INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr.No | Aim | Date | Page No | Sign |
| 1 | Creating Data Model using Cassandra |  |  |  |
| 2 | Conversion from different formats to HOURS format. |  |  |  |
| A. | Text delimited csv format. |  |  |  |
| B. | XML |  |  |  |
| C. | JSON |  |  |  |
| D. | MySQL Database |  |  |  |
| E. | Picture (JPEG) |  |  |  |
| F. | Video |  |  |  |
| G. | Audio |  |  |  |
| 3 | Utilities and Auditing |  |  |  |
| 4 | Retrieving Data |  |  |  |
| 5 | Assessing Data |  |  |  |
| 6 | Processing Data |  |  |  |
| 7 | Transforming Data |  |  |  |
| 8 | Organizing Data |  |  |  |
| 9 | Generating Reports |  |  |  |
| 10 | Data Visualization with Power BI |  |  |  |

**Practical 1**

**Creating Data Model using Cassandra.**

**Cassandra Data Model**

**Step-1:**

* Open a folder Datascience\apache-cassandra-3.11.4-bin\apache-cassandra-3.11.4\bin\cassandra.bat
* now open IDLE (PYTHON GUI)
* go to file -> open -> select (Datascience\apache-cassandra-3.11.4-bin\apache-cassandra-3.11.4\bin\select-cqlsh.py

-inside sqlsh.py -> run -> run module

## Step-2: command to Create keyspace:

create keyspace DATASCI WITH replication={'class':'SimpleStrategy','replication\_factor':3};

## step – 3: command to use keyspace run this command

cqlsh> use datasci;

## step – 4: command to create a new table

cqlsh:datasci> create table student( student\_id int PRIMARY KEY, student\_name text,

student\_city text, student\_fees varint, student\_phone varin

## Step – 5: command to display created keyspace list

Desc keyspace;

## Step – 6: command to alter keyspace

alter keyspace datasci with replication={'class':'SimpleStrategy','replication\_factor':2};

## step – 7 : command to display all the tables of the keyspaces

cqlsh:datasci> desc tables;

## step- 8: command to alter table

cqlsh:datasci> alter table student

add student\_gender text;

## step- 9: command to insert data into table

insert into student(student\_id,student\_city,student\_fees,student\_name,student\_phone) values(1,'Bhy',5000,'pooja',0939293939);

(you can only add one value at a time) **Step – 10: command to show the table** cqlsh:datasci> select \* from student; **step – 11: command to update table**

cqlsh:datasci> update student set student\_fees=200000,student\_name='hima' where student\_id=2;

## step – 12: command to refresh the table

cqlsh:datasci> truncate student;**step – 13: command to delete the specific column data from the table** cqlsh:datasci> delete student\_city from student where student\_id=2;

## PRACTICAL NO2

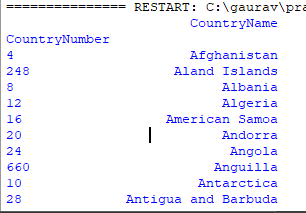
**A.Text delimited CSVto HORUS format**

import pandas as pd

sInputFileName='C:/VKHCG/05-DS/9999-Data/Country\_Code.csv' InputData=pd.read\_csv(sInputFileName,encoding="latin-1") ProcessData=InputData

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

ProcessData.sort\_values('CountryName', axis=0, ascending=True, inplace=True) print(ProcessData.head(10))



## B>XML to HORUS Format

# Utility Start XML to HORUS =================================

# Standard Tools import pandas as pd

import xml.etree.ElementTree as ET def df2xml(data):

header = data.columns root = ET.Element('root')

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

entry = ET.SubElement(root,'entry') for index in range(data.shape[1]):

schild=str(header[index])

child = ET.SubElementlement(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('Input Data Values ===================================')

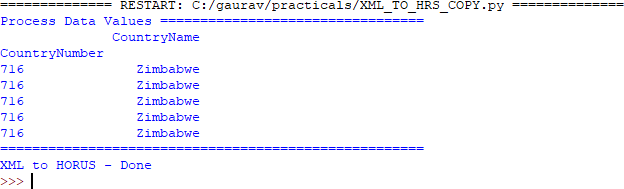
ProcessDataXML=InputData ProcessData=xml2df(ProcessDataXML)

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.head(5)) print('=====================================================')

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv' OutputData.to\_csv(sOutputFileName, index = False, encoding="latin-1") print('XML to HORUS - Done')



## C>JSON to HORUS Format

import pandas as pd

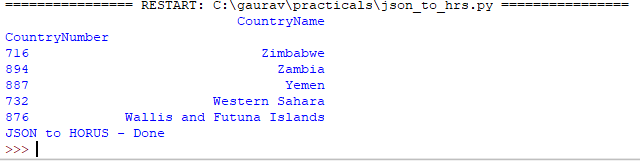
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country\_Code.json' InputData=pd.read\_json(sInputFileName, orient='index', encoding="latin-1") ProcessData=InputData

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

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True) print(ProcessData.head(5))

OutputData=ProcessData

sOutputFileName='c:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv' OutputData.to\_csv(sOutputFileName, index = False, encoding="latin-1") print('JSON to HORUS - Done')



## D>MySql Database to HORUS Format

import pandas as pd import sqlite3 as sq

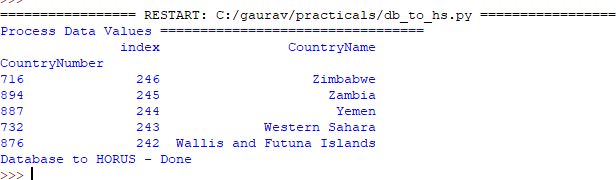
conn = sq.connect('C:/VKHCG/05-DS/9999-Data/utility.db') sSQL='select \* FROM ' + 'Country\_Code' + ';' InputData=pd.read\_sql\_query(sSQL, conn) ProcessData=InputData

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

ProcessData.sort\_values('CountryName', axis=0, ascending=False, inplace=True) print('Process Data Values =================================') print(ProcessData.head(5))

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv' OutputData.to\_csv(sOutputFileName, index = False, encoding="latin-1") print('Database to HORUS - Done')



## E>Picture (JPEG) to HORUS Format

import pandas as pd

import matplotlib.pyplot as plt import numpy as np

import imageio

sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/Angus.jpg' InputData = imageio.imread(sInputFileName, mode='RGBA') ProcessRawData=InputData.flatten()

y=InputData.shape[2] + 2 x=int(ProcessRawData.shape[0]/y)

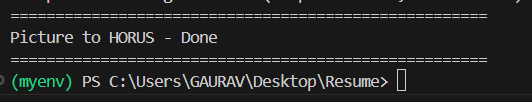
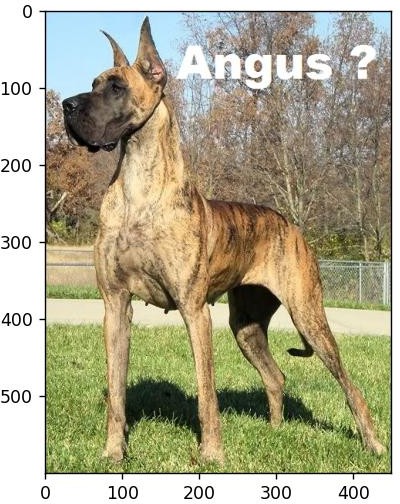
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y))) sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha'] ProcessData.columns=sColumns

ProcessData.index.names =['ID'] plt.imshow(InputData) plt.show()

OutputData=ProcessData

sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv' OutputData.to\_csv(sOutputFileName, index = False) print('=====================================================')

print('Picture to HORUS - Done') print('=====================================================')



## F>Video to HORUS Format

1st =====================================

import os import shutil import cv2

sInputFileName = 'C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/dog.mp4' sDataBaseDir = 'C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/temp'

if os.path.exists(sDataBaseDir): shutil.rmtree(sDataBaseDir)

if not os.path.exists(sDataBaseDir): os.makedirs(sDataBaseDir)

vidcap = cv2.VideoCapture(sInputFileName) if not vidcap.isOpened():

print('Error: Could not open video file') exit()

count = 0 while True:

success, image = vidcap.read() if not success:

break

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

cv2.imwrite(sFrame, image)

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

os.remove(sFrame) print('Removed: ', sFrame) continue

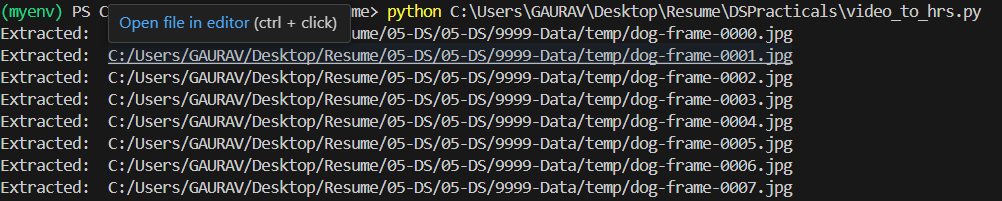
count += 1

if cv2.waitKey(10) == 27:

break

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

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



2nd part ==============================

import imageio import pandas as pd

import matplotlib.pyplot as plt import numpy as np

import os

sDataBaseDir='C:/Users/GAURAV/Desktop/Resume/05-DS/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)

InputData = imageio.imread(sInputFileName, mode='RGBA') 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 plt.imshow(InputData) plt.show()

ProcessData = [] if f == 1:

ProcessData=ProcessFrameData else:

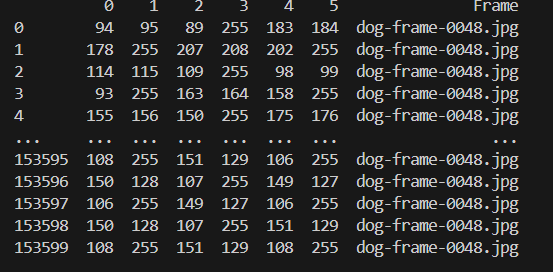
ProcessData=ProcessData.append(ProcessFrameData) if f > 0:

# ProcessData = pd.DataFrame(ProcessFrameData) print(ProcessData)

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

ProcessFrameData.index.names =['ID'] print('Rows: ',ProcessData.shape[0]) print('Columns :',ProcessData.shape[1])

ProcessData.to\_csv('C:/VKHCG/05-DS/9999-Data/HORUS-Movie-Frame.csv' , index = False) print('Processed ; ', f,' frames')



**G. Audio to HORUS Format** 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:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/2ch-sound.wav' 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:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-2ch.csv' OutputData.to\_csv(sOutputFileName, index = False) sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/4ch-sound.wav' 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:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-4ch.csv' OutputData.to\_csv(sOutputFileName, index = False) sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/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:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-6ch.csv' OutputData.to\_csv(sOutputFileName, index = False) #=============================================================

sInputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/8ch-sound.wav' print('=====================================================')

print('Processing : ', sInputFileName) print('=====================================================')

