**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Practical No.** | | **Description** | **Page**  **No.** | **Signature** |
| **1.** |  | **Conversion from different file formats to HORUS format.** |  |  |
|  | A | Conversion from Text-Delimited to HORUS format. |  |  |
|  | B | Conversion from XML to HORUS format. |  |  |
|  | C | Conversion from JSON to HORUS format. |  |  |
|  | D | Conversion from MYSQL database to HORUS format. |  |  |
|  | E | Conversion from picture (jpeg) to HORUS format. |  |  |
|  | F | Conversion from video to HORUS format. |  |  |
|  | G | Conversion from audio to HORUS format. |  |  |
| **2.** |  | **Utilities and Auditing** |  |  |
|  | A | Fixer Utilities  1. Removing leading and lagging spaces from data entry  2. Removing non-printable characters from data entry  3. Reformatting data entry to match specific formatting criteria |  |  |
|  | B | Data Binning and Bucketing |  |  |
|  | C | Averaging of data |  |  |
|  | D | Outlier Detection |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E | Logging: Write python/R program for basic logging in Data Science. |  |  |
| **3** |  | **Retrieving Data** |  |  |
|  | A | Perform data processing using R |  |  |
|  | B | Program to retrieve different attributes of data |  |  |
|  | C | Data Pattern  Example 1  Example 2 |  |  |
|  | D | Loading IP\_DATA\_ALL  Haversine distance calculation using Vermeulen dataset |  |  |
|  | E | Building a diagram for scheduling of jobs |  |  |
|  | F | Picking content for billboards |  |  |
| **4** |  | **Assessing Data** |  |  |
|  | A | Perform error management on the given data  1. Drop the columns where all elements are missing values  2. Drop the columns where any of the elements is missing values 3. Keep only the rows that contain a maximum of two non-missing values  4. Fill All Missing Values with the Mean, Median, Mode, Minimum, and Maximum of the Particular Numeric Column |  |  |
|  | B | Write python/R program to create the network routing diagram from the given data on routers in assess superstep. |  |  |
|  | C | Write python/R program to build directed acyclic graph.  Example 1: Company location DAG  Example 2: Customer location DAG  Example 3: DAG using GPS data |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | D | Write python/R program to pick the content for Billboards for the given data. |  |  |
| **5** |  | **Processing Data** |  |  |
|  | A | Build the time hub, links and satellites. |  |  |
|  | B | Golden Nominal |  |  |
|  | C | Vehicles as object hub and satellite using python and SQLite. |  |  |
|  | D | Human-Environment interaction |  |  |
| **6** |  | **Transforming Data** |  |  |
|  | A | Program to show the details of hub: person being born. |  |  |
|  | B | Building dimension Person, Time and fact PersonBornAtTime |  |  |
|  | C | Building a Data warehouse for transform superstep. |  |  |
|  | D | Write python program to demonstrate Simple Linear Regression. |  |  |
| **7** |  | **Organizing Data** |  |  |
|  | A | Write python/R program to perform the horizontal style subset or slice of the data warehouse data. |  |  |
|  | B | Write python/R program to perform the vertical style subset or slice of the data warehouse data. |  |  |
|  | C | Write python/R program to perform the island style subset or slice of the data warehouse data. |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | D | Write python/R program to perform the secure vault style subset or slice of the data warehouse data and attach the result to the person who performs the query. |  |  |
|  | E | Write python program to demonstrate Association rule mining. |  |  |
|  | F | Write python program to create network routing diagram in organizing superstep. |  |  |
| **8** |  | **Generating Data** |  |  |
|  | A | Write python/R program to perform data visualization to create following graphs using profit data.  1. Pie Graph  2. Double Pie Graph  3. Line Graph  4. Vertical Bar Graph  5. Horizontal Bar Graph  6. Area Graph  7. Scatter Graph  8. Hex Bin Graph |  |  |
|  | B | Write python/R program to perform data visualization to create following advanced graphs/plots for the data.  1. Kernel Density Estimation (KDE) Graph  2. Scatter Matrix  3. Andrew’s Curves  4. Parallel Coordinates  5. RADVIZ method  6. Lag Plot  7. Autocorrelation Plot  8. Bootstrap Plot  9. Contour Graphs  10. 3D Graph |  |  |
| **9** |  | **Data Visualization with Power BI.** |  |  |

Practical 1

The Homogeneous Ontology for Recursive Uniform Schema (HORUS) is used as an internal data format structure that enables the framework to reduce the permutations of transformations required by the framework. The use of HORUS methodology results in a hub-and-spoke data transformation approach. External data formats are converted to HORUS format, and then a HORUS format is transformed into any other external format. The basic concept is to take native raw data and then transform it first to a single format. That means that there is only one format for text files, one format for JSON or XML, one format for images and video. Therefore, to achieve any-to-any transformation coverage, the framework’s only requirements are a data- format-to-HORUS and HURUS-to- data-format converter.

