****

**Anjuman-I-Islam’s**

**AKBAR PEERBHOY COLLEGE OF COMMERCE AND ECONOMICS**M.S Road Do Taki, Mumbai- 400008

**Data Science**

**Submitted by**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Guided by – Prof. Hanna Saadya Shaikh**

**University of Mumbai**

**MSC IT (Information Technology) – Semester 1**

**Academic Year 2024-2025**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Practical No** |  | **Title** | **Date** | **Sign** |
| **Practical 1** |  | **Create a data model using Cassandra** | **08-10-23** |  |
| **Practical 2** |  | **Conversion from different formats to HORUS** | **14-10-23** |  |
|  | A | CSV to HORUS |  |  |
|  | B | XML to HORUS |  |  |
|  | C | JSON to HORUS |  |  |
|  | D | MySQL Database to HORUS |  |  |
|  | E | Picture to HORUS |  |  |
|  | F | Video to HORUS |  |  |
|  | G | Audio to HORUS |  |  |
| **Practical 3** |  | **Utilities and Auditing** | **21-10-23** |  |
|  | A | Fixer Utilities |  |  |
|  | B | Data Binning or Bucketing |  |  |
|  | C | Aggregation of Data |  |  |
|  | D | Outlier Detection |  |  |
|  | E | Audit |  |  |
| **Practical 4** |  | **Retrieving Data** | **28-10-23** |  |
|  | A | Data Processing |  |  |
|  | B | Retrieve different attributes of data |  |  |
|  | C | Data Pattern |  |  |
|  | D | Loading IP\_DATA\_ALL.csv |  |  |
|  | E | Building a diagram for scheduling of jobs |  |  |
|  | F | Connecting other Data Sources |  |  |
| **Practical 5** |  | **Assessing Data** | **04-11-23** |  |
| **Practical 6** |  | **Processing Data** | **11-11-23** |  |
| **Practical 7** |  | **Transforming Data** | **18-11-23** |  |
| **Practical 8** |  | **Organizing Data** | **25-11-23** |  |
|  | A | Horizontal Style |  |  |
|  | B | Vertical Style |  |  |
|  | C | Island Style |  |  |
|  | D | Secure Vault Style |  |  |
| **Practical 9** |  | **Reporting Data** | **02-12-23** |  |
|  | A | Create a network routing diagram |  |  |
|  | B | Directed Acyclic Graph |  |  |
|  | C | Graphics |  |  |

**Practical 1**

**Creating Data Model using Cassandra**

**Cassandra Data Model**

**Code:**

Create keyspace keyspace1 with replication = {'class':'SimpleStrategy','replication\_factor': 3};

Use keyspace1;

Create table dept ( dept\_id int PRIMARY KEY, dept\_name text, dept\_loc text);

Create table emp ( emp\_id int PRIMARY KEY, emp\_name text, dept\_id int, email text, phone text );

Insert into dept (dept\_id, dept\_name, dept\_loc) values (1001, 'Accounts', 'Mumbai');

Insert into dept (dept\_id, dept\_name, dept\_loc) values (1002, 'Marketing', 'Delhi');

Insert into dept (dept\_id, dept\_name, dept\_loc) values (1003, 'HR', 'Chennai');

Insert into emp ( emp\_id, emp\_name, dept\_id, email, phone ) values (1001, 'ABCD',1001, 'abcd@company.com', '1122334455');

Insert into emp ( emp\_id, emp\_name, dept\_id, email, phone ) values (1002, 'DEFG',1001, 'defg@company.com', '2233445566');

Insert into emp ( emp\_id, emp\_name, dept\_id, email, phone ) values (1003, 'GHIJ',1002, 'ghij@company.com', '3344556677');

Insert into emp ( emp\_id, emp\_name, dept\_id, email, phone ) values (1004, 'JKLM',1002, 'jklm@company.com', '4455667788');

Insert into emp ( emp\_id, emp\_name, dept\_id, email, phone ) values (1005, 'MNOP',1003, 'mnop@company.com', '5566778899');

Insert into emp ( emp\_id, emp\_name, dept\_id, email, phone ) values (1006, 'MNOP',1003, 'mnop@company.com', '5566778844');

select \* from emp;

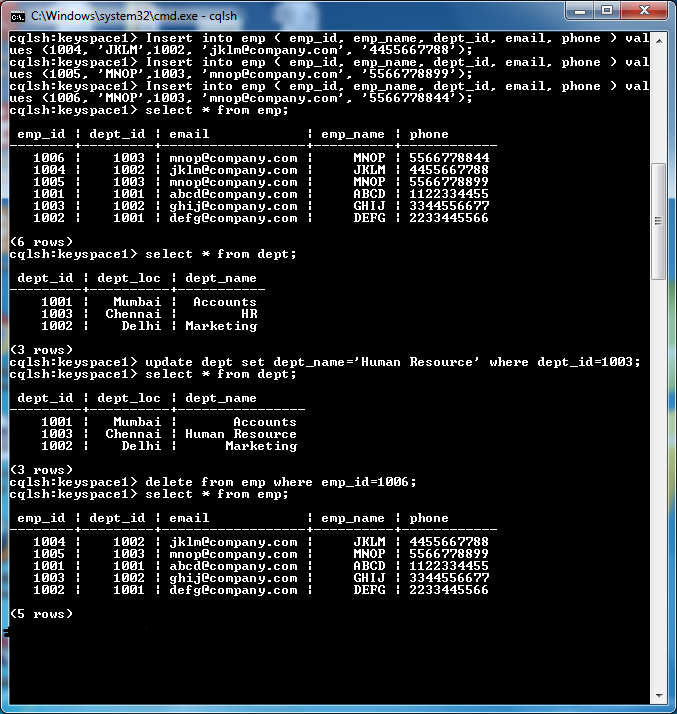
select \* from dept;

update dept set dept\_name='Human Resource' where dept\_id=1003;

delete from emp where emp\_id=1006;

select \* from emp;

**Output:**

****

**Practical 2**

**Conversion from different formats to HORUS formats**

1. **Csv to Horus**

**Code:**

import pandas as pd

# Input Agreement ============================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/Country\_Code.csv'

InputData=pd.read\_csv(sInputFileName,encoding="latin-1")

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

print(InputData)

print('=====================================================')

# Processing Rules ===========================================

ProcessData=InputData

# Remove columns ISO-2-Code and ISO-3-CODE

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

ProcessData.set\_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True)

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

print(ProcessData)

print('=====================================================')

# Output Agreement ===========================================

OutputData=ProcessData

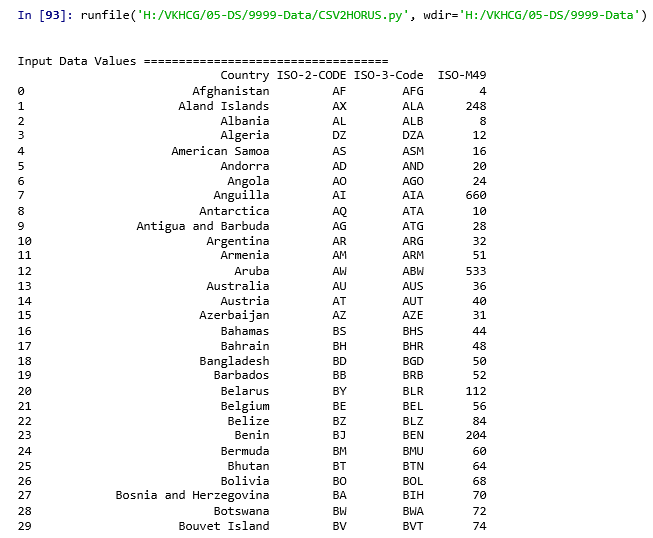
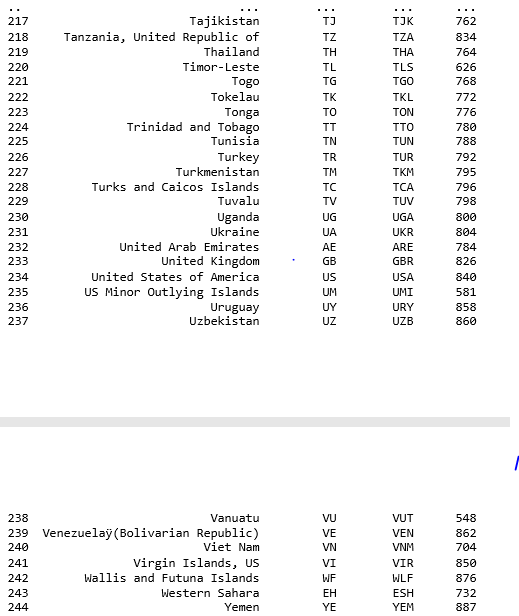
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'

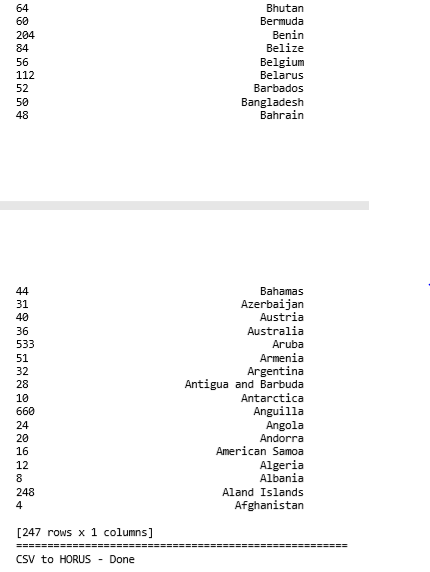
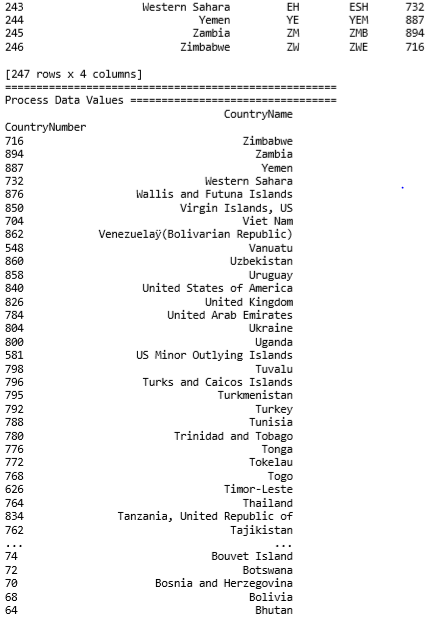
OutputData.to\_csv(sOutputFileName, index = False)

print('CSV to HORUS - Done')

# Utility done

**Output:**



1. **XML to Horus**

**Code:**

import pandas as pd

import xml.etree.ElementTree as ET

def df2xml(data):

header = data.columns

root = ET.Element('root')

for row in range(data.shape[0]):

entry = ET.SubElement(root,'entry')

for index in range(data.shape[1]):

schild=str(header[index])

child = ET.SubElement(entry, schild)

if str(data[schild][row]) != 'nan':

child.text = str(data[schild][row])

else:

child.text = 'n/a'

entry.append(child)

result = ET.tostring(root)

return result

def xml2df(xml\_data):

root = ET.XML(xml\_data)

all\_records = []

for i, child in enumerate(root):

record = {}

for subchild in child:

record[subchild.tag] = subchild.text

all\_records.append(record)

return pd.DataFrame(all\_records)

sInputFileName='C:/VKHCG/05-DS/9999-Data/Country\_Code.xml'