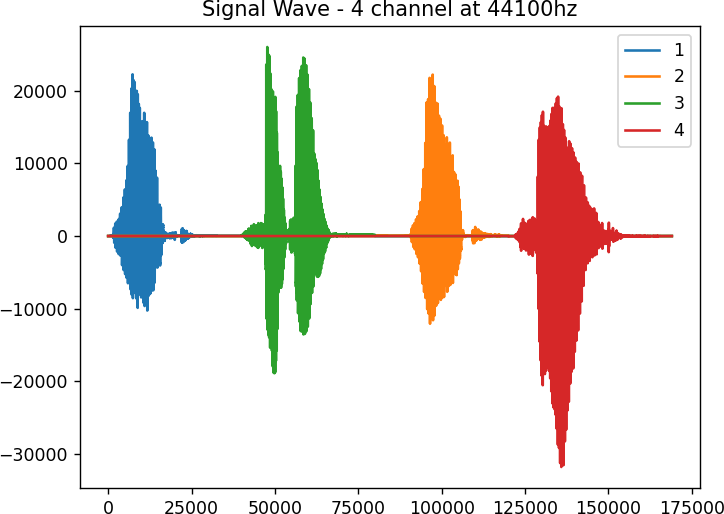
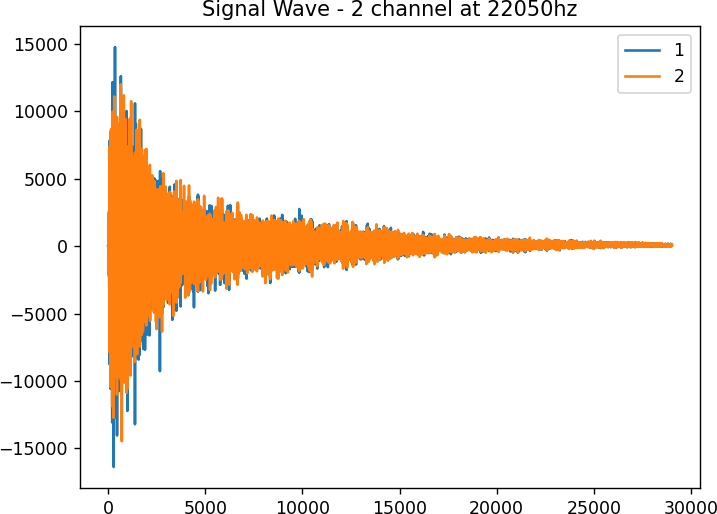
InputRate, InputData = wavfile.read(sInputFileName) show\_info("8 channel", InputData,InputRate) ProcessData=pd.DataFrame(InputData)

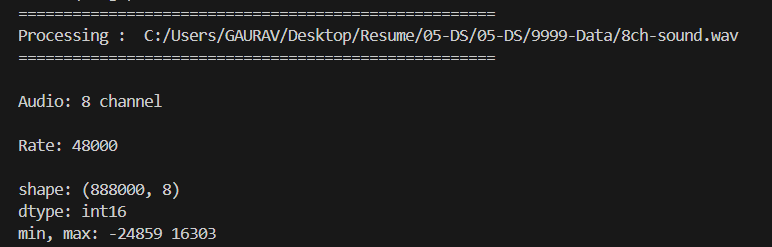
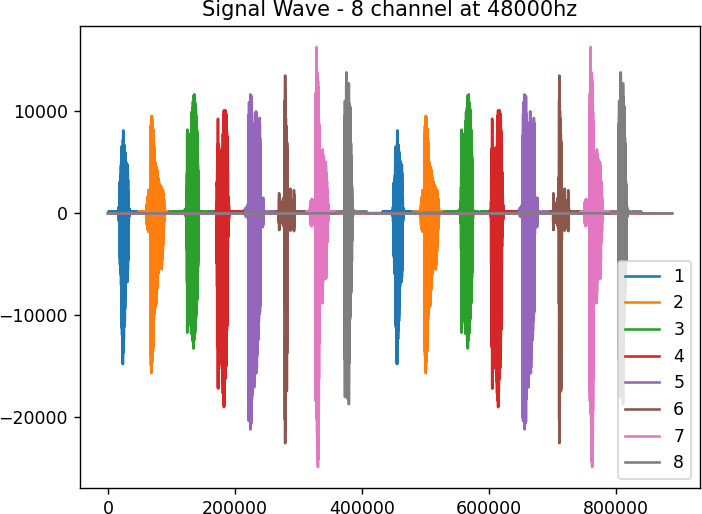
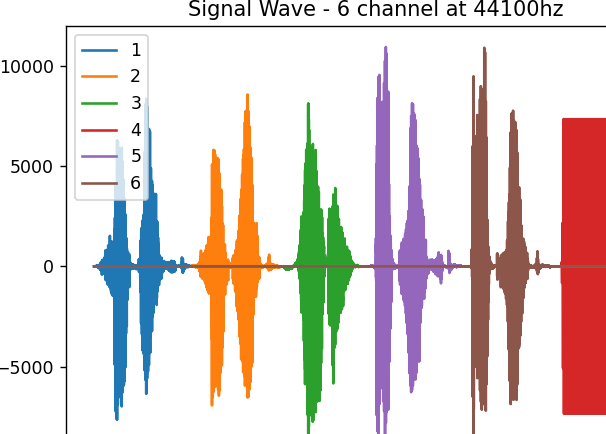
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']

ProcessData.columns=sColumns OutputData=ProcessData

sOutputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/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

Basic Utility Design

## Fixers Utilities:

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

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.

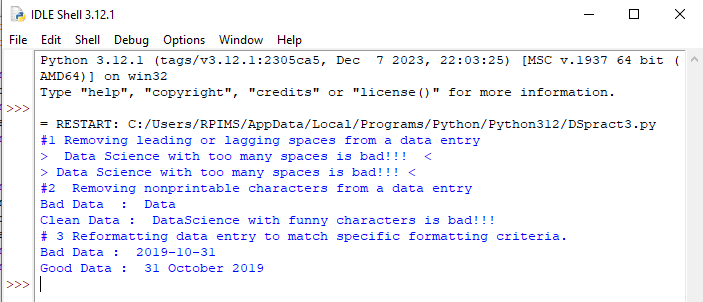
# Convert YYYY/MM/DD to DD Month YYYY

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)



## Data Binning or Bucketing

import numpy as np

import matplotlib.mlab as mlab import matplotlib.pyplot as plt np.random.seed(0)

# example data

mu = 90 # mean of distribution

sigma = 25 # standard deviation of distribution x = mu + sigma \* np.random.randn(5000) num\_bins = 25

fig, ax = plt.subplots()

# the histogram of the data

n, bins, patches = ax.hist(x, num\_bins, normed=1) # add a 'best fit' line

y = mlab.normpdf(bins, mu, sigma) ax.plot(bins, y, '--') ax.set\_xlabel('Example Data') ax.set\_ylabel('Probability density')

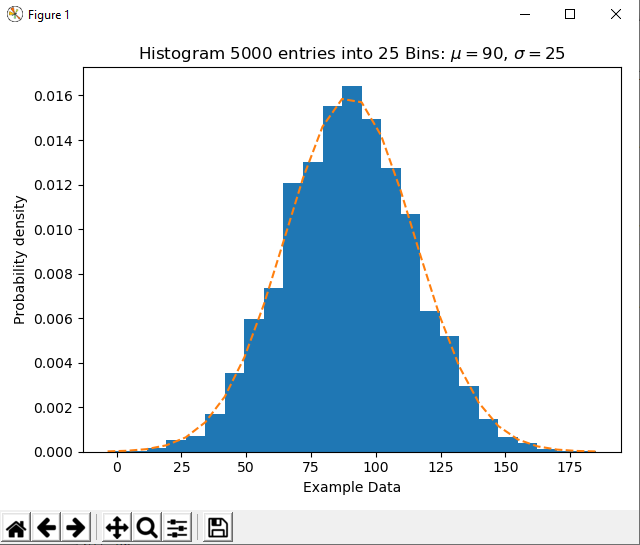
sTitle=r'Histogram ' + str(len(x)) + ' entries into ' + str(num\_bins) + ' Bins: $\mu=' + str(mu) + '$, $\sigma=' + str(sigma) + '$' ax.set\_title(sTitle)

fig.tight\_layout()

sPathFig='C:/VKHCG/05-DS/4000-UL/0200-DU/DU-Histogram.png' fig.savefig(sPathFig)

plt.show()

Output:



## Averaging of Data

**Input:**

import pandas as pd InputFileName='IP\_DATA\_CORE.csv' OutputFileName='Retrieve\_Router\_Location.csv' Base='C:/VKHCG'

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

sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName print('Loading :',sFileName)

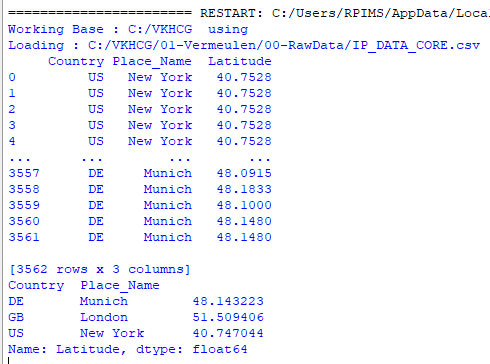
IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False, usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")

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:



1. **Outlier Detection** import pandas as pd InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv' Base='C:/VKHCG'

print('Working Base :',Base)

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)

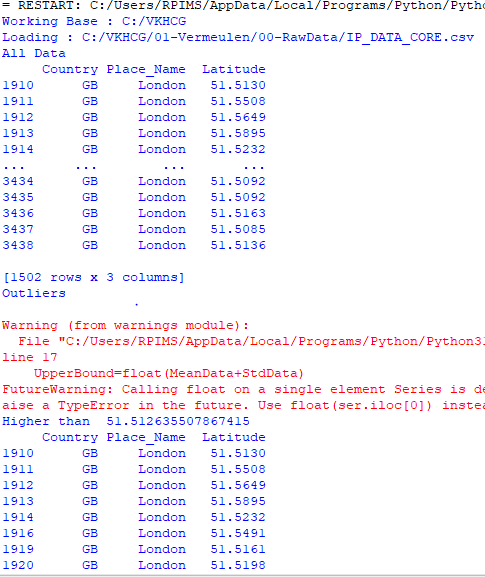
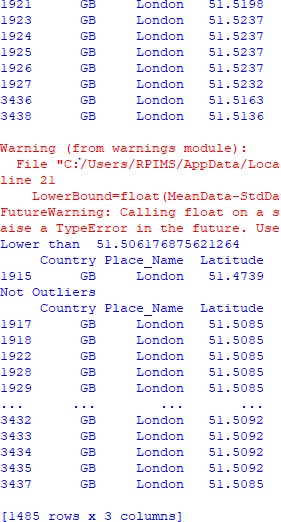
LowerBound=float(MeanData-StdData) print('Lower than ', LowerBound)

