**Practical 1:**

**Creating Data Model using Cassandra.**

**Cassandra Data Model**

**Step-1:**

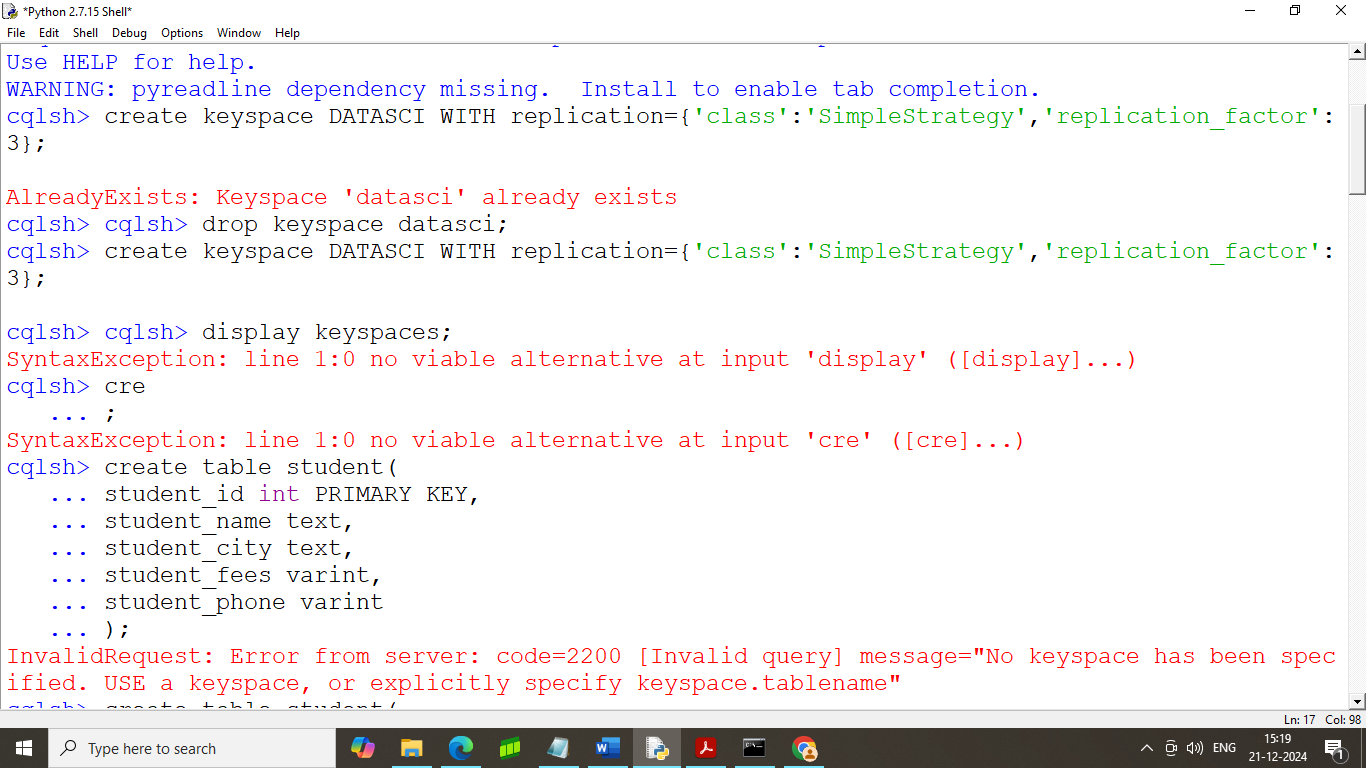
- Open a folder Datascience\apache-cassandra-3.11.4-bin\apache-cassandra-3.11.4\bin\cassandra.bat

- 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,

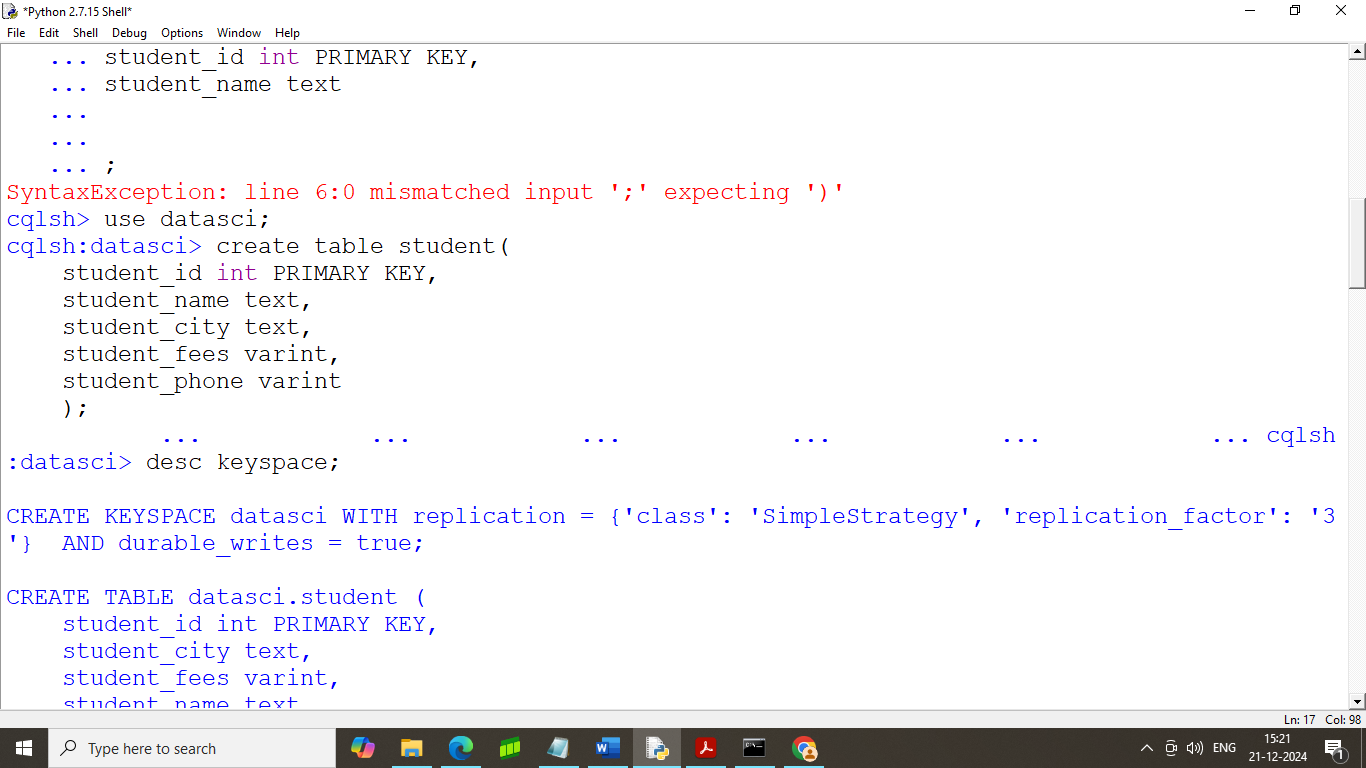
student\_name text,

student\_city text,

student\_fees varint,

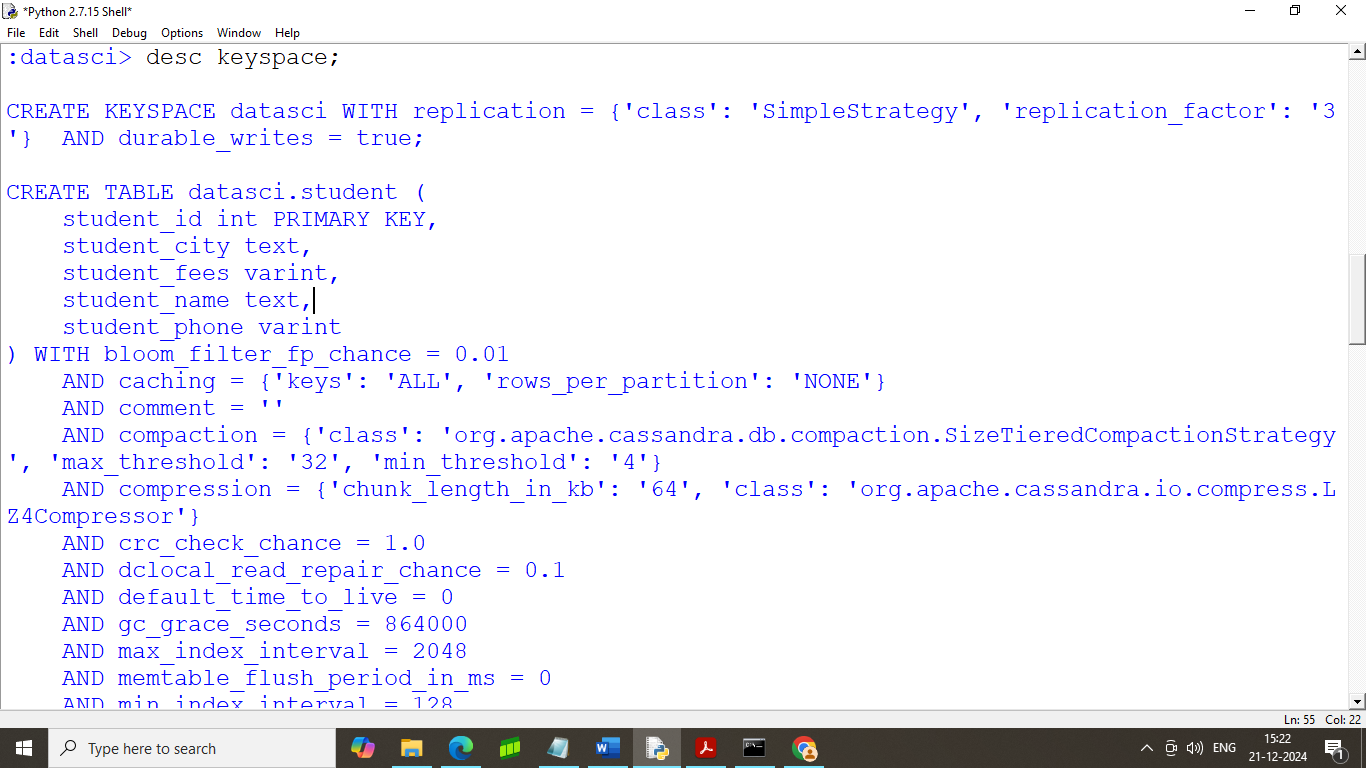
student\_phone varint

);



