emp_id = 1;

 emp_id | emp_city | emp_name | emp_phone | emp_sal
--------+----------+----------+-----------+---------
      1 |    Thane |   Nikhil |     12345 |   10000

(1 rows)
```

Input:

	A	B	C	D	E
1	Country	ISO-2-CODE	ISO-3-Code	ISO-M49	
2	Afghanistan	AF	AFG	4	
3	Aland Islands	AX	ALA	248	
4	Albania	AL	ALB	8	
5	Algeria	DZ	DZA	12	
6	American Samoa	AS	ASM	16	
7	Andorra	AD	AND	20	
8	Angola	AO	AGO	24	
9	Anguilla	AI	AIA	660	
10	Antarctica	AQ	ATA	10	
11	Antigua and Barbuda	AG	ATG	28	
12	Argentina	AR	ARG	32	
13	Armenia	AM	ARM	51	
14	Aruba	AW	ABW	533	
15	Australia	AU	AUS	36	
16	Austria	AT	AUT	40	
17	Azerbaijan	AZ	AZE	31	
18	Bahamas	BS	BHS	44	
19	Bahrain	BH	BHR	48	
20	Bangladesh	BD	BGD	50	
21	Barbados	BB	BRB	52	
22	Belarus	BY	BLR	112	
23	Belgium	BE	BEL	56	
24	Belize	BZ	BLZ	84	
25	Benin	BJ	BEN	204	
26	Bermuda	BM	BMU	60	

Country_Code

Code:

```python

import pandas as pd
import datetime as dt

print("Chetan Thabe",dt.datetime.now())

# Input Agreement ==========================================
sInputFileName='C:/Users/Data Science/Practical/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ==================================')
print(InputData)
# 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('CountryName', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryNumber', axis=0, ascending=False, inplace=True)
print(ProcessData)

print('Process Data Values ================================')
print(ProcessData)
print('====================================================')
# Output Agreement =========================================
OutputData=ProcessData

sOutputFileName='HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('CSV to HORUS - Done')
# Utility done =============================================

```

Output:

```
Chetan Thabe  2024-02-28 15:22:29.754318
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]
                                  CountryNumber
CountryName
Zambia                                      894
Yemen                                       887
Samoa                                       882
Wallis and Futuna Islands                   876
Venezuelaÿ(Bolivarian Republic)             862
...                                         ...
American Samoa                               16
Algeria                                      12
Antarctica                                   10
Albania                                       8
Afghanistan                                   4

[247 rows x 1 columns]
Process Data Values ================================
                                  CountryNumber
CountryName
Zambia                                      894
Yemen                                       887
Samoa                                       882
Wallis and Futuna Islands                   876
Venezuelaÿ(Bolivarian Republic)             862
...                                         ...
American Samoa                               16
Algeria                                      12
Antarctica                                   10
Albania                                       8
Afghanistan                                   4

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

Code:

```python
import pandas as pd
import xml.etree.ElementTree as ET

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:/DS Chetan/Country_code.Xml'
InputData = open(sInputFileName).read()

print(InputData)

ProcessDataXML=InputData
ProcessData=xml2df(ProcessDataXML)
print(ProcessData)

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
print(ProcessData)

ProcessData.set_index('CountryNumber', inplace=True)
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print(ProcessData)

OutputData=ProcessData
sOutputFileName='HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = True)
```

Output:

```
==================== RESTART: C:/DS Chetan/Practical2bXML.py ====================
Squeezed text (707 lines).
                      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]
                  CountryName CountryNumber
0                 Afghanistan             4
1               Aland Islands           248
2                     Albania             8
3                     Algeria            12
4               American Samoa            16
..                         ...           ...
242  Wallis and Futuna Islands           876
243             Western Sahara           732
244                      Yemen           887
245                     Zambia           894
246                   Zimbabwe           716

[247 rows x 2 columns]
                             CountryName
CountryNumber
716                            Zimbabwe
894                              Zambia
887                               Yemen
732                      Western Sahara
876           Wallis and Futuna Islands
...                                 ...
16                       American Samoa
12                              Algeria
```

Code:

```python
import pandas as pd
import datetime as dt

print("Chetan Thabe",dt.datetime.now())

# Input Agreement ==========================================
sInputFileName='C:/Users/Data Science/Practical/Country_Code.json'
InputData=pd.read_json(sInputFileName, orient='index',encoding="latin-1")

# Processing Rules ==========================================
ProcessData=InputData

print(ProcessData)

# 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)

OutputData=ProcessData

print(ProcessData)

sOutputFileName='C:/Users/Data Science/Practical/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('JSON to HORUS - Done')
# Utility done ==========================================
```

Input:

Practical2 > C > {} Country_Code.json > {} 3

```json
{
  "0": {
    "Country": "Afghanistan",
    "ISO-2-CODE": "AF",
    "ISO-3-Code": "AFG",
    "ISO-M49": 4
  },
  "1": {
    "Country": "Aland Islands",
    "ISO-2-CODE": "AX",
    "ISO-3-Code": "ALA",
    "ISO-M49": 248
  },
  "2": {
    "Country": "Albania",
    "ISO-2-CODE": "AL",
    "ISO-3-Code": "ALB",
    "ISO-M49": 8
  },
```

Output:

```
Chetan Thabe 2024-02-28 16:01:31.184367
                        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]
                              CountryName
CountryNumber
716                             Zimbabwe
894                               Zambia
887                                Yemen
732                       Western Sahara
876            Wallis and Futuna Islands
...                                  ...
16                        American Samoa
12                               Algeria
8                                Albania
248                        Aland Islands
4                            Afghanistan

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

Code:

```python

import pandas as pd
import sqlite3 as sq

sInputFileName='utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)

print('Input Data Values ==================================')
print(InputData)
print('====================================================')

ProcessData=InputData

ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)

ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)

ProcessData.set_index('CountryNumber', inplace=True)

ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)

print('Process Data Values ================================')
print(ProcessData)
print('====================================================')

OutputData=ProcessData

sOutputFileName='output/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)

print('Database to HORUS - Done')
```

Output:

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

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

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

Code:

```python

from skimage import io
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import datetime as dt

print("Chetan thabe",dt.datetime.now())
sInputFileName='D:\Data Science\Practical\Angus.jpg'
InputData = io.imread(sInputFileName)
InputData.shape

print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])

ProcessRawData=InputData.flatten()
ProcessRawData.shape

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']
ProcessData.columns=sColumns
ProcessData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('=====================================================')
print('Process Data Values ================================')
print('=====================================================')
OutputData = ProcessData
sOutputFileName = 'Image to HORUS.csv'
OutputData.to_csv(sOutputFileName, index = True)

```

Input:

Output:

```
Chetan thabe 2024-02-28 16:42:14.863018
X:  600
Y:  450
RGBA:  3
Rows:  162000
Columns : 5
=====================================================
Process Data Values ================================
=====================================================
```

	A	B	C	D	E	F	G	H
1	ID	XAxis	YAxis	Red	Green	Blue		
2	0	179	226	254	178	225		
3	1	253	178	225	253	178		
4	2	225	253	178	225	253		
5	3	178	225	253	178	225		
6	4	253	178	225	253	178		
7	5	225	253	178	225	253		
8	6	179	226	254	179	226		
9	7	254	179	226	254	179		
10	8	226	254	179	226	254		
11	9	179	226	254	179	226		
12	10	254	179	226	254	179		
13	11	226	254	179	226	254		
14	12	179	226	254	179	226		
15	13	254	179	226	254	179		
16	14	226	254	178	225	253		
17	15	178	225	253	178	225		
18	16	253	178	225	253	178		
19	17	225	253	178	225	253		
20	18	178	225	253	178	225		
21	19	253	178	225	255	178		
22	20	225	255	178	225	255		
23	21	179	226	255	178	225		
24	22	255	178	225	255	178		
25	23	225	255	178	225	255		
26	24	179	226	255	179	226		

Image to HORUS

Code:

```python
import os
import cv2
import time

sInputFileName = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical2\\F\\Dog.mp4'
sDataBaseDir = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical2\\F'

print('Converting movie to Frame')
vidcap = cv2.VideoCapture(sInputFileName)
success, image = vidcap.read()
print("success : ", success)

count = 0
while success:
    success, image = vidcap.read()

    if success:
        sFrame = os.path.join(sDataBaseDir, f'dog-frame{format(count, "04d")}.jpg')
        print('Extracted: ', sFrame)

        # Check if the image is not None before writing
        if image is not None:
            cv2.imwrite(sFrame, image)
        else:
            print('Empty image. Skipping...')

        # Add a small delay to ensure the file is fully saved before checking its size
        time.sleep(0.1)

        if os.path.getsize(sFrame) == 0:
            count -= 1
            os.remove(sFrame)
            print('Removed:', sFrame)

        if cv2.waitKey(10) == 27:
            # exit if escape is hit
            break

        count += 1

print('============')
print('Generated:', count, 'Frames')
print('============')
print('Movies to Frame Horus-Done')
```

Output:

```
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical> & C:/Users/11692/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/11692/Downloads/Msc.IT/Data Science/Practical/Practical2/F/
pract2f.py"
Converting movie to Frame
success :  True
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0000.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0001.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0002.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0003.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0004.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0005.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0006.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0007.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0008.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0009.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0010.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0011.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0012.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0013.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0014.jpg
```

```
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0373.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0374.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0375.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0376.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0377.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0378.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0379.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0380.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0381.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0382.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0383.jpg
Extracted:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical2\F\dog-frame0384.jpg
============
Generated: 385 Frames
============
Movies to Frame Horus-Done
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Input:

Code:

```python

from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import datetime as dt

print("Chetan Thabe",dt.datetime.now())

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()

inputFile='D:\\Msc.IT\\Data Science\\Practical\\Practical2\\G\\2ch-sound.wav'
print('=====================================================')
print('Processing : ', inputFile)
print('=====================================================')
InputRate, InputData = wavfile.read(inputFile)
show_info("2 channel", InputData,InputRate)

#Processdata
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns

#Output
OutputData=ProcessData
sOutputFileName='HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
```

```python

inputFile='D:\\Msc.IT\\Data Science\\Practical\\Practical2\\G\\4ch-sound.wav'
print('====================================================')
print('Processing : ', inputFile)
print('====================================================')
InputRate, InputData = wavfile.read(inputFile)
show_info("4 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)

inputFile='D:\\Msc.IT\\Data Science\\Practical\\Practical2\\G\\6ch-sound.wav'
print('====================================================')
print('Processing : ', inputFile)
print('====================================================')
InputRate, InputData = wavfile.read(inputFile)
show_info("6 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)

inputFile='D:\\Msc.IT\\Data Science\\Practical\\Practical2\\G\\8ch-sound.wav'
print('====================================================')
print('Processing : ', inputFile)
print('====================================================')
InputRate, InputData = wavfile.read(inputFile)
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='HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('====================================================')
print('Audio to HORUS - Done')
print('================================')
```

Output:

Figure 1
Signal Wave - 2 channel at 22050hz
15000
10000
5000
0
−5000
−10000
−15000
0
5000
10000
15000
20000
25000
30000
1
2
x=2.969e+04 y=1.066e+04

Figure 1
Signal Wave - 4 channel at 44100hz
20000
10000
0
−10000
−20000
−30000
0
25000
50000
75000
100000
125000
150000
175000
1
2
3
4

Figure 1
Signal Wave - 6 channel at 44100hz
1
2
3
4
5
6
10000
5000
0
−5000
−10000
0
50000
100000
150000
200000
250000

Figure 1
Signal Wave - 8 channel at 48000hz
1
2
3
4
5
6
7
8
10000
0
−10000
−20000
0
200000
400000
600000
800000
x=1.38e+05 y=1.072e+04

```
==================================================
Processing :  D:\Msc.IT\Data Science\Practical\Practical2\G\2ch-sound.wav
==================================================
----------------
Audio: 2 channel
----------------
Rate: 22050
----------------
shape: (29016, 2)
dtype: int16
min, max: -16384 14767
----------------
==================================================
Processing :  D:\Msc.IT\Data Science\Practical\Practical2\G\4ch-sound.wav
==================================================
----------------
Audio: 4 channel
----------------
Rate: 44100
----------------
shape: (169031, 4)
dtype: int16
min, max: -31783 26018
----------------
==================================================
Processing :  D:\Msc.IT\Data Science\Practical\Practical2\G\6ch-sound.wav
==================================================
----------------
Audio: 6 channel
----------------
Rate: 44100
----------------
shape: (257411, 6)
dtype: int16
min, max: -10018 10957
----------------
==================================================
Processing :  D:\Msc.IT\Data Science\Practical\Practical2\G\8ch-sound.wav
==================================================
----------------
Audio: 8 channel
----------------
Rate: 48000
----------------
shape: (888000, 8)
dtype: int16
min, max: -24859 16303
----------------
C:\Users\11692\AppData\Local\Programs\Python\Python312\Lib\tkinter\__init__.py:861: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  func(*args)
==================================================
Audio to HORUS - Done
================================
PS D:\Msc.IT\Data Science>
```

HORUS-Audio-2ch.csv
HORUS-Audio-4ch.csv
HORUS-Audio-6ch.csv
HORUS-Audio-8ch.csv

Code:

```python

import datetime as dt

print("Chetan thabe",dt.datetime.now())

print('#1 Removing lagging spaces from a data entry')
baddata = "Data Science with too many spaces is bad!!!           "

print('>',baddata,'<')

cleandata=baddata.strip()
print('>',cleandata,'<')

print('#2 Removing leading spaces from a data entry');
baddata = "         Data Science with too many spaces is bad!!! "
print('>',baddata,'<')

cleandata=baddata.strip()
print('>',cleandata,'<')

print('#3 Removing leading & lagging spaces from a data entry')
baddata = "         Data Science with too many spaces is bad!!!              "
print('>',baddata,'<')

cleandata=baddata.strip()
print('>',cleandata,'<')

print('#4 Reformatting data entry to match specific formatting criteria.')

baddate = dt.date(2021, 11, 8)
print(baddate)

baddata=format(baddate,'%Y-%m-%d')
print(baddata)

gooddate = dt.datetime.strptime(baddata,'%Y-%m-%d')
print(gooddate)

gooddata=format(gooddate,'%d-%b-%Y')
gooddata

# %b: Returns the first three characters of the month name. In our example, it returned "Nov"
# %d: Returns day of the month, from 1 to 31. In our example, it returned "15".
# %Y: Returns the year in four-digit format. In our example, it returned "2018".
# %H: Returns the hour. In our example, it returned "00".
# %M: Returns the minute, from 00 to 59. In our example, it returned "00".
# %S: Returns the second, from 00 to 59. In our example, it returned "00".
```

Output:

```
Chetan thabe 2024-02-28 16:49:21.406438
#1 Removing lagging spaces from a data entry
> Data Science with too many spaces is bad!!!            <
> Data Science with too many spaces is bad!!! <
#2 Removing leading spaces from a data entry
>          Data Science with too many spaces is bad!!!  <
> Data Science with too many spaces is bad!!! <
#3 Removing leading & lagging spaces from a data entry
>          Data Science with too many spaces is bad!!!              <
> Data Science with too many spaces is bad!!! <
#4 Reformatting data entry to match specific formatting criteria.
2021-11-08
2021-11-08
2021-11-08 00:00:00
'08-Nov-2021'
```

Code:

```python

import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import scipy.stats as stats
import datetime as dt

print("Chetan thabe",dt.datetime.now())
np.random.seed(0)
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)

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='DU-Histogram.png'
fig.savefig(sPathFig)
plt.show()