OutliersLower=AllData[AllData.Latitude<LowerBound] print(OutliersLower)

print('Not Outliers')

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

## Output:

## Audit

1. **Logging** 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() for hdlr in log.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='Apllication-'+ sCompany + '-' + sLayer + '-' + sLevel logger = logging.getLogger(sApp)

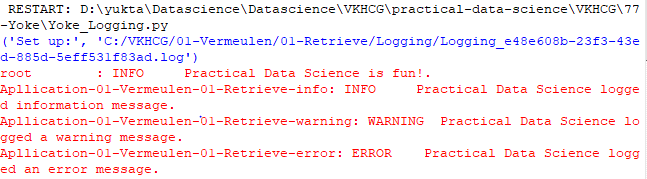
if sLevel == 'debug':

logger.debug('Practical Data Science logged a debugging message.') if sLevel == 'info':

logger.info('Practical Data Science logged information message.') if sLevel == 'warning':

logger.warning('Practical Data Science logged a warning message.') if sLevel == 'error':

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

**Output:**

**Practical 4**

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

**Code**

library(readr)

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

Output spec(IP\_DATA\_ALL)

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

IP\_DATA\_ALL\_FIX <- read.csv("E:/NIKHILESH/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/01-R/IP\_DATA\_ALL\_FIX.csv") 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)) setorder(hist\_country,'Country')

hist\_country\_with\_id=rowid\_to\_column(hist\_country, var = "RowIDCountry") View(hist\_country\_fix) IP\_DATA\_COUNTRY\_FREQ=data.table(with(IP\_DATA\_ALL\_FIX, table(Country))) View(IP\_DATA\_COUNTRY\_FREQ)

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)

Finding mean median range and quantile following are the commands are used- 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)

Finding the standard deviation of any column in table the commands will be –

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

## Program to retrieve different attributes of data.

Code- import sys import os

import pandas as pd

sFileName='E:/NIKHILESH/VKHCG/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv' print('Loading :',sFileName) IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1") sFileDir='E:/NIKHILESH/VKHCG/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('Fixed Data Set with ID') IP\_DATA\_ALL\_with\_ID=IP\_DATA\_ALL\_FIX IP\_DATA\_ALL\_with\_ID.index.names = ['RowID'] sFileName2=sFileDir + '/Retrieve\_IP\_DATA.csv'

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

Output-

## Data Pattern

Code

Write the program using r Studio 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

## Loading IP\_DATA\_ALL:

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

## Vermeulen PLC

Code import sys import os

import pandas as pd

from math import radians, cos, sin, asin, sqrt # Function to calculate haversine distance def haversine(lon1, lat1, lon2, lat2, stype):

# Convert decimal degrees to radians

lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2]) dlon = lon2 - lon1

dlat = lat2 - lat1

a = sin(dlat / 2)\*\*2 + cos(lat1) \* cos(lat2) \* sin(dlon / 2)\*\*2 c = 2 \* asin(sqrt(a))

# Determine the radius of Earth based on the unit type if stype == 'km':

r = 6371 # Radius of Earth in kilometers else:

r = 3956 # Radius of Earth in miles

# Calculate and return the distance d = round(c \* r, 3)

return d # File paths

sFileName = 'E:/NIKHILESH/VKHCG/01-Vermeulen/00-RawData/IP\_DATA\_CORE.csv' sFileDir = 'E:/NIKHILESH/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python'

# Check if output directory exists; create if not if not os.path.exists(sFileDir):

os.makedirs(sFileDir) # Load the CSV file

print('Loading:', sFileName) IP\_DATA\_ALL = pd.read\_csv(

sFileName, header=0, low\_memory=False,

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

)

# Process the data

IP\_DATA = IP\_DATA\_ALL.drop\_duplicates(subset=None, keep='first', inplace=False) IP\_DATA.rename(columns={'Place Name': 'Place\_Name'}, inplace=True)

IP\_DATA1 = IP\_DATA.copy() IP\_DATA1.insert(0, 'K', 1) IP\_DATA2 = IP\_DATA1.copy()

# Cross-join to calculate pairwise distances

IP\_CROSS = pd.merge(right=IP\_DATA1, left=IP\_DATA2, on='K') IP\_CROSS.drop('K', axis=1, inplace=True)

# Rename columns for clarity IP\_CROSS.rename(columns={

'Longitude\_x': 'Longitude\_from', 'Longitude\_y': 'Longitude\_to', 'Latitude\_x': 'Latitude\_from', 'Latitude\_y': 'Latitude\_to', 'Place\_Name\_x': 'Place\_Name\_from', 'Place\_Name\_y': 'Place\_Name\_to', 'Country\_x': 'Country\_from', 'Country\_y': 'Country\_to'

}, inplace=True)

# Calculate distances in kilometers and miles IP\_CROSS['DistanceBetweenKilometers'] = IP\_CROSS.apply(

lambda row: haversine( row['Longitude\_from'], row['Latitude\_from'], row['Longitude\_to'], row['Latitude\_to'], 'km'

),

axis=1

)

IP\_CROSS['DistanceBetweenMiles'] = IP\_CROSS.apply( lambda row: haversine(

row['Longitude\_from'], row['Latitude\_from'], row['Longitude\_to'], row['Latitude\_to'], 'miles'

),

axis=1

)

# Save the result to a CSV file print('Saving results...')

sFileName2 = os.path.join(sFileDir, 'Retrieve\_IP\_Routing.csv') IP\_CROSS.to\_csv(sFileName2, index=False, encoding="latin-1") print('### Done!! ############################################')

output –

See the file named Retrieve\_IP\_Routing.csv in C:\VKHCG\01-Vermeulen\01-Retrieve\01-EDS\02- Total Records: 22501

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

## Building a Diagram for the Scheduling of Jobs

Code import sys import os

import pandas as pd InputFileName='IP\_DATA\_CORE.csv' OutputFileName='Retrieve\_Router\_Location.csv'

sFileName='E:/NIKHILESH/VKHCG/01-Vermeulen/00-RawData/' + InputFileName print('Loading :',sFileName) IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False, usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1") IP\_DATA\_ALL.rename(columns={'Place Name': 'Place\_Name'}, inplace=True) sFileDir='E:/NIKHILESH/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

ROUTERLOC = IP\_DATA\_ALL.drop\_duplicates(subset=None, keep='first', inplace=False) print('Rows :',ROUTERLOC.shape[0])

print('Columns :',ROUTERLOC.shape[1]) sFileName2=sFileDir + '/' + OutputFileName

ROUTERLOC.to\_csv(sFileName2, index = False, encoding="latin-1") print('### Done!! ############################################')

output

## Understanding Your Online Visitor Data

Code import sys import os

import pandas as pd import gzip as gz

InputFileName='IP\_DATA\_ALL.csv' OutputFileName='Retrieve\_Online\_Visitor' CompanyIn= '01-Vermeulen' CompanyOut= '02-Krennwallner' Base='E:/NIKHILESH/VKHCG/'

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

print('Working Base :',Base, ' using ', sys.platform) print('################################') Base='E:/NIKHILESH/VKHCG/'

sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False,usecols=['Country','Place.Name','Latitude','Longitude','Fi rst.IP.Number','Last.IP.Number'])

IP\_DATA\_ALL.rename(columns={'Place Name': 'Place\_Name'}, inplace=True) IP\_DATA\_ALL.rename(columns={'First IP Number': 'First\_IP\_Number'}, inplace=True) IP\_DATA\_ALL.rename(columns={'Last IP Number': 'Last\_IP\_Number'}, inplace=True) sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

visitordata = IP\_DATA\_ALL.drop\_duplicates(subset=None, keep='first', inplace=False) visitordata10=visitordata.head(10)

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

sFileName2=sFileDir + '/' + OutputFileName + '.csv' visitordata.to\_csv(sFileName2, index = False) print('Store All:',sFileName2)

sFileName3=sFileDir + '/' + OutputFileName + '\_10.csv' visitordata10.to\_csv(sFileName3, index = False)

print('Store 10:',sFileName3) for z in ['gzip', 'bz2', 'xz']:

if z == 'gzip':

sFileName4=sFileName2 + '.gz' else:

sFileName4=sFileName2 + '.' + z visitordata.to\_csv(sFileName4, index = False, compression=z) print('Store :',sFileName4)

print('Export JSON')

for sOrient in ['split','records','index', 'columns','values','table']: sFileName2=sFileDir + '/' + OutputFileName + '\_' + sOrient + '.json'

visitordata.to\_json(sFileName2,orient=sOrient,force\_ascii=True) print('Store All:',sFileName2)

sFileName3=sFileDir + '/' + OutputFileName + '\_10\_' + sOrient + '.json' visitordata10.to\_json(sFileName3,orient=sOrient,force\_ascii=True) print('Store 10:',sFileName3)

sFileName4=sFileName2 + '.gz' file\_in = open(sFileName2, 'rb') file\_out = gz.open(sFileName4, 'wb') file\_out.writelines(file\_in) file\_in.close()

file\_out.close()

print('Store GZIP All:',sFileName4)

sFileName5=sFileDir + '/' + OutputFileName + '\_' + sOrient + '\_UnGZip.json' file\_in = gz.open(sFileName4, 'rb')

file\_out = open(sFileName5, 'wb') file\_out.writelines(file\_in) file\_in.close()

file\_out.close()

print('Store UnGZIP All:',sFileName5)

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

output

## XML processing

Code import sys import os

import pandas as pd

import xml.etree.ElementTree as ET def df2xml(data):

header = data.columns root = ET.Element('root')

for row in range(data.shape[0]): entry = ET.SubElement(root,'entry')

for index in range(data.shape[1]): schild=str(header[index])

child = ET.SubElement(entry, schild) if str(data[schild][row]) != 'nan':

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

child.text = 'n/a' entry.append(child) result = ET.tostring(root) return result

def xml2df(xml\_data):

root = ET.XML(xml\_data) all\_records = []

for i, child in enumerate(root):

record = {}

for subchild in child:

record[subchild.tag] = subchild.text all\_records.append(record)

return pd.DataFrame(all\_records) InputFileName='IP\_DATA\_ALL.csv' OutputFileName='Retrieve\_Online\_Visitor.xml' CompanyIn= '01-Vermeulen'

CompanyOut= '02-Krennwallner' if sys.platform == 'linux':

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

Base='E:/NIKHILESH/VKHCG/'