InputData = open(sInputFileName).read()

print('=====================================================')

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

print('=====================================================')

print(InputData)

print('=====================================================')

#=============================================================

# Processing Rules ===========================================

#=============================================================

ProcessDataXML=InputData

# XML to Data Frame

ProcessData=xml2df(ProcessDataXML)

# Remove columns ISO-2-Code and ISO-3-CODE

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

ProcessData.set\_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True)

print('=====================================================')

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

print('=====================================================')

print(ProcessData)

print('=====================================================')

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('=====================================================')

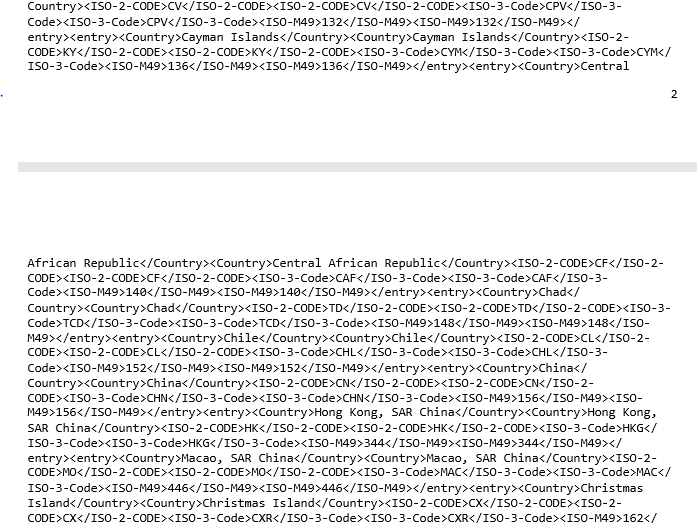
print('XML to HORUS - Done')

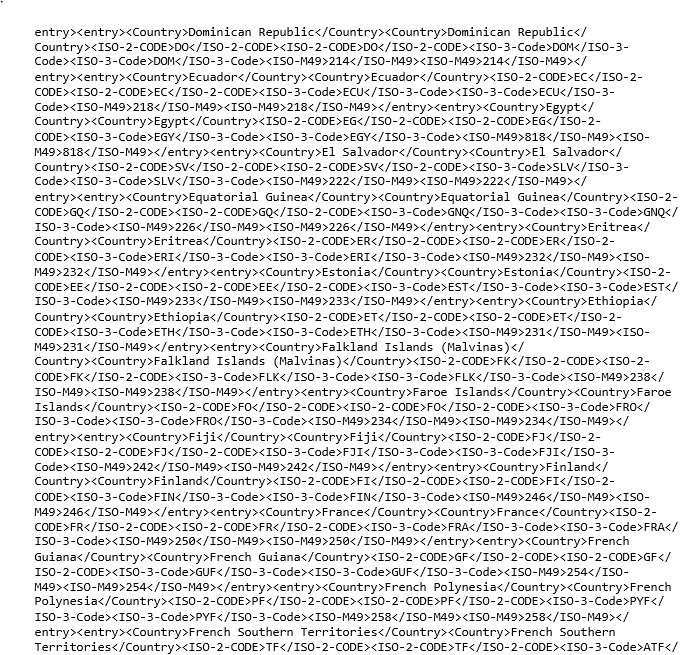
print('=====================================================')

# Utility done ===============================================

**Output:**







1. **JSON to HORUS Format**

**Code:**

# Utility Start JSON to HORUS =================================

# Standard Tools

#=============================================================

import pandas as pd

# Input Agreement ============================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/Country\_Code.json'

InputData=pd.read\_json(sInputFileName, orient='index', encoding="latin-1")

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

print(InputData)

print('=====================================================')

# Processing Rules ===========================================

ProcessData=InputData

# Remove columns ISO-2-Code and ISO-3-CODE

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

ProcessData.set\_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True)

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

print(ProcessData)

print('=====================================================')

# Output Agreement ===========================================

OutputData=ProcessData

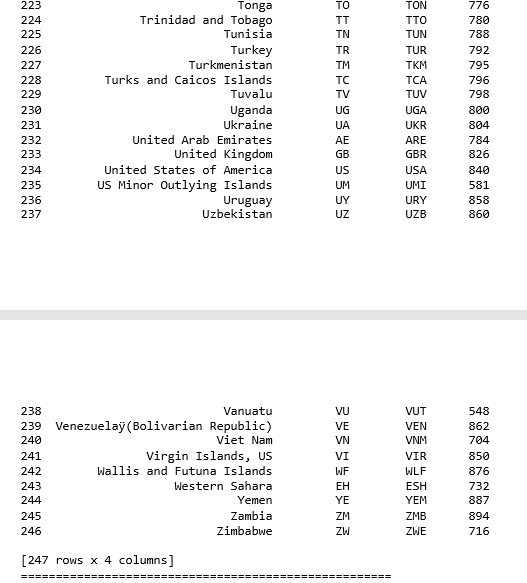
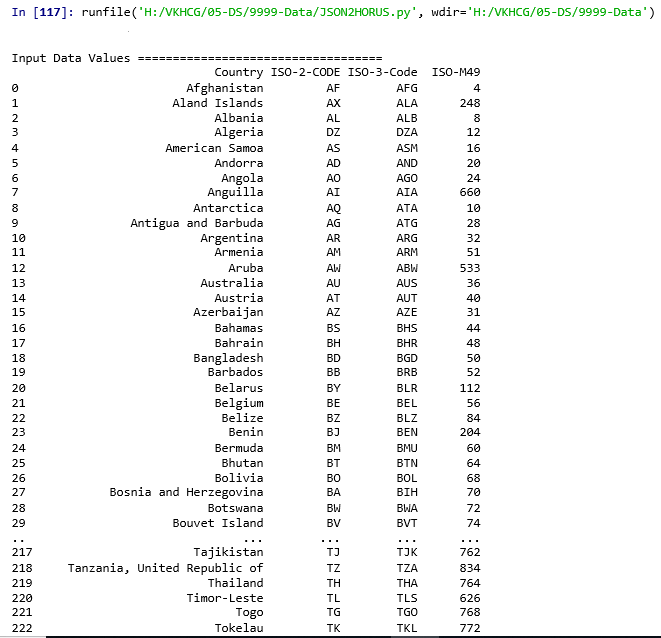
sOutputFileName='c:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv'

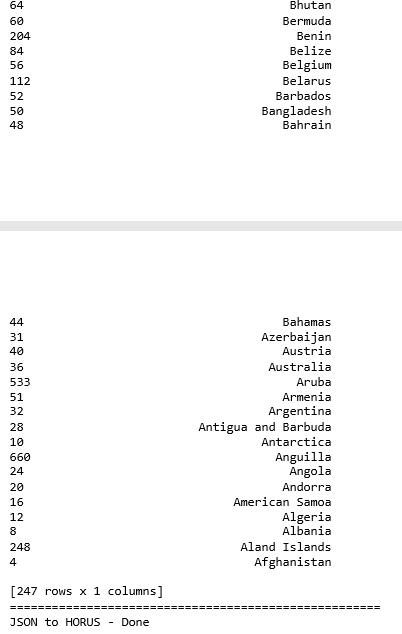
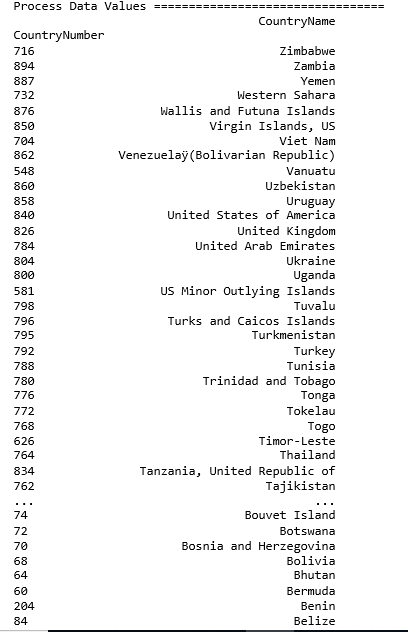
OutputData.to\_csv(sOutputFileName, index = False)

print('JSON to HORUS - Done')

# Utility done ===============================================

**Output:**





1. **MySql Database to HORUS Format**

**Code :**

import pandas as pd

import sqlite3 as sq

# Input Agreement ============================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/utility.db'

sInputTable='Country\_Code'

conn = sq.connect(sInputFileName)

sSQL='select \* FROM ' + sInputTable + ';'

InputData=pd.read\_sql\_query(sSQL, conn)

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

print(InputData)

print('=====================================================')

# Processing Rules ===========================================

ProcessData=InputData

# Remove columns ISO-2-Code and ISO-3-CODE

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

ProcessData.set\_index('CountryNumber', inplace=True)

# Sort data by CurrencyNumber

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True)

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

print(ProcessData)

print('=====================================================')

# Output Agreement ===========================================

OutputData=ProcessData

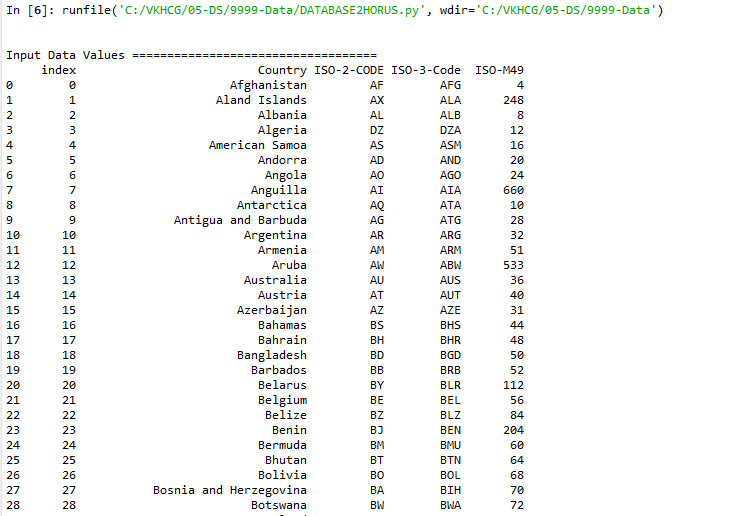
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'