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

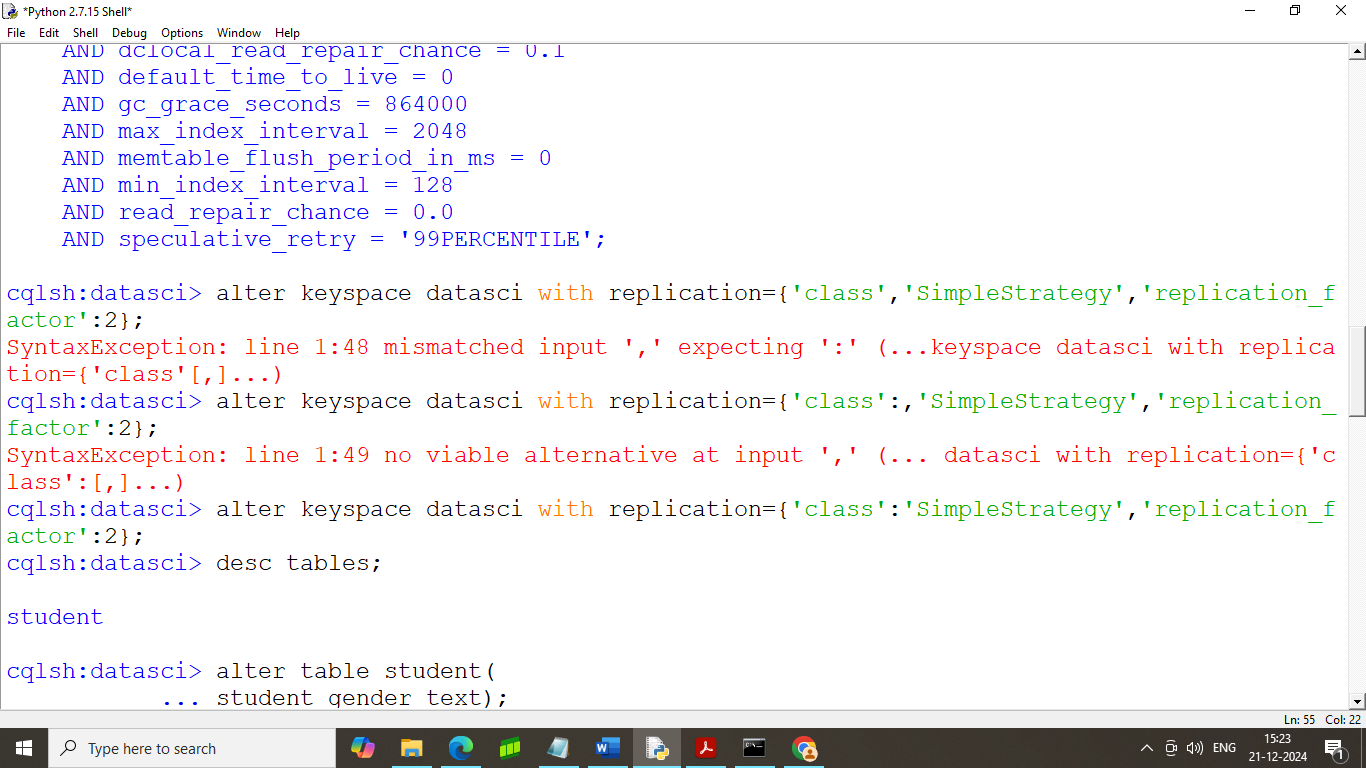
Desc keyspace;





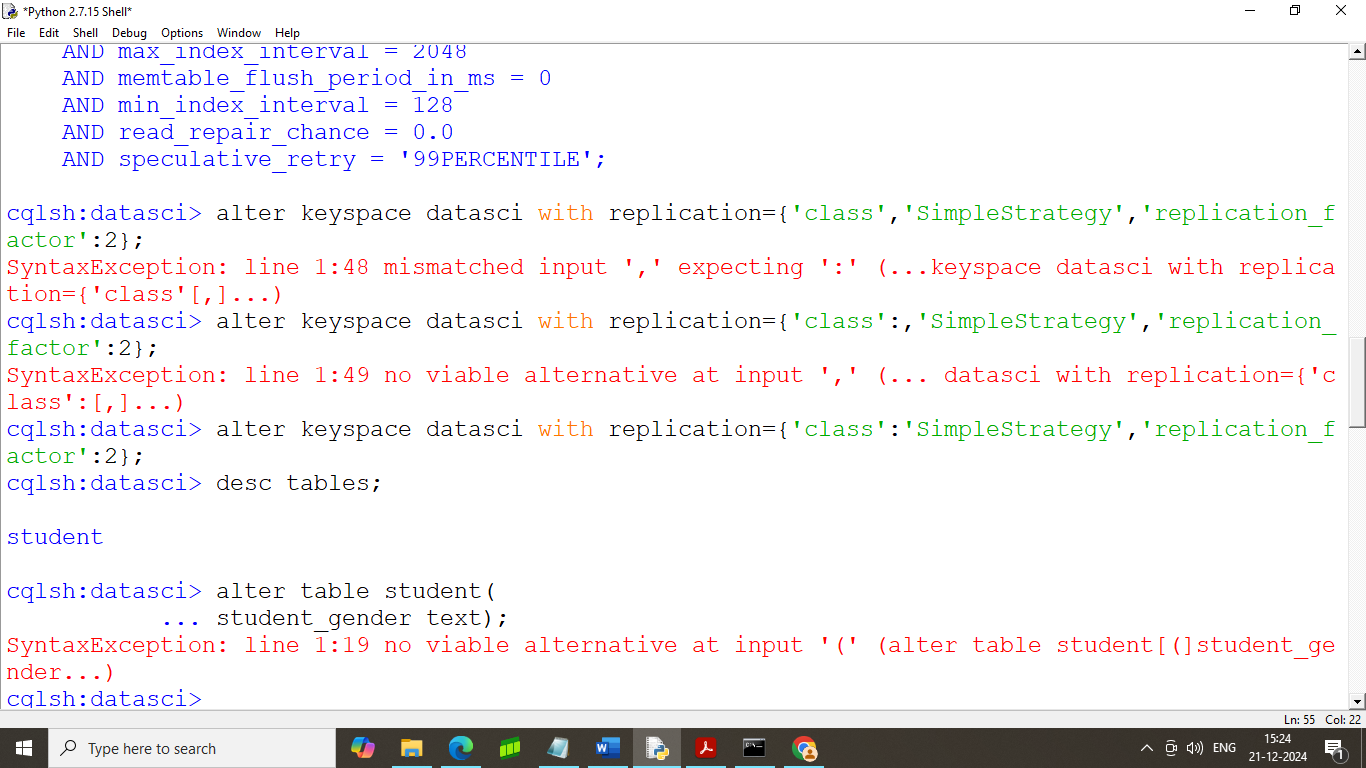
**Step – 6: command to alter keyspace**

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)

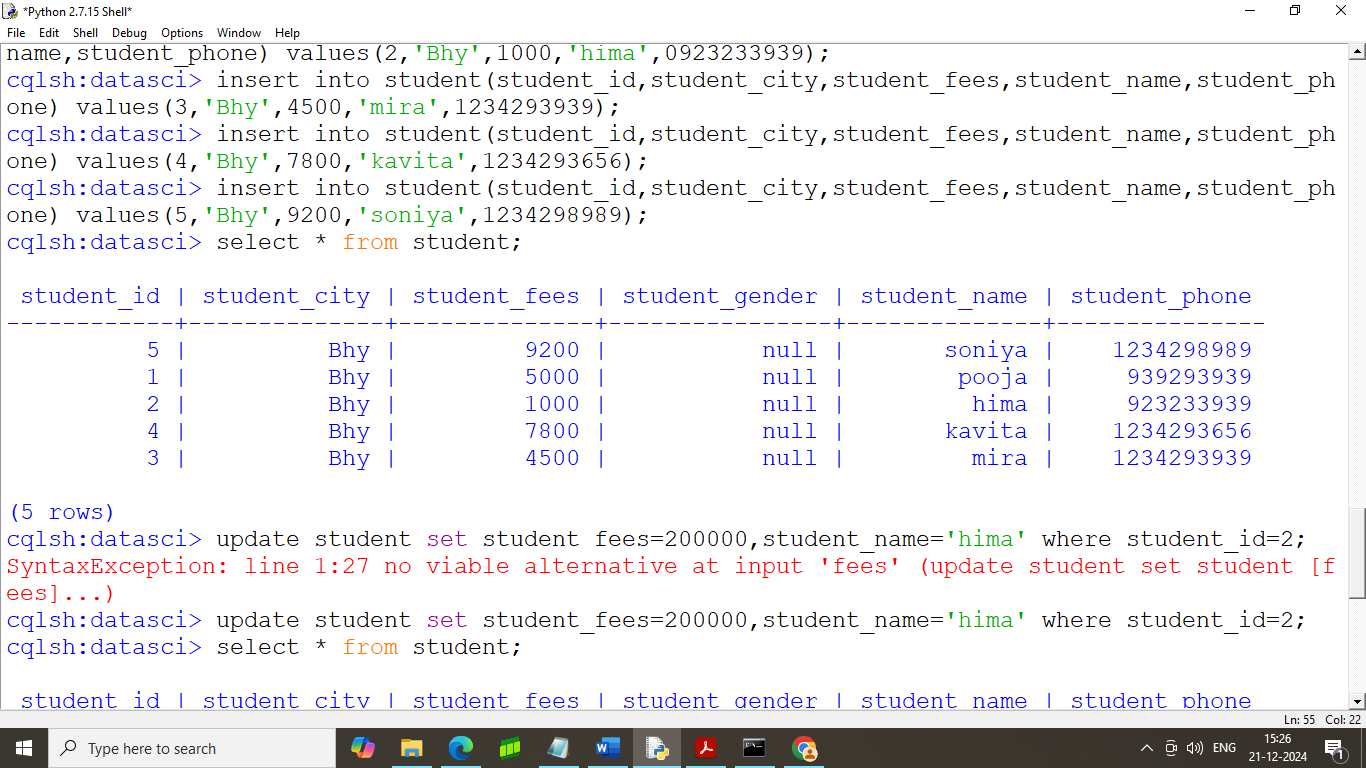
values(1,'Bhy',5000,'pooja',0939293939);