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

sFileName=Base + '/' + CompanyIn + '/00-RawData/' + InputFileName print('Loading :',sFileName) IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False) IP\_DATA\_ALL.rename(columns={'Place Name': 'Place\_Name'}, inplace=True)

IP\_DATA\_ALL.rename(columns={'First IP Number': 'First\_IP\_Number'}, inplace=True) IP\_DATA\_ALL.rename(columns={'Last IP Number': 'Last\_IP\_Number'}, inplace=True) IP\_DATA\_ALL.rename(columns={'Post Code': 'Post\_Code'}, inplace=True) sFileDir=Base + '/' + CompanyOut + '/01-Retrieve/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

visitordata = IP\_DATA\_ALL.head(10000) print('Original Subset Data Frame') print('Rows :',visitordata.shape[0]) print('Columns :',visitordata.shape[1]) print(visitordata)

print('Export XML') sXML=df2xml(visitordata) sFileName=sFileDir + '/' + OutputFileName file\_out = open(sFileName, 'wb') file\_out.write(sXML)

file\_out.close()

print('Store XML:',sFileName) xml\_data = open(sFileName).read() unxmlrawdata=xml2df(xml\_data) print('Raw XML Data Frame') print('Rows :',unxmlrawdata.shape[0])

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

unxmldata = unxmlrawdata.drop\_duplicates(subset=None, keep='first', inplace=False) print('Deduplicated XML Data Frame')

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

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

Output

## Adopt New Shipping Containers

Code

import sys import os

import pandas as pd

ContainerFileName = 'Retrieve\_Container.csv' BoxFileName = 'Retrieve\_Box.csv' ProductFileName = 'Retrieve\_Product.csv' Company = '03-Hillman'

Base = 'E:/NIKHILESH/10th .pdfVKHCG' print('Working Base :', Base, ' using ', sys.platform)

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

os.makedirs(sFileDir) containerLength = range(1, 21) containerWidth = range(1, 10) containerHeigth = range(1, 6) containerStep = 1

c = 0

# Initialize an empty DataFrame for containers ContainerFrame = pd.DataFrame()

for l in containerLength:

for w in containerWidth:

for h in containerHeigth:

containerVolume = (l / containerStep) \* (w / containerStep) \* (h / containerStep) c += 1

ContainerLine = { 'ShipType': 'Container',

'UnitNumber': 'C' + format(c, "06d"), 'Length': round(l, 4),

'Width': round(w, 4),

'Height': round(h, 4),

'ContainerVolume': round(containerVolume, 6)

}

ContainerRow = pd.DataFrame([ContainerLine])

ContainerFrame = pd.concat([ContainerFrame, ContainerRow], ignore\_index=True)

ContainerFrame.index.name = 'IDNumber' print('################')

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

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

sFileContainerName = sFileDir + '/' + ContainerFileName ContainerFrame.to\_csv(sFileContainerName, index=False) boxLength = range(1, 21)

boxWidth = range(1, 21) boxHeigth = range(1, 21) packThick = range(0, 6) boxStep = 10

b = 0

# Initialize an empty DataFrame for boxes BoxFrame = pd.DataFrame()

for l in boxLength: for w in boxWidth:

for h in boxHeigth: for t in packThick:

boxVolume = round((l / boxStep) \* (w / boxStep) \* (h / boxStep), 6)

productVolume = round(((l - t) / boxStep) \* ((w - t) / boxStep) \* ((h - t) / boxStep), 6) if productVolume > 0:

b += 1

BoxLine = { 'ShipType': 'Box',

'UnitNumber': 'B' + format(b, "06d"), 'Length': round(l / 10, 6),

'Width': round(w / 10, 6),

'Height': round(h / 10, 6),

'Thickness': round(t / 5, 6),

'BoxVolume': round(boxVolume, 9),

'ProductVolume': round(productVolume, 9)

}

BoxRow = pd.DataFrame([BoxLine])

BoxFrame = pd.concat([BoxFrame, BoxRow], ignore\_index=True)

BoxFrame.index.name = 'IDNumber' print('## Box####')

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

sFileBoxName = sFileDir + '/' + BoxFileName BoxFrame.to\_csv(sFileBoxName, index=False) productLength = range(1, 21)

productWidth = range(1, 21) productHeigth = range(1, 21) productStep = 10

p = 0

# Initialize an empty DataFrame for products ProductFrame = pd.DataFrame()

for l in productLength:

for w in productWidth:

for h in productHeigth:

productVolume = round((l / productStep) \* (w / productStep) \* (h / productStep), 6) if productVolume > 0:

p += 1

ProductLine = { 'ShipType': 'Product',

'UnitNumber': 'P' + format(p, "06d"), 'Length': round(l / 10, 6),

'Width': round(w / 10, 6),

'Height': round(h / 10, 6),

'ProductVolume': round(productVolume, 9)

}

ProductRow = pd.DataFrame([ProductLine])

ProductFrame = pd.concat([ProductFrame, ProductRow], ignore\_index=True)

ProductFrame.index.name = 'IDNumber' print('## Product')

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

sFileProductName = sFileDir + '/' + ProductFileName ProductFrame.to\_csv(sFileProductName, index=False)

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

output

**Global Post Codes** Code in r studio library(readr)

All\_Countries <- read\_delim("C:/VKHCG/03-Hillman/00-RawData/All\_Countries.txt", "\t", col\_names = FALSE,

col\_types = cols( X12 = col\_skip(), X6 = col\_skip(), X7 = col\_skip(), X8 = col\_skip(), X9 = col\_skip()),

na = "null", trim\_ws = TRUE) write.csv(All\_Countries,

file = "C:/VKHCG/03-Hillman/01-Retrieve/01-EDS/01-R/Retrieve\_All\_Countries.csv") output

The program will successfully uploaded a new file named Retrieve\_All\_Countries.csv, after removing column No. 6, 7, 8, 9 and 12 from All\_Countries.txt

## Program to connect to different data sources.

Code

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('## Data Values')

print(TestData)**\ Practical no 5 Assessing Data**

Perform error management on the given data using pandas package. Drop the Columns Where All Elements Are Missing Values

Code:- import sys import os

import pandas as pd

if sys.platform == 'linux': Base=os.path.expanduser('~') + 'VKHCG'

else:

Base='C:/VKHCG'

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

sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName print('Loading :',sFileName) RawData=pd.read\_csv(sFileName,header=0)

print('## Raw Data Values') print(RawData)

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

print('Columns :',RawData.shape[1]) sFileName=sFileDir + '/' + sInputFileName RawData.to\_csv(sFileName, index = False) TestData=RawData.dropna(axis=1, how='all') print('## Test Data Values')

print(TestData) print('## Data Profile')

print('Rows :',TestData.shape[0]) print('Columns :',TestData.shape[1]) sFileName=sFileDir + '/' + sOutputFileName TestData.to\_csv(sFileName, index = False) print('### Done!! #####################')

## Write Python / R program to create the network routing diagram from the given data On routers.

Code:- import sys import os

import pandas as pd pd.options.mode.chained\_assignment = None if sys.platform == 'linux':

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

Base='C:/VKHCG'

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

sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve\_Country\_Code.csv' sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve\_Router\_Location.csv' sInputFileName3='01-Retrieve/01-EDS/01-R/Retrieve\_IP\_DATA.csv' sOutputFileName='Assess-Network-Routing-Company.csv'

Company='01-Vermeulen' ### Import Country Data

sFileName=Base + '/' + Company + '/' + sInputFileName1 print('Loading :',sFileName)

CountryData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1") print('Loaded Country:',CountryData.columns.values)

## Assess Country Data

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)

### Import Company Data

sFileName=Base + '/' + Company + '/' + sInputFileName2

print('Loading :',sFileName) CompanyData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1") print('Loaded Company :',CompanyData.columns.values)

Code:-

####################Assess-Network-Routing- Customer.py ###################### import sys

import os

import pandas as pd pd.options.mode.chained\_assignment = None Base='C:/VKHCG'

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

sInputFileName=Base+'/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing- Customer.csv'

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

### Import Country Data sFileName=sInputFileName print('Loading :',sFileName)

CustomerData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1") print('Loaded Country:',CustomerData.columns.values)

print(CustomerData.head())

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

Output:-

Code:-

## Assess-Network-Routing-Node.py

import sys import os

import pandas as pd pd.options.mode.chained\_assignment = None

Base='C:/VKHCG'

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

sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve\_IP\_DATA.csv' sOutputFileName='Assess-Network-Routing-Node.csv'

Company='01-Vermeulen' ### Import IP Data

sFileName=Base + '/' + Company + '/' + sInputFileName print('Loading :',sFileName)

IPData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1") print('Loaded IP :', IPData.columns.values)

print('Changed :',IPData.columns.values) IPData.drop('RowID', axis=1, inplace=True) IPData.drop('ID', axis=1, inplace=True)

IPData.rename(columns={'Country': 'Country\_Code'}, inplace=True) IPData.rename(columns={'Place.Name': 'Place\_Name'}, inplace=True) IPData.rename(columns={'Post.Code': 'Post\_Code'}, inplace=True) IPData.rename(columns={'First.IP.Number': 'First\_IP\_Number'}, inplace=True) IPData.rename(columns={'Last.IP.Number': 'Last\_IP\_Number'}, inplace=True) print('To :',IPData.columns.values)

print('Change ',IPData.columns.values) for i in IPData.columns.values: j='Node\_'+i

IPData.rename(columns={i: j}, inplace=True) print('To ', IPData.columns.values)

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName print('Storing :', sFileName)

IPData.to\_csv(sFileName, index = False, encoding="latin-1") print('### Done!! #####################')

Output:-

## Write a Python / R program to build directed acyclic graph

Code:-

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

import sys import os

import sqlite3 as sq import pandas as pd

if sys.platform == 'linux':

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

Base='C:/VKHCG'

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

sInputFileName1='01-Retrieve/01-EDS/02-Python/Retrieve\_DE\_Billboard\_Locations.csv' sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve\_Online\_Visitor.csv' sOutputFileName='Assess-DE-Billboard-Visitor.csv'

Company='02-Krennwallner'

sDataBaseDir=Base + '/' + Company + '/02-Assess/SQLite' if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir) sDatabaseName=sDataBaseDir + '/krennwallner.db' conn = sq.connect(sDatabaseName)

### Import Billboard Data

sFileName=Base + '/' + Company + '/' + sInputFileName1 print('Loading :',sFileName)

BillboardRawData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1") BillboardRawData.drop\_duplicates(subset=None, keep='first', inplace=True)

BillboardData=BillboardRawData

print('Loaded Company :',BillboardData.columns.values) sTable='Assess\_BillboardData'

print('Storing :',sDatabaseName,' Table:',sTable) BillboardData.to\_sql(sTable, conn, if\_exists="replace") print(BillboardData.head())

print('Rows : ',BillboardData.shape[0]) ### Import Billboard Data

sFileName=Base + '/' + Company + '/' + sInputFileName2 print('Loading :',sFileName)