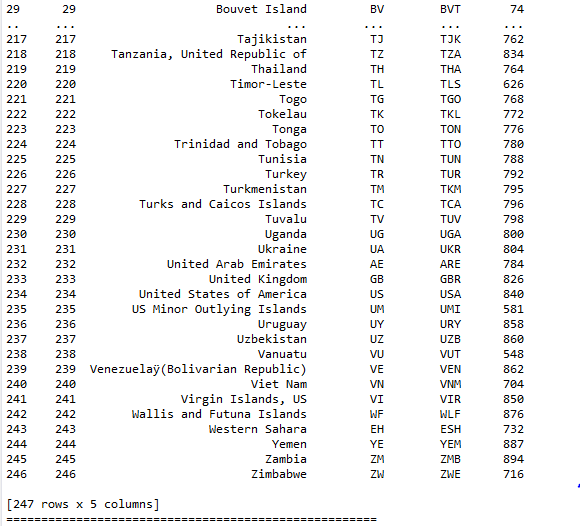
OutputData.to\_csv(sOutputFileName, index = False)

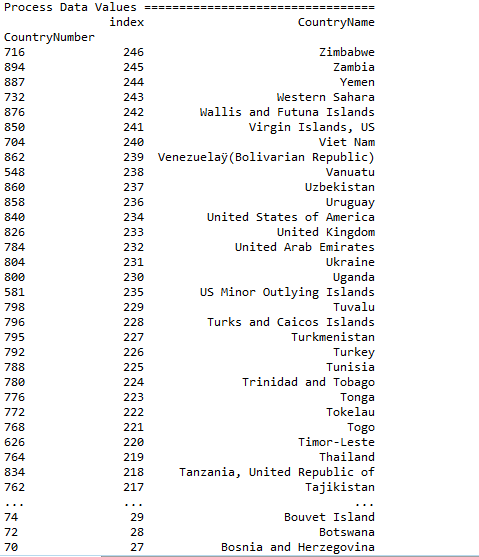
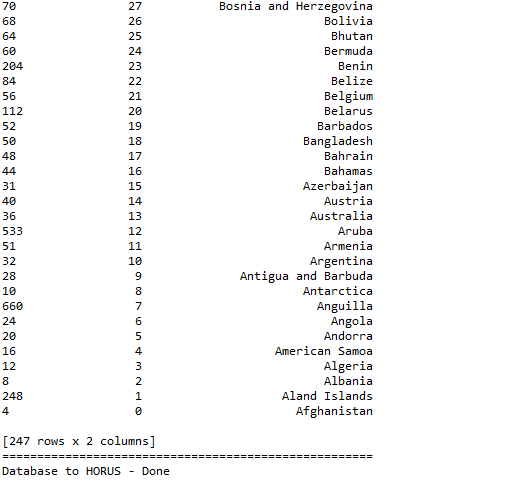
print('Database to HORUS - Done')

# Utility done ===============================================

**Output:**

****

****

** **

1. **Picture (JPEG) to HORUS Format**

**Code:**

# Utility Start Picture to HORUS =================================

# Standard Tools

#=============================================================

from scipy.misc import imread

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

# Input Agreement ============================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/Angus.jpg'

InputData = imread(sInputFileName, flatten=False, mode='RGBA')

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

print('X: ',InputData.shape[0])

print('Y: ',InputData.shape[1])

print('RGBA: ', InputData.shape[2])

print('=====================================================')

# Processing Rules ===========================================

ProcessRawData=InputData.flatten()

y=InputData.shape[2] + 2

x=int(ProcessRawData.shape[0]/y)

ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))

sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']

ProcessData.columns=sColumns

ProcessData.index.names =['ID']

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

print('Columns :',ProcessData.shape[1])

print('=====================================================')

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

print('=====================================================')

plt.imshow(InputData)

plt.show()

print('=====================================================')

# Output Agreement ===========================================

OutputData=ProcessData

print('Storing File')

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('=====================================================')

print('Picture to HORUS - Done')

print('=====================================================')

1. **i. Video to HORUS Format**

**Code:**

**Movies to Frames**

import os

import shutil

import cv2

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/dog.mp4'

sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp'

if os.path.exists(sDataBaseDir):

shutil.rmtree(sDataBaseDir)

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

print('=====================================================')

print('Start Movie to Frames')

print('=====================================================')

vidcap = cv2.VideoCapture(sInputFileName)

success,image = vidcap.read()

count = 0

while success:

success,image = vidcap.read()

sFrame=sDataBaseDir + str('/dog-frame-' + str(format(count, '04d'))+ '.jpg')

print('Extracted: ', sFrame)

cv2.imwrite(sFrame, image)

if os.path.getsize(sFrame) == 0:

count += -1

os.remove(sFrame)

print('Removed: ', sFrame)

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

break

count += 1

print('=====================================================')

print('Generated : ', count, ' Frames')

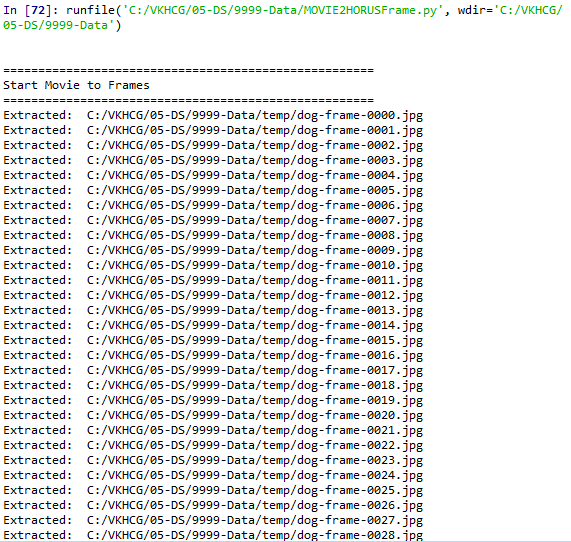
print('=====================================================')

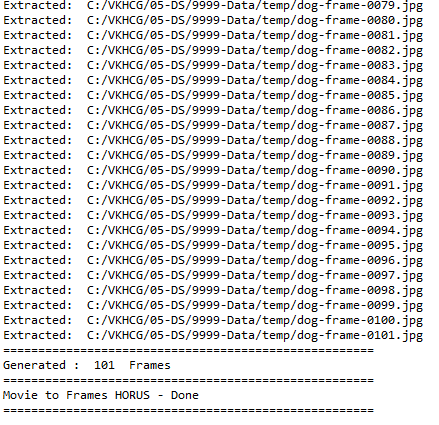
print('Movie to Frames HORUS - Done')

print('=====================================================')

# Utility done ==============================================**=**

**Output:**

****

****

1. **Frames to Horus**

from scipy.misc import imread

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import os

# Input Agreement ============================================

sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp'

f=0

for file in os.listdir(sDataBaseDir):

if file.endswith(".jpg"):

f += 1

sInputFileName=os.path.join(sDataBaseDir, file)

print('Process : ', sInputFileName)

InputData = imread(sInputFileName, flatten=False, mode='RGBA')

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

print('X: ',InputData.shape[0])

print('Y: ',InputData.shape[1])

print('RGBA: ', InputData.shape[2])

print('=====================================================')

# Processing Rules ===========================================

ProcessRawData=InputData.flatten()

y=InputData.shape[2] + 2

x=int(ProcessRawData.shape[0]/y)

ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))

ProcessFrameData['Frame']=file

print('=====================================================')

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

print('=====================================================')

plt.imshow(InputData)

plt.show()

if f == 1:

ProcessData=ProcessFrameData

else:

ProcessData=ProcessData.append(ProcessFrameData)

if f > 0:

sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha','FrameName']

ProcessData.columns=sColumns

print('=====================================================')

ProcessFrameData.index.names =['ID']

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

print('Columns :',ProcessData.shape[1])

print('=====================================================')

# Output Agreement ===========================================

1. **Audio to HORUS Format**

**Code:**

# Utility Start Audio to HORUS ===============================

# Standard Tools

#=============================================================

from scipy.io import wavfile

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

#=============================================================

def show\_info(aname, a,r):

print ('----------------')

print ("Audio:", aname)

print ('----------------')

print ("Rate:", r)

print ('----------------')

print ("shape:", a.shape)

print ("dtype:", a.dtype)

print ("min, max:", a.min(), a.max())

print ('----------------')

plot\_info(aname, a,r)

#=============================================================

def plot\_info(aname, a,r):

sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz'

plt.title(sTitle)

sLegend=[]

for c in range(a.shape[1]):

sLabel = 'Ch' + str(c+1)

sLegend=sLegend+[str(c+1)]

plt.plot(a[:,c], label=sLabel)

plt.legend(sLegend)

plt.show()

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("2 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2']

ProcessData.columns=sColumns

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'

OutputData.to\_csv(sOutputFileName, index = False)

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("4 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2','Ch3', 'Ch4']

ProcessData.columns=sColumns

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'

OutputData.to\_csv(sOutputFileName, index = False)

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("6 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']

ProcessData.columns=sColumns

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'

OutputData.to\_csv(sOutputFileName, index = False)

#=============================================================

sInputFileName='C:/VKHCG/05-DS/9999-Data/8ch-sound.wav'

print('=====================================================')

print('Processing : ', sInputFileName)

print('=====================================================')

InputRate, InputData = wavfile.read(sInputFileName)

show\_info("8 channel", InputData,InputRate)

ProcessData=pd.DataFrame(InputData)

sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']

ProcessData.columns=sColumns

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'

OutputData.to\_csv(sOutputFileName, index = False)

print('=====================================================')

print('Audio to HORUS - Done')

**Practical 3**

**Utilities and Auditing**

1. **Fixers Utilities:**

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

#---------------------------- Program to Demonstrate Fixers utilities -------------------

import string

import datetime as dt

**# 1 Removing leading or lagging spaces from a data entry**

print('#1 Removing leading or lagging spaces from a data entry');

baddata = " Data Science with too many spaces is bad!!! "

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

cleandata=baddata.strip()

print('>',cleandata,'<')

**# 2 Removing nonprintable characters from a data entry**

print('#2 Removing nonprintable characters from a data entry')

printable = set(string.printable)

baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"

cleandata=''.join(filter(lambda x: x in string.printable,baddata))

print('Bad Data : ',baddata);

print('Clean Data : ',cleandata)

**# 3 Reformatting data entry to match specific formatting criteria**.

# Convert YYYY/MM/DD to DD Month YYYY

print('# 3 Reformatting data entry to match specific formatting criteria.')

baddate = dt.date(2019, 10, 31)

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

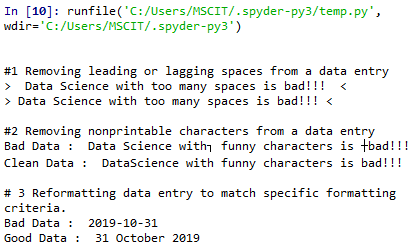
gooddate = dt.datetime.strptime(baddata,'%Y-%m-%d')

gooddata=format(gooddate,'%d %B %Y')

print('Bad Data : ',baddata)

print('Good Data : ',gooddata)

**Output:**



1. **Averaging of Data**

**Code:**

import pandas as pd

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

InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv'

Base='C:/VKHCG'

print('################################')

print('Working Base :',Base, ' using ')

print('################################')

sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False,

usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")

IP\_DATA\_ALL.rename(columns={'Place Name': 'Place\_Name'}, inplace=True)