(you can only add one value at a time)



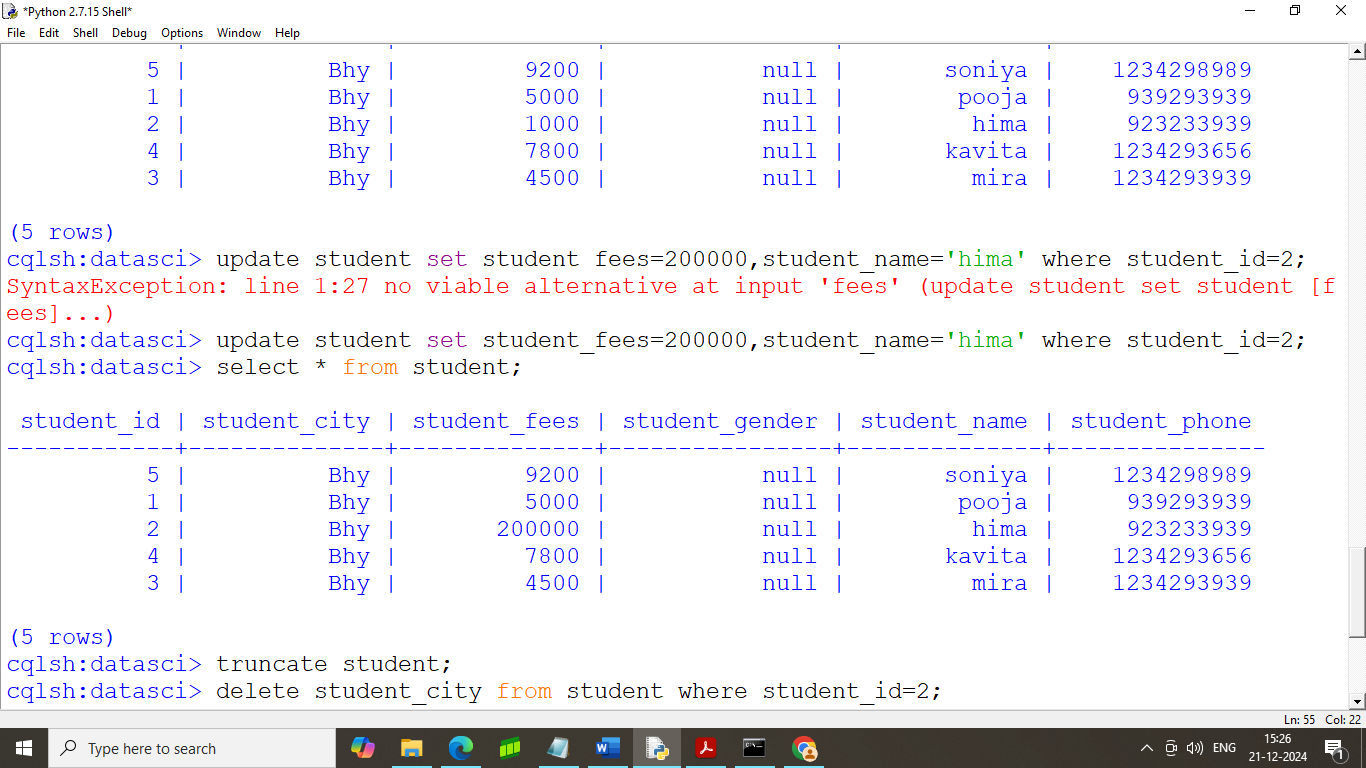
**Step – 10: command to show the table**

cqlsh:datasci> select \* from student;



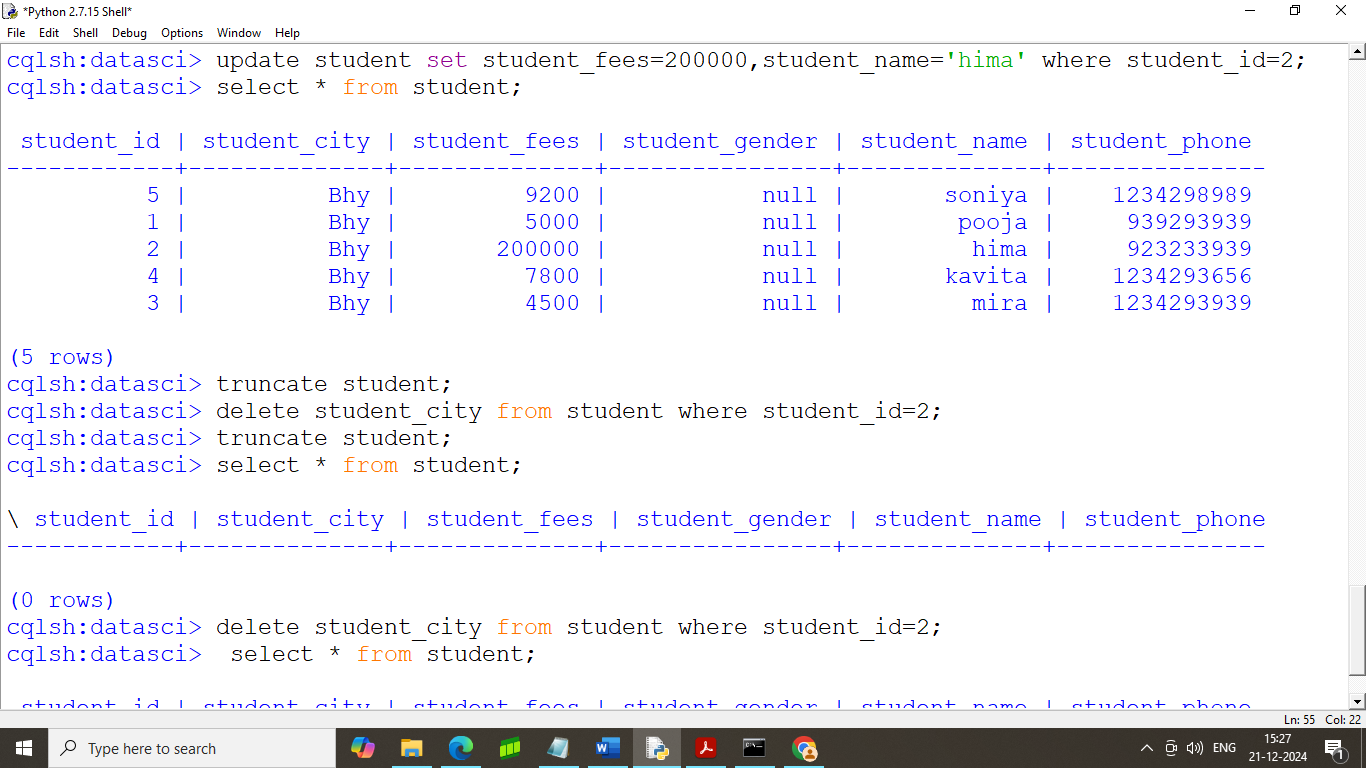
**step – 11: command to update table**

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



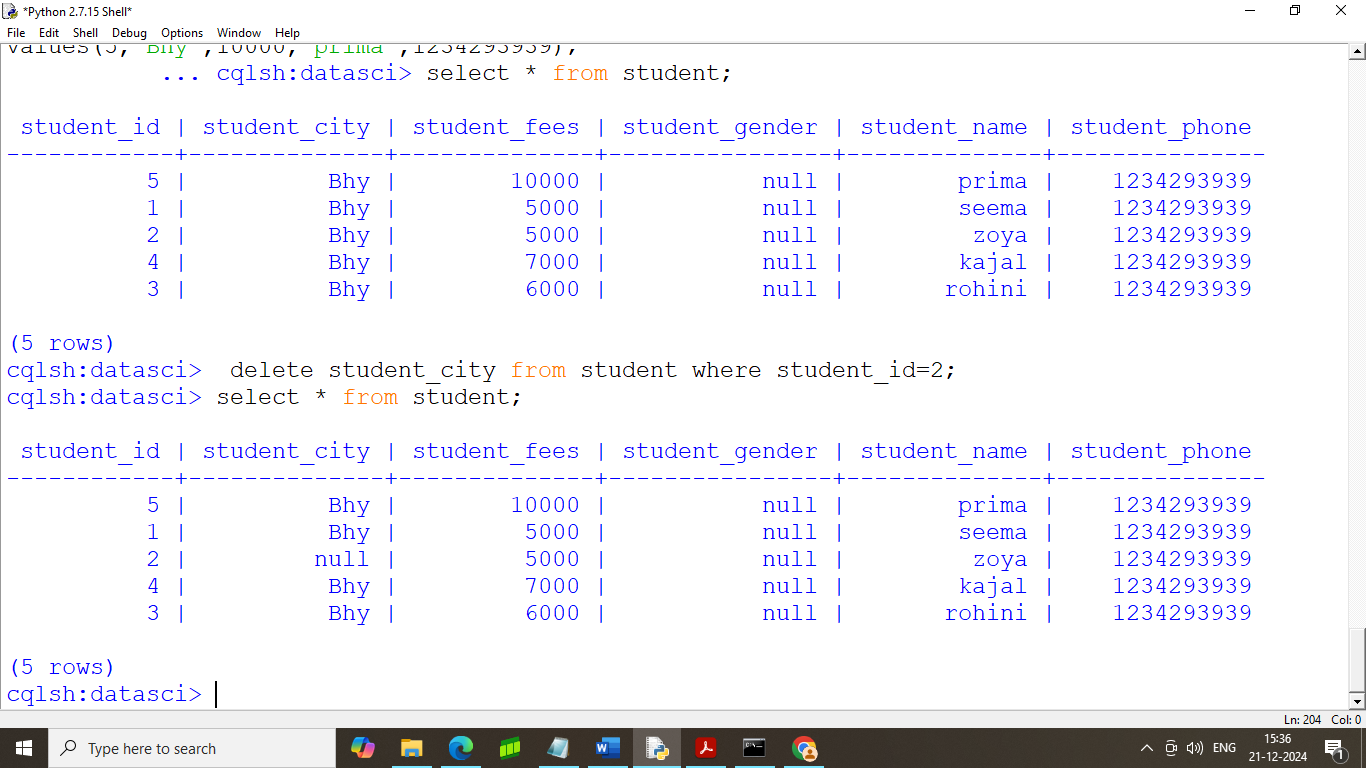
**step – 12: command to refresh the table**