VisitorRawData=pd.read\_csv(sFileName,header=0,low\_memory=False, encoding="latin-1") VisitorRawData.drop\_duplicates(subset=None, keep='first', inplace=True) VisitorData=VisitorRawData[VisitorRawData.Country=='DE']

print('Loaded Company :',VisitorData.columns.values) sTable='Assess\_VisitorData'

print('Storing :',sDatabaseName,' Table:',sTable) VisitorData.to\_sql(sTable, conn, if\_exists="replace") print(VisitorData.head())

print('Rows : ',VisitorData.shape[0]) sTable='Assess\_BillboardVisitorData'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="select distinct"

sSQL=sSQL+ " A.Country AS BillboardCountry," sSQL=sSQL+ " A.Place\_Name AS BillboardPlaceName," sSQL=sSQL+ " A.Latitude AS BillboardLatitude, " sSQL=sSQL+ " A.Longitude AS BillboardLongitude," sSQL=sSQL+ " B.Country AS VisitorCountry," sSQL=sSQL+ " B.Place\_Name AS VisitorPlaceName," sSQL=sSQL+ " B.Latitude AS VisitorLatitude, " sSQL=sSQL+ " B.Longitude AS VisitorLongitude,"

sSQL=sSQL+ " (B.Last\_IP\_Number - B.First\_IP\_Number) \* 365.25 \* 24 \* 12 AS VisitorYearRate" sSQL=sSQL+ " from"

sSQL=sSQL+ " Assess\_BillboardData as A" sSQL=sSQL+ " JOIN "

sSQL=sSQL+ " Assess\_VisitorData as B" sSQL=sSQL+ " ON "

sSQL=sSQL+ " A.Country = B.Country" sSQL=sSQL+ " AND "

sSQL=sSQL+ " A.Place\_Name = B.Place\_Name;" BillboardVistorsData=pd.read\_sql\_query(sSQL, conn) sTable='Assess\_BillboardVistorsData'

print('Storing :',sDatabaseName,' Table:',sTable) BillboardVistorsData.to\_sql(sTable, conn, if\_exists="replace") print(BillboardVistorsData.head())

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

sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python' if not os.path.exists(sFileDir):

os.makedirs(sFileDir) print('Storing :', sFileName)

sFileName=sFileDir + '/' + sOutputFileNameBillboardVistorsData.to\_csv(sFileName, index = False

# Practical 5 Assessing Data

import pandas as pd

def assess\_data\_quality(df): """

Function to assess data quality of a pandas DataFrame.

Outputs summary statistics, missing values, duplicates, and data types. """

print("\n--- Data Summary ---") print(df.describe(include='all'))

print("\n--- Missing Values ---") missing\_values = df.isnull().sum() print(missing\_values[missing\_values > 0])

print("\n--- Duplicates ---") duplicate\_count = df.duplicated().sum()

print(f"Number of duplicate rows: {duplicate\_count}")

print("\n--- Data Types ---") print(df.dtypes)

print("\n--- Unique Values per Column ---") for col in df.columns:

print(f"{col}: {df[col].nunique()} unique values")

print("\n--- Potential Inconsistencies (Categorical Columns) ---") for col in df.select\_dtypes(include='object').columns:

print(f"\nColumn '{col}' unique values:") print(df[col].value\_counts())

# Example usage:

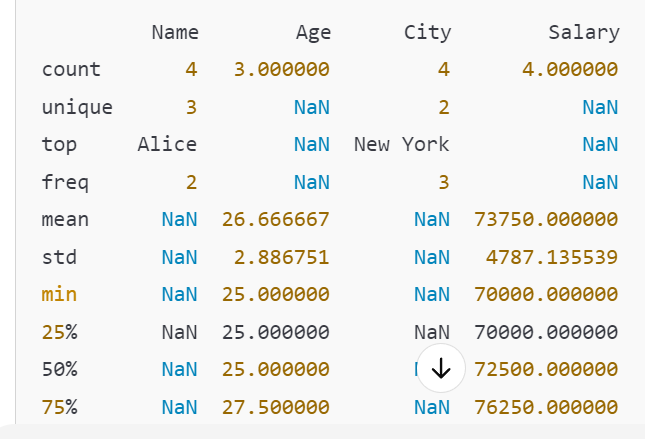
# Load sample data (replace with your dataset) df = pd.DataFrame({

'Name': ['Alice', 'Bob', 'Charlie', 'Alice'], 'Age': [25, 30, None, 25],

'City': ['New York', 'Los Angeles', 'New York', 'New York'], 'Salary': [70000, 80000, 75000, 70000]

})

assess\_data\_quality(df)

Output:

# Practical 6 Build the time hub ,links , and satelites.

**#code**

import datetime import time import threading

class TimeHub:

def init (self):

self.time\_links = []

def add\_time\_link(self, link): self.time\_links.append(link)

def distribute\_time(self):

current\_time = datetime.datetime.utcnow() for link in self.time\_links:

link.receive\_time(current\_time) class TimeLink:

def init (self, satellite):

self.satellite = satellite

def receive\_time(self, current\_time): self.satellite.update\_time(current\_time)

class Satellite:

def init (self, name):

self.name = name self.current\_time = None

def update\_time(self, current\_time): self.current\_time = current\_time

def get\_time(self):

return self.current\_time def main():

time\_hub = TimeHub()

satellite1 = Satellite("Satellite\_1") satellite2 = Satellite("Satellite\_2") link1 = TimeLink(satellite1)

link2 = TimeLink(satellite2) time\_hub.add\_time\_link(link1) time\_hub.add\_time\_link(link2) def sync\_time():

while True: time\_hub.distribute\_time() time.sleep(1)

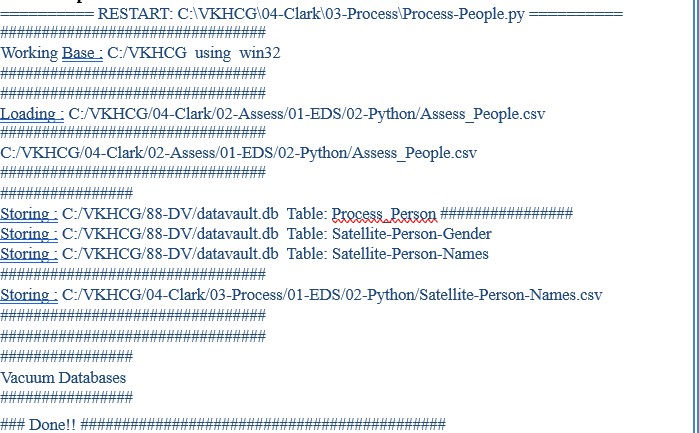
sync\_thread = threading.Thread(target=sync\_time) sync\_thread.daemon = True

sync\_thread.start()

time.sleep(5)

print(f"{satellite1.name} Time: {satellite1.get\_time()}") print(f"{satellite2.name} Time: {satellite2.get\_time()}")

if name == " main ":

main() output

# Practical 7 - Transform-Gunnarsson\_is\_Born.py

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('Working Base :',Base, ' using ', sys.platform) Company='01-Vermeulen' InputDir='00- RawData' InputFileName='VehicleData.csv'

sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite' if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir) sDatabaseName=sDataBaseDir + '/Vermeulen.db' conn1 = sq.connect(sDatabaseName)

sDataVaultDir=Base + '/88-DV'

if not os.path.exists(sDataVaultDir):

os.makedirs(sDataVaultDir) sDatabaseName=sDataVaultDir + '/datavault.db' conn2 = sq.connect(sDatabaseName)

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir): os.makedirs(sDataWarehouseDir)

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db' conn3 = sq.connect(sDatabaseName)

print('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('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)

IDZoneNumber=str(uuid.uuid4()) sDateTimeKey=BirthDateZoneStr.replace(' ','-').replace(':','-') TimeLine=[('ZoneBaseKey', ['UTC']),

('IDNumber', [IDZoneNumber]),

('DateTimeKey', [sDateTimeKey]), ('UTCDateTimeValue', [BirthDateZoneUTC]), ('Zone', [BirthZone]), ('DateTimeValue', [BirthDateStr])]

TimeFrame = pd.DataFrame.from\_dict(dict(TimeLine))

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

sTable = 'Hub-Time-Gunnarsson'

print('Storing :',sDatabaseName,'\n Table:',sTable) TimeHubIndex.to\_sql(sTable, conn2, if\_exists="replace") sTable = 'Dim-Time-Gunnarsson' TimeHubIndex.to\_sql(sTable, conn3,

if\_exists="replace") TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']] TimeSatelliteIndex=TimeSatellite.set\_index(['IDNumber'],inplace=False) BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')

sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson' print('Storing :',sDatabaseName,'\n Table:',sTable) TimeSatelliteIndex.to\_sql(sTable, conn2, if\_exists="replace") sTable = 'Dim-Time-' + BirthZoneFix + '-Gunnarsson'

TimeSatelliteIndex.to\_sql(sTable, conn3, if\_exists="replace") print('Person Category') FirstName =

'Guðmundur' LastName = 'Gunnarsson' print('Name:',FirstName,LastName) print('Birth Date:',BirthDateLocal) print('Birth Zone:',BirthZone)

print('UTC Birth Date:',BirthDateZoneStr) IDPersonNumber=str(uuid.uuid4()) PersonLine=[('IDNumber', [IDPersonNumber]),

('FirstName', [FirstName]),

('LastName', [LastName]),

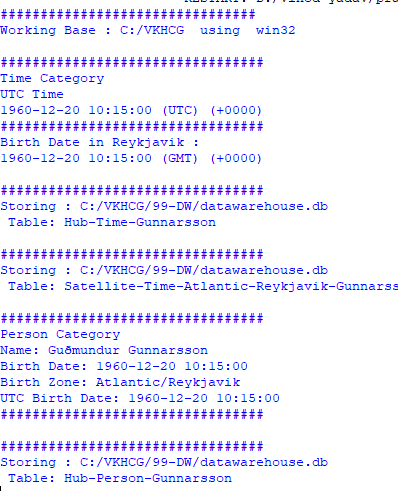
('Zone', ['UTC']),

('DateTimeValue', [BirthDateZoneStr])] PersonFrame = pd.DataFrame.from\_dict(dict(PersonLine)) TimeHub=PersonFrame TimeHubIndex=TimeHub.set\_index(['IDNumber'],inplace=False)

sTable = 'Hub-Person-Gunnarsson'

print('Storing :',sDatabaseName,'\n Table:',sTable) TimeHubIndex.to\_sql(sTable, conn2, if\_exists="replace") sTable = 'Dim-Person-Gunnarsson' TimeHubIndex.to\_sql(sTable, conn3, if\_exists="replace")