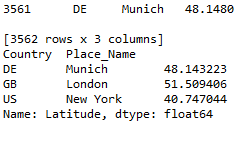
AllData=IP\_DATA\_ALL[['Country', 'Place\_Name','Latitude']]

print(AllData)

MeanData=AllData.groupby(['Country', 'Place\_Name'])['Latitude'].mean()

print(MeanData)

**Output:**



**D. Outlier Detection**

**Code:**

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

# -\*- coding: utf-8 -\*-

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

import pandas as pd

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

InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv'

Base='C:/VKHCG'

print('################################')

print('Working Base :',Base)

print('################################')

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

sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False,

usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")

IP\_DATA\_ALL.rename(columns={'Place Name': 'Place\_Name'}, inplace=True)

LondonData=IP\_DATA\_ALL.loc[IP\_DATA\_ALL['Place\_Name']=='London']

AllData=LondonData[['Country', 'Place\_Name','Latitude']]

print('All Data')

print(AllData)

MeanData=AllData.groupby(['Country', 'Place\_Name'])['Latitude'].mean()

StdData=AllData.groupby(['Country', 'Place\_Name'])['Latitude'].std()

print('Outliers')

UpperBound=float(MeanData+StdData)

print('Higher than ', UpperBound)

OutliersHigher=AllData[AllData.Latitude>UpperBound]

print(OutliersHigher)

LowerBound=float(MeanData-StdData)

print('Lower than ', LowerBound)

OutliersLower=AllData[AllData.Latitude<LowerBound]

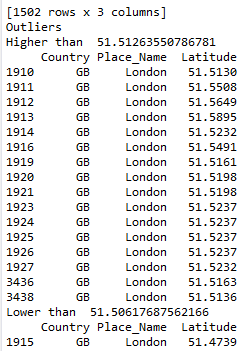
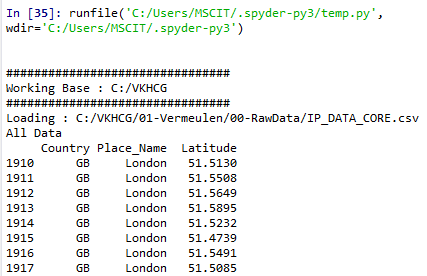
print(OutliersLower)

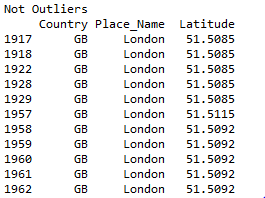
print('Not Outliers')

OutliersNot=AllData[(AllData.Latitude>=LowerBound) & (AllData.Latitude<=UpperBound)]

print(OutliersNot)

**Output:**





1. **Logging**

**Code:**

import sys

import os

import logging

import uuid

import shutil

import time

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']

sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','

06-Report']

sLevels=['debug','info','warning','error']

for sCompany in sCompanies:

sFileDir=Base + '/' + sCompany

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

for sLayer in sLayers:

log = logging.getLogger() # root logger

for hdlr in log.handlers[:]: # remove all old handlers

log.removeHandler(hdlr)

sFileDir=Base + '/' + sCompany + '/' + sLayer + '/Logging'

if os.path.exists(sFileDir):

shutil.rmtree(sFileDir)

time.sleep(2)

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

skey=str(uuid.uuid4())

sLogFile=Base + '/' + sCompany + '/' + sLayer + '/Logging/

Logging\_'+skey+'.log'

print('Set up:',sLogFile)

logging.basicConfig(level=logging.DEBUG,

format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',

datefmt='%m-%d %H:%M',

filename=sLogFile,

filemode='w')

console = logging.StreamHandler()

console.setLevel(logging.INFO)

formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')

console.setFormatter(formatter)

logging.getLogger('').addHandler(console)

logging.info('Practical Data Science is fun!.')

for sLevel in sLevels:

sApp='Application-'+ sCompany + '-' + sLayer + '-' + sLevel

logger = logging.getLogger(sApp)

if sLevel == 'debug':

logger.debug('Practical Data Science logged a debugging message.')

if sLevel == 'info':

logger.info('Practical Data Science logged information message.')

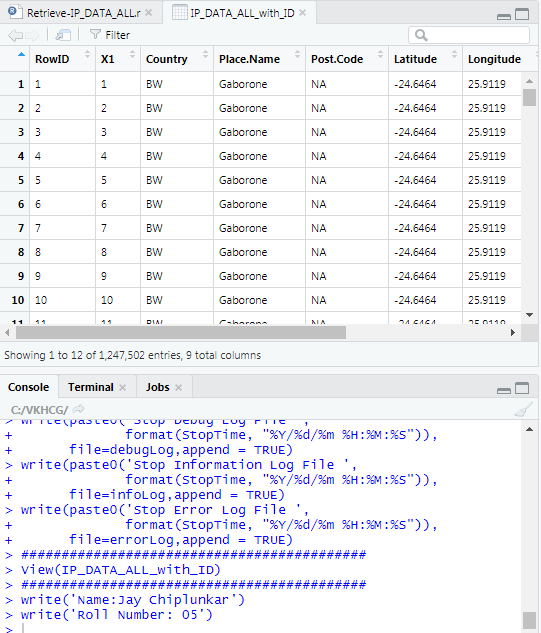
if sLevel == 'warning':

logger.warning('Practical Data Science logged a warning message.')

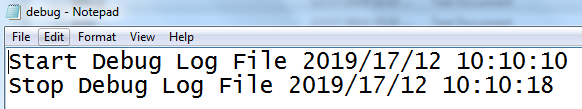
if sLevel == 'error':

logger.error('Practical Data Science logged an error message.')

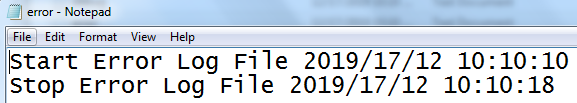
**Output:**



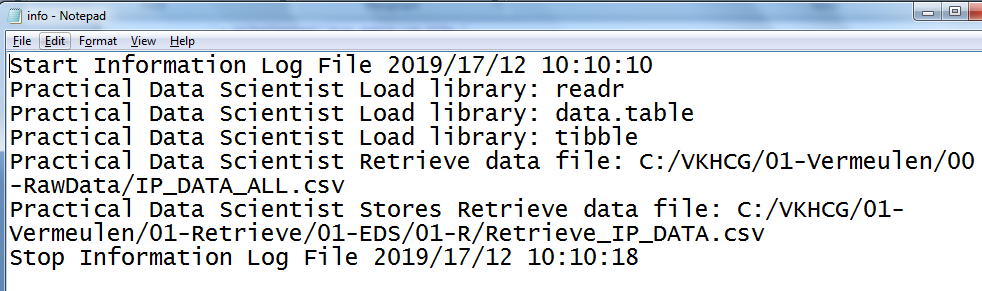
Debug.txt



Error.txt



Info.txt



**Practical 4**

**Retrieve Superstep**

1. **Perform the following data processing using R.**

**Code:**

library(readr)

IP\_DATA\_ALL <- read\_csv("C:/VKHCG/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv")

View(IP\_DATA\_ALL)

spec(IP\_DATA\_ALL)

IP\_DATA\_ALL\_FIX=set\_tidy\_names(IP\_DATA\_ALL, syntactic = TRUE, quiet = TRUE)

library(tibble)

set\_tidy\_names(IP\_DATA\_ALL, syntactic = TRUE, quiet = FALSE)

sapply(IP\_DATA\_ALL\_FIX, typeof)

library(data.table)

hist\_country=data.table(Country=unique(IP\_DATA\_ALL\_FIX[is.na(IP\_DATA\_ALL\_FIX ['Country']) == 0, ]$Country

))

IP\_DATA\_COUNTRY\_FREQ=data.table(with(IP\_DATA\_ALL\_FIX, table(Country)))

View(IP\_DATA\_COUNTRY\_FREQ)

sapply(IP\_DATA\_ALL\_FIX[,'Latitude'], min, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX[,'Country'], min, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX[,'Latitude'], max, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX[,'Country'], max, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX [,'Latitude'], mean, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX [,'Latitude'], median, na.rm=TRUE)

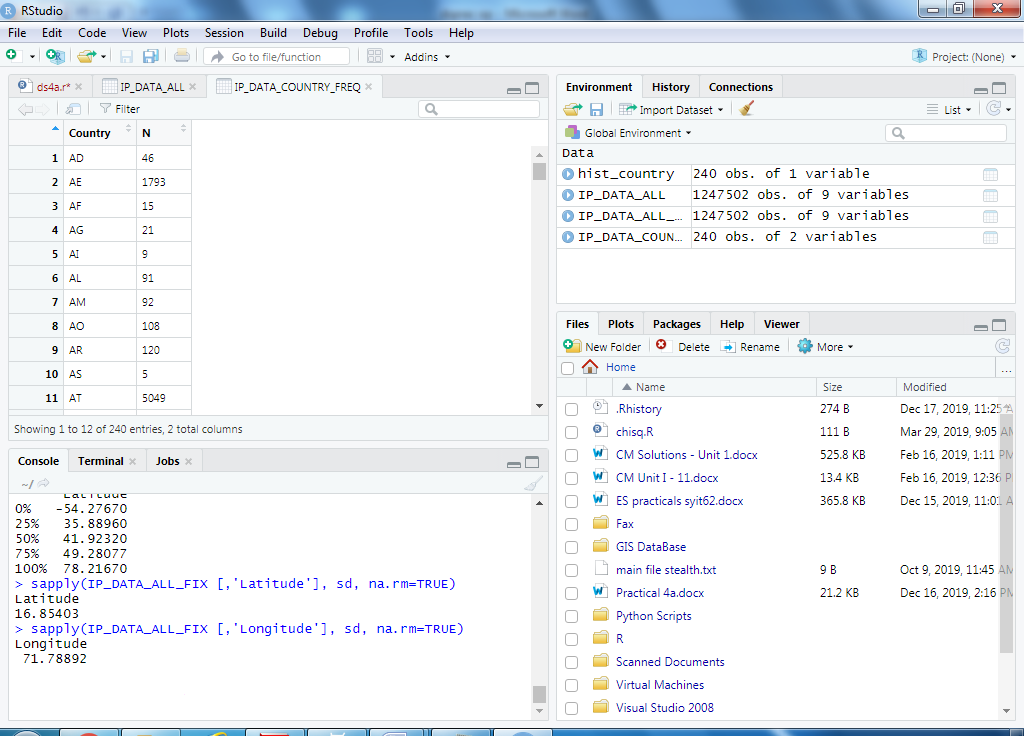
sapply(IP\_DATA\_ALL\_FIX [,'Latitude'], range, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX [,'Latitude'], quantile, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX [,'Latitude'], sd, na.rm=TRUE)

sapply(IP\_DATA\_ALL\_FIX [,'Longitude'], sd, na.rm=TRUE)

**Output:**



1. **Retrive\_IP\_DATA\_ALL.py**

**Code:**

import sys

import os

import pandas as pd

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

Base='C:/VKHCG'

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

sFileName=Base + '/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv'

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1")

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

sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

print('Rows:', IP\_DATA\_ALL.shape[0])

print('Columns:', IP\_DATA\_ALL.shape[1])

print('### Raw Data Set #####################################')

for i in range(0,len(IP\_DATA\_ALL.columns)):

print(IP\_DATA\_ALL.columns[i],type(IP\_DATA\_ALL.columns[i]))

print('### Fixed Data Set ###################################')

IP\_DATA\_ALL\_FIX=IP\_DATA\_ALL

for i in range(0,len(IP\_DATA\_ALL.columns)):

cNameOld=IP\_DATA\_ALL\_FIX.columns[i] + ' '

cNameNew=cNameOld.strip().replace(" ", ".")