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

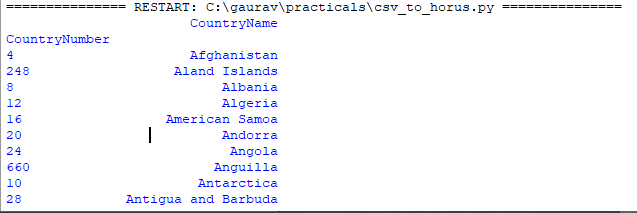
ProcessData.drop(['ISO-3-Code', 'ISO-2-CODE'], axis=1,inplace=True)

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))

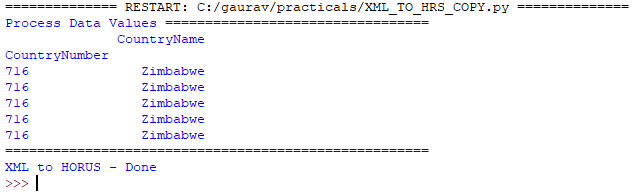
print('=====================================================')

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)

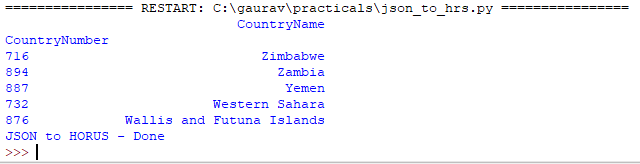
print(ProcessData.head(5))

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 =================================')

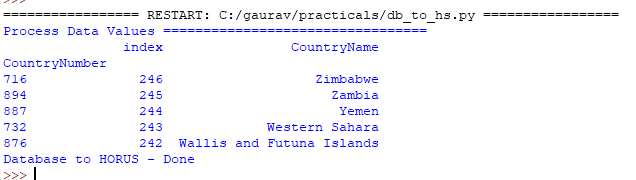
print(ProcessData.head(5))

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

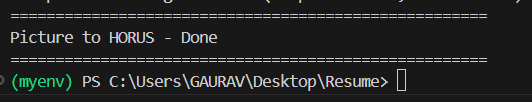
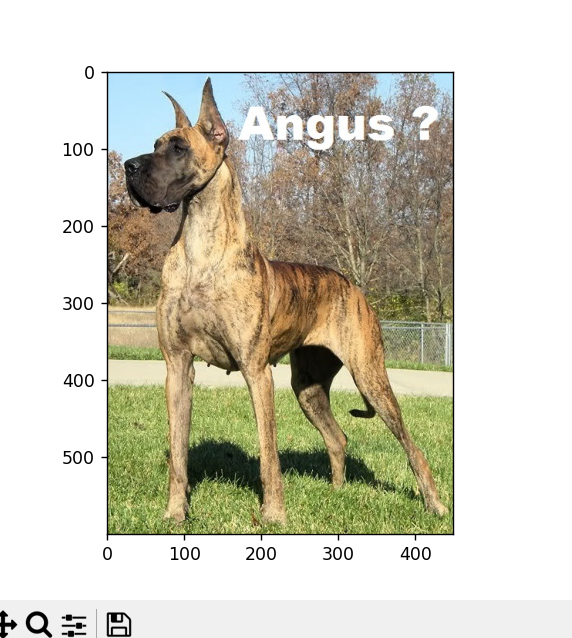
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'

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:

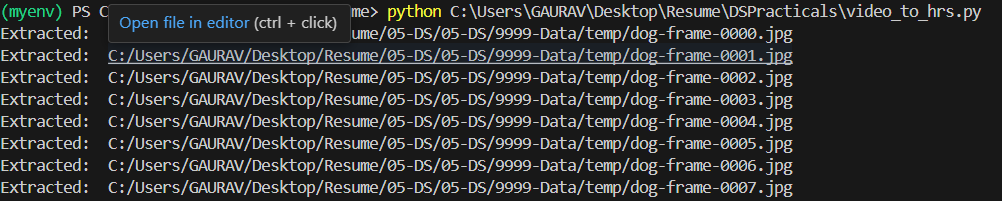
        break

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

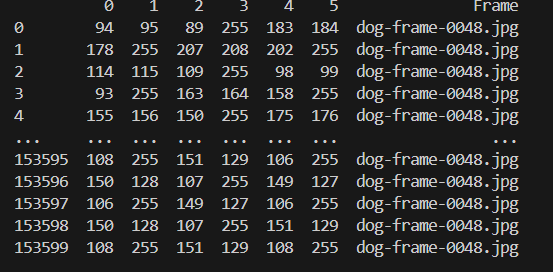
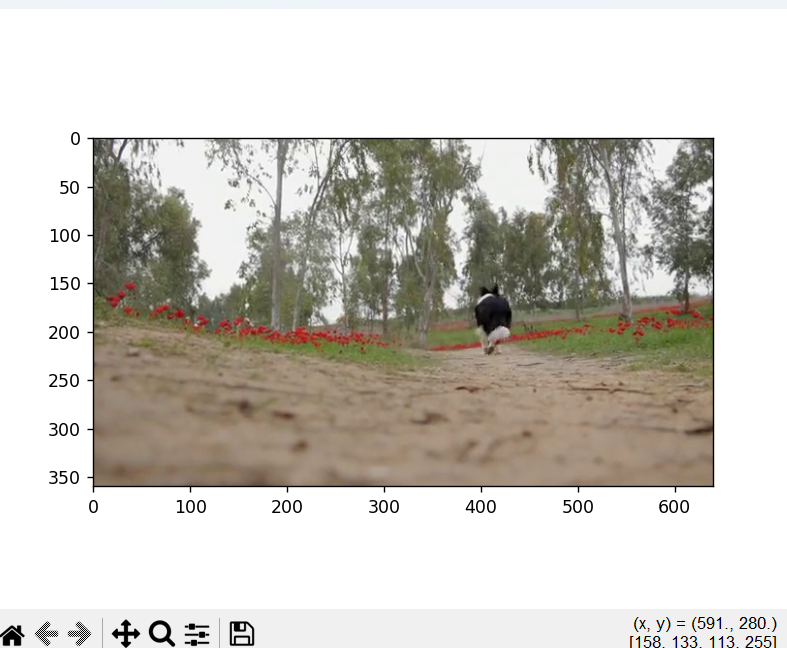
    ProcessFrameData.index.names =['ID']