**OUTPUT**



# Practical 8 : Organize-Horizontal

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

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1) sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT PersonID,\

Height,\ Weight,\ bmi,\ Indicator\

FROM [Dim-BMI]\ WHERE \

Height > 1.5 \ and Indicator = 1\ ORDER BY \

Height,\ Weight;" PersonFrame1=pd.read\_sql\_query(sSQL, conn1) DimPerson=PersonFrame1

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

sTable = 'Dim-BMI-Horizontal'

print('Storing :',sDatabaseName,'\n Table:',sTable) DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace") sTable = 'Dim-BMI-Horizontal'

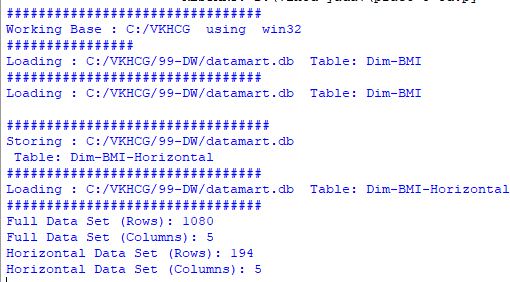
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \*

FROM [Dim-BMI];"

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

print('Full Data Set (Rows):', PersonFrame0.shape[0]) print('Full Data Set (Columns):', PersonFrame0.shape[1]) print('Horizontal Data Set (Rows):', PersonFrame2.shape[0]) print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

**OUTPUT**

****

# Practical 8 : Organize-Vertical

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

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1) sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \

Height,\ Weight,\ Indicator\

FROM [Dim-BMI];"

PersonFrame1=pd.read\_sql\_query(sSQL, conn1) DimPerson=PersonFrame1 DimPersonIndex=DimPerson.set\_index(['Indicator'],inplace=False) sTable = 'Dim-BMI-Vertical')

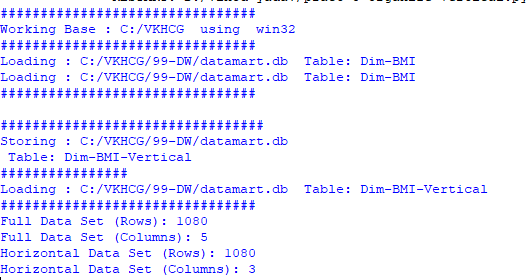
print('Storing :',sDatabaseName,'\n Table:',sTable) DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace") sTable = 'Dim-BMI-Vertical'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \* FROM [Dim-BMI-Vertical];" PersonFrame2=pd.read\_sql\_query(sSQL, conn2)

print('Full Data Set (Rows):', PersonFrame0.shape[0]) print('Full Data Set (Columns):', PersonFrame0.shape[1])

print('Horizontal Data Set (Rows):', PersonFrame2.shape[0]) print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

**OUTPUT**

****

# Practical 8 : Organize-island

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

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1) sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \

Height,\ Weight,\ Indicator\ FROM [Dim-BMI]\

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

DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace") sTable = 'Dim-BMI-Vertical'

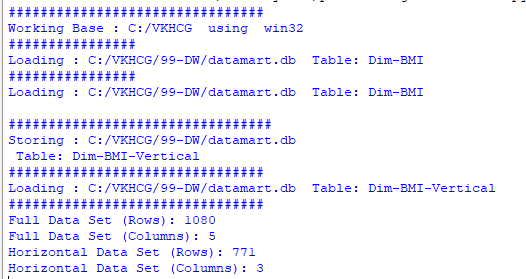
print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \* FROM [Dim-BMI-Vertical];"

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

print('Full Data Set (Rows):', PersonFrame0.shape[0]) print('Full Data Set (Columns):', PersonFrame0.shape[1])

print('Horizontal Data Set (Rows):', PersonFrame2.shape[0]) print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

OUTPUT



## Practical 8 : Organize-secure-vault

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

sTable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \* FROM [Dim-BMI];"

PersonFrame0=pd.read\_sql\_query(sSQL, conn1) STable = 'Dim-BMI'

print('Loading :',sDatabaseName,' Table:',sTable) sSQL="SELECT \

Height,\ Weight,\ Indicator,\ CASE Indicator\

WHEN 1 THEN 'Pip'\

WHEN 2 THEN 'Norman'\ WHEN 3 THEN

'Grant'\ ELSE 'Sam'\

END AS Name\ FROM [Dim-BMI]\

WHERE Indicator > 2\ ORDER BY \ Height,\ Weigh

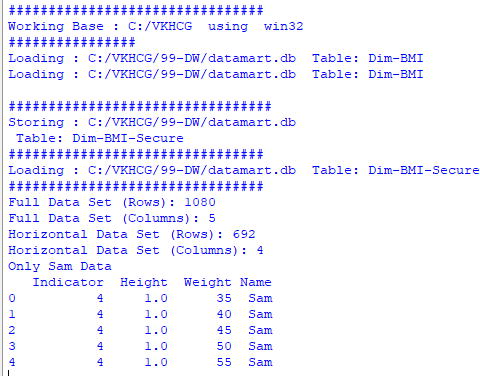
PersonFrame1=pd.read\_sql\_query(sSQL, conn1) DimPerson=PersonFrame1

DimPersonIndex=DimPerson.set\_index(['Indicator'],inplace=False) sTable = 'Dim-BMI-Secure'

print('Storing :',sDatabaseName,'\n Table:',sTable) DimPersonIndex.to\_sql(sTable,

conn2, if\_exists="replace") sTable = 'Dim-BMI-Secure'

**OUTPUT**

****

**Practical No.9**

**A) Generating Reports**

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'

sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Customer.csv' sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.gml' sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-Customer.png' Company='01-Vermeulen'

sFileName=Base + '/' + Company + '/' + sInputFileName CustomerDataRaw=pd.read\_csv(sFileName, header=0, low\_memory=False, encoding="latin-1") CustomerData=CustomerDataRaw.head(100)

G=nx.Graph()

for i in range(CustomerData.shape[0]):

for j in range(CustomerData.shape[0]): Node0=CustomerData['Customer\_Country\_Name'][i] Node1=CustomerData['Customer\_Country\_Name'][j] if Node0 != Node1:

G.add\_edge(Node0,Node1)

for i in range(CustomerData.shape[0]): Node0=CustomerData['Customer\_Country\_Name'][i]

Node1=CustomerData['Customer\_Place\_Name'][i] + '('+ CustomerData['Customer\_Country\_Name'][i] + ')' Node2='('+ "{:.9f}".format(CustomerData['Customer\_Latitude'][i]) + ')('+

"{:.9f}".format(CustomerData['Customer\_Longitude'][i]) + ')' if Node0 != Node1:

G.add\_edge(Node0,Node1) if Node1 != Node2:

G.add\_edge(Node1,Node2)

sFileName=Base + '/' + Company + '/' + sOutputFileName1 nx.write\_gml(G, sFileName)

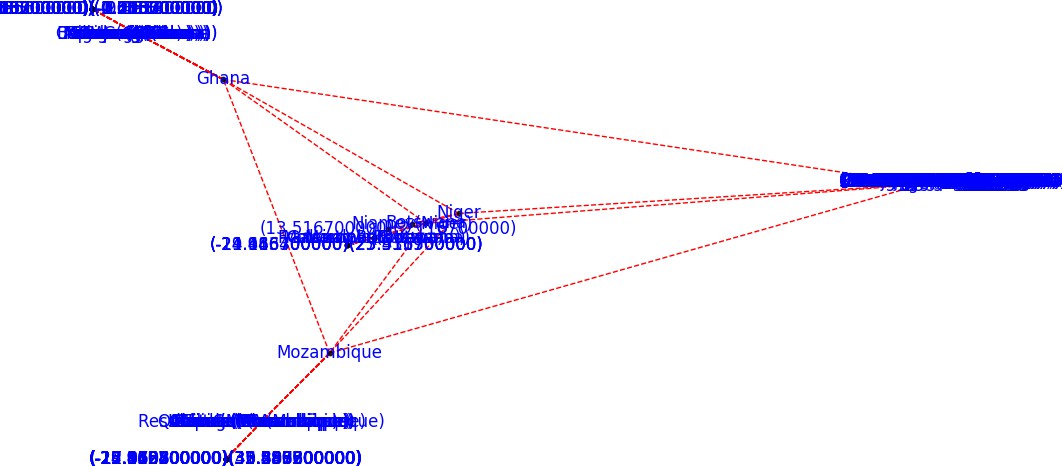
sFileName=Base + '/' + Company + '/' + sOutputFileName2 plt.figure(figsize=(25, 25))

pos=nx.spectral\_layout(G, dim=2)

nx.draw\_networkx\_nodes(G, pos, node\_color='k', node\_size=10, alpha=0.8) nx.draw\_networkx\_edges(G, pos, edge\_color='r', arrows=False, style='dashed') nx.draw\_networkx\_labels(G, pos, font\_size=12, font\_family='sans-serif', font\_color='b') plt.axis('off')

plt.savefig(sFileName, dpi=600) plt.show()