```

Output:

```
Chetan thabe 2024-02-28 16:54:11.947414
```

InPut:

```
ID,Country,Place Name,Post Code,Latitude,Longitude,First IP Number,Last IP Number
1,US,New York,10017,40.7528,-73.9725,204276480,204276735
2,US,New York,10017,40.7528,-73.9725,301984864,301985791
3,US,New York,10017,40.7528,-73.9725,404678736,404679039
4,US,New York,10017,40.7528,-73.9725,411592704,411592959
5,US,New York,10017,40.7528,-73.9725,416784384,416784639
6,US,New York,10017,40.7528,-73.9725,643347456,643348479
7,US,New York,10017,40.7528,-73.9725,643738112,643738623
8,US,New York,10017,40.7528,-73.9725,644459264,644459519
9,US,New York,10017,40.7528,-73.9725,644663808,644664319
10,US,New York,10017,40.7528,-73.9725,645591040,645591551
11,US,New York,10017,40.7528,-73.9725,789912128,789912191
12,US,New York,10017,40.7528,-73.9725,1064738304,1064738305
13,US,New York,10017,40.7528,-73.9725,1064769840,1064769847
14,US,New York,10017,40.7528,-73.9725,1075999744,1075999999
15,US,New York,10017,40.7528,-73.9725,1077213184,1077215231
16,US,New York,10017,40.7528,-73.9725,1081304704,1081304831
17,US,New York,10017,40.7528,-73.9725,1094561072,1094561103
18,US,New York,10017,40.7528,-73.9725,1104610304,1104611327
19,US,New York,10017,40.7528,-73.9725,1106442240,1106442751
20,US,New York,10017,40.7528,-73.9725,1117246720,1117246975
21,US,New York,10017,40.7528,-73.9725,1121484224,1121484239
22,US,New York,10017,40.7528,-73.9725,1136663552,1136663559
23,US,New York,10017,40.7528,-73.9725,1146290432,1146290559
24,US,New York,10017,40.7528,-73.9725,1161871662,1161871679
25,US,New York,10017,40.7528,-73.9725,1170316288,1170316543
26,US,New York,10017,40.7528,-73.9725,1207534336,1207534591
27,US,New York,10017,40.7528,-73.9725,1208299008,1208299263
28,US,New York,10017,40.7528,-73.9725,1208875656,1208875663
29,US,New York,10017,40.7528,-73.9725,1242109952,1242110463
30,US,New York,10017,40.7528,-73.9725,1249254144,1249254271
31,US,New York,10017,40.7528,-73.9725,1625352704,1625352959
32,US,New York,10017,40.7528,-73.9725,1625839104,1625839359
```

Code:

```
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import scipy.stats as stats
import datetime as dt

print("Chetan thabe",dt.datetime.now())
np.random.seed(0)
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)

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='DU-Histogram.png'
fig.savefig(sPathFig)
plt.show()
```

Output:

```
Chetan thabe 2024-02-28 16:57:10.162284
     Country Place_Name   Latitude
0         US   New York  51.512636
1         US   New York  40.752800
2         US   New York  40.752800
3         US   New York  40.752800
4         US   New York  40.752800
...      ...        ...        ...
3557      DE     Munich  48.091500
3558      DE     Munich  48.183300
3559      DE     Munich  48.100000
3560      DE     Munich  48.148000
3561      DE     Munich  48.148000

[3562 rows x 3 columns]
Country  Place_Name
DE       Munich        48.143223
GB       London        51.509381
US       New York      40.754824
Name: Latitude, dtype: float64
```

Input:

	A	B	C	D	E	F	G	H
1	ID	Country	Place Name	Post Code	Latitude	Longitude	First IP Number	Last IP Number
2	1	US	New York	10017	40.7528	-73.9725	204276480	204276735
3	2	US	New York	10017	40.7528	-73.9725	301984864	301985791
4	3	US	New York	10017	40.7528	-73.9725	404678736	404679039
5	4	US	New York	10017	40.7528	-73.9725	411592704	411592959
6	5	US	New York	10017	40.7528	-73.9725	416784384	416784639
7	6	US	New York	10017	40.7528	-73.9725	643347456	643348479
8	7	US	New York	10017	40.7528	-73.9725	643738112	643738623
9	8	US	New York	10017	40.7528	-73.9725	644459264	644459519
10	9	US	New York	10017	40.7528	-73.9725	644663808	644664319
11	10	US	New York	10017	40.7528	-73.9725	645591040	645591551
12	11	US	New York	10017	40.7528	-73.9725	789912128	789912191
13	12	US	New York	10017	40.7528	-73.9725	1064738304	1064738305
14	13	US	New York	10017	40.7528	-73.9725	1064769840	1064769847
15	14	US	New York	10017	40.7528	-73.9725	1075999744	1075999999
16	15	US	New York	10017	40.7528	-73.9725	1077213184	1077215231
17	16	US	New York	10017	40.7528	-73.9725	1081304704	1081304831

Code:

```python

import pandas as pd
import datetime as dt
print("Chetan thabe",dt.datetime.now())

InputFileName='/content/IP_DATA_CORE.csv'
OutputFileName='/content/Outliers.csv'
Ip_Data=pd.read_csv(InputFileName,header=0,low_memory=False,encoding="latin-1")
Ip_Data.shape
Ip_Data_All=Ip_Data[['Country','Place Name','Latitude','Longitude']]
Ip_Data_All.shape
LondonData=Ip_Data_All.loc[Ip_Data_All['Place Name']=='London']
LondonData.shape
LondonData
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()
MeanData
StdData
UpperBound=float(MeanData+StdData)
UpperBound
LowerBound=float(MeanData-StdData)
LowerBound
print('Lower Than ',LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Greater Than ',UpperBound)
OutliersUpper=AllData[AllData.Latitude>UpperBound]
print(OutliersUpper)
OutliersUpper.shape
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound)&(AllData.Latitude<=UpperBound)]
print(OutliersNot)
OutliersNot.to_csv(OutputFileName,index=False)
```

Output:

```
Chetan thabe 2024-02-28 17:00:58.765371
All Data
     Country Place Name   Latitude
1910      GB     London  51.513000
1911      GB     London  51.512636
1912      GB     London  51.564900
1913      GB     London  51.589500
1914      GB     London  51.523200
...      ...        ...        ...
3434      GB     London  51.509200
3435      GB     London  51.509200
3436      GB     London  51.516300
3437      GB     London  51.508500
3438      GB     London  51.513600
```

```
[1502 rows x 3 columns]
Lower Than  51.506332299883894
     Country Place Name  Latitude
1915      GB     London   51.4739
Greater Than  51.51242926537576
     Country Place Name   Latitude
1910      GB     London  51.513000
1911      GB     London  51.512636
1912      GB     London  51.564900
1913      GB     London  51.589500
1914      GB     London  51.523200
1916      GB     London  51.549100
1919      GB     London  51.516100
1920      GB     London  51.519800
1921      GB     London  51.519800
1923      GB     London  51.523700
1924      GB     London  51.523700
1925      GB     London  51.523700
1926      GB     London  51.523700
1927      GB     London  51.523200
3436      GB     London  51.516300
3438      GB     London  51.513600
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]
```

Input:

	A	B	C	D	E	F	G	H
1	ID	Country	Place Name	Post Code	Latitude	Longitude	First IP Number	Last IP Number
2	1	US	New York	10017	40.7528	-73.9725	204276480	204276735
3	2	US	New York	10017	40.7528	-73.9725	301984864	301985791
4	3	US	New York	10017	40.7528	-73.9725	404678736	404679039
5	4	US	New York	10017	40.7528	-73.9725	411592704	411592959
6	5	US	New York	10017	40.7528	-73.9725	416784384	416784639
7	6	US	New York	10017	40.7528	-73.9725	643347456	643348479
8	7	US	New York	10017	40.7528	-73.9725	643738112	643738623
9	8	US	New York	10017	40.7528	-73.9725	644459264	644459519
10	9	US	New York	10017	40.7528	-73.9725	644663808	644664319
11	10	US	New York	10017	40.7528	-73.9725	645591040	645591551
12	11	US	New York	10017	40.7528	-73.9725	789912128	789912191
13	12	US	New York	10017	40.7528	-73.9725	1064738304	1064738305
14	13	US	New York	10017	40.7528	-73.9725	1064769840	1064769847
15	14	US	New York	10017	40.7528	-73.9725	1075999744	1075999999
16	15	US	New York	10017	40.7528	-73.9725	1077213184	1077215231
17	16	US	New York	10017	40.7528	-73.9725	1081304704	1081304831

Code:

```python

import logging
import pandas as pd
import datetime as dt

print("Chetan thabe",dt.datetime.now())
Log_Format ='%(levelname)s %(asctime)s - %(message)s'
logging.basicConfig(filename = "DSlogging.log", level = logging.DEBUG, format = Log_Format)

#Creating an object
logger=logging.getLogger()

#test the logger
now = dt.datetime.now()
logger.info("second message - "+str(now))

InputFile='/content/IP_DATA_CORE.csv'
InputData = pd.read_csv(InputFile, encoding="latin-1")
logger.info("Attempting to read file - Jan 2022")

print(InputData.head())
logger.info("Browsing file data - Jan 2022")
logger.warning('Protocol problem')
#Reading the log file

with open("/content/DSlogging.log",'r') as f:
    for line in f:
        print(line)

```

Output:

```
Chetan thabe 2024-02-28 17:05:45.959370
   ID Country Place Name Post Code   Latitude  Longitude  First IP Number  \
0   1      US   New York     10017  51.512636   -73.9725        204276480
1   2      US   New York     10017  40.752800   -73.9725        301984864
2   3      US   New York     10017  40.752800   -73.9725        404678736
3   4      US   New York     10017  40.752800   -73.9725        411592704
4   5      US   New York     10017  40.752800   -73.9725        416784384

   Last IP Number
0       204276735
1       301985791
2       404679039
3       411592959
4       416784639
INFO 2024-01-15 13:06:23,453 - second message - 2024-01-15 13:06:23.453226

INFO 2024-01-15 13:06:23,469 - Attempting to read file - Jan 2022

INFO 2024-01-15 13:06:23,481 - Browsing file data - Jan 2022

WARNING 2024-01-15 13:06:23,481 - Protocol problem
```

Code:

```
#install.packages('readr')
#install.packages('data.table')
#install.packages('e1071')

# 4A - Perform following data processing using R

library(readr)
IP_DATA_ALL = read_csv("IP_DATA_ALL(small).csv")
# View(IP_DATA_ALL)
# spec(IP_DATA_ALL)

library(tibble)
set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
spec(IP_DATA_ALL)
IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = TRUE)
View(IP_DATA_ALL_FIX)
#Unique Identifier of Each Data Entry
IP_DATA_ALL_with_ID= rowid_to_column(IP_DATA_ALL_FIX, var = "RowID")
View(IP_DATA_ALL_with_ID)
#Data Type of Each Data Column
sapply(IP_DATA_ALL_with_ID, typeof)
```

```
#histogram
library(data.table)
hist_country=data.table(Country=unique(IP_DATA_ALL_with_ID[is.na(IP_DATA_ALL_with_ID ['Country']) == 0,
]$Country))
setorder(hist_country,'Country')
hist_country_with_id=rowid_to_column(hist_country, var = "RowIDCountry")
View(hist_country_with_id)
IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_with_ID, table(Country)))
View(IP_DATA_COUNTRY_FREQ)

# Profiling the data lake,
#frequency of lattitude
library(data.table)
hist_latitude =data.table(Latitude=unique(IP_DATA_ALL_with_ID [is.na(IP_DATA_ALL_with_ID ['Latitude']) == 0,
]$Latitude))
setkeyv(hist_latitude, 'Latitude')
setorder(hist_latitude)
hist_latitude_with_id=rowid_to_column(hist_latitude, var = "RowID")
View(hist_latitude_with_id)
IP_DATA_Latitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,
                                      table(Latitude)))
View(IP_DATA_Latitude_FREQ)

# similar process for longitude
library(data.table)
hist_longitude =data.table(Longitude=unique(IP_DATA_ALL_with_ID [is.na(IP_DATA_ALL_with_ID ['Longitude']) == 0,
```

```
library(data.table)
hist_longitude =data.table(Longitude=unique(IP_DATA_ALL_with_ID [is.na(IP_DATA_ALL_with_ID ['Longitude']) == 0,
]$Longitude))
setkeyv(hist_longitude, 'Longitude')
setorder(hist_longitude)
hist_longitude_with_id=rowid_to_column(hist_longitude, var = "RowID")
View(hist_longitude_with_id)
IP_DATA_Longitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,
                                       table(Longitude)))
View(IP_DATA_Longitude_FREQ)

#Minimum Value
min(hist_country$Country)

# Maximum Value
max(hist_country$Country)

# mean
sapply(hist_latitude_with_id[,'Latitude'], mean, na.rm=TRUE)
sapply(hist_longitude_with_id[,'Longitude'], mean, na.rm=TRUE)

#Median
sapply(hist_latitude_with_id[,'Latitude'], median, na.rm=TRUE)

# Mode
IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_with_ID, table(Country)))
```

```
# Mode
IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_with_ID, table(Country)))
setorder(IP_DATA_COUNTRY_FREQ,-N)
IP_DATA_COUNTRY_FREQ[1,'Country']
IP_DATA_Latitude_FREQ=data.table(with(IP_DATA_ALL_with_ID,
                                      table(Latitude)))
setorder(IP_DATA_Latitude_FREQ,-N)
IP_DATA_Latitude_FREQ[1,'Latitude']

# Range
sapply(hist_latitude_with_id[,'Latitude'], range, na.rm=TRUE)

# Quartiles
sapply(hist_latitude_with_id[,'Latitude'], quantile, na.rm=TRUE)

#Standard Deviation
sapply(hist_latitude_with_id[,'Latitude'], sd, na.rm=TRUE)

#Skewness
library(e1071)
View(skewness(hist_latitude_with_id$Latitude, na.rm = FALSE, type = 2))

#missing values
missing_country=data.table(Country=unique(IP_DATA_ALL_with_ID[is.na(IP_DATA_ALL_with_ID ['Country']) == 1, ]))
View(missing_country)
```

Output:

```
Rows: 124 Columns: 8
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): Country, Place Name
dbl (6): ID, Post Code, Latitude, Longitude, First IP Number, Last IP Number

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
New names:
Place Name -> Place.Name
Post Code -> Post.Code
First IP Number -> First.IP.Number
Last IP Number -> Last.IP.Number
```

A spec_tbl_df: 124 × 8

ID	Country	Place.Name	Post.Code	Latitude	Longitude	First.IP.Number	Last.IP.Number
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	BW	Gaborone	NA	-24.6464	25.9119	692781056	692781567
2	BW	Gaborone	NA	-24.6464	25.9119	692781824	692783103
3	BW	Gaborone	NA	-24.6464	25.9119	692909056	692909311
4	BW	Gaborone	NA	-24.6464	25.9119	692909568	692910079
18	NE	Niamey	NA	13.5167	2.1167	701203456	701203711

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

A spec_tbl_df: 124 × 8

ID	Country	Place.Name	Post.Code	Latitude	Longitude	First.IP.Number	Last.IP.Number
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	BW	Gaborone	NA	-24.6464	25.9119	692781056	692781567
2	BW	Gaborone	NA	-24.6464	25.9119	692781824	692783103
3	BW	Gaborone	NA	-24.6464	25.9119	692909056	692909311
4	BW	Gaborone	NA	-24.6464	25.9119	692909568	692910079

A spec_tbl_df: 124 × 9

RowID	ID	Country	Place.Name	Post.Code	Latitude	Longitude	First.IP.Number	Last.IP.Number
<int>	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	BW	Gaborone	NA	-24.6464	25.9119	692781056	692781567
2	2	BW	Gaborone	NA	-24.6464	25.9119	692781824	692783103
3	3	BW	Gaborone	NA	-24.6464	25.9119	692909056	692909311
4	4	BW	Gaborone	NA	-24.6464	25.9119	692909568	692910079
5	18	NE	Niamey	NA	13.5167	2.1167	701203456	701203711
6	19	NE	Niamey	NA	13.5167	2.1167	758886912	758887167
7	20	NE	Niamey	NA	13.5167	2.1167	1347294153	1347294160
8	21	NE	Niamey	NA	13.5167	2.1167	1755108096	1755108351
9	22	NE	Niamey	NA	13.5167	2.1167	1755828480	1755828735
10	23	MZ	Maputo	NA	-25.9653	32.5892	692883456	692883967

RowID: 'integer' ID: 'double' Country: 'character' Place.Name: 'character' Post.Code: 'double' Latitude: 'double' Longitude: 'double' First.IP.Number: 'double' Last.IP.Number: 'double'

A data.table: 7 × 2

RowIDCountry	Country
<int>	<chr>
1	BW
2	DZ
3	EG
4	GH
5	MZ
6	NE
7	US

A data.table: 7 × 2

Country	N
<chr>	<int>
BW	4
DZ	24
EG	7
GH	13
MZ	7
NE	5
US	64

A data.table: 19 × 2

RowID	Latitude
<int>	<dbl>
1	-25.9653
2	-24.6464
3	5.5500
4	5.6167
5	6.6833
6	13.5167
7	26.0410
8	26.5745
9	30.0771
10	33.5092
11	33.7480
12	36.1647
13	36.1787

A data.table: 19 × 2

Latitude	N
<chr>	<int>
-25.9653	7
-24.6464	4
5.55	7
5.6167	5
6.6833	1
13.5167	5
26.041	6
26.5745	6
30.0771	7
33.5092	4
33.748	9
36.1647	8
36.1787	8