    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

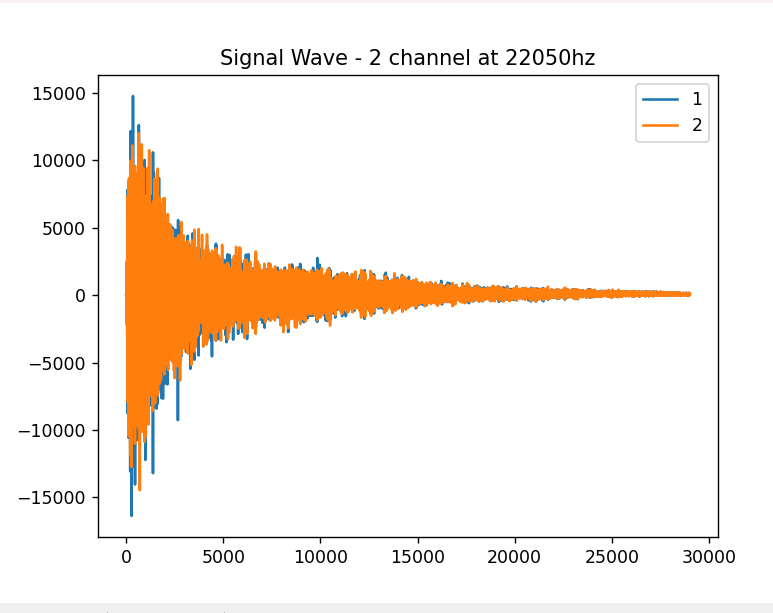
OutputData=ProcessData

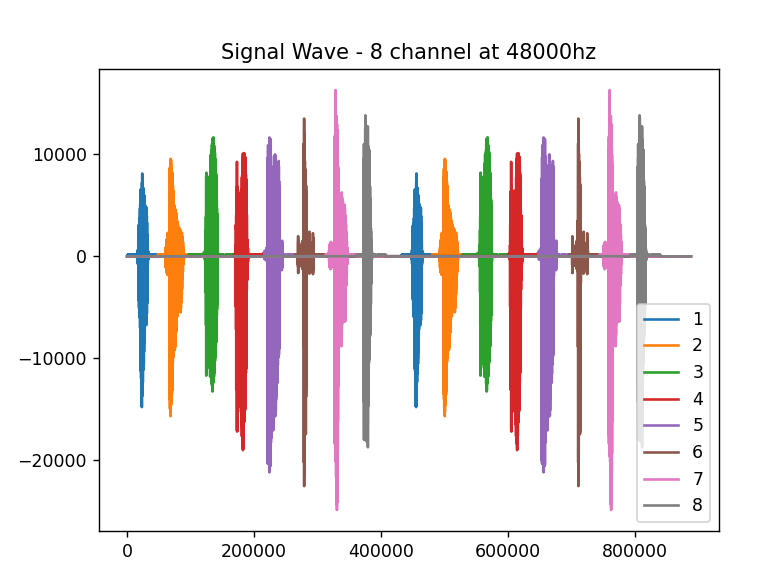
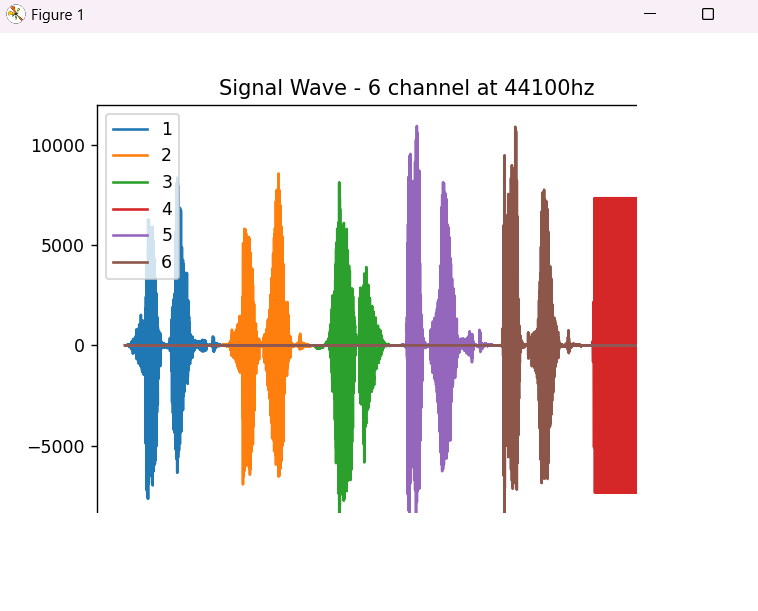
sOutputFileName='C:/Users/GAURAV/Desktop/Resume/05-DS/05-DS/9999-Data/HORUS-Audio-8ch.csv'

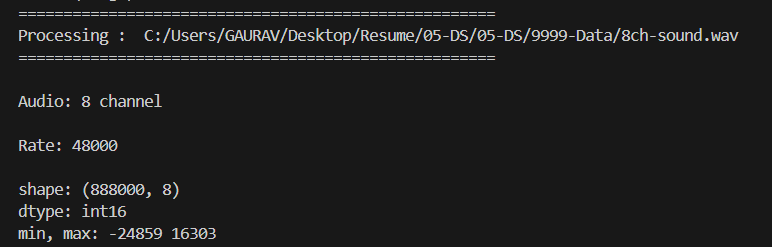
OutputData.to\_csv(sOutputFileName, index = False)

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

print('Audio to HORUS - Done')







**Practical 3: Utilities and Auditing**

Basic Utility Design

1. **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.')

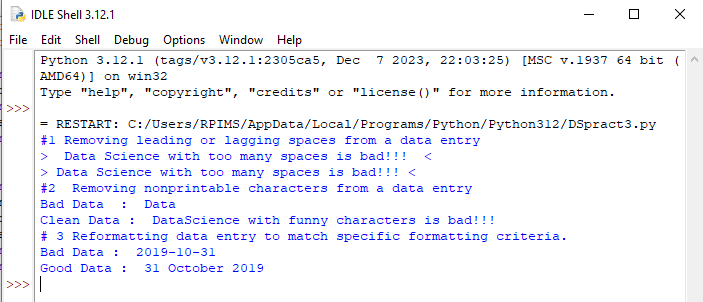
baddate = dt.date(2019, 10, 31)

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)



B. 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)

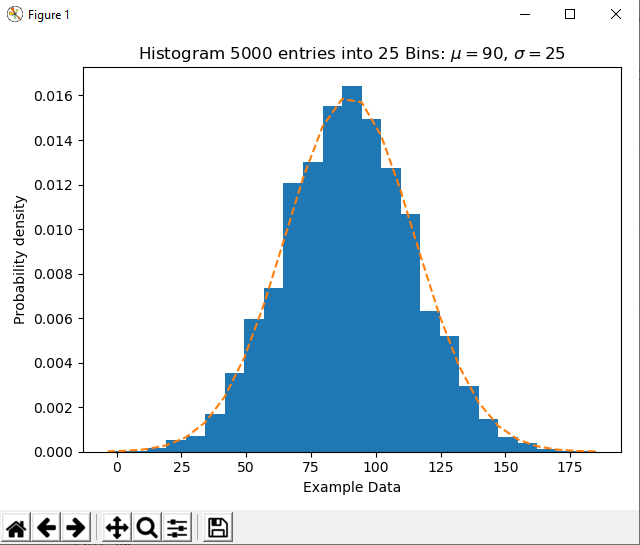
fig.tight\_layout()

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

fig.savefig(sPathFig)

plt.show()

Output:



C. 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)

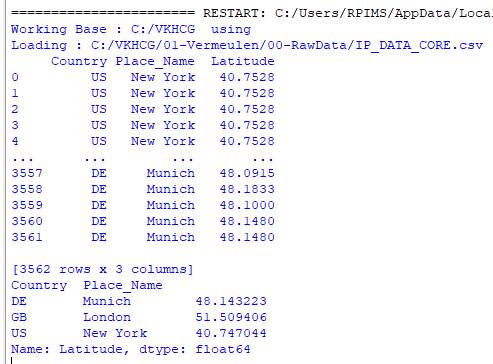
AllData=IP\_DATA\_ALL[['Country', 'Place\_Name','Latitude']]

print(AllData)

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

print(MeanData)

**Output:**



D. 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]

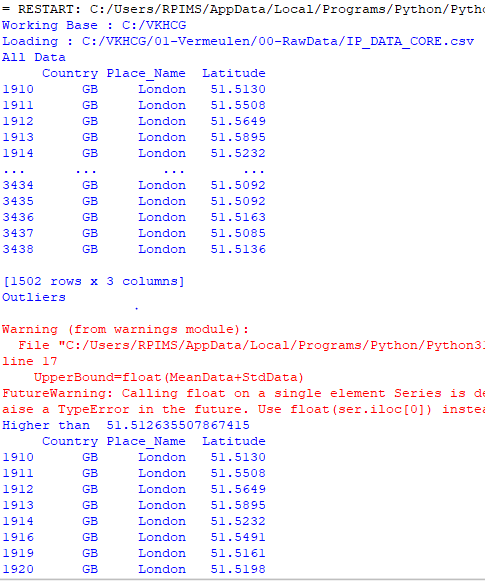
print(OutliersLower)

print('Not Outliers')

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

print(OutliersNot)

**Output:**

# **Audit**

**E. 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.')

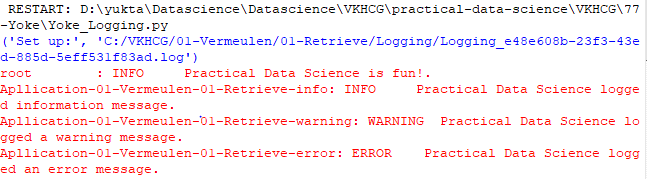
if sLevel == 'warning':

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

if sLevel == 'error':

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

**Output:**



# **Practical 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))

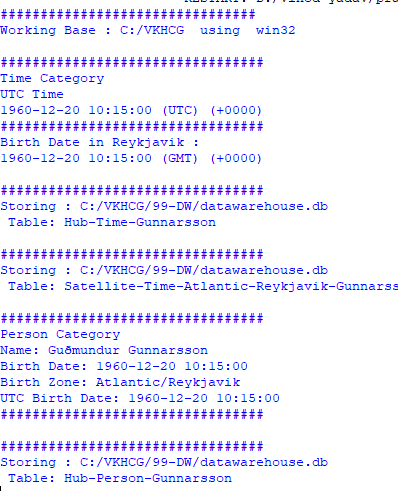
TimeHub=PersonFrame TimeHubIndex=TimeHub.set\_index(['IDNumber'],inplace=False)

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 7 - Transform-Gunnarsson-Sun-Model.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'

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('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\_dict(dict(TimeLine))

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

sTable = 'Dim-Time'

print('Storing :',sDatabaseName,'\n Table:',sTable) DimTimeIndex.to\_sql(sTable, conn1, if\_exists="replace") DimTimeIndex.to\_sql(sTable, conn2, if\_exists="replace")

print('Dimension Person') FirstName = 'Guðmundur' LastName = 'Gunnarsson'

IDPersonNumber=str(uuid.uuid4()) PersonLine=[('PersonID', [IDPersonNumber]),

('FirstName', [FirstName]),

('LastName', [LastName]),

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

('DateTimeValue', [BirthDateZoneStr])] PersonFrame = pd.DataFrame.from\_dict(dict(PersonLine))

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

sTable = 'Dim-Person'

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

DimPersonIndex.to\_sql(sTable, conn1, if\_exists="replace") DimPersonIndex.to\_sql(sTable, conn2, if\_exists="replace")

print('Fact - Person - time')

IDFactNumber=str(uuid.uuid4()) PersonTimeLine=[('IDNumber', [IDFactNumber]),

('IDPersonNumber', [IDPersonNumber]), ('IDTimeNumber', [IDTimeNumber])]

PersonTimeFrame = pd.DataFrame.from\_dict(dict(PersonTimeLine))

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

sTable = 'Fact-Person-Time'

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

FctPersonTimeIndex.to\_sql(sTable, conn1, if\_exists="replace") FctPersonTimeIndex.to\_sql(sTable, conn2, 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")