IP\_DATA\_ALL\_FIX.columns.values[i] = cNameNew

print(IP\_DATA\_ALL.columns[i],type(IP\_DATA\_ALL.columns[i]))

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

#print(IP\_DATA\_ALL\_FIX.head())

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

print('Fixed Data Set with ID')

IP\_DATA\_ALL\_with\_ID=IP\_DATA\_ALL\_FIX

IP\_DATA\_ALL\_with\_ID.index.names = ['RowID']

#print(IP\_DATA\_ALL\_with\_ID.head())

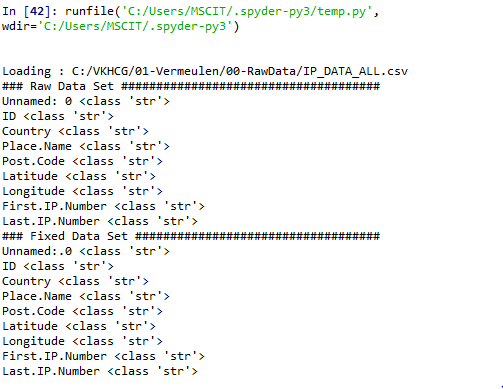
sFileName2=sFileDir + '/Retrieve\_IP\_DATA.csv'

IP\_DATA\_ALL\_with\_ID.to\_csv(sFileName2, index = True, encoding="latin-1")

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

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

**Output:**



1. **Data Pattern**

**Code:**

library(readr)

library(data.table)

FileName=paste0('c:/VKHCG/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv')

IP\_DATA\_ALL <- read\_csv(FileName)

hist\_country=data.table(Country=unique(IP\_DATA\_ALL$Country))

pattern\_country=data.table(Country=hist\_country$Country,

PatternCountry=hist\_country$Country)

oldchar=c(letters,LETTERS)

newchar=replicate(length(oldchar),"A")

for (r in seq(nrow(pattern\_country))){

s=pattern\_country[r,]$PatternCountry;

for (c in seq(length(oldchar))){

s=chartr(oldchar[c],newchar[c],s)

};

for (n in seq(0,9,1)){

s=chartr(as.character(n),"N",s)

};

s=chartr(" ","b",s)

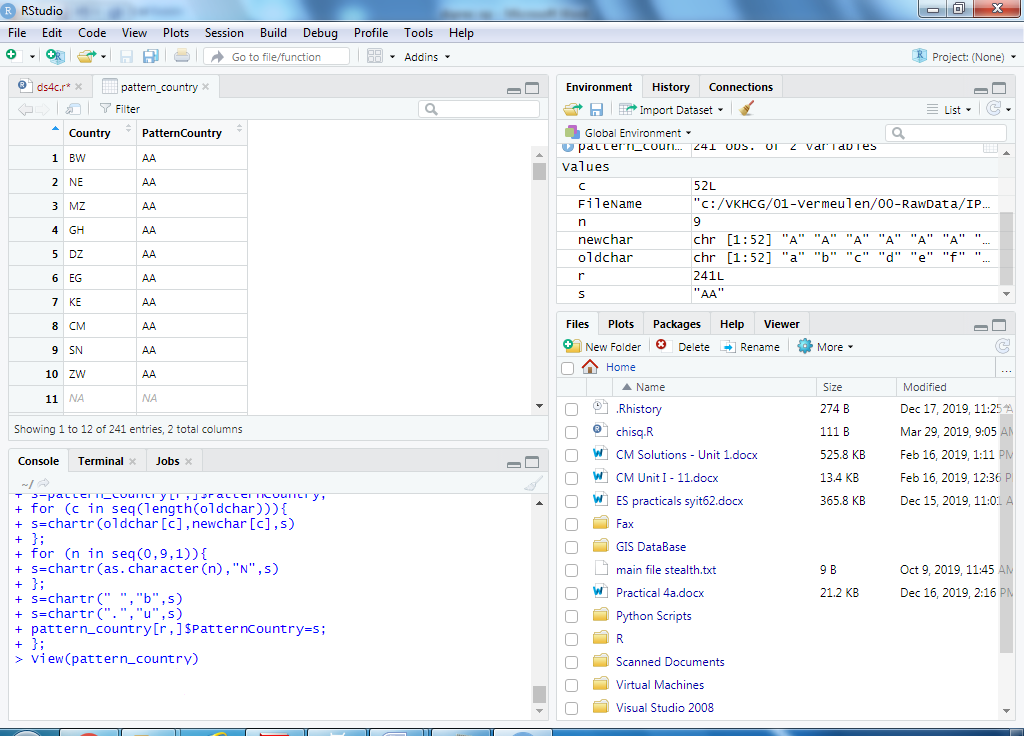
s=chartr(".","u",s)

pattern\_country[r,]$PatternCountry=s;

};

View(pattern\_country)

**Output:**



1. **Loading IP\_DATA\_ALL**

**Code:**

##############Retrieve-IP\_DATA\_ALL.py########################

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

Base='C:/VKHCG'

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

sFileName=Base + '/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv'

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1")

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

sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

print('Rows:', IP\_DATA\_ALL.shape[0])

print('Columns:', IP\_DATA\_ALL.shape[1])

print('### Raw Data Set #####################################')

for i in range(0,len(IP\_DATA\_ALL.columns)):

print(IP\_DATA\_ALL.columns[i],type(IP\_DATA\_ALL.columns[i]))

print('### Fixed Data Set ###################################')

IP\_DATA\_ALL\_FIX=IP\_DATA\_ALL

for i in range(0,len(IP\_DATA\_ALL.columns)):

cNameOld=IP\_DATA\_ALL\_FIX.columns[i] + ' '

cNameNew=cNameOld.strip().replace(" ", ".")

IP\_DATA\_ALL\_FIX.columns.values[i] = cNameNew

print(IP\_DATA\_ALL.columns[i],type(IP\_DATA\_ALL.columns[i]))

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

#print(IP\_DATA\_ALL\_FIX.head())

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

print('Fixed Data Set with ID')

IP\_DATA\_ALL\_with\_ID=IP\_DATA\_ALL\_FIX

IP\_DATA\_ALL\_with\_ID.index.names = ['RowID']

#print(IP\_DATA\_ALL\_with\_ID.head())

sFileName2=sFileDir + '/Retrieve\_IP\_DATA.csv'

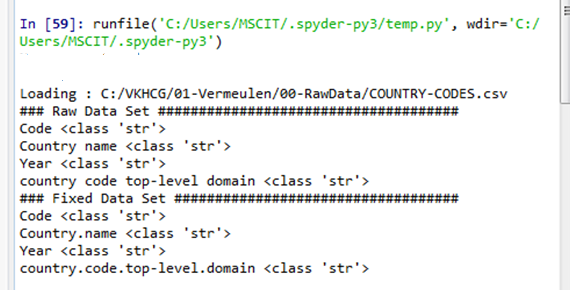
IP\_DATA\_ALL\_with\_ID.to\_csv(sFileName2, index = True, encoding="latin-1")

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

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

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

**Output:**



1. **Program to connect to different data sources.**

**SQLite**:

**Code:**

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

# -\*- coding: utf-8 -\*-

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

import sqlite3 as sq

import pandas as pd

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

Base='C:/VKHCG'

sDatabaseName=Base + '/01-Vermeulen/00-RawData/SQLite/vermeulen.db'

conn = sq.connect(sDatabaseName)

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

sFileName='C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve\_IP\_DATA.csv'

print('Loading :',sFileName)

IP\_DATA\_ALL\_FIX=pd.read\_csv(sFileName,header=0,low\_memory=False)

IP\_DATA\_ALL\_FIX.index.names = ['RowIDCSV']

sTable='IP\_DATA\_ALL'

print('Storing :',sDatabaseName,' Table:',sTable)

IP\_DATA\_ALL\_FIX.to\_sql(sTable, conn, if\_exists="replace")

print('Loading :',sDatabaseName,' Table:',sTable)

TestData=pd.read\_sql\_query("select \* from IP\_DATA\_ALL;", conn)

print('################')

print('## Data Values')

print('################')

print(TestData)

print('################')

print('## Data Profile')

print('################')

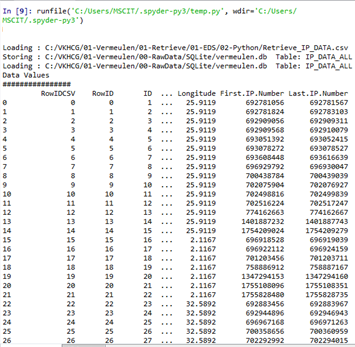
print('Rows :',TestData.shape[0])

print('Columns :',TestData.shape[1])

print('################')

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

**Output:**



**Practical 5**

**Assessing Data**

**5A1. Drop the Columns Where All Elements Are Missing Values**

**Code:**

################### Assess-Good-Bad-01.py########################

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

Base='C:/VKHCG'

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

print('################################')

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

print('################################')

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

sInputFileName='Good-or-Bad.csv'

sOutputFileName='Good-or-Bad-01.csv'

Company='01-Vermeulen'

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

Base='C:/VKHCG'

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

### Import Warehouse

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

sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName

print('Loading :',sFileName)

RawData=pd.read\_csv(sFileName,header=0)

print('################################')

print('## Raw Data Values')

print('################################')

print(RawData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns :',RawData.shape[1])

print('################################')

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

sFileName=sFileDir + '/' + sInputFileName

RawData.to\_csv(sFileName, index = False)

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

TestData=RawData.dropna(axis=1, how='all')

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

print('################################')

print('## Test Data Values')

print('################################')

print(TestData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns :',TestData.shape[1])

print('################################')

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

sFileName=sFileDir + '/' + sOutputFileName

TestData.to\_csv(sFileName, index = False)

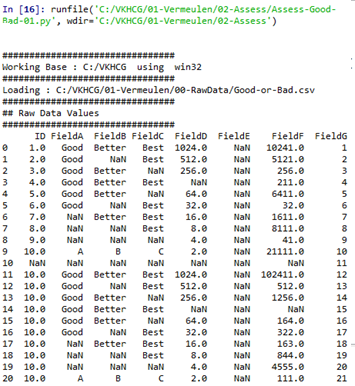
################################################################