## Output:

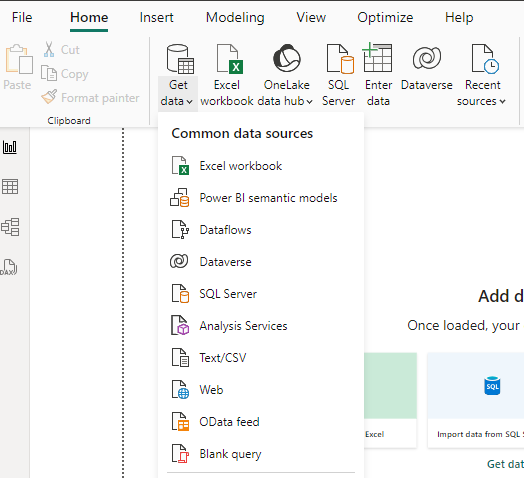


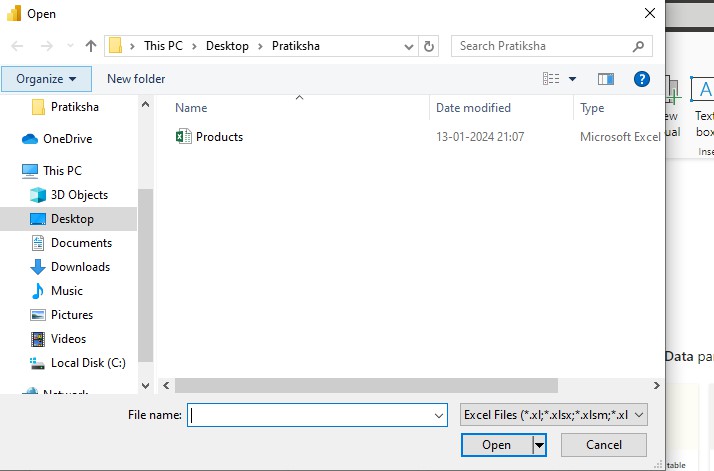
**Practical No. 10**

Data Visualization with power bi Case Study : Sales Data

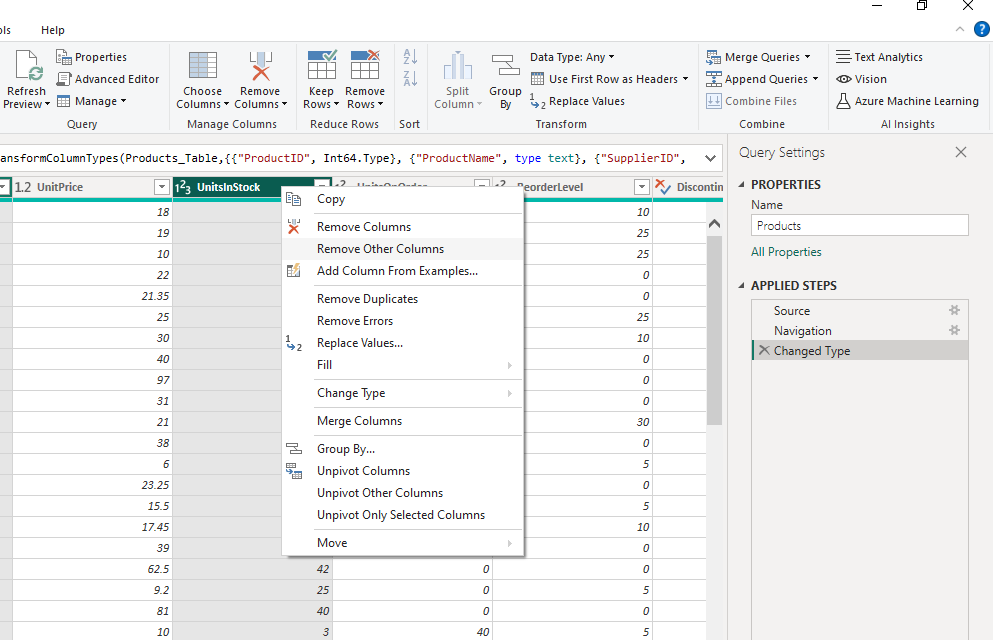
Step 1: connect to an Excel Workbook

1. Launch power Bi Desktop.
2. From the Home Ribbon, Select Get Data  Select Excel Workbook .
3. In the Open File Dialog Box, Select the Product.xlsx file.

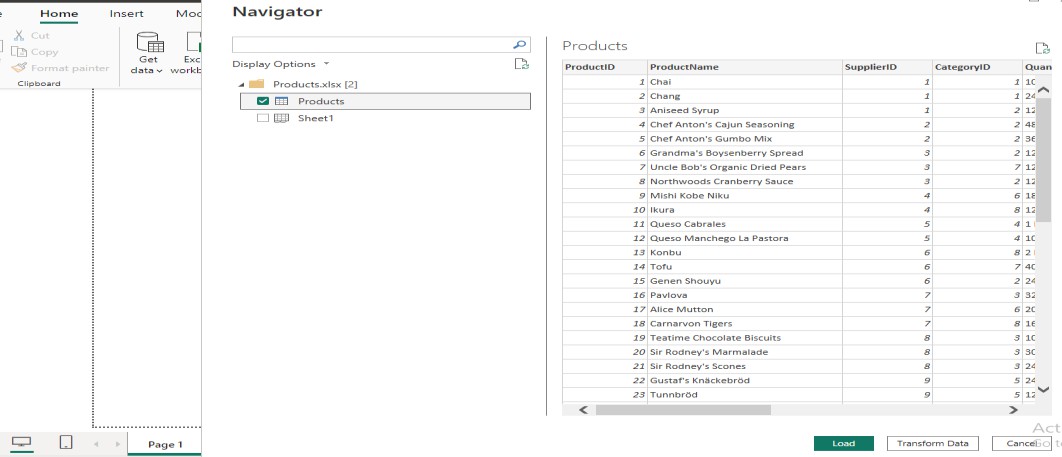


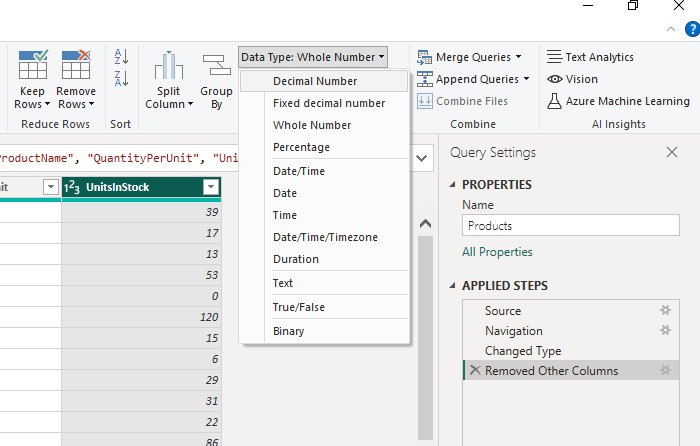


1. In the Open File Dialog Box, Select the Product.xlsx file.
2. Click on Products Check Box. & We Will see the product Table. Select The Transform Data.
3. In Query Editor , Select the ProductID, ProductName, QuantityPerUnit, and unitsInStock Columns. ( Use Ctrl + Click to select more than one column ).



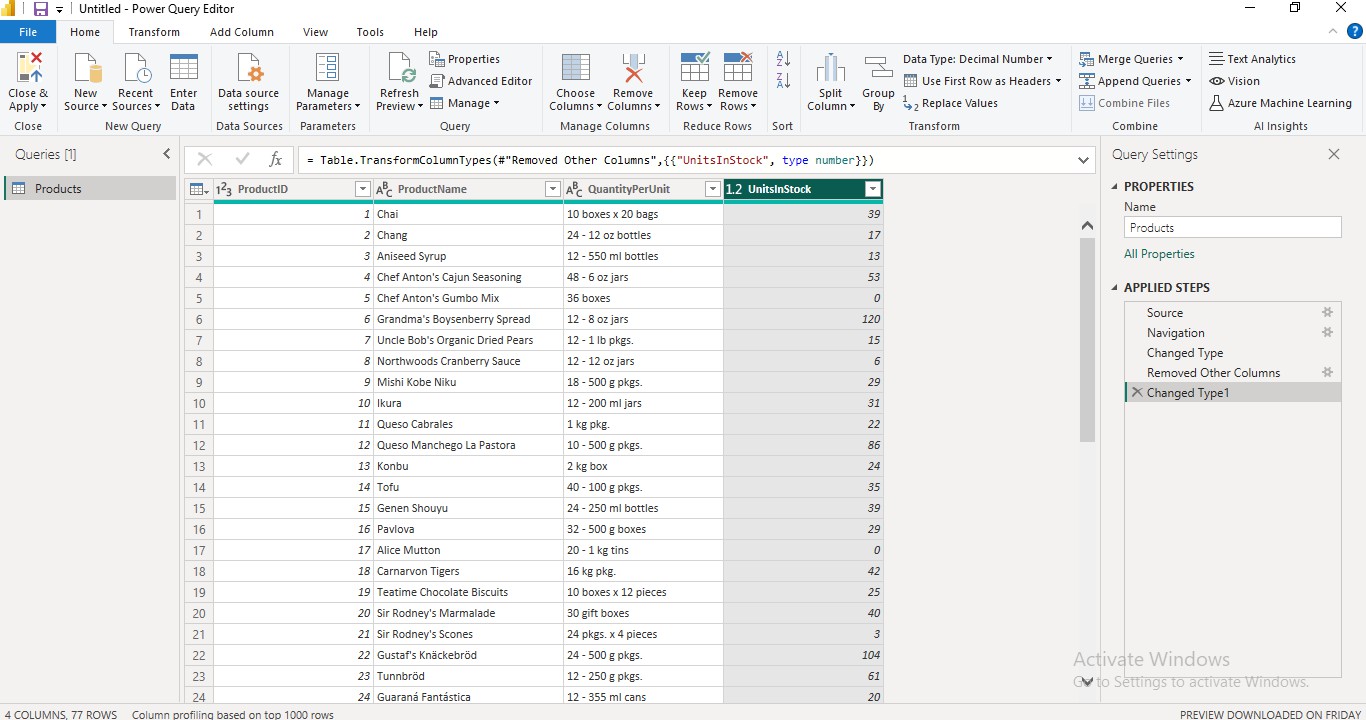
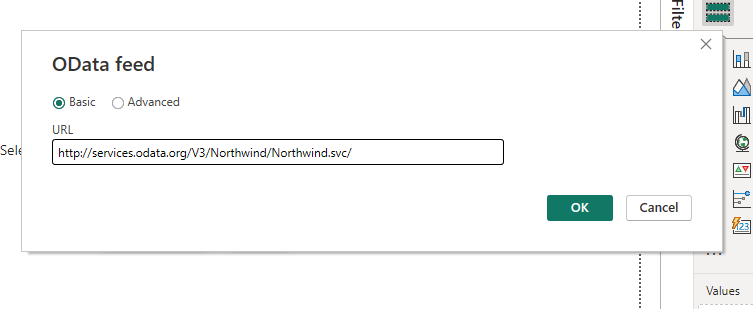
1. Right Click on Column Header and Click Remove Other Columns.