A data.table: 19 × 2

RowID	Longitude
<int>	<dbl>
1	-122.2091
2	-115.3152
3	-112.1009
4	-84.3858
5	-81.9438
6	-81.7416
7	-78.8781
8	-77.5550
9	-72.8883
10	-1.6167
11	-0.2167
12	-0.0167
13	1.3347

A data.table: 19 × 2

Longitude	N
<chr>	<int>
-122.2091	6
-115.3152	8
-112.1009	4
-84.3858	9
-81.9438	6
-81.7416	6
-78.8781	12
-77.555	6
-72.8883	7
-1.6167	1
-0.2167	7
-0.0167	5
1.3347	8

'BW'
'US'
Latitude: 23.2449578947368
Longitude: -38.3436421052632
Latitude: 33.5092

A data.table: 1 × 1

Country
<chr>
US

A data.table: 1 × 1

Latitude
<chr>
42.8864

A matrix: 2 × 1 of type dbl

Latitude
-25.9653
43.1790

A matrix: 5 × 1 of type dbl

	Latitude
0%	-25.9653
25%	10.1000
50%	33.5092
75%	36.7400
100%	43.1790

Latitude: 21.0888149461672
-1.38568522369322

A data.table: 0 × 9

Country.RowID	Country.ID	Country.Country	Country.Place.Name	Country.Post.Code	Country.Latitude	Country.Longitude	Country.First.IP.Number
<int>	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>

Country.Longitude	Country.First.IP.Number	Country.Last.IP.Number
<dbl>	<dbl>	<dbl>

Input:

```
ID,Country,Place Name,Post Code,Latitude,Longitude,First IP Number,Last IP Number
1,BW,Gaborone,NA,-24.6464,25.9119,692781056,692781567
2,BW,Gaborone,NA,-24.6464,25.9119,692781824,692783103
3,BW,Gaborone,NA,-24.6464,25.9119,692909056,692909311
4,BW,Gaborone,NA,-24.6464,25.9119,692909568,692910079
18,NE,Niamey,NA,13.5167,2.1167,701203456,701203711
19,NE,Niamey,NA,13.5167,2.1167,758886912,758887167
20,NE,Niamey,NA,13.5167,2.1167,1347294153,1347294160
21,NE,Niamey,NA,13.5167,2.1167,1755108096,1755108351
22,NE,Niamey,NA,13.5167,2.1167,1755828480,1755828735
23,MZ,Maputo,NA,-25.9653,32.5892,692883456,692883967
30,MZ,Maputo,NA,-25.9653,32.5892,702519296,702519551
31,MZ,Maputo,NA,-25.9653,32.5892,774162694,774162698
32,MZ,Maputo,NA,-25.9653,32.5892,1754252544,1754252799
33,MZ,Maputo,NA,-25.9653,32.5892,1755839232,1755839487
```

Code:

```python
import sys
import os
import pandas as pd
import datetime as dt


print("Vignesh Nadar ",dt.datetime.now())
sFileName= 'C:/Users/11692/Downloads/Msc.IT/Data Science/Practical/Practical4/IP_DATA_ALL(small).csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")

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

print('\n#################### Raw Data Set ####################################')
for i in range(0,len(IP_DATA_ALL.columns)):
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))

print('\n#################### Fixed Data Set ##################################')
IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
    cNameOld=IP_DATA_ALL_FIX.columns[i] + '  '
    cNameNew=cNameOld.strip().replace(" ", ".")
    IP_DATA_ALL_FIX.columns.values[i] = cNameNew
    print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))

print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ['RowID']
print(IP_DATA_ALL_with_ID.head())

sFileName2= 'Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
```

Output:

```
Country <class 'str'>
Place Name <class 'str'>
Post Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First IP Number <class 'str'>
Last IP Number <class 'str'>

#################### Fixed Data Set ####################################
ï»¿ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
       ï»¿ID Country Place.Name  Post.Code  Latitude  Longitude  First.IP.Number  Last.IP.Number
RowID
0          1      BW   Gaborone        NaN  -24.6464    25.9119        692781056       692781567
1          2      BW   Gaborone        NaN  -24.6464    25.9119        692781824       692783103
2          3      BW   Gaborone        NaN  -24.6464    25.9119        692909056       692909311
3          4      BW   Gaborone        NaN  -24.6464    25.9119        692909568       692910079
4         18      NE     Niamey        NaN   13.5167     2.1167        701203456       701203711
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical4>
```

Retrieve_IP_DATA.csv

```
RowID,ï»¿ID,Country,Place.Name,Post.Code,Latitude,Longitude,First.IP.Number,Last.IP.Number
0,1,BW,Gaborone,,-24.6464,25.9119,692781056,692781567
1,2,BW,Gaborone,,-24.6464,25.9119,692781824,692783103
2,3,BW,Gaborone,,-24.6464,25.9119,692909056,692909311
3,4,BW,Gaborone,,-24.6464,25.9119,692909568,692910079
4,18,NE,Niamey,,13.5167,2.1167,701203456,701203711
5,19,NE,Niamey,,13.5167,2.1167,758886912,758887167
6,20,NE,Niamey,,13.5167,2.1167,1347294153,1347294160
7,21,NE,Niamey,,13.5167,2.1167,1755108096,1755108351
8,22,NE,Niamey,,13.5167,2.1167,1755828480,1755828735
9,23,MZ,Maputo,,-25.9653,32.5892,692883456,692883967
10,30,MZ,Maputo,,-25.9653,32.5892,702519296,702519551
11,31,MZ,Maputo,,-25.9653,32.5892,774162694,774162698
12,32,MZ,Maputo,,-25.9653,32.5892,1754252544,1754252799
13,33,MZ,Maputo,,-25.9653,32.5892,1755839232,1755839487
14,34,MZ,Maputo,,-25.9653,32.5892,1762662400,1762662911
15,35,MZ,Maputo,,-25.9653,32.5892,1762693120,1762695167
16,39,GH,Tema,,5.6167,-0.0167,701105152,701106175
17,40,GH,Tema,,5.6167,-0.0167,701640704,701642652
```

Code:

```
infoLog = paste0('info.log')
UserName='Practical Data Scientist Vignesh'
library(readr)
library(data.table)
library(tibble)
StartTime=Sys.time()
write(paste0('Start Information Log File ',
  format(StartTime, "%Y/%d/%m %H:%M:%S")),
  file=infoLog,append = FALSE)
FileName=paste0('IP_DATA_ALL(small).csv')
write(paste0(UserName,' Retrieve data file: ', FileName),
      file=infoLog,append = TRUE)
IP_DATA_ALL <- read_csv(FileName)
head(IP_DATA_ALL)
IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = TRUE)
head(IP_DATA_ALL_FIX)
IP_DATA_ALL_with_ID=rowid_to_column(IP_DATA_ALL_FIX, var = "RowID")
head(IP_DATA_ALL_with_ID)
FileNameOut=paste0('Retrieve_IP_DATA.csv')
fwrite(IP_DATA_ALL_with_ID, FileNameOut)
write(paste0(UserName,' Stores Retrieve data file: ', FileNameOut),
      file=infoLog,append = TRUE)
StopTime=Sys.time()
write(paste0('Stop Information Log File ',
             format(StopTime, "%Y/%d/%m %H:%M:%S")),
      file=infoLog,append = TRUE)
```

Output:

```
Rows: 124 Columns: 8
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): Country, Place Name
dbl (6): ID, Post Code, Latitude, Longitude, First IP Number, Last IP Number

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

A tibble: 6 × 8

ID	Country	Place Name	Post Code	Latitude	Longitude	First IP Number	Last IP Number
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	BW	Gaborone	NA	-24.6464	25.9119	692781056	692781567
2	BW	Gaborone	NA	-24.6464	25.9119	692781824	692783103
3	BW	Gaborone	NA	-24.6464	25.9119	692909056	692909311
4	BW	Gaborone	NA	-24.6464	25.9119	692909568	692910079
18	NE	Niamey	NA	13.5167	2.1167	701203456	701203711
19	NE	Niamey	NA	13.5167	2.1167	758886912	758887167

A tibble: 6 × 8

A tibble: 6 × 8

ID	Country	Place.Name	Post.Code	Latitude	Longitude	First.IP.Number	Last.IP.Number
<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	BW	Gaborone	NA	-24.6464	25.9119	692781056	692781567
2	BW	Gaborone	NA	-24.6464	25.9119	692781824	692783103
3	BW	Gaborone	NA	-24.6464	25.9119	692909056	692909311
4	BW	Gaborone	NA	-24.6464	25.9119	692909568	692910079
18	NE	Niamey	NA	13.5167	2.1167	701203456	701203711
19	NE	Niamey	NA	13.5167	2.1167	758886912	758887167

A tibble: 6 × 9

RowID	ID	Country	Place.Name	Post.Code	Latitude	Longitude	First.IP.Number	Last.IP.Number
<int>	<dbl>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	BW	Gaborone	NA	-24.6464	25.9119	692781056	692781567
2	2	BW	Gaborone	NA	-24.6464	25.9119	692781824	692783103
3	3	BW	Gaborone	NA	-24.6464	25.9119	692909056	692909311
4	4	BW	Gaborone	NA	-24.6464	25.9119	692909568	692910079
5	18	NE	Niamey	NA	13.5167	2.1167	701203456	701203711
6	19	NE	Niamey	NA	13.5167	2.1167	758886912	758887167

Code:

```
import sys
import os
import pandas as pd

#i. Drop the Columns Where All Elements Are Missing Values
InputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\Good-or-Bad.csv'
OutputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\Good-or-Bad.csv'
print('Loading :',InputFileName)
RawData=pd.read_csv(InputFileName,header=0)
RawData.shape
print('**Raw Data Values**')
print('********************')
print(RawData)
print('********************')
print('**Data Profile**')
print('********************')
print('Rows:',RawData.shape[0])
print('Columns :',RawData.shape[1])
TestData=RawData.dropna(axis=1,how='all')
TestData.shape
print('**Test Data Values**')
print('********************')
print(TestData)
print('********************')
print('**Data Profile**')
print('********************')
print('Rows:',TestData.shape[0])
print('Columns :',TestData.shape[1])
TestData.to_csv(OutputFileName,index=False)

```

Output:

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

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

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

Code:

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

#ii.Drop the Columns Where Any of the Elements Is Missing Values
InputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\Good-or-Bad.csv'
OutputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\Any_of_the_elements_misssing.csv'
print('Loading :',InputFileName)
RawData=pd.read_csv(InputFileName,header=0)
RawData.shape
print('**Raw Data Values**')
print('********************')
print(RawData)
print('********************')
print('**Data Profile**')
print('********************')
print('Rows:',RawData.shape[0])
print('Columns :',RawData.shape[1])
TestData=RawData.dropna(axis=1,how='any')
TestData.shape
print('**Test Data Values**')
print('********************')
print(TestData)
print('********************')
print('**Data Profile**')
print('********************')
print('Rows:',TestData.shape[0])
print('Columns :',TestData.shape[1])
TestData.to_csv(OutputFileName,index=False)
```

Output:

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

```
**Test Data Values**
********************
    FieldG
0        1
1        2
2        3
3        4
4        5
5        6
6        7
7        8
8        9
9       10
10      11
11      12
12      13
13      14
14      15
15      16
16      17
17      18
18      19
19      20
20      21
********************
**Data Profile**
********************
Rows: 21
Columns : 1
```

Code:

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

#iii.Keep Only the Rows That Contain a Maximum of Two Missing Values
InputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\Good-or-Bad.csv'
OutputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\More_Than_2_Values_Missing.csv'
print("Loading :",InputFileName)
RawData=pd.read_csv(InputFileName,header=0)
print('**Raw Data Values**')
print('********************')
print(RawData)
print('********************')
print('**Data Profile**')
print('********************')
print('Rows:',RawData.shape[0])
print('Columns :',RawData.shape[1])
TestData=RawData.dropna(thresh=2)
print('**Test Data Values**')
print('********************')
print(TestData)
print('********************')
print('**Data Profile**')
print('********************')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
TestData.to_csv(OutputFileName,index=False)
```

Output:

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

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

Code:

```
import sys
import os
import pandas as pd

#iii.Replacing missing values with mean, median, min, max
InputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\Good-or-Bad.csv'
OutputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5\\Good-or-Bad-04.csv'
InputFileDir ='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5'
sOutputFileNameA='Good-or-Bad-05-A.csv'
sOutputFileNameB='Good-or-Bad-05-B.csv'
sOutputFileNameC='Good-or-Bad-05-C.csv'
sOutputFileNameD='Good-or-Bad-05-D.csv'
sOutputFileNameE='Good-or-Bad-05-E.csv'
OutputFileDir = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical5'
sInputFileName = os.path.join(InputFileDir, InputFileName)
print('Loading', sInputFileName)
RawData=pd.read_csv(sInputFileName,header=0)
print(RawData)
RawData.shape

# # Replace with mean
print("################## MEAN ##################")
# Separate numeric and non-numeric columns
numeric_columns = RawData.select_dtypes(include=['number']).columns
non_numeric_columns = list(set(RawData.columns) - set(numeric_columns))
# Replace missing values in numeric columns with mean
TestData = RawData.copy()
TestData[numeric_columns] = TestData[numeric_columns].fillna(TestData[numeric_columns].mean())
# For non-numeric columns, replace missing values with a placeholder (e.g., 'NA')
TestData[non_numeric_columns] = TestData[non_numeric_columns].fillna('NA')
print(TestData)
sOutputFileName= os.path.join(OutputFileDir, sOutputFileNameA)
TestData.to_csv(sOutputFileName, index = False)
```

Output:

```
Loading C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical5\Good-or-Bad.csv
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good     NaN   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better    NaN   256.0     NaN     256.0       3
3    4.0   Good  Better   Best     NaN     NaN     211.0       4
4    5.0   Good  Better    NaN    64.0     NaN    6411.0       5
5    6.0   Good     NaN   Best    32.0     NaN      32.0       6
6    7.0    NaN  Better   Best    16.0     NaN    1611.0       7
7    8.0    NaN     NaN   Best     8.0     NaN    8111.0       8
8    9.0    NaN     NaN    NaN     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10   NaN    NaN     NaN    NaN     NaN     NaN       NaN      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good     NaN   Best   512.0     NaN     512.0      13
13  10.0   Good  Better    NaN   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best     NaN     NaN       NaN      15
15  10.0   Good  Better    NaN    64.0     NaN     164.0      16
16  10.0   Good     NaN   Best    32.0     NaN     322.0      17
17  10.0    NaN  Better   Best    16.0     NaN     163.0      18
18  10.0    NaN     NaN   Best     8.0     NaN     844.0      19
19  10.0    NaN     NaN    NaN     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
```

```
################## MEAN ###################
       ID FieldA  FieldB FieldC       FieldD  FieldE         FieldF  FieldG
0    1.00   Good  Better   Best  1024.000000     NaN   10241.000000       1
1    2.00   Good      NA   Best   512.000000     NaN    5121.000000       2
2    3.00   Good  Better     NA   256.000000     NaN     256.000000       3
3    4.00   Good  Better   Best   213.111111     NaN     211.000000       4
4    5.00   Good  Better     NA    64.000000     NaN    6411.000000       5
5    6.00   Good      NA   Best    32.000000     NaN      32.000000       6
6    7.00     NA  Better   Best    16.000000     NaN    1611.000000       7
7    8.00     NA      NA   Best     8.000000     NaN    8111.000000       8
8    9.00     NA      NA     NA     4.000000     NaN      41.000000       9
9   10.00      A       B      C     2.000000     NaN   21111.000000      10
10   7.75     NA      NA     NA   213.111111     NaN    8604.421053      11
11  10.00   Good  Better   Best  1024.000000     NaN  102411.000000      12
12  10.00   Good      NA   Best   512.000000     NaN     512.000000      13
13  10.00   Good  Better     NA   256.000000     NaN    1256.000000      14
14  10.00   Good  Better   Best   213.111111     NaN    8604.421053      15
15  10.00   Good  Better     NA    64.000000     NaN     164.000000      16
16  10.00   Good      NA   Best    32.000000     NaN     322.000000      17
17  10.00     NA  Better   Best    16.000000     NaN     163.000000      18
18  10.00     NA      NA   Best     8.000000     NaN     844.000000      19
19  10.00     NA      NA     NA     4.000000     NaN    4555.000000      20
20  10.00      A       B      C     2.000000     NaN     111.000000      21
```

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

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