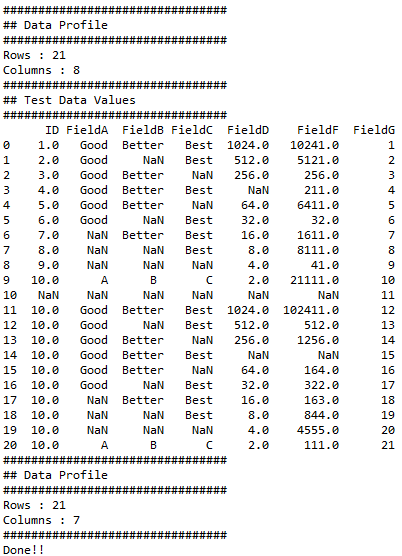
print('################################')

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

print('################################')

**Output:**





**5A2. Drop the Columns Where Any of the Elements Is Missing Values**

**Code:**

################## Assess-Good-Bad-02.py###########################

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

Base='C:/VKHCG'

sInputFileName='Good-or-Bad.csv'

sOutputFileName='Good-or-Bad-02.csv'

Company='01-Vermeulen'

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

Base='C:/VKHCG'

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

print('################################')

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

print('################################')

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

### Import Warehouse

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

sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName

print('Loading :',sFileName)

RawData=pd.read\_csv(sFileName,header=0)

print('################################')

print('## Raw Data Values')

print('################################')

print(RawData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns :',RawData.shape[1])

print('################################')

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

sFileName=sFileDir + '/' + sInputFileName

RawData.to\_csv(sFileName, index = False)

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

TestData=RawData.dropna(axis=1, how='any')

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

print('################################')

print('## Test Data Values')

print('################################')

print(TestData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns :',TestData.shape[1])

print('################################')

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

sFileName=sFileDir + '/' + sOutputFileName

TestData.to\_csv(sFileName, index = False)

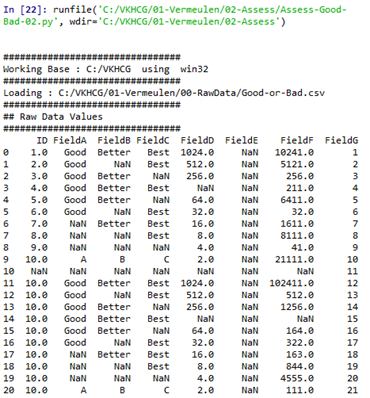
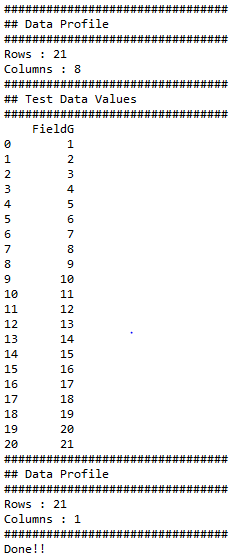
################################################################

print('################################')

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

print('################################')

**Output:**

**** 

**5A3. Keep Only the Rows That Contain a Maximum of Two Missing Values**

**Code:**

##################### Assess-Good-Bad-03.py ################

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

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

sInputFileName='Good-or-Bad.csv'

sOutputFileName='Good-or-Bad-03.csv'

Company='01-Vermeulen'

Base='C:/VKHCG'

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

print('################################')

print('Working Base :',Base, ' using Windows ~~~~')

print('################################')

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

### Import Warehouse

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

sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName

print('Loading :',sFileName)

RawData=pd.read\_csv(sFileName,header=0)

print('################################')

print('## Raw Data Values')

print('################################')

print(RawData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns :',RawData.shape[1])

print('################################')

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

sFileName=sFileDir + '/' + sInputFileName

RawData.to\_csv(sFileName, index = False)

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

TestData=RawData.dropna(thresh=2)

print('################################')

print('## Test Data Values')

print('################################')

print(TestData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns :',TestData.shape[1])

print('################################')

sFileName=sFileDir + '/' + sOutputFileName

TestData.to\_csv(sFileName, index = False)

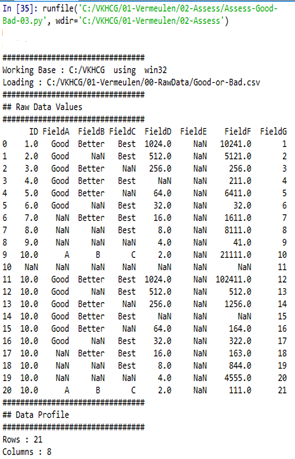
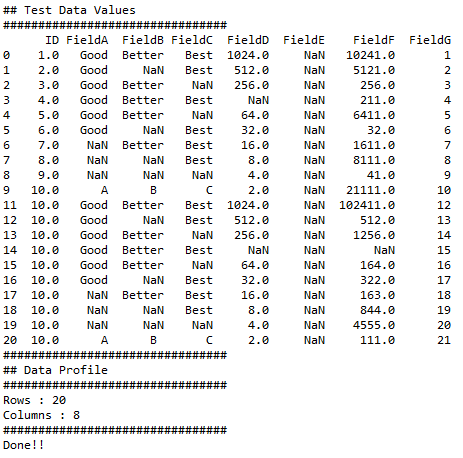
################################################################

print('################################')

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

print('################################')

**Output:**

**5A4. Fill All Missing Values with the Mean, Median, Mode, Minimum, and Maximum of the Particular Numeric Column**

**Code:**

# -\*- coding: utf-8 -\*################################################################ Import sys import os

import pandas as pd

################################################################ Base='C:/VKHCG'

sInputFileName='Good-or-Bad.csv'

sOutputFileNameA='Good-or-Bad-04-A.csv'

sOutputFileNameB='Good-or-Bad-04-B.csv'

sOutputFileNameC='Good-or-Bad-04-C.csv'

sOutputFileNameD='Good-or-Bad-04-D.csv'

sOutputFileNameE='Good-or-Bad-04-E.csv'

Company='01-Vermeulen' ################################################################

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################') ################################################################

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

### Import Warehouse ################################################################ sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName print('Loading:',sFileName)

RawData=pd.read\_csv(sFileName,header=0)

print('################################')

print('## Raw Data Values')

print('################################')

print(RawData) print('################################')

print('## Data Profile')

print('################################')

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

print('Columns:',RawData.shape[1])

print('################################') ################################################################ sFileName=sFileDir + '/' + sInputFileName

RawData.to\_csv(sFileName,index=False) ################################################################ TestData=RawData.fillna(RawData.mean()) ################################################################ print('################################')

print('## Test Data Values- Mean')

print('################################')

print(TestData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns:',TestData.shape[1])

print('################################') ################################################################ sFileName=sFileDir + '/' + sOutputFileNameA

TestData.to\_csv(sFileName,index=False)

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

## This is the important action! The rest of this code snippet

## Only supports this action.

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

TestData=RawData.fillna(RawData.median()) ################################################################ print('################################')

print('## Test Data Values - Median')

print('################################')

print(TestData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns:',TestData.shape[1])

print('################################') ################################################################ sFileName=sFileDir + '/' + sOutputFileNameB

TestData.to\_csv(sFileName,index=False) ################################################################ ################################################################ TestData=RawData.fillna(RawData.mode()) ################################################################ print('################################')

print('## Test Data Values - Mode')

print('################################')

print(TestData)

print('################################')

print('## Data Profile')

print('################################')

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

print('Columns:',TestData.shape[1])

print('################################')

################################################################ sFileName=sFileDir + '/' + sOutputFileNameC

TestData.to\_csv(sFileName,index=False) ################################################################ ################################################################ TestData=RawData.fillna(RawData.min()) ################################################################ print('################################')

print('## Test Data Values - Minumum')

print('################################')

print(TestData) print('################################')

print('## Data Profile')

print('################################')

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

print('Columns:',TestData.shape[1])

print('################################') ################################################################ sFileName=sFileDir + '/' + sOutputFileNameD

TestData.to\_csv(sFileName,index=False) ################################################################ ################################################################ TestData=RawData.fillna(RawData.max()) ################################################################ print('################################')

print('## Test Data Values - Maximum')

print('################################')

print(TestData)

print('################################')

print('##DataProfile')

print('################################')

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

print('Columns:',TestData.shape[1])

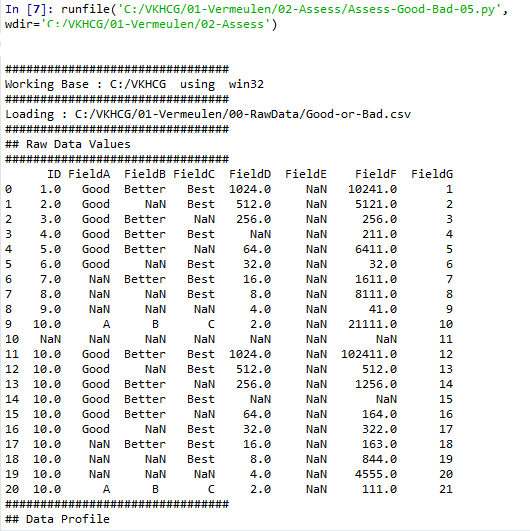
print('################################') ################################################################

sFileName=sFileDir + '/' + sOutputFileNameE

TestData.to\_csv(sFileName,index=False) ################################################################ ################################################################ print('################################')

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

**Output:**



**Practical 6**

**Processing Data**

**Hubs, Links, Satellite**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

from datetime import datetime

from datetime import timedelta

from pytz import timezone, all\_timezones

import pandas as pd

import sqlite3 as sq

from pandas.io import sql

import uuid

pd.options.mode.chained\_assignment = None

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################')

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

Company='01-Vermeulen'

InputDir='00-RawData'

InputFileName='VehicleData.csv'

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

sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

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

sDatabaseName=sDataBaseDir + '/Hillman.db'

conn1 = sq.connect(sDatabaseName)

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

sDataVaultDir=Base + '/88-DV'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

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

sDatabaseName=sDataVaultDir + '/datavault.db'

conn2 = sq.connect(sDatabaseName)

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

base = datetime(2018,1,1,0,0,0)

numUnits=10\*365\*24

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

date\_list = [base - timedelta(hours=x) for x in range(0, numUnits)]

t=0

for i in date\_list:

now\_utc=i.replace(tzinfo=timezone('UTC'))

sDateTime=now\_utc.strftime("%Y-%m-%d %H:%M:%S")

print(sDateTime)

sDateTimeKey=sDateTime.replace(' ','-').replace(':','-')

t+=1

IDNumber=str(uuid.uuid4())

TimeLine=[('ZoneBaseKey', ['UTC']),

('IDNumber', [IDNumber]),

('nDateTimeValue', [now\_utc]),

('DateTimeValue', [sDateTime]),

('DateTimeKey', [sDateTimeKey])]

if t==1:

TimeFrame = pd.DataFrame.from\_items(TimeLine)

else:

TimeRow = pd.DataFrame.from\_items(TimeLine)

TimeFrame = TimeFrame.append(TimeRow)

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

TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]

TimeHubIndex=TimeHub.set\_index(['IDNumber'],inplace=False)

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

TimeFrame.set\_index(['IDNumber'],inplace=True)