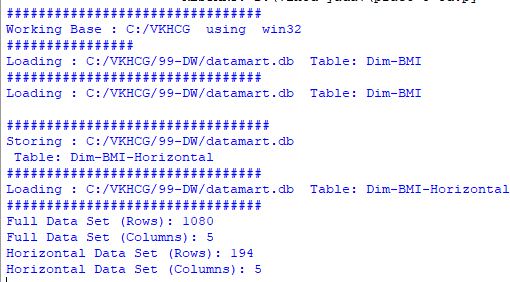
sTable = 'Dim-BMI-Horizontal'

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")

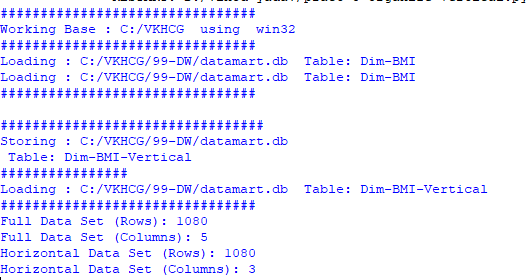
sTable = 'Dim-BMI-Vertical'

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,\ Weight;"

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

DimPerson=PersonFrame1 DimPersonIndex=DimPerson.set\_index(['Indicator'],inplace=False)

sTable = 'Dim-BMI-Secure'

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

sTable = 'Dim-BMI-Secure'

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

sSQL="SELECT \* FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"

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

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

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

print(PersonFrame2.head())

**OUTPUT**

**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:**



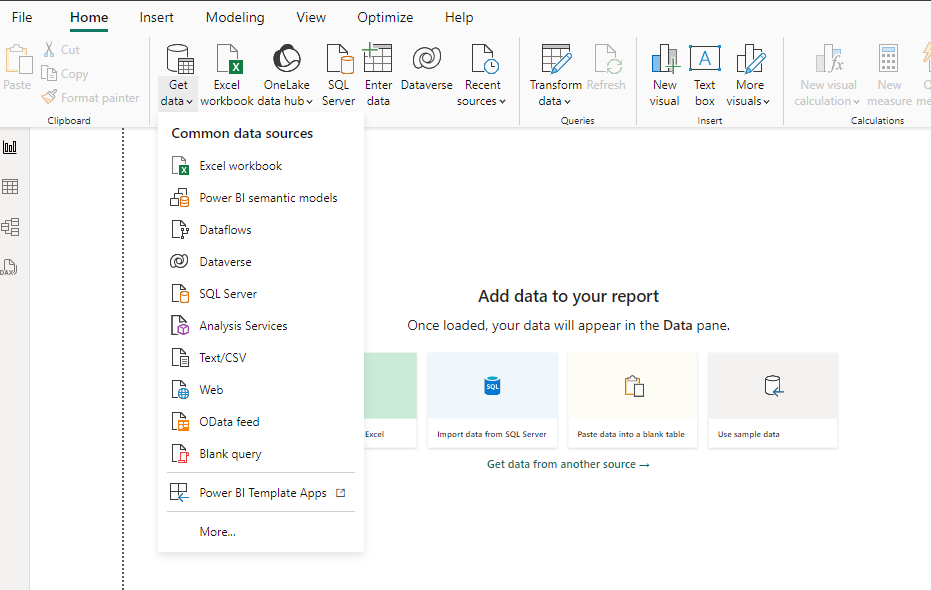
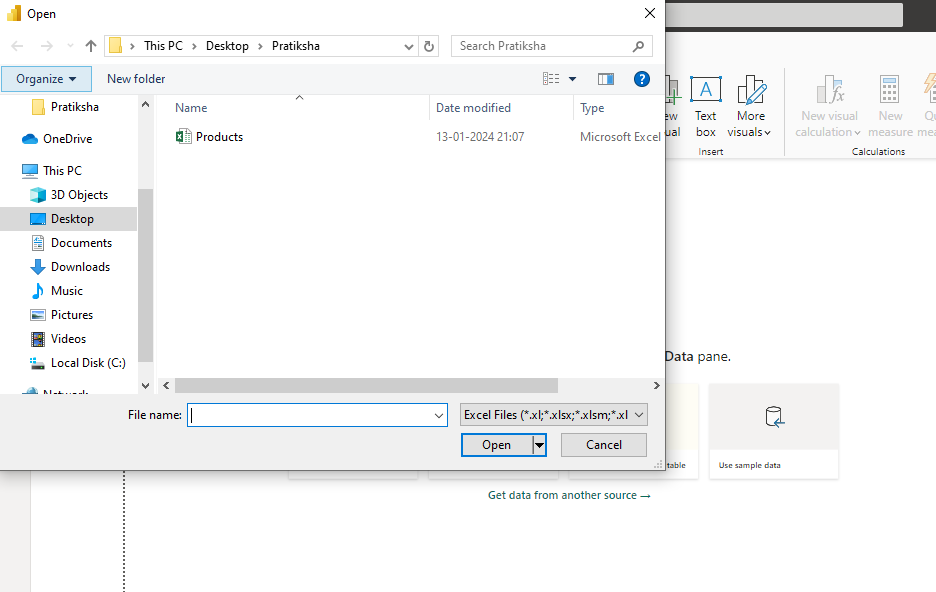
**Practical No. 10**

Data Visualization with power bi

Case Study : Sales Data

Step 1: connect to an Excel Workbook

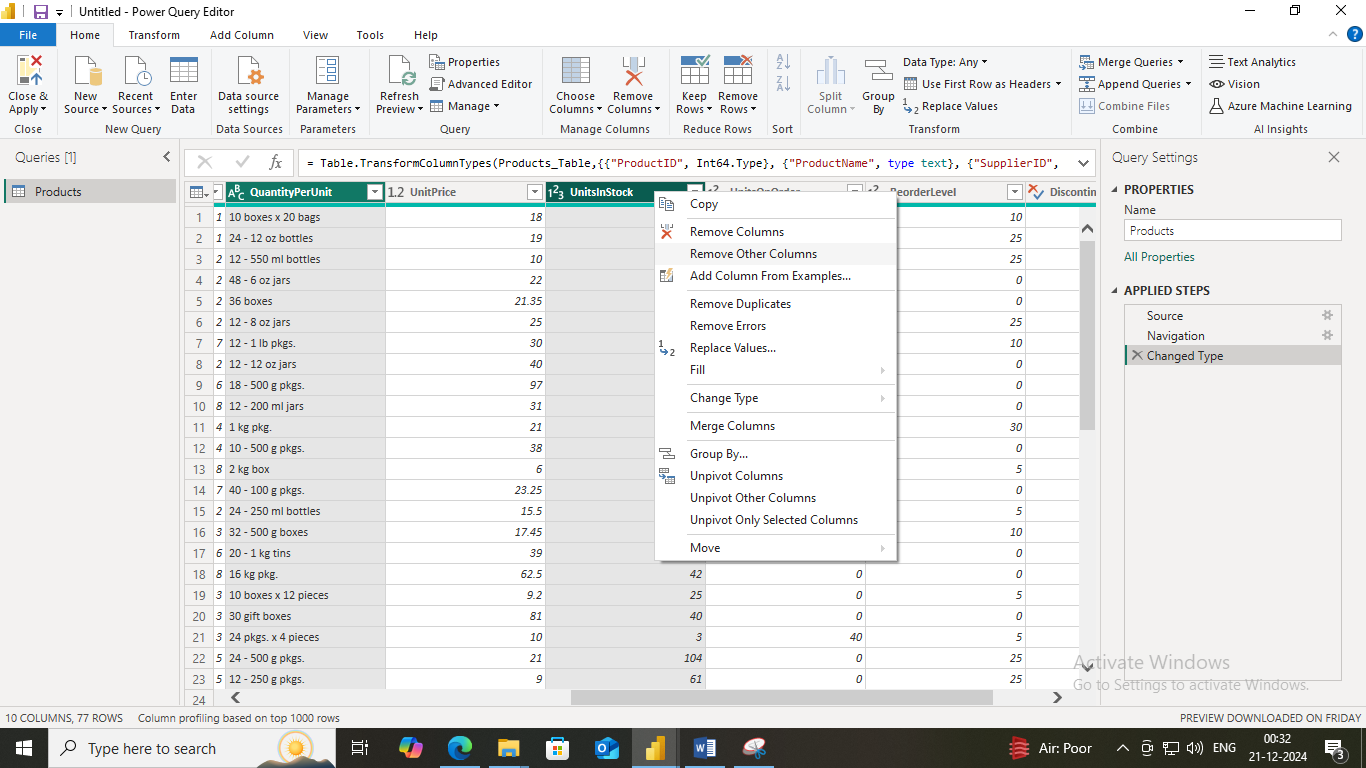
1. Launch power Bi Desktop.
2. From the Home Ribbon, Select Get Data 🡪 Select Excel Workbook .
3. In the Open File Dialog Box, Select the Product.xlsx file.