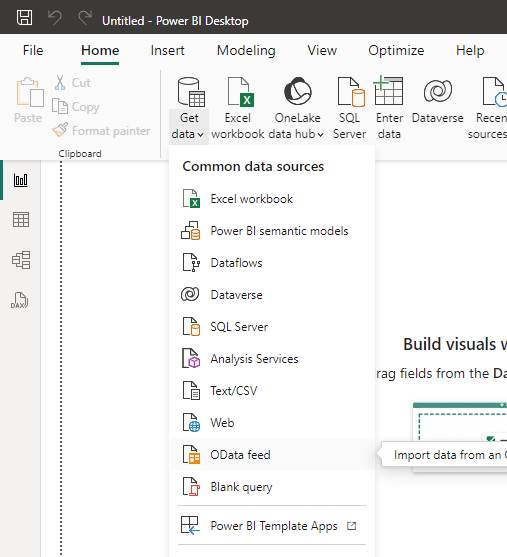


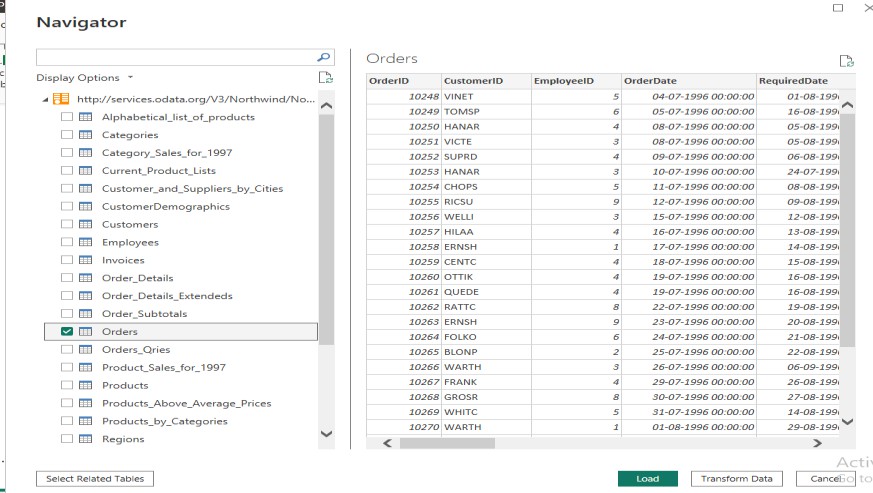
1. Select Close & Apply from Home Ribbon.
2. Another Window on  Select Get Data & Select the OData feed. And Copy the link given below. & Paste it to OData feed URL Box and Click Ok.

<http://services.odata.org/V3/Northwind/Northwind.svc/>

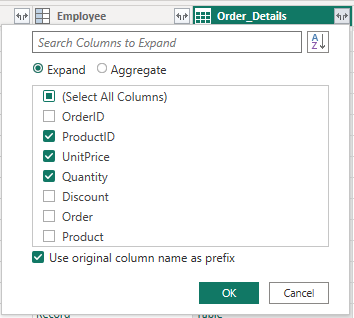


1. We will see the below screen like this then select from navigator orders checkbox. & Click on Transform Data.

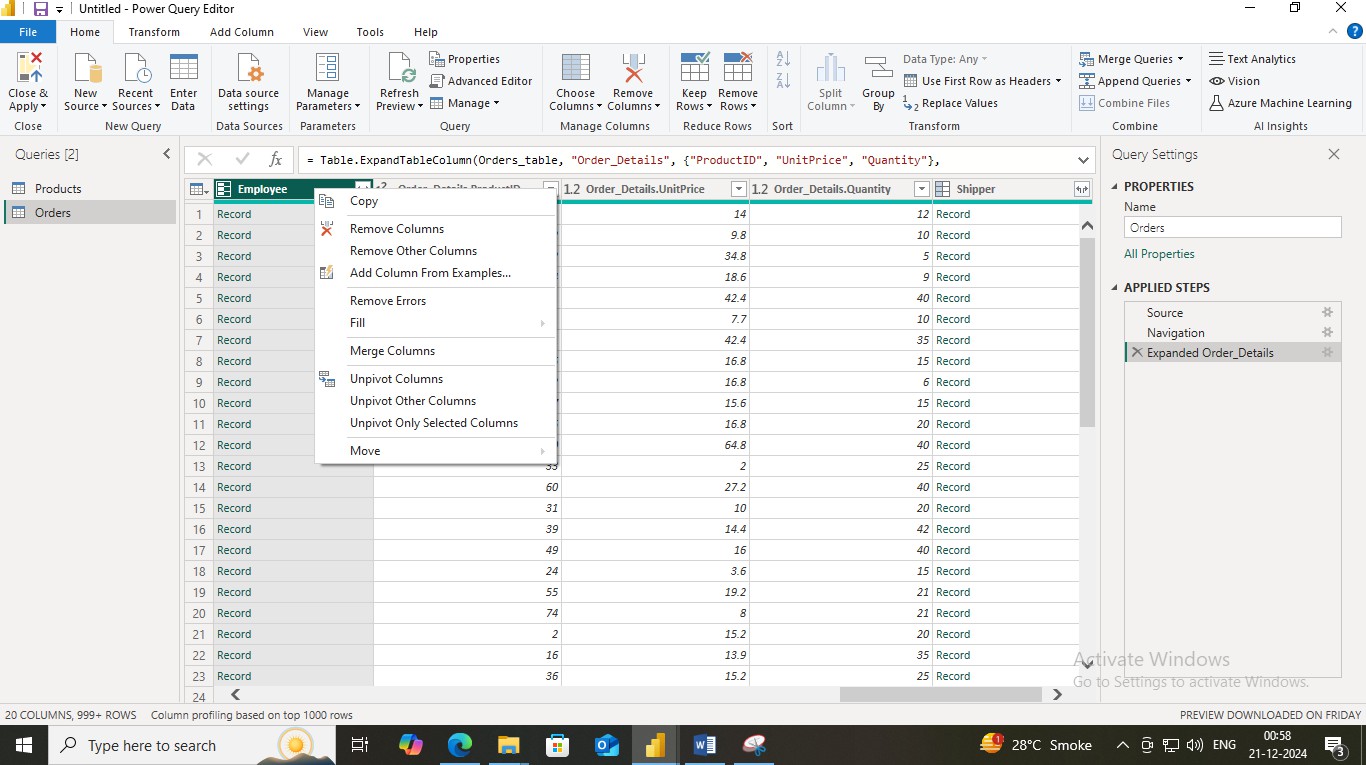




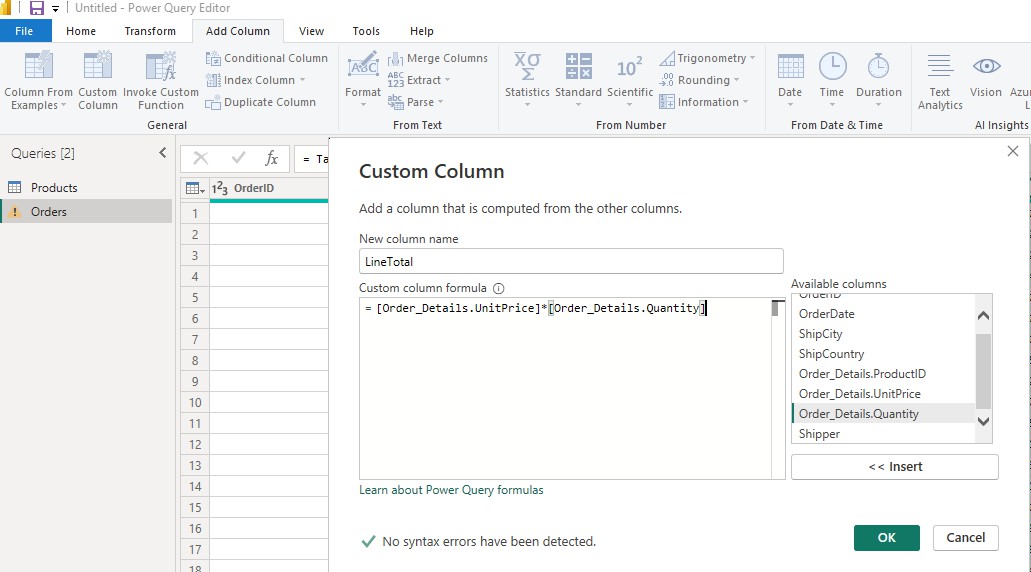
1. Expand the Order\_Details column & select the ProductID, UnitPrice, Quantity & Click OK.



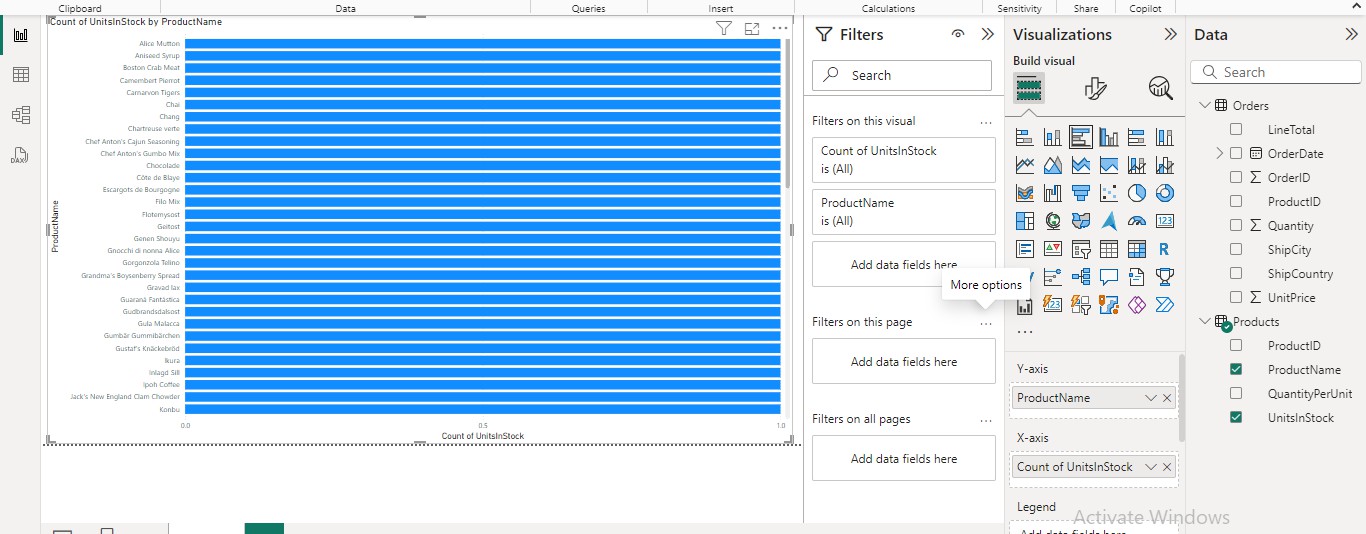
1. Remove other Column to only display column of interest, In this step you remove all Column except OrderID, OrderDate, ShipCity, Order\_Details.ProductID, Order\_Details.Unitprice, Order\_Details.Quantity, Shipper columns. & Remove Columns.



1. From Add Column Ribbon Select Custom Column. Add New Name in new Column name LineTotal. From Available column Select order\_Details.Unitprice and Click insert Add “\*” and select Order\_Details.Quantity and insert  Ok. We Will see the a New Column Name LineTotal Appears.



1. In Query Editor, drag the LineTotal Column to the left , After ShipCountry.  Double Click on Order\_Details.ProductID, Order\_Details.Unitprice, Order\_Details.Quantity change name to Only ProductID, Unitprice, Quantity.
2. From Home Ribbon , Select Close and apply. We Will get new Window of Power Bi. Select From Data Paneel From Products select ProductName And UnitInStock. If output is not seen then Change X-axis and Y-axis from Visualizations.



1. For Orders Select Map from Visualizations. And From data Column Select From Orders Select LineTotal And ShipCity.