```
################### MAX ####################
      ID FieldA  FieldB FieldC  FieldD  FieldE    FieldF  FieldG
0    1.0   Good  Better   Best  1024.0     NaN   10241.0       1
1    2.0   Good      NA   Best   512.0     NaN    5121.0       2
2    3.0   Good  Better     NA   256.0     NaN     256.0       3
3    4.0   Good  Better   Best  1024.0     NaN     211.0       4
4    5.0   Good  Better     NA    64.0     NaN    6411.0       5
5    6.0   Good      NA   Best    32.0     NaN      32.0       6
6    7.0     NA  Better   Best    16.0     NaN    1611.0       7
7    8.0     NA      NA   Best     8.0     NaN    8111.0       8
8    9.0     NA      NA     NA     4.0     NaN      41.0       9
9   10.0      A       B      C     2.0     NaN   21111.0      10
10  10.0     NA      NA     NA  1024.0     NaN  102411.0      11
11  10.0   Good  Better   Best  1024.0     NaN  102411.0      12
12  10.0   Good      NA   Best   512.0     NaN     512.0      13
13  10.0   Good  Better     NA   256.0     NaN    1256.0      14
14  10.0   Good  Better   Best  1024.0     NaN  102411.0      15
15  10.0   Good  Better     NA    64.0     NaN     164.0      16
16  10.0   Good      NA   Best    32.0     NaN     322.0      17
17  10.0     NA  Better   Best    16.0     NaN     163.0      18
18  10.0     NA      NA   Best     8.0     NaN     844.0      19
19  10.0     NA      NA     NA     4.0     NaN    4555.0      20
20  10.0      A       B      C     2.0     NaN     111.0      21
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Code:

```python
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:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9'
print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')
sInputFileName='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\Assess-Network-Routing-Company.csv'
sOutputFileName1='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\Organise-Network-Routing-Company.gml'
sOutputFileName2='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\Organise-Network-Routing-Company.png'
#Company='01-Vermeulen'
### Import Country Data
sFileName=sInputFileName
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('################################')
print(CompanyData.head())
print(CompanyData.shape)

G=nx.Graph()
for i in range(CompanyData.shape[0]):
    for j in range(CompanyData.shape[0]):
        Node0=CompanyData['Company_Country_Name'][i]
        Node1=CompanyData['Company_Country_Name'][j]
    if Node0 != Node1:
        G.add_edge(Node0,Node1)

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

print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
sFileName=sOutputFileName1
```

```python
sFileName=sOutputFileName1
print('################################')
print('Storing :',sFileName)
print('################################')
nx.write_gml(G, sFileName)
sFileName=sOutputFileName2
print('################################')
print('Storing Graph Image:',sFileName)
print('################################')
plt.figure(figsize=(15, 15))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
print('################################')
print('### Done!! ####################')
print('################################')
```

Output:

```
################################
Working Base : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9  using  win32
################################
################################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Assess-Network-Routing-Company.csv
################################
################################
  Company_Country_Code Company_Place_Name  Company_Latitude  Company_Longitude      Company_Country_Name
0                   US           New York           40.7528           -73.9725  United States of America
1                   US           New York           40.7214           -74.0052  United States of America
2                   US           New York           40.7662           -73.9862  United States of America
3                   US           New York           40.7449           -73.9782  United States of America
4                   US           New York           40.7605           -73.9933  United States of America
(150, 5)
Nodes: 6
Edges: 5
################################
Storing : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Organise-Network-Routing-Company.gml
################################
################################
Storing Graph Image: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Organise-Network-Routing-Company.png
################################
```

Organise-Network-Routing-Company.png
Practical9 > Organise-Network-Routing-Company.png
London(United Kingdom)
United Kingdom
Germany
United States of America
Munich(Germany)
New York(United States of Am

Code:

```python
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
sDataBaseDir='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical7'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)

sDatabaseName = os.path.join(sDataBaseDir, 'Vermeulen.db')
conn1 = sq.connect(sDatabaseName)
sDataVaultDir='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical7'
if not os.path.exists(sDataVaultDir):
    os.makedirs(sDataVaultDir)

sDatabaseName='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:

```
#################################
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
#################################
Birth Date in Reykjavik :
1960-12-20 10:15:00 (GMT) (+0000)
#################################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

PRACTICAL NO 8

Code:

```python
import sys
import os
import pandas as pd
import sqlite3 as sq
DWare='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical8\\datawarehouse.db'
DMart='C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical8\\datamart.db'
conn1=sq.connect(DWare)
conn2=sq.connect(DMart)
Table='Dim-BMI'
print('Loading into: ',DMart,' Table: ',Table)
print('From : ',DWare)
SQL="SELECT * FROM[Dim-BMI];"
PersonFrame0=pd.read_sql_query(SQL,conn1)
print("Profile")
PersonFrame0.shape
PersonFrame0
SQL="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(SQL,conn1)
PersonFrame1.shape
PersonFrame1
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
Table='Dim-BMI-Horizontal'
print('\n**********')
print('Storing in:',DMart,'\n Table:',Table)
print('\n**********')
DimPersonIndex.to_sql(Table,conn2,if_exists="replace")
SQL="SELECT * FROM [Dim-BMI-Horizontal];"
PersonFrame2=pd.read_sql_query(SQL,conn2)
PersonFrame2.shape
```

```python
PersonFrame2.shape
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:

```
Loading into:  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical8\datamart.db  Table:  Dim-BMI
From :  C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical8\datawarehouse.db
Profile

**********
Storing in: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical8\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
**********
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Code:

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

pd.options.mode.chained_assignment = None

if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\BaseFile'

print('################################')
print('Working Base:', Base, ' using', sys.platform)
print('################################')

sInputFileName = 'Assess-Network-Routing-Customer.csv'
sOutputFileName = 'Assess-Network-Routing-Customer.gml'
Company = '01-Vermeulen'

### Import Country Data

sFileName = os.path.join(Base, Company, sInputFileName)
print('################################')
print('Loading:', sFileName)
print('################################')

# Check if the file exists before reading
if os.path.exists(sFileName):
    CustomerData = pd.read_csv(sFileName, header=0, low_memory=False, encoding="latin-1")
    print('Loaded Country:', CustomerData.columns.values)
    print('################################')

    print(CustomerData.head())

else:
    print(f"File not found: {sFileName}")

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

Output:

```
################################
Working Base: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile  using win32
################################
################################
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Assess-Network-Routing-Customer.csv
################################
Loaded Country: ['Customer_Country_Code' 'Customer_Place_Name' 'Customer_Latitude'
 'Customer_Longitude' 'Customer_Country_Name']
################################
  Customer_Country_Code Customer_Place_Name  Customer_Latitude  Customer_Longitude Customer_Country_Name
0                    BW            Gaborone           -24.6464             25.9119              Botswana
1                    BW          Francistown          -21.1667             27.5167              Botswana
2                    BW                 Maun          -19.9833             23.4167              Botswana
3                    BW           Molepolole          -24.4167             25.5333              Botswana
4                    NE               Niamey           13.5167              2.1167                 Niger
################################
### Done!! ####################
################################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Code:

```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd

if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\BaseFile'

print('################################')
print('Working Base:', Base, ' using', sys.platform)
print('################################')

sInputFileName = 'Retrieve_Router_Location.csv'
sOutputFileName = 'Assess-DAG-Company-GPS.png'
Company = '01-Vermeulen'

# Import Company Data
sFileName = os.path.join(Base, Company, sInputFileName)
print('################################')
print('Loading:', sFileName)
print('################################')
CompanyData = pd.read_csv(sFileName, header=0, low_memory=False, encoding="latin-1")
print('Loaded Company:', CompanyData.columns.values)
print('################################')

print(CompanyData)
print('################################')
print('Rows:', CompanyData.shape[0])
print('################################')

G = nx.Graph()

print('################################')
for i in range(CompanyData.shape[0]):
    nLatitude = round(CompanyData['Latitude'][i], 1)
    nLongitude = round(CompanyData['Longitude'][i], 1)

    sLatitude = f'{abs(nLatitude)} S' if nLatitude < 0 else f'{nLatitude} N'
    sLongitude = f'{abs(nLongitude)} W' if nLongitude < 0 else f'{nLongitude} E'

    sGPS = f'{sLatitude}-{sLongitude}'
    G.add_node(sGPS)

print('################################')
for n1 in G.nodes():
    for n2 in G.nodes():
        if n1 != n2:
```

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

print('################################')
print("Nodes of graph:")
print(G.nodes())
print("Edges of graph:")
print(G.edges())
print('################################')

# Create directory if it doesn't exist
sFileDir = os.path.join(Base, Company, '02-Assess', '01-EDS', '02-Python')
os.makedirs(sFileDir, exist_ok=True)

# Save and display the graph
sFileName = os.path.join(sFileDir, sOutputFileName)
print('################################')
print('Storing:', sFileName)
print('################################')
pos = nx.circular_layout(G, dim=2, scale=2)
nx.draw(G, pos=pos, node_color='r', edge_color='b', with_labels=True, node_size=4000, font_size=9)
plt.savefig(sFileName)  # save as png
plt.show()  # display
print('################################')
```

Output:

```
##############################
Working Base: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile  using win32
##############################
##############################
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Retrieve_Router_Location.csv
##############################
Loaded Company: ['Country' 'Place_Name' 'Latitude' 'Longitude']
##############################
    Country Place_Name  Latitude  Longitude
0        US   New York   40.7528   -73.9725
1        US   New York   40.7214   -74.0052
2        US   New York   40.7662   -73.9862
3        US   New York   40.7449   -73.9782
4        US   New York   40.7605   -73.9933
..      ...        ...       ...        ...
145      DE     Munich   48.1475    11.4635
146      DE     Munich   48.0915    11.5392
147      DE     Munich   48.1833    11.7500
148      DE     Munich   48.1000    11.4667
149      DE     Munich   48.1480    11.7434

[150 rows x 4 columns]
##############################
Rows: 150
##############################
##############################
##############################
Link: 40.8 N-74.0 W to 40.7 N-74.0 W
Link: 40.8 N-74.0 W to 40.9 N-73.9 W
Link: 40.8 N-74.0 W to 40.8 N-73.9 W
Link: 40.8 N-74.0 W to 48.2 N-11.6 E
Link: 40.8 N-74.0 W to 48.1 N-11.6 E
Link: 40.8 N-74.0 W to 48.1 N-11.5 E
Link: 40.8 N-74.0 W to 48.2 N-11.5 E
Link: 40.8 N-74.0 W to 48.1 N-11.7 E
Link: 40.8 N-74.0 W to 48.2 N-11.4 E
Link: 40.8 N-74.0 W to 51.5 N-0.1 W
Link: 40.8 N-74.0 W to 51.6 N-0.1 E
Link: 40.8 N-74.0 W to 51.6 N-0.1 W
Link: 40.8 N-74.0 W to 51.6 N-0.2 W
Link: 40.8 N-74.0 W to 51.5 N-0.2 W
Link: 40.8 N-74.0 W to 48.2 N-11.8 E
Link: 40.7 N-74.0 W to 40.8 N-74.0 W
Link: 40.7 N-74.0 W to 40.9 N-73.9 W
Link: 40.7 N-74.0 W to 40.8 N-73.9 W
Link: 40.7 N-74.0 W to 48.2 N-11.6 E
```

```
Link: 48.2 N-11.8 E to 48.1 N-11.6 E
Link: 48.2 N-11.8 E to 48.1 N-11.5 E
Link: 48.2 N-11.8 E to 48.2 N-11.5 E
Link: 48.2 N-11.8 E to 48.1 N-11.7 E
Link: 48.2 N-11.8 E to 48.2 N-11.4 E
Link: 48.2 N-11.8 E to 51.5 N-0.1 W
Link: 48.2 N-11.8 E to 51.6 N-0.1 E
Link: 48.2 N-11.8 E to 51.6 N-0.1 W
Link: 48.2 N-11.8 E to 51.6 N-0.2 W
Link: 48.2 N-11.8 E to 51.5 N-0.2 W
##############################
##############################
Nodes of graph:
['40.8 N-74.0 W', '40.7 N-74.0 W', '40.9 N-73.9 W', '40.8 N-73.9 W', '48.2 N-11.6 E', '48.1 N-11.6 E', '48.1 N-11.5 E', '48.2 N-11.5 E', '48.1 N-11.7 E', '48.2 N-11.4 E', '51.5 N-0.1
 W', '51.6 N-0.1 E', '51.6 N-0.1 W', '51.6 N-0.2 W', '51.5 N-0.2 W', '48.2 N-11.8 E']
Edges of graph:
[('40.8 N-74.0 W', '40.7 N-74.0 W'), ('40.8 N-74.0 W', '40.9 N-73.9 W'), ('40.8 N-74.0 W', '40.8 N-73.9 W'), ('40.8 N-74.0 W', '48.2 N-11.6 E'), ('40.8 N-74.0 W', '48.1 N-11.6 E'), (
'40.8 N-74.0 W', '48.1 N-11.5 E'), ('40.8 N-74.0 W', '48.2 N-11.5 E'), ('40.8 N-74.0 W', '48.1 N-11.7 E'), ('40.8 N-74.0 W', '48.2 N-11.4 E'), ('40.8 N-74.0 W', '51.5 N-0.1 W'), ('40
.8 N-74.0 W', '51.6 N-0.1 E'), ('40.8 N-74.0 W', '51.6 N-0.1 W'), ('40.8 N-74.0 W', '51.6 N-0.2 W'), ('40.8 N-74.0 W', '51.5 N-0.2 W'), ('40.8 N-74.0 W', '48.2 N-11.8 E'), ('40.7 N-7
4.0 W', '40.9 N-73.9 W'), ('40.7 N-74.0 W', '40.8 N-73.9 W'), ('40.7 N-74.0 W', '48.2 N-11.6 E'), ('40.7 N-74.0 W', '48.1 N-11.6 E'), ('40.7 N-74.0 W', '48.1 N-11.5 E'), ('40.7 N-74.
0 W', '48.2 N-11.5 E'), ('40.7 N-74.0 W', '48.1 N-11.7 E'), ('40.7 N-74.0 W', '48.2 N-11.4 E'), ('40.7 N-74.0 W', '51.5 N-0.1 W'), ('40.7 N-74.0 W', '51.6 N-0.1 E'), ('40.7 N-74.0 W'
, '51.6 N-0.1 W'), ('40.7 N-74.0 W', '51.6 N-0.2 W'), ('40.7 N-74.0 W', '51.5 N-0.2 W'), ('40.7 N-74.0 W', '48.2 N-11.8 E'), ('40.9 N-73.9 W', '40.8 N-73.9 W'), ('40.9 N-73.9 W', '48
.2 N-11.6 E'), ('40.9 N-73.9 W', '48.1 N-11.6 E'), ('40.9 N-73.9 W', '48.1 N-11.5 E'), ('40.9 N-73.9 W', '48.2 N-11.5 E'), ('40.9 N-73.9 W', '48.1 N-11.7 E'), ('40.9 N-73.9 W', '48.2
 N-11.4 E'), ('40.9 N-73.9 W', '51.5 N-0.1 W'), ('40.9 N-73.9 W', '51.6 N-0.1 E'), ('40.9 N-73.9 W', '51.6 N-0.1 W'), ('40.9 N-73.9 W', '51.6 N-0.2 W'), ('40.9 N-73.9 W', '51.5 N-0.2
 W'), ('40.9 N-73.9 W', '48.2 N-11.8 E'), ('40.8 N-73.9 W', '48.2 N-11.6 E'), ('40.8 N-73.9 W', '48.1 N-11.6 E'), ('40.8 N-73.9 W', '48.1 N-11.5 E'), ('40.8 N-73.9 W', '48.2 N-11.5 E
'), ('40.8 N-73.9 W', '48.1 N-11.7 E'), ('40.8 N-73.9 W', '48.2 N-11.4 E'), ('40.8 N-73.9 W', '51.5 N-0.1 W'), ('40.8 N-73.9 W', '51.6 N-0.1 E'), ('40.8 N-73.9 W', '51.6 N-0.1 W'), (
'40.8 N-73.9 W', '51.6 N-0.2 W'), ('40.8 N-73.9 W', '51.5 N-0.2 W'), ('40.8 N-73.9 W', '48.2 N-11.8 E'), ('48.2 N-11.6 E', '48.1 N-11.6 E'), ('48.2 N-11.6 E', '48.1 N-11.5 E'), ('48.
2 N-11.6 E', '48.2 N-11.5 E'), ('48.2 N-11.6 E', '48.1 N-11.7 E'), ('48.2 N-11.6 E', '48.2 N-11.4 E'), ('48.2 N-11.6 E', '51.5 N-0.1 W'), ('48.2 N-11.6 E', '51.6 N-0.1 E'), ('48.2 N-
11.6 E', '51.6 N-0.1 W'), ('48.2 N-11.6 E', '51.6 N-0.2 W'), ('48.2 N-11.6 E', '51.5 N-0.2 W'), ('48.2 N-11.6 E', '48.2 N-11.8 E'), ('48.1 N-11.6 E', '48.1 N-11.5 E'), ('48.1 N-11.6
E', '48.2 N-11.5 E'), ('48.1 N-11.6 E', '48.1 N-11.7 E'), ('48.1 N-11.6 E', '48.2 N-11.4 E'), ('48.1 N-11.6 E', '51.5 N-0.1 W'), ('48.1 N-11.6 E', '51.6 N-0.1 E'), ('48.1 N-11.6 E',
'51.6 N-0.1 W'), ('48.1 N-11.6 E', '51.6 N-0.2 W'), ('48.1 N-11.6 E', '51.5 N-0.2 W'), ('48.1 N-11.6 E', '48.2 N-11.8 E'), ('48.1 N-11.5 E', '48.2 N-11.5 E'), ('48.1 N-11.5 E', '48.1
 N-11.7 E'), ('48.1 N-11.5 E', '48.2 N-11.4 E'), ('48.1 N-11.5 E', '51.5 N-0.1 W'), ('48.1 N-11.5 E', '51.6 N-0.1 E'), ('48.1 N-11.5 E', '51.6 N-0.1 W'), ('48.1 N-11.5 E', '51.6 N-0.
2 W'), ('48.1 N-11.5 E', '51.5 N-0.2 W'), ('48.1 N-11.5 E', '48.2 N-11.8 E'), ('48.2 N-11.5 E', '48.1 N-11.7 E'), ('48.2 N-11.5 E', '48.2 N-11.4 E'), ('48.2 N-11.5 E', '51.5 N-0.1 W'
), ('48.2 N-11.5 E', '51.6 N-0.1 E'), ('48.2 N-11.5 E', '51.6 N-0.1 W'), ('48.2 N-11.5 E', '51.6 N-0.2 W'), ('48.2 N-11.5 E', '51.5 N-0.2 W'), ('48.2 N-11.5 E', '48.2 N-11.8 E'), ('4
8.1 N-11.7 E', '48.2 N-11.4 E'), ('48.1 N-11.7 E', '51.5 N-0.1 W'), ('48.1 N-11.7 E', '51.6 N-0.1 E'), ('48.1 N-11.7 E', '51.6 N-0.1 W'), ('48.1 N-11.7 E', '51.6 N-0.2 W'), ('48.1 N-
11.7 E', '51.5 N-0.2 W'), ('48.1 N-11.7 E', '48.2 N-11.8 E'), ('48.2 N-11.4 E', '51.5 N-0.1 W'), ('48.2 N-11.4 E', '51.6 N-0.1 E'), ('48.2 N-11.4 E', '51.6 N-0.1 W'), ('48.2 N-11.4 E
', '51.6 N-0.2 W'), ('48.2 N-11.4 E', '51.5 N-0.2 W'), ('48.2 N-11.4 E', '48.2 N-11.8 E'), ('51.5 N-0.1 W', '51.6 N-0.1 E'), ('51.5 N-0.1 W', '51.6 N-0.1 W'), ('51.5 N-0.1 W', '51.6
N-0.2 W'), ('51.5 N-0.1 W', '51.5 N-0.2 W'), ('51.5 N-0.1 W', '48.2 N-11.8 E'), ('51.6 N-0.1 E', '51.6 N-0.1 W'), ('51.6 N-0.1 E', '51.6 N-0.2 W'), ('51.6 N-0.1 E', '51.5 N-0.2 W'),
('51.6 N-0.1 E', '48.2 N-11.8 E'), ('51.6 N-0.1 W', '51.6 N-0.2 W'), ('51.6 N-0.1 W', '51.5 N-0.2 W'), ('51.6 N-0.1 W', '48.2 N-11.8 E'), ('51.6 N-0.2 W', '51.5 N-0.2 W'), ('51.6 N-0
.2 W', '48.2 N-11.8 E'), ('51.5 N-0.2 W', '48.2 N-11.8 E')]
##############################
##############################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\02-Assess\01-EDS\02-Python\Assess-DAG-Company-GPS.png
##############################
```