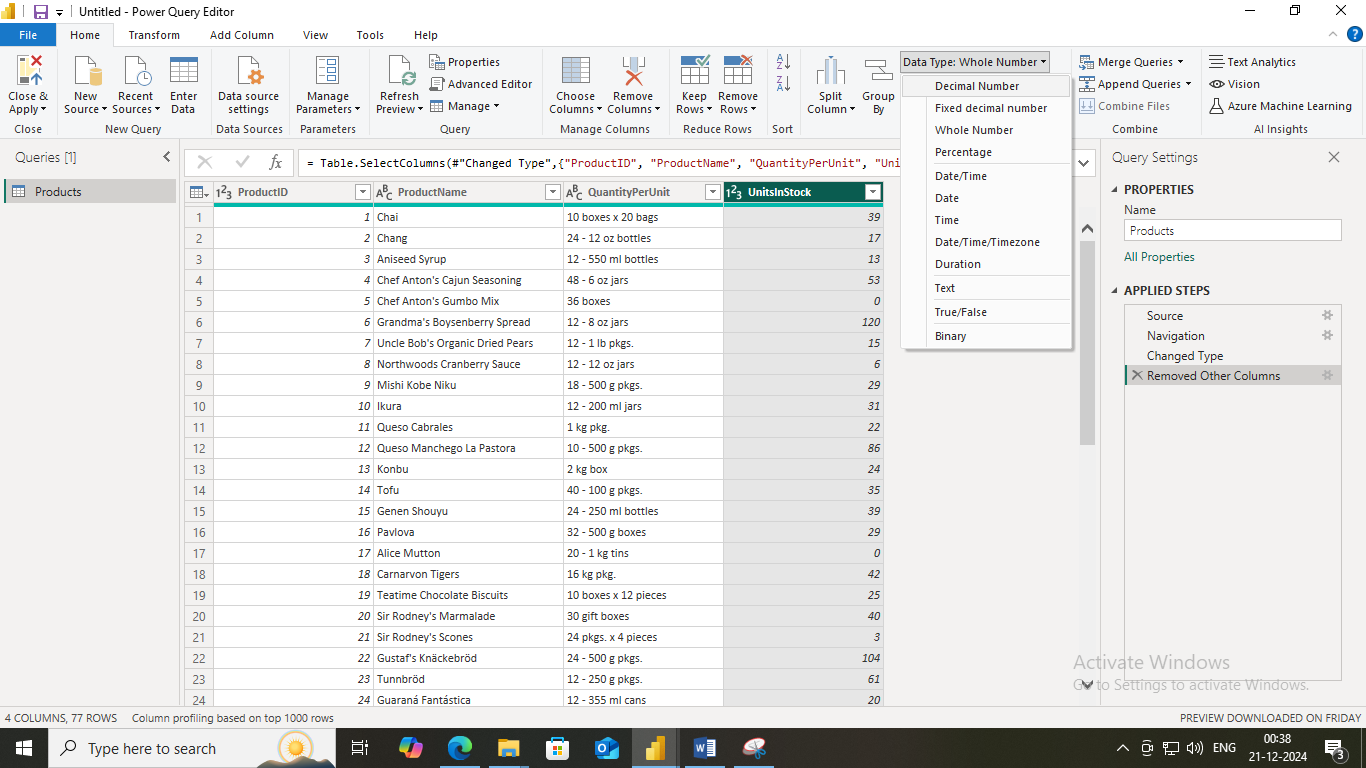
1. In the Open File Dialog Box, Select the Product.xlsx file.



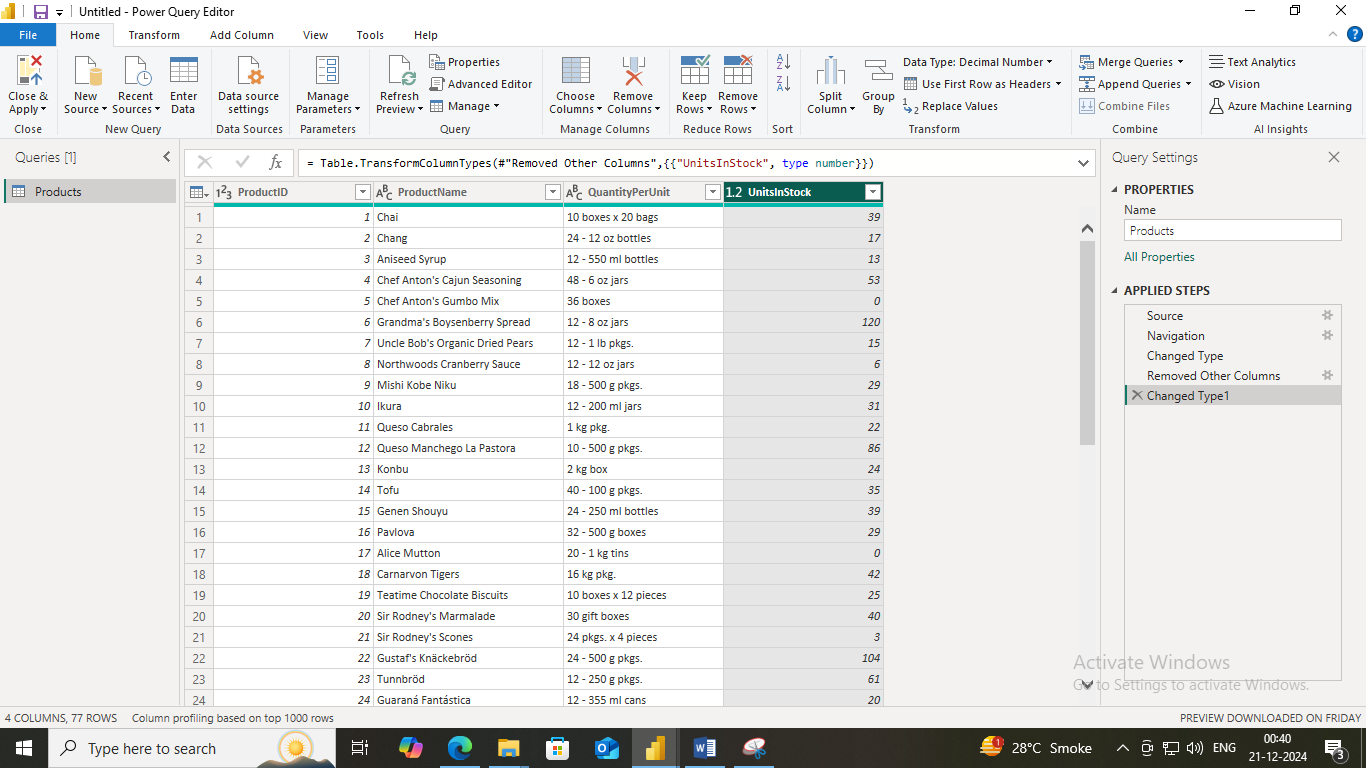
1. Click on Products Check Box. & We Will see the product Table. Select The Transform Data.
2. In Query Editor , Select the ProductID, ProductName, QuantityPerUnit, and unitsInStock Columns. ( Use Ctrl + Click to select more than one column ).



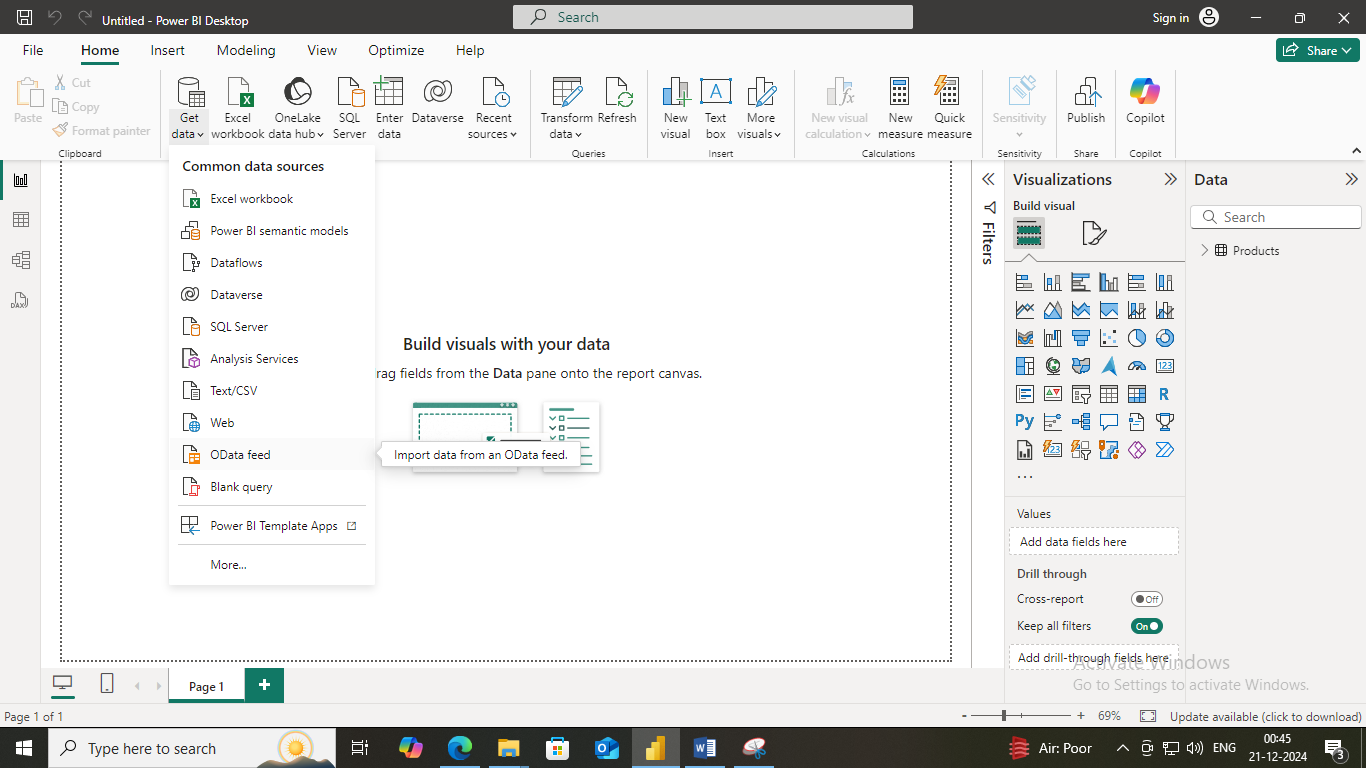
1. Right Click on Column Header and Click Remove Other Columns.