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

sTable = 'Process-Time'

print('Storing :',sDatabaseName,' Table:',sTable)

TimeHubIndex.to\_sql(sTable, conn1, if\_exists="replace")

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

sTable = 'Hub-Time'

print('Storing :',sDatabaseName,' Table:',sTable)

TimeHubIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

active\_timezones=all\_timezones

z=0

for zone in active\_timezones:

t=0

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

now\_date=TimeFrame['nDateTimeValue'][j]

DateTimeKey=TimeFrame['DateTimeKey'][j]

now\_utc=now\_date.replace(tzinfo=timezone('UTC'))

sDateTime=now\_utc.strftime("%Y-%m-%d %H:%M:%S")

now\_zone = now\_utc.astimezone(timezone(zone))

sZoneDateTime=now\_zone.strftime("%Y-%m-%d %H:%M:%S")

print(sZoneDateTime)

t+=1

z+=1

IDZoneNumber=str(uuid.uuid4())

TimeZoneLine=[('ZoneBaseKey', ['UTC']),

('IDZoneNumber', [IDZoneNumber]),

('DateTimeKey', [DateTimeKey]),

('UTCDateTimeValue', [sDateTime]),

('Zone', [zone]),

('DateTimeValue', [sZoneDateTime])]

if t==1:

TimeZoneFrame = pd.DataFrame.from\_items(TimeZoneLine)

else:

TimeZoneRow = pd.DataFrame.from\_items(TimeZoneLine)

TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)

TimeZoneFrameIndex=TimeZoneFrame.set\_index(['IDZoneNumber'],inplace=False)

sZone=zone.replace('/','-').replace(' ','')

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

sTable = 'Process-Time-'+sZone

print('Storing :',sDatabaseName,' Table:',sTable)

TimeZoneFrameIndex.to\_sql(sTable, conn1, if\_exists="replace")

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

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

sTable = 'Satellite-Time-'+sZone

print('Storing :',sDatabaseName,' Table:',sTable)

TimeZoneFrameIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

print('################')

print('Vacuum Databases')

sSQL="VACUUM;"

sql.execute(sSQL,conn1)

sql.execute(sSQL,conn2)

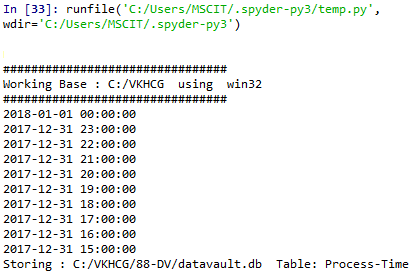
print('################')

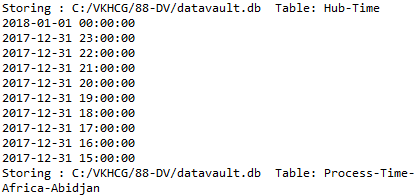
#################################################################

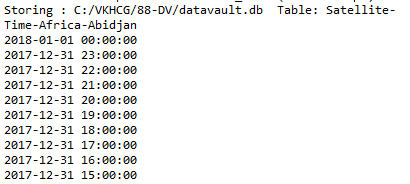
print('### Done!! ############################################')

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

**Output:**

****

****

****

**Practical 7**

**Transforming Data**

**Sun Model**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

from datetime import datetime

from pytz import timezone

import pandas as pd

import sqlite3 as sq

import uuid

pd.options.mode.chained\_assignment = None

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################')

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

Company='01-Vermeulen'

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

sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

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

sDatabaseName=sDataBaseDir + '/Vermeulen.db'

conn1 = sq.connect(sDatabaseName)

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

sDataWarehousetDir=Base + '/99-DW'

if not os.path.exists(sDataWarehousetDir):

os.makedirs(sDataWarehousetDir)

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

sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'

conn2 = sq.connect(sDatabaseName)

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

print('\n#################################')

print('Time Dimension')

BirthZone = 'Atlantic/Reykjavik'

BirthDateUTC = datetime(1960,12,20,10,15,0)

BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))

BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")

BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")

BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))

BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")

BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")

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

IDTimeNumber=str(uuid.uuid4())

TimeLine=[('TimeID', [IDTimeNumber]),

('UTCDate', [BirthDateZoneStr]),

('LocalTime', [BirthDateLocal]),

('TimeZone', [BirthZone])]

TimeFrame = pd.DataFrame.from\_items(TimeLine)

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

DimTime=TimeFrame

DimTimeIndex=DimTime.set\_index(['TimeID'],inplace=False)

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

sTable = 'Dim-Time'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimTimeIndex.to\_sql(sTable, conn1, if\_exists="replace")

DimTimeIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

print('\n#################################')

print('Dimension Person')

print('\n#################################')

FirstName = 'Guðmundur'

LastName = 'Gunnarsson'

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

IDPersonNumber=str(uuid.uuid4())

PersonLine=[('PersonID', [IDPersonNumber]),

('FirstName', [FirstName]),

('LastName', [LastName]),

('Zone', ['UTC']),

('DateTimeValue', [BirthDateZoneStr])]

PersonFrame = pd.DataFrame.from\_items(PersonLine)

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

DimPerson=PersonFrame

DimPersonIndex=DimPerson.set\_index(['PersonID'],inplace=False)

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

sTable = 'Dim-Person'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn1, if\_exists="replace")

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

print('\n#################################')

print('Fact - Person - time')

print('\n#################################')

IDFactNumber=str(uuid.uuid4())

PersonTimeLine=[('IDNumber', [IDFactNumber]),

('IDPersonNumber', [IDPersonNumber]),

('IDTimeNumber', [IDTimeNumber])]

PersonTimeFrame = pd.DataFrame.from\_items(PersonTimeLine)

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

FctPersonTime=PersonTimeFrame

FctPersonTimeIndex=FctPersonTime.set\_index(['IDNumber'],inplace=False)

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

sTable = 'Fact-Person-Time'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

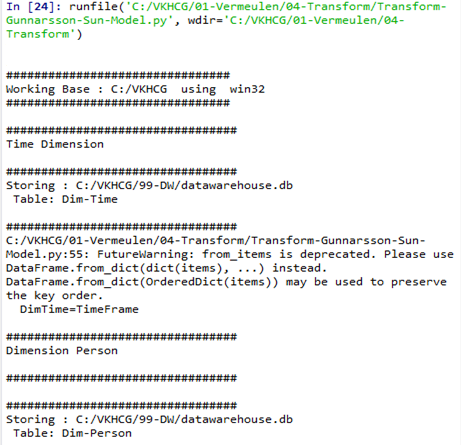
print('\n#################################')

FctPersonTimeIndex.to\_sql(sTable, conn1, if\_exists="replace")

FctPersonTimeIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

**Output:**

****



**Practical 8A**

**Organizing Data**

1. **Horizontal Style**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################')

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

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

Company='01-Vermeulen'

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

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

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

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn1 = sq.connect(sDatabaseName)

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

sDatabaseName=sDataWarehouseDir + '/datamart.db'

conn2 = sq.connect(sDatabaseName)

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

print('################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1)

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

print('################################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

print('################################')

sSQL="SELECT PersonID,\

Height,\

Weight,\

bmi,\

Indicator\

FROM [Dim-BMI]\

WHERE \

Height > 1.5 \

and Indicator = 1\

ORDER BY \

Height,\

Weight;"

PersonFrame1=pd.read\_sql\_query(sSQL, conn1)

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

DimPerson=PersonFrame1

DimPersonIndex=DimPerson.set\_index(['PersonID'],inplace=False)

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

sTable = 'Dim-BMI-Horizontal'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

print('################################')

sTable = 'Dim-BMI-Horizontal'

print('Loading :',sDatabaseName,' Table:',sTable)

print('################################')

sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame2=pd.read\_sql\_query(sSQL, conn2)

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

print('################################')

print('Full Data Set (Rows):', PersonFrame0.shape[0])

print('Full Data Set (Columns):', PersonFrame0.shape[1])

print('################################')

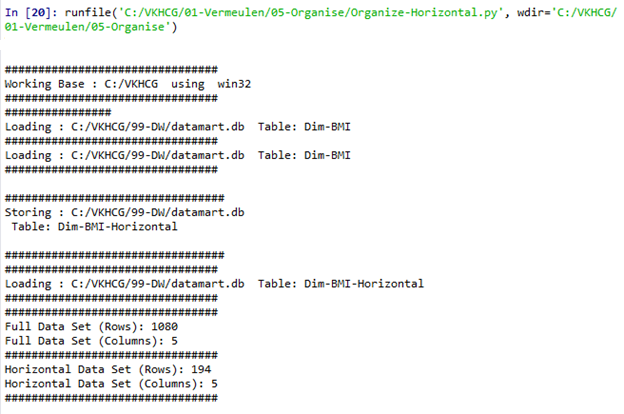
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])

print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

print('################################')

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

**Output:**



**Practical 8B**

**Vertical style**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################')

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

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

Company='01-Vermeulen'

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

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

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

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn1 = sq.connect(sDatabaseName)

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

sDatabaseName=sDataWarehouseDir + '/datamart.db'

conn2 = sq.connect(sDatabaseName)

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

print('################################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1)

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

print('################################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

print('################################')

sSQL="SELECT \

Height,\

Weight,\

Indicator\

FROM [Dim-BMI];"

PersonFrame1=pd.read\_sql\_query(sSQL, conn1)

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

DimPerson=PersonFrame1

DimPersonIndex=DimPerson.set\_index(['Indicator'],inplace=False)

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

sTable = 'Dim-BMI-Vertical'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

print('################')

sTable = 'Dim-BMI-Vertical'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \* FROM [Dim-BMI-Vertical];"

PersonFrame2=pd.read\_sql\_query(sSQL, conn2)

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

print('################################')

print('Full Data Set (Rows):', PersonFrame0.shape[0])

print('Full Data Set (Columns):', PersonFrame0.shape[1])

print('################################')

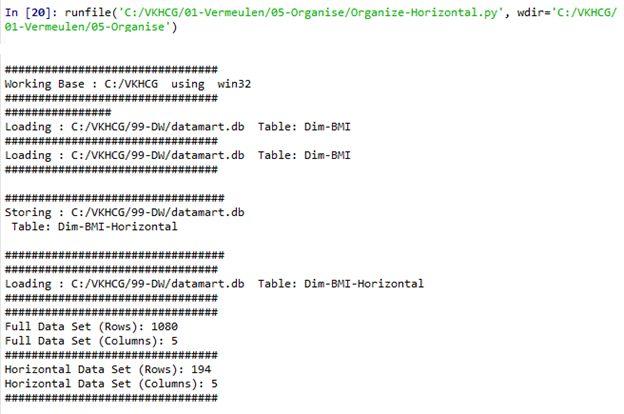
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])

print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

print('################################')

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

**Output:**

****

**Practical 8C**

**Island Style**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################')

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

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

Company='01-Vermeulen'

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

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

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

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn1 = sq.connect(sDatabaseName)

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

sDatabaseName=sDataWarehouseDir + '/datamart.db'

conn2 = sq.connect(sDatabaseName)