Code:

```python
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd

if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\BaseFile'

print('################################')
print('Working Base:', Base, ' using ', sys.platform)
print('################################')

sInputFileName = 'Retrieve_Router_Location.csv'
sOutputFileName1 = 'Assess-DAG-Company-Country.png'
sOutputFileName2 = 'Assess-DAG-Company-Country-Place.png'
Company = '01-Vermeulen'

# Import Company Data
sFileName = os.path.join(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 = f"{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('################################')
```

```python
print('##############################')

print('##############################')
print("Nodes of graph:")
print(G1.nodes())
print("Edges of graph:")
print(G1.edges())
print('##############################')

sFileDir = os.path.join(Base, Company)
os.makedirs(sFileDir, exist_ok=True)

sFileName = os.path.join(sFileDir, sOutputFileName1)
print('##############################')
print('Storing:', sFileName)
print('##############################')
nx.draw(G1, pos=nx.spectral_layout(G1), node_color='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 = os.path.join(Base, Company)
os.makedirs(sFileDir, exist_ok=True)

sFileName = os.path.join(sFileDir, sOutputFileName2)
print('##############################')
print('Storing:', sFileName)
print('##############################')
nx.draw(G2, pos=nx.spectral_layout(G2), node_color='r', edge_color='b', with_labels=True, node_size=8000, font_size=12)
plt.savefig(sFileName)  # save as png
plt.show()  # display
```

Output:

```
##############################
Working Base: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile  using  win32
##############################
##############################
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Retrieve_Router_Location.csv
##############################
Loaded Company: ['Country' 'Place_Name' 'Latitude' 'Longitude']
##############################
    Country Place_Name  Latitude  Longitude
0        US   New York   40.7528   -73.9725
1        US   New York   40.7214   -74.0052
2        US   New York   40.7662   -73.9862
3        US   New York   40.7449   -73.9782
4        US   New York   40.7605   -73.9933
..      ...        ...       ...        ...
145      DE     Munich   48.1475    11.4635
146      DE     Munich   48.0915    11.5392
147      DE     Munich   48.1833    11.7500
148      DE     Munich   48.1000    11.4667
149      DE     Munich   48.1480    11.7434

[150 rows x 4 columns]
##############################
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')]
##############################
##############################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Assess-DAG-Company-Country.png
##############################
```

```
Nodes of graph:
['US', 'DE', 'GB']
Edges of graph:
[('US', 'DE'), ('US', 'GB'), ('DE', 'US'), ('DE', 'GB'), ('GB', 'US'), ('GB', 'DE')]
##############################
##############################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Assess-DAG-Company-Country.png
##############################
##############################
Link: New York-US  to  Munich-DE
Link: New York-US  to  London-GB
Link: Munich-DE  to  New York-US
Link: Munich-DE  to  London-GB
Link: London-GB  to  New York-US
Link: London-GB  to  Munich-DE
##############################
##############################
Nodes of graph:
['New York-US', 'Munich-DE', 'London-GB']
Edges of graph:
[('New York-US', 'Munich-DE'), ('New York-US', 'London-GB'), ('Munich-DE', 'New York-US'), ('Munich-DE', 'London-GB'), ('London-GB', 'New York-US'), ('London-GB', 'Munich-DE')]
##############################
##############################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Assess-DAG-Company-Country-Place.png
##############################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Code:

```python

import sys
import os
import pandas as pd

pd.options.mode.chained_assignment = None

if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\BaseFile'

print('################################')
print('Working Base :',Base, ' using ', sys.platform)
print('################################')

sInputFileName1='Retrieve_Country_Code.csv'
sInputFileName2='Retrieve_Router_Location.csv'
sInputFileName3='Retrieve_IP_DATA.csv'

sOutputFileName='Assess-Network-Routing-Company.csv'
Company='01-Vermeulen'


### Import Country Data

sFileName = os.path.join(Base, Company, sInputFileName1)
print('################################')
print('Loading :',sFileName)
print('################################')
CountryData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('################################')

## Assess Country Data

print('################################')
print('Changed :',CountryData.columns.values)
CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inplace=True)
print('To :',CountryData.columns.values)
print('################################')


### Import Company Data
sFileName = os.path.join(Base, Company, sInputFileName2)
print('################################')
```

```python
### Import Company Data
sFileName = os.path.join(Base, Company, sInputFileName2)
print('################################')
print('Loading :',sFileName)
print('################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('################################')

## Assess Company Data

print('################################')
print('Changed :',CompanyData.columns.values)
CompanyData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CompanyData.columns.values)
print('################################')


### Import Customer Data
sFileName = os.path.join(Base, Company, sInputFileName3)
print('################################')
print('Loading :',sFileName)
print('################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('################################')
print('Loaded Customer :',CustomerRawData.columns.values)
print('################################')

CustomerData=CustomerRawData.dropna(axis=0, how='any')
print('################################')
print('Remove Blank Country Code')
print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])
print('################################')

print('################################')
print('Changed :',CustomerData.columns.values)
CustomerData.rename(columns={'Country': 'Country_Code'}, inplace=True)
print('To :',CustomerData.columns.values)
print('################################')

print('################################')
print('Merge Company and Country Data')
print('################################')
CompanyNetworkData=pd.merge(
        CompanyData,
        CountryData,
        how='inner',
```

```python
CompanyNetworkData=pd.merge(
        CompanyData,
        CountryData,
        how='inner',
        on='Country_Code'
        )

print('################################')
print('Change ',CompanyNetworkData.columns.values)
for i in CompanyNetworkData.columns.values:
    j='Company_'+i
    CompanyNetworkData.rename(columns={i: j}, inplace=True)
print('To ', CompanyNetworkData.columns.values)
print('################################')


sFileDir = os.path.join(Base, Company)
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing :', sFileName)
print('################################')
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")


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

Output:

```
##############################
Working Base : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile  using  win32
##############################
##############################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Retrieve_Country_Code.csv
##############################
Loaded Country: ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
##############################
##############################
Changed : ['RowID' 'Country' 'ISO-2-CODE' 'ISO-3-Code' 'ISO-M49']
To : ['Country_Name' 'Country_Code']
##############################
##############################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Retrieve_Router_Location.csv
##############################
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
##############################
##############################
Changed : ['Country' 'Place_Name' 'Latitude' 'Longitude']
To : ['Country_Code' 'Place_Name' 'Latitude' 'Longitude']
##############################
##############################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Retrieve_IP_DATA.csv
##############################
##############################
Loaded Customer : ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
##############################
##############################
Remove Blank Country Code
Reduce Rows from 1247502  to  1150323
##############################
##############################
Changed : ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
To : ['RowID' 'ID' 'Country_Code' 'Place.Name' 'Post.Code' 'Latitude'
 'Longitude' 'First.IP.Number' 'Last.IP.Number']
##############################
##############################
Merge Company and Country Data
##############################
##############################
Change  ['Country_Code' 'Place_Name' 'Latitude' 'Longitude' 'Country_Name']
To  ['Company_Country_Code' 'Company_Place_Name' 'Company_Latitude'
 'Company_Longitude' 'Company_Country_Name']
##############################
```

```
Change  ['Country_Code' 'Place_Name' 'Latitude' 'Longitude' 'Country_Name']
To  ['Company_Country_Code' 'Company_Place_Name' 'Company_Latitude'
 'Company_Longitude' 'Company_Country_Name']
##############################
##############################
Storing : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen/Assess-Network-Routing-Company.csv
##############################
##############################
### Done!! ##################
##############################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

```
Company_Country_Code,Company_Place_Name,Company_Latitude,Company_Longitude,Company_Country_Name
US,New York,40.7528,-73.9725,United States of America
US,New York,40.7214,-74.0052,United States of America
US,New York,40.7662,-73.9862,United States of America
US,New York,40.7449,-73.9782,United States of America
US,New York,40.7605,-73.9933,United States of America
US,New York,40.7588,-73.968,United States of America
US,New York,40.7637,-73.9727,United States of America
US,New York,40.7553,-73.9924,United States of America
US,New York,40.7308,-73.9975,United States of America
US,New York,40.7694,-73.9609,United States of America
US,New York,40.733,-74.0078,United States of America
US,New York,40.7505,-73.9931,United States of America
US,New York,40.7517,-73.9972,United States of America
US,New York,40.7082,-74.0132,United States of America
US,New York,40.7267,-73.9981,United States of America
US,New York,40.7168,-73.9861,United States of America
US,New York,40.7317,-73.9885,United States of America
US,New York,40.7584,-73.9794,United States of America
US,New York,40.7592,-73.9778,United States of America
```

Code:

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

pd.options.mode.chained_assignment = None

# Determine the base directory based on the operating system
if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\BaseFile'

print('################################')
print('Working Base:', Base, ' using', sys.platform)
print('################################')

# File names and output settings
sInputFileName = 'Retrieve_IP_DATA.csv'
sOutputFileName = 'Assess-Network-Routing-Node.csv'
Company = '01-Vermeulen'

# Import IP Data
sFileName = os.path.join(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('################################')

# Column renaming and dropping unnecessary columns
print('################################')
print('Changed:', IPData.columns.values)
IPData.drop(['RowID', 'ID'], axis=1, inplace=True)
column_name_mapping = {
    'Country': 'Country_Code',
    'Place.Name': 'Place_Name',
    'Post.Code': 'Post_Code',
    'First.IP.Number': 'First_IP_Number',
    'Last.IP.Number': 'Last_IP_Number'
}
IPData.rename(columns=column_name_mapping, inplace=True)
print('To:', IPData.columns.values)
print('################################')

# Prefixing column names with 'Node_'
print('################################')
print('Change', IPData.columns.values)
IPData.rename(columns=lambda x: 'Node_' + x, inplace=True)
print('To', IPData.columns.values)
```

```python
print('To', IPData.columns.values)
print('################################')

# Create the output directory if it doesn't exist
sFileDir = os.path.join(Base, Company)
os.makedirs(sFileDir, exist_ok=True)

# Save the modified IP data to a new CSV file
sFileName = os.path.join(sFileDir, sOutputFileName)
print('################################')
print('Storing:', sFileName)
print('################################')
IPData.to_csv(sFileName, index=False, encoding="latin-1")
print('################################')
print('### Done!! ####################')
print('################################')
```

Output:

```
##############################
Working Base: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile  using win32
##############################
##############################
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Retrieve_IP_DATA.csv
##############################
Loaded IP: ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
##############################
##############################
Changed: ['RowID' 'ID' 'Country' 'Place.Name' 'Post.Code' 'Latitude' 'Longitude'
 'First.IP.Number' 'Last.IP.Number']
To: ['Country_Code' 'Place_Name' 'Post_Code' 'Latitude' 'Longitude'
 'First_IP_Number' 'Last_IP_Number']
##############################
##############################
Change ['Country_Code' 'Place_Name' 'Post_Code' 'Latitude' 'Longitude'
 'First_IP_Number' 'Last_IP_Number']
To ['Node_Country_Code' 'Node_Place_Name' 'Node_Post_Code' 'Node_Latitude'
 'Node_Longitude' 'Node_First_IP_Number' 'Node_Last_IP_Number']
##############################
##############################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\BaseFile\01-Vermeulen\Assess-Network-Routing-Node.csv
##############################
##############################
### Done!! ####################
##############################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