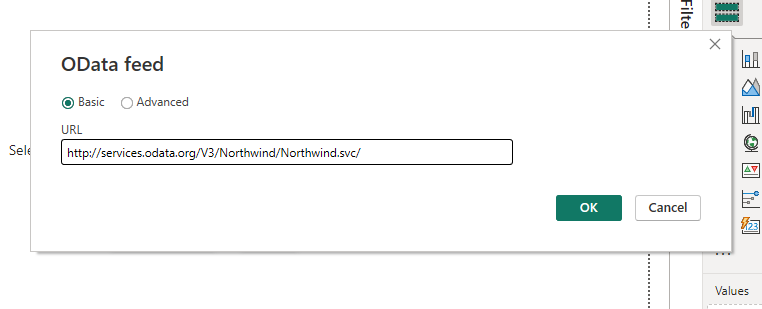


1. Select Close & Apply from Home Ribbon.

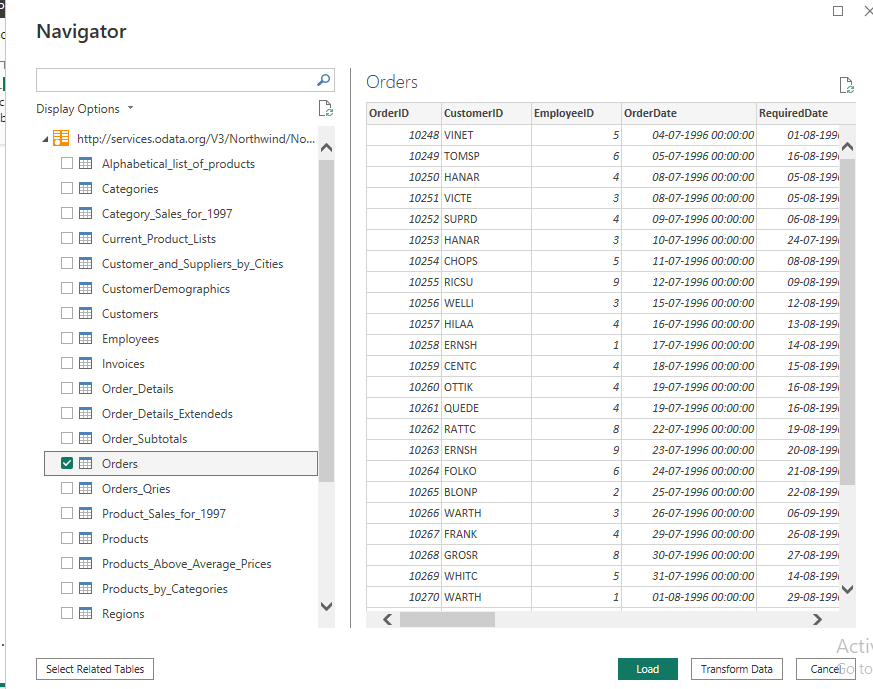


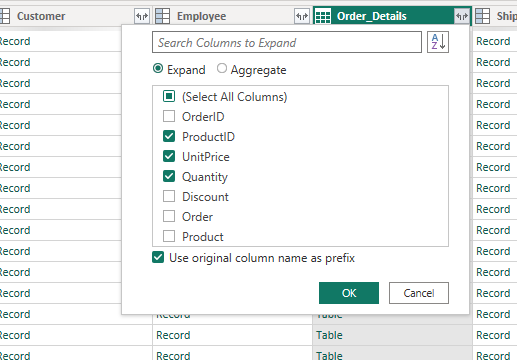
1. Another Window on 🡪 Select Get Data & Select the OData feed. And Copy the link given below. & Paste it to OData feed URL Box and Click Ok.

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

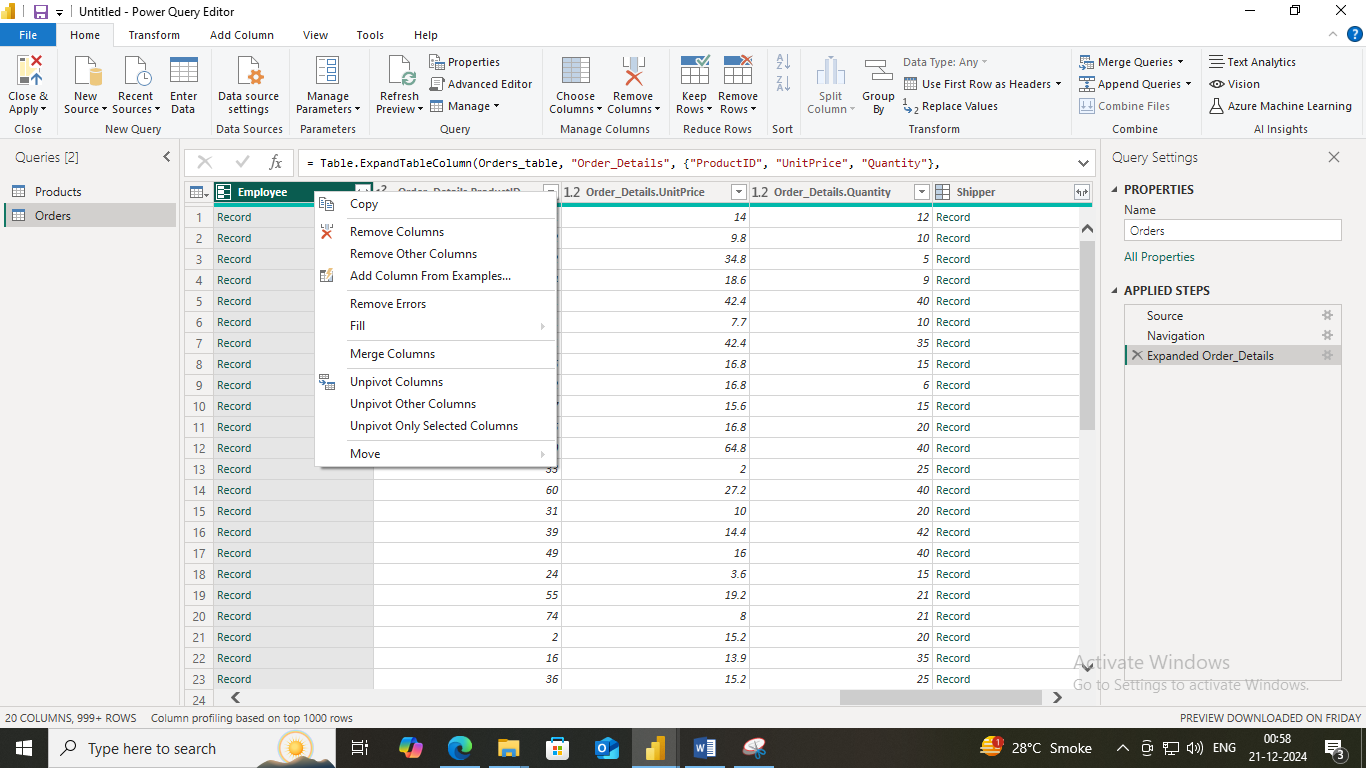


1. We will see the below screen like this then select from navigator orders checkbox. & Click on Transform Data.

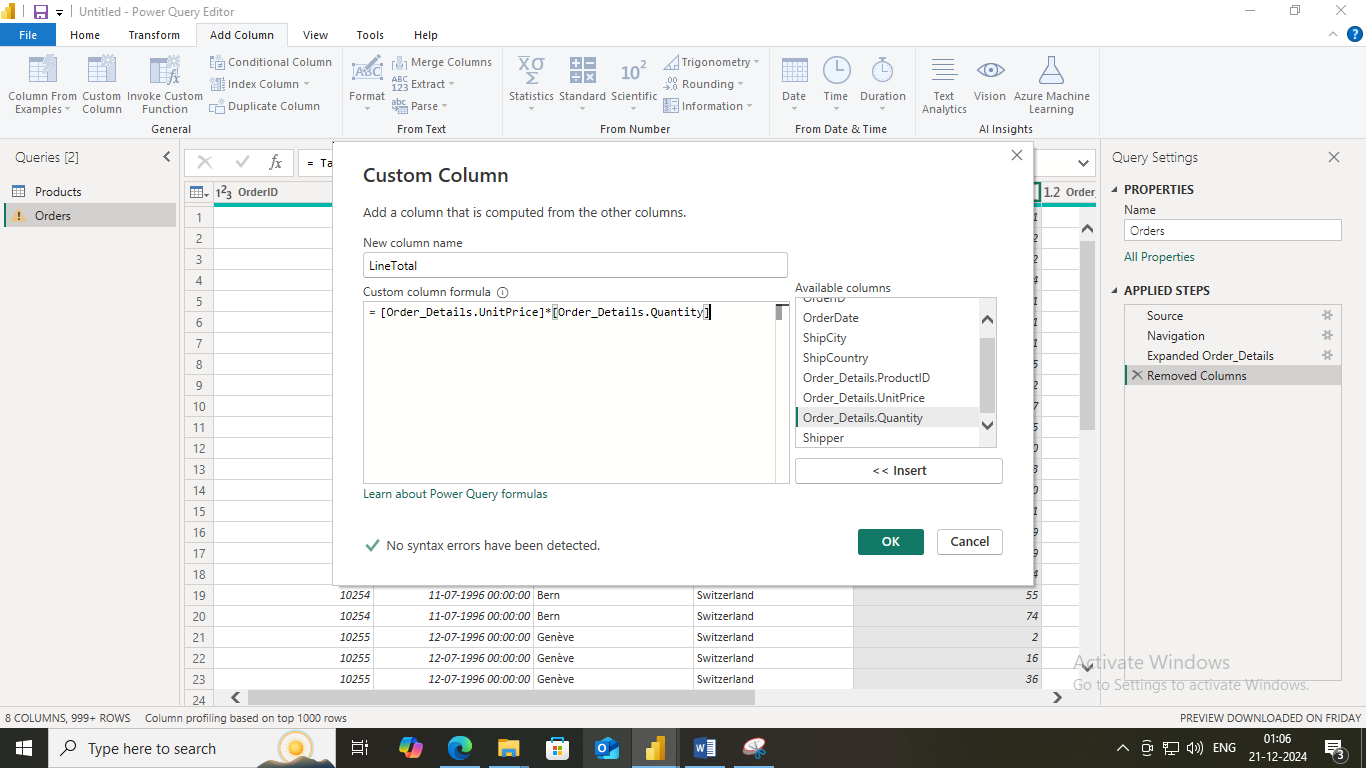


1. Expand the Order\_Details column & select the ProductID, UnitPrice, Quantity & Click OK.

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.