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

print('################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1)

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

print('################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \

Height,\

Weight,\

Indicator\

FROM [Dim-BMI]\

WHERE Indicator > 2\

ORDER BY \

Height,\

Weight;"

PersonFrame1=pd.read\_sql\_query(sSQL, conn1)

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

DimPerson=PersonFrame1

DimPersonIndex=DimPerson.set\_index(['Indicator'],inplace=False)

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

sTable = 'Dim-BMI-Vertical'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

print('################################')

sTable = 'Dim-BMI-Vertical'

print('Loading :',sDatabaseName,' Table:',sTable)

print('################################')

sSQL="SELECT \* FROM [Dim-BMI-Vertical];"

PersonFrame2=pd.read\_sql\_query(sSQL, conn2)

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

print('################################')

print('Full Data Set (Rows):', PersonFrame0.shape[0])

print('Full Data Set (Columns):', PersonFrame0.shape[1])

print('################################')

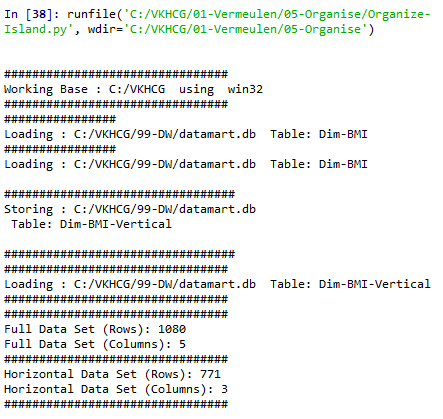
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])

print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

print('################################')

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

**Output:**

****

**Practical 8D**

**Secure Vault Style**

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

# -\*- coding: utf-8 -\*-

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

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################')

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

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

Company='01-Vermeulen'

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

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

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

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn1 = sq.connect(sDatabaseName)

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

sDatabaseName=sDataWarehouseDir + '/datamart.db'

conn2 = sq.connect(sDatabaseName)

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

print('################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1)

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

print('################')

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable)

sSQL="SELECT \

Height,\

Weight,\

Indicator,\

CASE Indicator\

WHEN 1 THEN 'Pip'\

WHEN 2 THEN 'Norman'\

WHEN 3 THEN 'Grant'\

ELSE 'Sam'\

END AS Name\

FROM [Dim-BMI]\

WHERE Indicator > 2\

ORDER BY \

Height,\

Weight;"

PersonFrame1=pd.read\_sql\_query(sSQL, conn1)

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

DimPerson=PersonFrame1

DimPersonIndex=DimPerson.set\_index(['Indicator'],inplace=False)

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

sTable = 'Dim-BMI-Secure'

print('\n#################################')

print('Storing :',sDatabaseName,'\n Table:',sTable)

print('\n#################################')

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

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

print('################################')

sTable = 'Dim-BMI-Secure'

print('Loading :',sDatabaseName,' Table:',sTable)

print('################################')

sSQL="SELECT \* FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"

PersonFrame2=pd.read\_sql\_query(sSQL, conn2)

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

print('################################')

print('Full Data Set (Rows):', PersonFrame0.shape[0])

print('Full Data Set (Columns):', PersonFrame0.shape[1])

print('################################')

print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])

print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

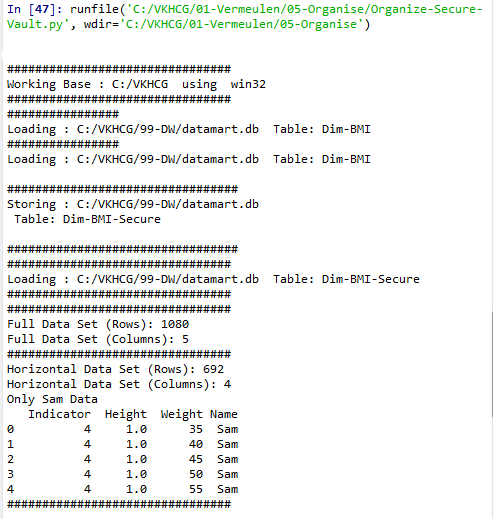
print('Only Sam Data')

print(PersonFrame2.head())

print('################################')

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

**Output:**

****

**Practical 9**

**Reporting Data**

1. **Create a Network Routing Diagram**

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

import sys

import os

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

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

pd.options.mode.chained\_assignment = None

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + 'VKHCG'

else:

Base='C:/VKHCG'

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

print('################################')

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

print('################################')

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

sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv'

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

sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company.gml'

sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company.png'

Company='01-Vermeulen'

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

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

### Import Country Data

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

sFileName=Base + '/' + Company + '/' + sInputFileName

print('################################')

print('Loading :',sFileName)

print('################################')

CompanyData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1")

print('################################')

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

print(CompanyData.head())

print(CompanyData.shape)

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

G=nx.Graph()

for i in range(CompanyData.shape[0]):

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

Node0=CompanyData['Company\_Country\_Name'][i]

Node1=CompanyData['Company\_Country\_Name'][j]

if Node0 != Node1:

G.add\_edge(Node0,Node1)

for i in range(CompanyData.shape[0]):

Node0=CompanyData['Company\_Country\_Name'][i]

Node1=CompanyData['Company\_Place\_Name'][i] + '('+ CompanyData['Company\_Country\_Name'][i] + ')'

if Node0 != Node1:

G.add\_edge(Node0,Node1)

print('Nodes:', G.number\_of\_nodes())

print('Edges:', G.number\_of\_edges())

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

sFileName=Base + '/' + Company + '/' + sOutputFileName1

print('################################')

print('Storing :',sFileName)

print('################################')

nx.write\_gml(G, sFileName)

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

sFileName=Base + '/' + Company + '/' + sOutputFileName2

print('################################')

print('Storing Graph Image:',sFileName)

print('################################')

plt.figure(figsize=(15, 15))

pos=nx.spectral\_layout(G,dim=2)

nx.draw\_networkx\_nodes(G,pos, node\_color='k', node\_size=10, alpha=0.8)

nx.draw\_networkx\_edges(G, pos,edge\_color='r', arrows=False, style='dashed')

nx.draw\_networkx\_labels(G,pos,font\_size=12,font\_family='sans-serif',font\_color='b')

plt.axis('off')

plt.savefig(sFileName,dpi=600)

plt.show()

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

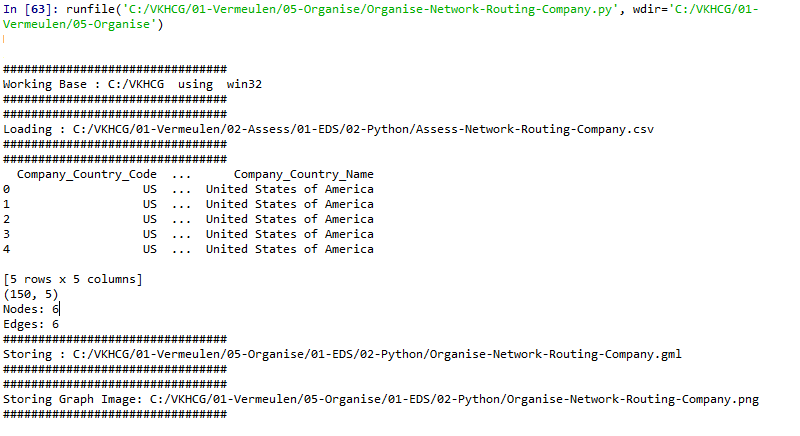
print('################################')

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

print('################################')

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

**Output:**

****

**Practical 9B**

**Directed Acyclic Graph**

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

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:/VKHCG'

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

print('################################')

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

print('################################')

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

sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve\_Router\_Location.csv'

sOutputFileName1='Assess-DAG-Company-Country.png'

sOutputFileName2='Assess-DAG-Company-Country-Place.png'

Company='01-Vermeulen'

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

### Import Company Data

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

sFileName=Base + '/' + Company + '/' + sInputFileName

print('################################')

print('Loading :',sFileName)

print('################################')

CompanyData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1")

print('Loaded Company :',CompanyData.columns.values)

print('################################')

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

print(CompanyData)

print('################################')

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

print('################################')

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

G1=nx.DiGraph()

G2=nx.DiGraph()

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

for i in range(CompanyData.shape[0]):

G1.add\_node(CompanyData['Country'][i])

sPlaceName= CompanyData['Place\_Name'][i] + '-' + CompanyData['Country'][i]

G2.add\_node(sPlaceName)

print('################################')

for n1 in G1.nodes():

for n2 in G1.nodes():

if n1 != n2:

print('Link :',n1,' to ', n2)

G1.add\_edge(n1,n2)

print('################################')

print("Nodes of graph: ")

print(G1.nodes())

print("Edges of graph: ")

print(G1.edges())

print('################################')

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

sFileName=sFileDir + '/' + sOutputFileName1

print('################################')

print('Storing :', sFileName)

print('################################')

nx.draw(G1,pos=nx.spectral\_layout(G1),

nodecolor='r',edge\_color='g',

with\_labels=True,node\_size=8000,

font\_size=12)

plt.savefig(sFileName) # save as png

plt.show() # display

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

print('################################')

for n1 in G2.nodes():

for n2 in G2.nodes():

if n1 != n2:

print('Link :',n1,' to ', n2)

G2.add\_edge(n1,n2)

print('################################')

print("Nodes of graph: ")

print(G2.nodes())

print("Edges of graph: ")

print(G2.edges())

print('################################')

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

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

sFileName=sFileDir + '/' + sOutputFileName2

print('################################')

print('Storing :', sFileName)

print('################################')

nx.draw(G2,pos=nx.spectral\_layout(G2),

nodecolor='r',edge\_color='b',

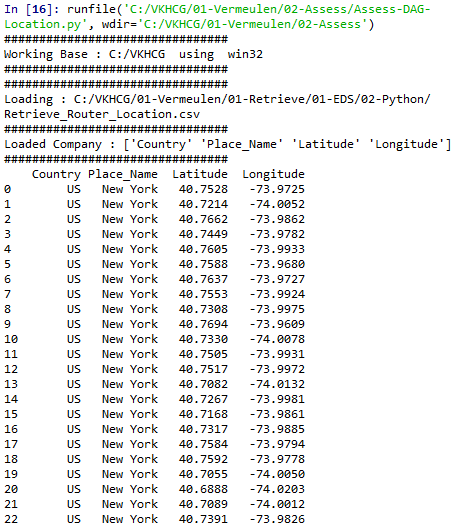
with\_labels=True,node\_size=8000,

font\_size=12)

plt.savefig(sFileName) # save as png

plt.show() # display

**Output:**

****

**Practical 9 C**

**Graphics**

**CODE**

import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

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

if sys.platform == 'linux':

Base=os.path.expanduser('~') + '/VKHCG'

else:

Base='C:/VKHCG'

print('################################')

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

print('################################')

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'Conversion Percent',

'URL', 'Stats URL'],

dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

explode = (0, 0, 0, 0, 0, 0.1)

labels=os\_new['Warehouse Location']