```csv
Node_Country_Code,Node_Place_Name,Node_Post_Code,Node_Latitude,Node_Longitude,Node_First_IP_Number,Node_Last_IP_Number
BW,Gaborone,,-24.6464,25.9119,692781056,692781567
BW,Gaborone,,-24.6464,25.9119,692781824,692783103
BW,Gaborone,,-24.6464,25.9119,692909056,692909311
BW,Gaborone,,-24.6464,25.9119,692909568,692910079
BW,Gaborone,,-24.6464,25.9119,693051392,693052415
BW,Gaborone,,-24.6464,25.9119,693078272,693078527
BW,Gaborone,,-24.6464,25.9119,693608448,693616639
BW,Gaborone,,-24.6464,25.9119,696929792,696930047
BW,Gaborone,,-24.6464,25.9119,700438784,700439039
BW,Gaborone,,-24.6464,25.9119,702075904,702076927
BW,Gaborone,,-24.6464,25.9119,702498816,702499839
BW,Gaborone,,-24.6464,25.9119,702516224,702517247
BW,Gaborone,,-24.6464,25.9119,774162663,774162667
BW,Gaborone,,-24.6464,25.9119,1401887232,1401887743
BW,Gaborone,,-24.6464,25.9119,1754209024,1754209279
NE,Niamey,,13.5167,2.1167,696918528,696919039
NE,Niamey,,13.5167,2.1167,696922112,696924159
NE,Niamey,,13.5167,2.1167,701203456,701203711
NE,Niamey,,13.5167,2.1167,758886912,758887167
```

Code:

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

# Set the option to avoid chained assignment warning
pd.options.mode.chained_assignment = None

# Determine the base directory based on the operating system
if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\Krenwaller'

print('################################')
print('Working Base:', Base, ' using', sys.platform)
print('################################')

# Input and output file names, and company identifier
sInputFileName1 = 'Retrieve_DE_Billboard_Locations.csv'
sInputFileName2 = 'Retrieve_Online_Visitor.csv'
sOutputFileName = 'Assess-DE-Billboard-Visitor.csv'
Company = '02-Krennwallner'

# Create the directory structure if it doesn't exist
sDataBaseDir = os.path.join(Base, Company)
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)

# Database setup
sDatabaseName = os.path.join(sDataBaseDir, 'krennwallner.db')
conn = sq.connect(sDatabaseName)

# Import Billboard Data
sFileName = os.path.join(Base, Company, sInputFileName1)
print('################################')
print('Loading:', sFileName)
print('################################')
BillboardRawData = pd.read_csv(sFileName, header=0, low_memory=False, encoding="latin-1")
BillboardRawData.drop_duplicates(subset=None, keep='first', inplace=True)
BillboardData = BillboardRawData.copy()
print('Loaded Company:', BillboardData.columns.values)
print('################################')

# Store Billboard Data in SQLite Database
print('################')
sTable = 'Assess_BillboardData'
print('Storing:', sDatabaseName, ' Table:', sTable)
BillboardData.to_sql(sTable, conn, if_exists="replace")
print('################')
```

```
BillboardData.to_sql(sTable, conn, if_exists="replace")
print('################')

print(BillboardData.head())
print('################################')
print('Rows:', BillboardData.shape[0])
print('################################')

# Import Visitor Data
sFileName = os.path.join(Base, Company, sInputFileName2)
print('################################')
print('Loading:', sFileName)
print('################################')
VisitorRawData = pd.read_csv(sFileName, header=0, low_memory=False, encoding="latin-1")
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData = VisitorRawData[VisitorRawData.Country == 'DE'].copy()
print('Loaded Company:', VisitorData.columns.values)
print('################################')

# Store Visitor Data in SQLite Database
print('################')
sTable = 'Assess_VisitorData'
print('Storing:', sDatabaseName, ' Table:', sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('################')

print(VisitorData.head())
print('################################')
print('Rows:', VisitorData.shape[0])
print('################################')

# Merge Billboard and Visitor Data
print('################')
sTable = 'Assess_BillboardVisitorData'
print('Loading:', sDatabaseName, ' Table:', sTable)
sSQL = "SELECT DISTINCT"
sSQL += " A.Country AS BillboardCountry,"
sSQL += " A.Place_Name AS BillboardPlaceName,"
sSQL += " A.Latitude AS BillboardLatitude, "
sSQL += " A.Longitude AS BillboardLongitude,"
sSQL += " B.Country AS VisitorCountry,"
sSQL += " B.Place_Name AS VisitorPlaceName,"
sSQL += " B.Latitude AS VisitorLatitude, "
sSQL += " B.Longitude AS VisitorLongitude,"
sSQL += " (B.Last_IP_Number - B.First_IP_Number) * 365.25 * 24 * 12 AS VisitorYearRate"
sSQL += " FROM"
sSQL += " Assess_BillboardData AS A"
sSQL += " JOIN "
```

```
sSQL += " Assess_BillboardData AS A"
sSQL += " JOIN "
sSQL += " Assess_VisitorData AS B"
sSQL += " ON "
sSQL += " A.Country = B.Country"
sSQL += " AND "
sSQL += " A.Place_Name = B.Place_Name;"

BillboardVistorsData = pd.read_sql_query(sSQL, conn)
print('################')

# Store Merged Data in SQLite Database
print('################')
sTable = 'Assess_BillboardVistorsData'
print('Storing:', sDatabaseName, ' Table:', sTable)
BillboardVistorsData.to_sql(sTable, conn, if_exists="replace")
print('################')

print(BillboardVistorsData.head())
print('################################')
print('Rows:', BillboardVistorsData.shape[0])
print('################################')

# Create the output directory if it doesn't exist
sFileDir = os.path.join(Base, Company)
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)

# Save the merged data to a CSV file
print('################################')
print('Storing:', sOutputFileName)
print('################################')
sFileName = os.path.join(sFileDir, sOutputFileName)
BillboardVistorsData.to_csv(sFileName, index=False)

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

Output:

```
################################
Working Base: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller  using win32
################################
################################
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\Retrieve_DE_Billboard_Locations.csv
################################
Loaded Company: ['Country' 'Place_Name' 'Latitude' 'Longitude']
################################
################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_BillboardData
################
  Country         Place_Name  Latitude  Longitude
0      DE               Lake   51.7833     8.5667
1      DE               Horb   48.4333     8.6833
2      DE         Hardenberg   51.1000     7.7333
3      DE  Horn-bad Meinberg   51.9833     8.9667
4      DE              Winkel   51.5500    13.3833
################################
Rows: 8873
################################
################################
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\Retrieve_Online_Visitor.csv
################################
Loaded Company: ['Country' 'Place_Name' 'Latitude' 'Longitude' 'First_IP_Number'
 'Last_IP_Number']
################################
################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_VisitorData
################
       Country Place_Name  Latitude  Longitude  First_IP_Number  Last_IP_Number
368678      DE       Lake   51.7833     8.5667       1418011904      1418012159
368679      DE       Lake   51.7833     8.5667       1528171008      1528171263
368680      DE       Lake   51.7833     8.5667       1528177920      1528178175
408664      DE       Horb   48.4333     8.6833        777246720       777246975
408665      DE       Horb   48.4333     8.6833       1339562496      1339562751
################################
Rows: 75999
################################
################
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_BillboardVisitorData
################
################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_BillboardVistorsData
################
  BillboardCountry BillboardPlaceName  BillboardLatitude  BillboardLongitude VisitorCountry VisitorPlaceName  VisitorLatitude  VisitorLongitude  VisitorYearRate
0               DE               Lake            51.7833              8.5667             DE             Lake          51.7833            8.5667       26823960.0
```

```
Loading: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_BillboardVisitorData
################
################
Storing: C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_BillboardVistorsData
################
  BillboardCountry BillboardPlaceName  BillboardLatitude  BillboardLongitude VisitorCountry VisitorPlaceName  VisitorLatitude  VisitorLongitude  VisitorYearRate
0               DE               Lake            51.7833              8.5667             DE             Lake          51.7833            8.5667       26823960.0
1               DE               Horb            48.4333              8.6833             DE             Horb          48.4333            8.6833       26823960.0
2               DE               Horb            48.4333              8.6833             DE             Horb          48.4333            8.6833       53753112.0
3               DE               Horb            48.4333              8.6833             DE             Horb          48.4333            8.6833      107611416.0
4               DE               Horb            48.4333              8.6833             DE             Horb          48.4333            8.6833       13359384.0
################################
Rows: 181235
################################
################################
Storing: Assess-DE-Billboard-Visitor.csv
################################
### Done!! ##############################################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Code:

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

# Determine the base directory based on the operating system
if sys.platform == 'linux':
    Base = os.path.expanduser('~') + '/VKHCG'
else:
    Base = 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\Krenwaller'

print('################################')
print('Working Base :', Base, ' using ', sys.platform)
print('################################')

Company = '02-Krennwallner'
sInputFileName = 'Retrieve_Online_Visitor.csv'

sDataBaseDir = os.path.join(Base, Company)
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)

sDatabaseName = os.path.join(sDataBaseDir, 'krennwallner.db')
conn = sq.connect(sDatabaseName)

# Import Country Data
sFileName = os.path.join(Base, Company, sInputFileName)
print('################################')
print('Loading :', sFileName)
print('################################')
VisitorRawData = pd.read_csv(sFileName,
                             header=0,
                             low_memory=False,
                             encoding="latin-1",
                             skip_blank_lines=True)
VisitorRawData.drop_duplicates(subset=None, keep='first', inplace=True)
VisitorData = VisitorRawData
print('Loaded Company :', VisitorData.columns.values)
print('################################')

# Storing Data in SQLite Table
print('################')
sTable = 'Assess_Visitor'
print('Storing :', sDatabaseName, ' Table:', sTable)
VisitorData.to_sql(sTable, conn, if_exists="replace")
print('################')

print(VisitorData.head())
print('################################')
```

```python
print(VisitorData.head())
print('##############################')
print('Rows : ', VisitorData.shape[0])
print('##############################')

# Creating Views
print('################')
sView = 'Assess_Visitor_UseIt'
print('Creating :', sDatabaseName, ' View:', sView)
sSQL = f"DROP VIEW IF EXISTS {sView};"
sql.execute(sSQL, conn)

sSQL = f"CREATE VIEW {sView} AS SELECT A.Country, A.Place_Name, A.Latitude, A.Longitude, (A.Last_IP_Number - A.First_IP_Number) AS UsesIt FROM Assess_Visitor as A WHERE Country is not null AND Place_Name is not null;"
sql.execute(sSQL, conn)

# Creating Total Visitors Location View
print('################')
sView = 'Assess_Total_Visitors_Location'
print('Creating :', sDatabaseName, ' View:', sView)
sSQL = f"DROP VIEW IF EXISTS {sView};"
sql.execute(sSQL, conn)

sSQL = f"CREATE VIEW {sView} AS SELECT Country, Place_Name, SUM(UsesIt) AS TotalUsesIt FROM Assess_Visitor_UseIt GROUP BY Country, Place_Name ORDER BY TotalUsesIt DESC LIMIT 10;"
sql.execute(sSQL, conn)

# Creating Total Visitors GPS View
print('################')
sView = 'Assess_Total_Visitors_GPS'
print('Creating :', sDatabaseName, ' View:', sView)
sSQL = f"DROP VIEW IF EXISTS {sView};"
sql.execute(sSQL, conn)

sSQL = f"CREATE VIEW {sView} AS SELECT Latitude, Longitude, SUM(UsesIt) AS TotalUsesIt FROM Assess_Visitor_UseIt GROUP BY Latitude, Longitude ORDER BY TotalUsesIt DESC LIMIT 10;"
sql.execute(sSQL, conn)

# Query and Display Top Data from Views
sTables = ['Assess_Total_Visitors_Location', 'Assess_Total_Visitors_GPS']
for sTable in sTables:
    print('################')
    print('Loading :', sDatabaseName, ' Table:', sTable)
    sSQL = f"SELECT * FROM {sTable};"
    TopData = pd.read_sql_query(sSQL, conn)
    print('################')
    print(TopData)
    print('################')
    print('##############################')
```

```python
    sSQL = f"SELECT * FROM {sTable};"
    TopData = pd.read_sql_query(sSQL, conn)
    print('################')
    print(TopData)
    print('################')
    print('##############################')
    print('Rows : ', TopData.shape[0])
    print('##############################')

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

Output:

```
##############################
Working Base : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller  using  win32
##############################
##############################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\Retrieve_Online_Visitor.csv
##############################
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude' 'First_IP_Number'
 'Last_IP_Number']
##############################
###############
Storing : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_Visitor
###############
  Country Place_Name  Latitude  Longitude  First_IP_Number  Last_IP_Number
0      BW   Gaborone  -24.6464    25.9119        692781056       692781567
1      BW   Gaborone  -24.6464    25.9119        692781824       692783103
2      BW   Gaborone  -24.6464    25.9119        692909056       692909311
3      BW   Gaborone  -24.6464    25.9119        692909568       692910079
4      BW   Gaborone  -24.6464    25.9119        693051392       693052415
##############################
Rows :  1247502
##############################
###############
Creating : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  View: Assess_Visitor_UseIt
c:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\Assess-Visitors.py:59: FutureWarning: `pandas.io.sql.execute` is deprecated and will be removed in the fu
ture version.
  sql.execute(sSQL, conn)
c:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\Assess-Visitors.py:62: FutureWarning: `pandas.io.sql.execute` is deprecated and will be removed in the fu
ture version.
  sql.execute(sSQL, conn)
###############
Creating : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  View: Assess_Total_Visitors_Location
c:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\Assess-Visitors.py:69: FutureWarning: `pandas.io.sql.execute` is deprecated and will be removed in the fu
ture version.
  sql.execute(sSQL, conn)
c:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\Assess-Visitors.py:72: FutureWarning: `pandas.io.sql.execute` is deprecated and will be removed in the fu
ture version.
  sql.execute(sSQL, conn)
###############
Creating : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  View: Assess_Total_Visitors_GPS
c:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\Assess-Visitors.py:79: FutureWarning: `pandas.io.sql.execute` is deprecated and will be removed in the fu
ture version.
  sql.execute(sSQL, conn)
c:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\Assess-Visitors.py:82: FutureWarning: `pandas.io.sql.execute` is deprecated and will be removed in the fu
ture version.
  sql.execute(sSQL, conn)
###############
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_Total_Visitors_Location
###############
  Country     Place_Name  TotalUsesIt
0      CN        Beijing     53139475
1      US       Palo Alto    33682341
2      US  Fort Huachuca     33472427
3      JP           Tokyo    31404799
```

```
  Country     Place_Name  TotalUsesIt
0      CN        Beijing     53139475
1      US       Palo Alto    33682341
2      US  Fort Huachuca     33472427
3      JP           Tokyo    31404799
4      US      Cambridge     25598851
5      US      San Diego     17751367
6      CN      Guangzhou     17563744
7      US         Newark     17270604
8      US        Raleigh     17167484
9      US         Durham     16914033
###############
##############################
Rows :  10
##############################
###############
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Krenwaller\02-Krennwallner\krennwallner.db  Table: Assess_Total_Visitors_GPS
###############
   Latitude  Longitude  TotalUsesIt
0   39.9289   116.3883     53139732
1   37.3762  -122.1826     33551404
2   31.5273  -110.3607     33472427
3   35.6427   139.7677     31439772
4   23.1167   113.2500     17577053
5   42.3646   -71.1028     16890698
6   40.7355   -74.1741     16813373
7   42.3223   -83.1763     16777212
8   35.7977   -78.6253     16761084
9   32.8072  -117.1649     16747680
###############
##############################
Rows :  10
##############################
### Done!! ##########################################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Code:

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

# Determine the base directory based on the operating system
base_directory = os.path.expanduser('~') + '/VKHCG' if sys.platform == 'linux' else 'C:\\Users\\11692\\Downloads\\Msc.IT\\Data Science\\Practical\\Practical9\\Clark'
print('################################')
print('Working Base :', base_directory, ' using ', sys.platform)
print('################################')

Company = '04-Clark'
sInputFileName1 = 'Retrieve-Country-Currency.csv'
sInputFileName2 = 'Retrieve_Euro_EchangeRates.csv'

sDataBaseDir = os.path.join(base_directory, Company)
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)

sDatabaseName = os.path.join(sDataBaseDir, 'clark.db')
conn = sq.connect(sDatabaseName)

# Import Country Data
sFileName1 = os.path.join(base_directory, sInputFileName1)
print('################################')
print('Loading :', sFileName1)
print('################################')
CountryRawData = pd.read_csv(sFileName1, header=0, low_memory=False, encoding="latin-1")
CountryRawData.drop_duplicates(subset=None, keep='first', inplace=True)
CountryData = CountryRawData
print('Loaded Company :', CountryData.columns.values)
print('################################')

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

print(CountryData.head())
print('################################')
print('Rows : ', CountryData.shape[0])
print('################################')

# Import Forex Data
sFileName2 = os.path.join(base_directory, sInputFileName2)
print('################################')
print('Loading :', sFileName2)
print('################################')
ForexRawData = pd.read_csv(sFileName2, header=0, low_memory=False, encoding="latin-1")
```

```python
print('################################')
ForexRawData = pd.read_csv(sFileName2, header=0, low_memory=False, encoding="latin-1")
ForexRawData.drop_duplicates(subset=None, keep='first', inplace=True)
ForexData = ForexRawData.head(5)
print('Loaded Company :', ForexData.columns.values)
print('################################')

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

print(ForexData.head())
print('################################')
print('Rows : ', ForexData.shape[0])
print('################################')

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

for c in range(CodeData.shape[0]):
    print('################')
    sTable = 'Assess_Forex_2xCountry_' + CodeData['CodeIn'][c]
    print('Loading :', sDatabaseName, ' Table:', sTable)
    sSQL = "select distinct"
    sSQL = sSQL + " A.Date,"
    sSQL = sSQL + " A.CodeIn,"
    sSQL = sSQL + " B.Country as CountryIn,"
    sSQL = sSQL + " B.Currency as CurrencyNameIn,"
    sSQL = sSQL + " A.CodeOut,"
    sSQL = sSQL + " C.Country as CountryOut,"
    sSQL = sSQL + " C.Currency as CurrencyNameOut,"
    sSQL = sSQL + " A.Rate"
    sSQL = sSQL + " from"
    sSQL = sSQL + " Assess_Forex as A"
    sSQL = sSQL + " JOIN"
    sSQL = sSQL + " Assess_Country as B"
    sSQL = sSQL + " ON A.CodeIn = B.CurrencyCode"
    sSQL = sSQL + " JOIN"
    sSQL = sSQL + " Assess_Country as C"
    sSQL = sSQL + " ON A.CodeOut = C.CurrencyCode"
    sSQL = sSQL + " WHERE"
```

```python
        sSQL = sSQL + " ON A.CodeIn = B.CurrencyCode"
        sSQL = sSQL + " JOIN"
        sSQL = sSQL + " Assess_Country as C"
        sSQL = sSQL + " ON A.CodeOut = C.CurrencyCode"
        sSQL = sSQL + " WHERE"
        sSQL = sSQL + " A.CodeIn ='" + CodeData['CodeIn'][c] + "';"
        ForexData = pd.read_sql_query(sSQL, conn).head(1000)
        print('################')
        if ForexData is not None and not ForexData.empty:
            print(ForexData)
            print('################')
            sTable = 'Assess_Forex_' + CodeData['CodeIn'][c]
            print('Storing :', sDatabaseName, ' Table:', sTable)
            ForexData.to_sql(sTable, conn, if_exists="replace")
            print('################')
            print('################################')
            print('Rows : ', ForexData.shape[0])
            print('################################')

    # Close the SQLite connection
    conn.close()

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

Output:

```
################################
Working Base : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark  using  win32
################################
################################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\Retrieve-Country-Currency.csv
################################
Loaded Company : ['Country' 'Currency' 'CurrencyCode']
################################
################
Storing : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\04-Clark\clark.db  Table: Assess_Country
################
                    Country        Currency CurrencyCode
0               Afghanistan  Afghan afghani          AFN
1  Akrotiri and Dhekelia (UK)   European euro          EUR
2     Aland Islands (Finland)   European euro          EUR
3                   Albania    Albanian lek          ALL
4                   Algeria  Algerian dinar          DZD
################################
Rows :  253
################################
################################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\Retrieve_Euro_EchangeRates.csv
################################
Loaded Company : ['Date' 'CodeIn' 'CodeOut' 'Rate']
################################
################
Storing : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\04-Clark\clark.db  Table: Assess_Forex
################
         Date CodeIn CodeOut      Rate
0  2017-05-05    AUD     AUD  1.000000
1  2017-05-05    AUD     BGN  1.318635
2  2017-05-05    AUD     BRL  2.354167
3  2017-05-05    AUD     CAD  1.017665
4  2017-05-05    AUD     CHF  0.730785
################################
Rows :  5
################################
################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\04-Clark\clark.db  Table: Assess_Forex
################
################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\04-Clark\clark.db  Table: Assess_Forex_2xCountry_AUD
################
          Date CodeIn                   CountryIn     CurrencyNameIn CodeOut                         CountryOut    CurrencyNameOut      Rate
0   2017-05-05    AUD                   Australia  Australian dollar     AUD                          Australia  Australian dollar  1.000000
1   2017-05-05    AUD                   Australia  Australian dollar     AUD         Christmas Island (Australia)  Australian dollar  1.000000
2   2017-05-05    AUD                   Australia  Australian dollar     AUD  Cocos (Keeling) Islands (Australia)  Australian dollar  1.000000
```

```
################
Loading : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\04-Clark\clark.db  Table: Assess_Forex_2xCountry_AUD
################
          Date CodeIn                  CountryIn     CurrencyNameIn CodeOut                          CountryOut    CurrencyNameOut      Rate
0   2017-05-05    AUD                  Australia  Australian dollar     AUD                            Australia  Australian dollar  1.000000
1   2017-05-05    AUD                  Australia  Australian dollar     AUD         Christmas Island (Australia)  Australian dollar  1.000000
2   2017-05-05    AUD                  Australia  Australian dollar     AUD  Cocos (Keeling) Islands (Australia)  Australian dollar  1.000000
3   2017-05-05    AUD                  Australia  Australian dollar     AUD                               Kiribati  Australian dollar  1.000000
4   2017-05-05    AUD                  Australia  Australian dollar     AUD                                  Nauru  Australian dollar  1.000000
..         ...    ...                        ...                ...     ...                                    ...                ...       ...
79  2017-05-05    AUD                      Nauru  Australian dollar     CHF                           Switzerland        Swiss franc  0.730785
80  2017-05-05    AUD  Norfolk Island (Australia)  Australian dollar     CHF                         Liechtenstein        Swiss franc  0.730785
81  2017-05-05    AUD  Norfolk Island (Australia)  Australian dollar     CHF                           Switzerland        Swiss franc  0.730785
82  2017-05-05    AUD                     Tuvalu  Australian dollar     CHF                         Liechtenstein        Swiss franc  0.730785
83  2017-05-05    AUD                     Tuvalu  Australian dollar     CHF                           Switzerland        Swiss franc  0.730785

[84 rows x 8 columns]
################
Storing : C:\Users\11692\Downloads\Msc.IT\Data Science\Practical\Practical9\Clark\04-Clark\clark.db  Table: Assess_Forex_AUD
################
##############################
Rows :  84
##############################
### Done!! ##########################################
PS C:\Users\11692\Downloads\Msc.IT\Data Science\Practical>
```

Input & Out

C17

	A	B	C
1	Company Name	Years	Profit
2	A	2001	75000
3	B	2001	100000
4	C	2001	30000
5	D	2001	150000
6	E	2001	300000
7	A	2002	100000
8	B	2002	180000
9	C	2002	100000
10	D	2002	200000
11	E	2002	500000
12	A	2003	200000
13	B	2003	100000
14	C	2003	125000
15	D	2003	130000
16	E	2003	225000
17			
18			
19			

Untitled - Power BI Desktop
Search
Sign in
File
Home
Insert
Modeling
View
Help
Paste
Cut
Copy
Format painter
Clipboard
Get data
Excel workbook
Data hub
SQL Server
Enter data
Dataverse
Recent sources
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Sensitivity
Publish
Share
Common data sources
Excel workbook
Power BI datasets
Dataflows
Dataverse
SQL Server
Analysis Services
Text/CSV
Web
OData feed
Blank query
Power BI Template Apps
More...
Add data to your report
Once loaded, your data will appear in the Fields pane.
Import data from a text or CSV file.
Import data from SQL Server
Paste data into a blank table
Try a sample dataset
Get data from another source →
Visualizations
Build visual
Values
Add data fields here
Drill through
Cross-report
Off
Keep all filters
On
Add drill-through fields here
Filters
Fields
Page 1
Page 1 of 1
74%
Update available (click to download)

Untitled - Power BI Desktop
Search
Sign in
File
Home
Insert
Modeling
View
Help
Format
Data / Drill
Paste
Cut
Copy
Format painter
Clipboard
Get data
Excel workbook
Data hub
SQL Server
Enter data
Dataverse
Recent sources
Data
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Sensitivity
Publish
Share
Sum of Years and Sum of Profit by Company Name
Sum of Years
Sum of Profit
Company Name
Company Name D
Sum of Profit 480000
Filters
Search
Filters on this visual
Company Name is (All)
Sum of Profit is (All)
Sum of Years is (All)
Add data fields here
Filters on this page
Add data fields here
Filters on all pages
Add data fields here
Visualizations
Build visual
Column y-axis
Sum of Years
Sum of Profit
Line y-axis
Add data fields here
Column legend
Add data fields here
Small multiples
Add data fields here
Fields
Search
Data
Company Name
Profit
Years
Page 1
Page 1 of 1
53%
Update available (click to download)

SOFT COMPUTING TECHNIQUES

INDEX

Sr. No.	Practical name	Date	Sign
1A	Design a simple neural network model		
1B	Calculate the output of neural net using both binary and bipolar sigmoidal function		
2A	Generate AND/NOT function using McCulloch-Pitts neural network		
2B	Generate XOR function using McCulloch-Pitts neural network		
3A	Write a program to implement Hebb's rule		
3B	Write a program to implement delta rule		
4A	Write a program for Back Propogation Algorithm		
4B	Write a program for Error Back Propogation Algorithm		
7A	Write a program for Linear separation		
7B	Write a program for Hopfield network model for associative memory		
8A	Membership and Identity Operators | in, not in		
8B	Membership and Identity Operators | is, is not		
9A	Find ratios using fuzzy logic		
9B	Solve Tipping problem using fuzzy logic		
10A	Implementation of Simple genetic algorithm		

Practical 1A

Aim: Design a simple linear neural network model

Code:

```
w=float(input("Enter value for weight:"))
b=float(input("Enter value for bias:"))
x=float(input("Enter value for input:"))
yin=float(b+(w*x))
print("The Net input is",yin)
```

Output:

```
Enter value for weight:0.3
Enter value for bias:0.2
Enter value for input:0.2
The Net input is 0.26
```

Practical 1B

Aim: Calculate the output of neural net using both binary and bipolar sigmoidal function.

Code:

```
# w will take weight & x will take the input
w = [ ]
x = [ ]
n = int(input("Enter the number of input neurons: "))
bias=float(input("Enter bias: "))
# taking the value of input and their weight
for i in range(0,n):
    a = float(input("Enter the input: "))
    x.append(a)
    b = float(input("Enter the weight: "))
    w.append(b)
print("The given input are: ",x)
print("The given weight are: ",w)
print("The Bias value is: ",bias)
yin=0.0
yin=yin+bias
for i in range(0,n):
    yin = yin + (w[i]*x[i])
print("The net input is ",yin)
#Binary
thres=float(input("enter threshold value for binary and bipolar :"))
if(yin>=thres):
    output_binary=1
else:
    output_binary=0
print("Final Output(binary)is",output_binary)
#Bipolar
if(yin>=thres):
    output_bipolar=1
else:
    output_bipolar=-1
print("Final Output(bipolar)is",output_bipolar)
```

Output:

```
= RESTART: C:/M.SC/Soft Computing/Coding/1B.py
Enter the number of input neurons: 3
Enter bias: 0.1
Enter the input: 1
Enter the weight: 0.1
Enter the input: 2
Enter the weight: 0.2
Enter the input: 3
Enter the weight: 0.3
The given input are:  [1.0, 2.0, 3.0]
The given weight are:  [0.1, 0.2, 0.3]
The Bias value is:  0.1
The net input is  1.5
enter threshold value for binary and bipolar :1
Final Output(binary)is 1
Final Output(bipolar)is 1
```

Practical 2A

Aim: Generate AND/NOT function using McCulloch-Pitts neural net.

Code:

```
num_ip=int(input("Enter the number of inputs:"))
print("For the",num_ip,"inputs calculate the net input using yin=x1w1+x2w2")
theta=1
x1=[]
x2=[]
for i in range(0,num_ip):
    a=int(input("Enter the input x1:"))
    x1.append(a)
    b=int(input("Enter the input x2:"))
    x2.append(b)
print("x1=",x1)
print("x2=",x2)
print("Value of theta is 1")
print("Case1: For calculating the net input we will take weights w1=w2=1")
w1=w2=1
case_y1=[]
case_yin1=[]
print("x1 w1 x2 w2 case_y1 case_yin1")
for i in range(0,num_ip):
    case_y1.append(x1[i]*w1+x2[i]*w2)
    if(case_y1[i]>=theta):
       case_yin1.append(1)
    else:
       case_yin1.append(0)
    print(x1[i]," ", w1, " ",x2[i]," ",w2," ",case_y1[i]," ", case_yin1[i])
print("From the calculated net inputs its not possible to fire neuron from the given inputs so these weights are not suitable")
print("Case2: For calculating the net input we will take weights w1=1,w2=-1")
w1=1
w2=-1
case_y2=[]
case_yin2=[]
print("x1 w1 x2 w2 case_y2 case_yin2")
for i in range(0,num_ip):
    case_y2.append(x1[i]*w1+x2[i]*w2)
    if(case_y2[i]>=theta):
       case_yin2.append(1)
    else:
       case_yin2.append(0)
    print(x1[i]," ", w1, " ",x2[i]," ",w2," ",case_y2[i]," ", case_yin2[i])
print("From the calculated net inputs it is possible to fire neuron from the given inputs so these weights are not suitable")
```

Output:

```
================= RESTART: C:/M.SC/Soft Computing/Coding/2A.py =================
Enter the number of inputs:4
For the 4 inputs calculate the net input using yin=x1w1+x2w2
Enter the input x1:1
Enter the input x2:1
Enter the input x1:1
Enter the input x2:0
Enter the input x1:0
Enter the input x2:1
Enter the input x1:0
Enter the input x2:0
x1= [1, 1, 0, 0]
x2= [1, 0, 1, 0]
Value of theta is 1
Case1: For calculating the net input we will take weights w1=w2=1
x1 w1 x2 w2 case_y1 case_yin1
1   1   1   1   2   1
1   1   0   1   1   1
0   1   1   1   1   1
0   1   0   1   0   0
From the calculated net inputs its not possible to fire neuron from the given inputs so these weights are not suitable
Case2: For calculating the net input we will take weights w1=1,w2=-1
x1 w1 x2 w2 case_y2 case_yin2
1   1   1   -1   0   0
1   1   0   -1   1   1
0   1   1   -1   -1   0
0   1   0   -1   0   0
From the calculated net inputs it is possible to fire neuron from the given inputs so these weights are not suitable
```

Practical 2B

Aim: Generate XOR function using McCulloch-Pitts neural net

Code:

```
import pandas as pd
print("Enter Weights")
w11=int(input('Weights w11='))
w12=int(input('Weights w12='))
w21=int(input('Weights w21='))
w22=int(input('Weights w22='))
v1=int(input('Weights v1='))
v2=int(input('Weights v2='))
print('Enter threshold value')
theta=int(input('theta='))
import numpy as np
x1=np.array([0,0,1,1])
x2=np.array([0,1,0,1])
z=np.array([0,1,1,0])
con=1
y1=np.zeros((4,))
y2=np.zeros((4,))
y=np.zeros((4,))
print(x1)
print(x2)
print(z)
print(y1)
print(y2)
print(y)
while con==1:
     zin1=np.zeros((4,))
     zin2=np.zeros((4,))
     zin1=x1*w11+x2*w12
     zin2=x1*w21+x2*w22
     print(zin1)
     print(zin2)
     for i in range(0,4):
         if(zin1[i]>=theta):
           y1[i]=1
```

```
        else:
           y1[i]=0
        if(zin2[i]>=theta):
           y2[i]=1
        else:
           y2[i]=0
    yin=np.array([])
    yin=y1*v1+y2*v2
    for i in range(0,4):
        if(yin[i]>=theta):
           y[i]=1
        else:
           y[i]=0
    print("yin",yin)
    print("Output of Net:")
    y=y.astype(int)
    print("y",y)
    print("z",z)
    if(np.array_equal(y,z)):
     con=0
    else:
       print("Net is not able to learn use another set of weigths and thresholds")
       print("Enter Weights")
       w11=int(input('Weights w11='))
       w12=int(input('Weights w12='))
       w21=int(input('Weights w21='))
       w22=int(input('Weights w22='))
       v1=int(input('Weights v1='))
       v2=int(input('Weights v2='))
       theta=int(input('theta='))
print("MP Output:")
print("Weights of neuron z1")
print(w11)
print(w21)
print("Weights of neuron z2")
print(w21)
print(w22)
print("Weights of neuron Y")
print(v1)
print(v2)
print("Threshold")
print(theta)
```

Output:

```
Enter Weights
Weights w11=1
Weights w12=-1
Weights w21=-1
Weights w22=1
Weights v1=1
Weights v2=1
Enter threshold value
theta=1
[0 0 1 1]
[0 1 0 1]
[0 1 1 0]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[ 0 -1  1  0]
[ 0  1 -1  0]
yin [0. 1. 1. 0.]
Output of Net:
y [0 1 1 0]
z [0 1 1 0]
MP Output:
Weights of neuron z1
1
-1
Weights of neuron z2
-1
1
Weights of neuron Y
1
1
Threshold
1
```

Practical 3A

Aim: Write a program to implement Hebb's rule.

Code:

```
import numpy as np
x1=np.array([1,1,1,-1,1,-1,1,1,1])
x2=np.array([1,1,1,1,-1,1,1,1,1])
b=0
y=np.array([1,-1])
wtold=np.zeros([9,])
wtnew=np.zeros([9,])
wtold=wtold.astype(int)
wtnew=wtnew.astype(int)
print("Expected output:",y)
print("First input with target =1")
for i in range(0,9):
    wtold[i]=wtold[i]+x1[i]*y[0]
    wtnew=wtold
    b=b+y[0]
print("Second input with target =-1")
for i in range(0,9):
    wtnew[i]=wtold[i]+x2[i]*y[1]
    wtnew=wtold
    b=b+y[1]
print("New weights are",wtnew)
print("Final bias is:",b)
```

Output:

```
================= RESTART: C:/M.SC/Soft Computing/Coding/3A.py =================
Expected output: [ 1 -1]
First input with target =1
Second input with target =-1
New weights are [ 0  0  0 -2  2 -2  0  0  0]
Final bias is: 0
```

Practical 3B

Aim: Write a program to implement of delta rule.

Code:

```
import numpy as np
import time
x=np.zeros([3,])
weights=np.zeros([3,])
desired=np.zeros([3,])
actual=np.zeros([3,])
for i in range(0,3):
    x[i]=float(input("Inital inputs:"))
for i in range(0,3):
    weights[i]=float(input("Inital weights:"))
for i in range(0,3):
    desired[i]=float(input("Desired Output:"))
a=float(input("Enter learning rate:"))
actual=x*weights
print("actual",actual)
print("desired",desired)
while True:
    if np.array_equal(desired,actual):
       break
    else:
       for i in range(0,3):
           weights[i]=weights[i]+a*(desired[i]-actual[i])
    actual=x*weights
    print("Weights ",weights)
    print("actual",actual)
    print("desired",desired)
print("*"*30)
print("Final Output")
print("Corrected Weights ",weights)
print("actual",actual)
print("desired",desired)
```

Output:

```
= RESTART: C:/M.SC/Soft Computing/Coding/3B.py
Inital inputs:1
Inital inputs:1
Inital inputs:1
Inital weights:1
Inital weights:1
Inital weights:1
Desired Output:2
Desired Output:3
Desired Output:4
Enter learning rate:1
actual [1. 1. 1.]
desired [2. 3. 4.]
Weights  [2. 3. 4.]
actual [2. 3. 4.]
desired [2. 3. 4.]
******************************
Final Output
Corrected Weights  [2. 3. 4.]
actual [2. 3. 4.]
desired [2. 3. 4.]
```

Practical 4A

Aim: Write a program for Back Propagation Algorithm

Code:

```
import numpy as np
import decimal
import math
v1 = np.array([0.6,0.3])
v2 = np.array([-0.1,0.4])
w = np.array([-0.2,0.4,0.1])
b1 = 0.3
b2 = 0.5
x1 = 0
x2 = 1
alpha = 0.25
print('Calculate net input to z1 layer')
zin1 = round(b1+x1*v1[0]+x2*v2[0],4)
print("z1 : ",zin1)
print('Calculate net input to z2 layer')
zin2 = round(b2+x1*v1[1]+x2*v2[1],4)
print("z2 : ",zin2)
print('Now applying activation function')
z1 = 1/(1+math.exp(-zin1))
z1 = round(z1,4)
z2 = 1/(1+math.exp(-zin2))
z2 = round(z2,4)
print('z1 : ',z1,'\n z2 : ',z2)
print('Calculate net input to output layer')
yin = w[0]+z1*w[1]+z2*w[2]
print('yin : ',yin)
y = 1/(1+math.exp(-yin))
print('y = ',y)
fyin = y*(1-y)
dk = (1-y)*fyin
print('dk : ',dk)
dw1 = alpha*dk*z1
dw2 = alpha*dk*z2
dw0 = alpha*dk
print('Computing error portion in delta')
din1 = dk*w[1]
```

```
din2 = dk*w[2]
print('din1 : ',din1)
print('din2 : ',din2)
print('Error in delta')
fzin1 = z1*(1-z1)
print('fzin1 : ',fzin1)
d1 = din1*fzin1
fzin2 = z2*(1-z2)
print('fzin2 : ',fzin2)
d2 = din2*fzin2
print('d1 : ',d1)
print('d2 : ',d2)
print('Changes in weights between input and hidden layer')
dv11 = alpha*d1*x1
dv21 = alpha*d1*x2
dv01 = alpha*d1
dv12 = alpha*d2*x1
dv22 = alpha*d2*x2
dv02 = alpha*d2
print('dv11 : ',dv11)
print('dv21 : ',dv21)
print('dv01 : ',dv01)
print('dv12 : ',dv12)
print('dv22 : ',dv22)
print('dv02 : ',dv02)
print('Final weight of network')
v1[0] = v1[0]+dv11
v1[1] = v1[1]+dv12
print('v1 : ',v1)
v2[0] = v2[0]+dv21
v2[1] = v2[1]+dv22
print('v2 : ',v2)
w[0] = w[0]+dw0
w[1] = w[1]+dw1
w[2] = w[2]+dw2
b1 = b1 + dv01
b2 = b2 + dv01
print('w : ',w)
print('bias b1 = ',b1," b2 = ",b2)
```

Output:

```
Calculate net input to z1 layer
z1 :  0.2
Calculate net input to z2 layer
z2 :  0.9
Now applying activation function
z1 :  0.5498
 z2 :  0.7109
Calculate net input to output layer
yin :  0.09101
y =  0.5227368084248941
dk :  0.11906907074145694
Computing error portion in delta
din1 :  0.04762762829658278
din2 :  0.011906907074145694
Error in delta
fzin1 :  0.24751996
fzin2 :  0.20552119000000002
d1 :  0.011788788650865037
d2 :  0.0024471217110978417
Changes in weights between input and hidden layer
dv11 :  0.0
dv21 :  0.0029471971627162592
dv01 :  0.0029471971627162592
dv12 :  0.0
dv22 :  0.0006117804277744604
dv02 :  0.0006117804277744604
Final weight of network
v1 :  [0.6 0.3]
v2 :  [-0.0970528   0.40061178]
w :  [-0.17023273  0.41636604  0.12116155]
bias b1 =  0.30294719716271623  b2 =  0.5029471971627163
```

Practical 4B

Aim: Write a program for error Backpropagation algorithm.

Code:

```
import math
a0 = -1
t = -1
w10 = float(input("Enter weight for first network : "))
b10 = float(input("Enter bias for first network : "))
w20 = float(input("Enter weight for second network : "))
b20 = float(input("Enter bias for second network : "))
c = float(input("Enter learning Coefficient : "))
n1 = float(w10*c+b10)
a1 = math.tanh(n1)
n2 = float(w20*c+b20)
a2 = math.tanh(n2)
e = t-a2
s2 = -2*(1-a2*a2)*e
s1 = (1-a1*a1)*w20*s2
w21 = w20-(c*s2*a1)
w11 = w10-(c*s1*a0)
b21 = b20-(c*s2)
b11 = b10-(c*s1)
print('Updated weight at first network w11 ',w11)
print('Updated weight at Second network w21 ',w21)
print('Updated bias at first network b10 ',b10)
print('Updated bias at first network b20 ',b20)
```

Output:

```
= RESTART: C:/M.SC/Soft Computing/Coding/4B.py
Enter weight for first network : 11
Enter bias for first network : 0.5
Enter weight for second network : 12
Enter bias for second network : 0.7
Enter learning Coefficient : 12
Updated weight at first network w11  11.0
Updated weight at Second network w21  12.0
Updated bias at first network b10  0.5
Updated bias at first network b20  0.7
```

Practical 7A

Aim: Write a program for Linear separation.

Code:

```
import numpy as np
import matplotlib.pyplot as plt

def create_distance_function(a, b, c):
    """Returns a function that calculates the distance from a point to a line."""
    def distance(x, y):
        """Return tuple(d, pos):
           d is the distance,
           If pos == -1 point is below the line,
           0 on the line and +1 if above the line
        """
        nom = a * x + b * y + c
        if nom == 0:
            pos = 0
        elif (nom < 0 and b < 0) or (nom > 0 and b > 0):
            pos = -1
        else:
            pos = 1
        return (np.absolute(nom) / np.sqrt(a ** 2 + b ** 2), pos)

    return distance

points = [(3.5, 1.8), (1.1, 3.9)]
fig, ax = plt.subplots()
ax.set_xlabel("sweetness")
ax.set_ylabel("sourness")
ax.set_xlim([-1, 6])
ax.set_ylim([-1, 8])
X = np.arange(-0.5, 5, 0.1)

size = 10
for index, (x, y) in enumerate(points):
    if index == 0:
        ax.plot(x, y, "o", color="darkorange", markersize=size)
    else:
        ax.plot(x, y, "oy", markersize=size)
```

```
step = 0.05
for x in np.arange(0, 1 + step, step):
    slope = np.tan(np.arccos(x))
    dist4line1 = create_distance_function(slope, -1, 0)
    Y = slope * X
    results = []
    for point in points:
        results.append(dist4line1(*point))
    if results[0][1] != results[1][1]:
        ax.plot(X, Y, "g-")
    else:
        ax.plot(X, Y, "r-")

plt.show()
```

Output:

Practical 7B

Aim: Write a program for Hopfield network model for associative memory

Code:

```
File Edit Format Run Options Window Help
import matplotlib.pyplot as plt
from neurodynex3.hopfield_network import network, pattern_tools, plot_tools

class PatternFactory(pattern_tools.PatternFactory):
    def create_checkerboard(self):
        pw = np.ones((self.pattern_length, self.pattern_length), dtype=int)
        for i in range(self.pattern_length):
            for j in range(self.pattern_length):
                pw[i, j] = (-1) ** (i + j)
        return pw

pattern_size = 5
hopfield_net = network.HopfieldNetwork(nr_neurons=pattern_size**2)
factory = PatternFactory(pattern_size, pattern_size)

checkboard = factory.create_checkerboard()
pattern_list = [checkboard]
pattern_list.extend(factory.create_random_pattern_list(nr_patterns=3, on_probability=0.5))

plot_tools.plot_pattern_list(pattern_list)
overlap_matrix = pattern_tools.compute_overlap_matrix(pattern_list)
plot_tools.plot_overlap_matrix(overlap_matrix)  # Let matplotlib create the axes automatically
plt.show()

hopfield_net.store_patterns(pattern_list)
noisy_init_state = pattern_tools.flip_n(checkboard, nr_of_flips=4)
hopfield_net.set_state_from_pattern(noisy_init_state)

# Increase the number of steps for simulation
nr_steps = 10

# Run the simulation for a larger number of steps
states = hopfield_net.run_with_monitoring(nr_steps=nr_steps)
states_as_patterns = factory.reshape_patterns(states)
plot_tools.plot_state_sequence_and_overlap(states_as_patterns, pattern_list, reference_idx=0, suptitle="Network Dynamics")

plt.show()
```

Output:

pattern overlap m(i,k)
pattern i
pattern k
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
1
2
3
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

S0
S1
S2
S3
S4
m = 0.68
m = 1.0
m = 1.0
m = 1.0
m = 1.0
1.0
0.5
0.0
−0.5
−1.0
0 1 2 3

Practical 8A

Aim: Membership and Identity Operators | in, not in.

Code:

```
def overlapping(list1,list2):
    c=0
    d=0
    for i in list1:
        c+=1
    for i in list2:
        d+=1
    for i in range(0,c):
        for j in range(0,d):
            if(list1[i]==list2[j]):
                return 1
    return 0
list1 = [1,2,3,4,5]
list2 = [6,7,8,9]
if(overlapping(list1,list2)):
   print("overlapping")
else:
   print("not overlapping")
```

Output:

```
================= RESTART: C:/M.SC/Soft Computing/Coding/8A.py =================
not overlapping
```

Practical 8A

Aim: Membership and Identity Operators is, is not

Code:

```
x=5
if(type(x) is int):
    print("true")
else:
    print("false")
x=5.2
if(type(x) is not int):
    print("true")
else:
    print("false")
```

Output:

```
================= RESTART: C:/M.SC/Soft Computing/Coding/8B.py =================
true
true
```

Practical 9A

Aim: Find ratios using fuzzy logic

Code:

```
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
s1="I Love ArtificialIntelligenece"
s2="I am loving ArtificialIntelligence"
print("Fuzzywuzyy Ratio:",fuzz.ratio(s1,s2))
print("Fuzzywuzyy Ratio:",fuzz.partial_ratio(s1,s2))
print("Fuzzywuzyy Ratio:",fuzz.token_sort_ratio(s1,s2))
print("Fuzzywuzyy Ratio:",fuzz.token_set_ratio(s1,s2))
print("Fuzzywuzyy Ratio:",fuzz.WRatio(s1,s2))
query="artificial intelligence"
choices=["artificial intelligence","arts intelligence"," a intelligence"]
print("List of ratios")
print(process.extract(query,choices),'\n')
print("Best among the above list: ",process.extractOne(query,choices))
```

Output:

```
================= RESTART: C:/M.SC/Soft Computing/Coding/9A.py =================
Fuzzywuzyy Ratio: 84
Fuzzywuzyy Ratio: 87
Fuzzywuzyy Ratio: 88
Fuzzywuzyy Ratio: 88
Fuzzywuzyy Ratio: 88
List of ratios
[('artificial intelligence', 100), (' a intelligence', 86), ('arts intelligence'
, 80)]

Best among the above list:  ('artificial intelligence', 100)
```

Practical 9B

Aim: Solve Tipping problem using fuzzy logic

Code:

```
#!pip install scikit-fuzzy
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
quality=ctrl.Antecedent(np.arange(0,11,1),'quality')
service=ctrl.Antecedent(np.arange(0,11,1),'service')
tip=ctrl.Consequent(np.arange(0,26,1),'tip')
quality.automf(3) # automembership function
service.automf(3)
tip['low']=fuzz.trimf(tip.universe,[0,0,13]) #trigural membership function
tip['medium']=fuzz.trimf(tip.universe,[0,13,25])
tip['high']=fuzz.trimf(tip.universe,[13,25,25])
quality['average'].view()
service['poor'].view()
tip['high'].view()
rule1=ctrl.Rule(quality['poor']|service['poor'],tip['low'])
rule2=ctrl.Rule(quality['average']|service['average'],tip['medium'])
rule3=ctrl.Rule(quality['good']|service['good'],tip['high'])
tipping_ctrl=ctrl.ControlSystem([rule1,rule2,rule3])
tipping=ctrl.ControlSystemSimulation(tipping_ctrl)
tipping.input['quality']=6.5
tipping.input['service']=9.8
tipping.compute()
print(tipping.output['tip'])
```

Output:

Quality(average):

Service(poor):

tip(high):

Output:

```
= RESTART: C:/M.SC/Soft Computing/Coding/9B.py
14.79822137450634
```

Practical 10A

Aim: Implementation of Simple genetic algorithm

Code:

```
import numpy as np
import pygad

X = np.array([4, -2, 3.5, 5, -11, -4.7])
desired_y = 44

def fitness_function(instance, solution, solution_idx):
    output = np.sum(solution * X)
    fitness = 1.0 / np.abs(output - desired_y)
    return fitness

num_generations = 50
num_parents_mating = 4
sol_per_pop = 8
num_genes = len(X)
init_range_low = -2
init_range_high = 5
parent_selection_type = "sss"
keep_parents = 1
crossover_type = "single_point"
mutation_type = "random"
mutation_percent_genes = 10

# Define the initial population
initial_population = np.random.uniform(low=init_range_low, high=init_range_high, size=(sol_per_pop, num_genes))

# Create an instance of the GA class
ga_instance = pygad.GA(num_generations=num_generations,
                       num_parents_mating=num_parents_mating,
                       fitness_func=fitness_function,
                       sol_per_pop=sol_per_pop,
                       num_genes=num_genes,
                       initial_population=initial_population,
                       parent_selection_type=parent_selection_type,
                       keep_parents=keep_parents,
                       crossover_type=crossover_type,
```

```
                       mutation_type=mutation_type,
                       mutation_percent_genes=mutation_percent_genes)

# Run the GA
ga_instance.run()

# Get the best solution
solution, solution_fitness, solution_idx = ga_instance.best_solution()

print("Parameters of the best solution:", solution)
print("Fitness value of the best solution:", solution_fitness)
print("Index of the best solution:", solution_idx)
```

Output:

```
Parameters of the best solution: [-0.29092073 -1.52560282  3.65355998  3.53501592 -1.07219487  0.02913768]
Fitness value of the best solution: 137.76627768821123
Index of the best solution: 0
```