Aim: -Write Python Program to convert from the following formats to HORUS format.

A]. Text delimited CSV to HORUS format.

Code: -

import pandas as pd

# Input Agreement

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

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

print('Input Data Values')

print(InputData)



# Processing Rules

ProcessData=InputData

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

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

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

# Sort data by CurrencyNumber

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

print('Process Data Values ')

print(ProcessData)

OutputData=ProcessData

sOutputFileName='B:/1MscIT material/Data science/raw/HORUS-CSV-Country1.csv'

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

print('CSV to HORUS - Done')

print("Bhavik")

A screenshot of a computer

Description automatically generated

B]. XML to HORUS Format

Code: -

# Utility Start XML to HORUS

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)

# Input Agreement

sInputFileName='B:/1MscIT material/Data science/raw/Country\_Code.xml'

InputData = open(sInputFileName).read()

print('Input Data Values')

print(InputData)



# Processing Rules

ProcessDataXL=InputData

# XML to Data Frame

ProcessData=xml2df(ProcessDataXML)

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

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

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

# Sort data by CurrencyNumber

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

print('Process Data Values')

print(ProcessData)

A computer screen shot of a black screen

Description automatically generated

# Output Agreement

OutputData=ProcessData

sOutputFileName='B:/1MscIT material/Data science/raw/HORUS-XML-Country.csv'

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

print('XML to HORUS - Done')

print("BHavik")

C]. JSON to HORUS Format

Code: -

# Utility Start JSON to HORUS

import pandas as pd

# Input Agreement

sInputFileName='B:/1MscIT material/Data science/raw/Country\_Code.json'

InputData=pd.read\_json(sInputFileName,

orient='index',

encoding="latin-1")

print('Input Data Values')

print(InputData)



# Processing Rules

ProcessData=InputData

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

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

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

# Sort data by CurrencyNumber

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

print('Process Data Values')

print(ProcessData)

A screenshot of a computer

Description automatically generated

# Output Agreement

OutputData=ProcessData

sOutputFileName='B:/1MscIT material/Data science/raw/HORUS-JSON-Country.csv'

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

print('JSON to HORUS - Done')

print("Bhavik")

D]. MySql Database to HORUS Format

Code: -

# Utility Start Database to HORUS

import pandas as pd

import sqlite3 as sq

# Input Agreement

sInputFileName='B:/1MscIT material/Data science/raw/utility.db'

sInputTable='Country\_Code'

conn = sq.connect(sInputFileName)

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

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

print('Input Data Values')

print(InputData)

A screenshot of a computer

Description automatically generated

# Processing Rules

ProcessData=InputData

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

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

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

# Rename Country and ISO-M49

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

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

# Set new Index

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

# Sort data by CurrencyNumber

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

print('Process Data Values ')

print(ProcessData)

A screenshot of a computer

Description automatically generated

# Output Agreement

OutputData=ProcessData

sOutputFileName='B:/1MscIT material/Data science/raw/DATABSE-HORUS-Country.csv'

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

print('Database to HORUS - Done')

print("Bhavik")

# Utility done

E]. Picture (JPEG) to HORUS Format

Code: -

from matplotlib.pyplot import imread

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

# Input Agreement

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

InputData = imread(sInputFileName)

A dog standing in the grass

Description automatically generated

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

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

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

# Processing Rules

ProcessRawData=InputData.flatten()

y=InputData.shape[2] + 3

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

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

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

ProcessData.columns=sColumns

ProcessData.index.names =['ID']

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

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

A screenshot of a computer

Description automatically generated

print('Process Data Values ')

plt.imshow(InputData)

plt.show()

# Output Agreement

OutputData=ProcessData

print('Storing File')

A screenshot of a computer

Description automatically generated

sOutputFileName='B:/1MscIT material/Data science/raw/HORUS-Picture.csv'

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

print('Picture to HORUS - Done')

print("Bhavik")

Practical 2: Utilities and Auditing

Basic Utility Design

Load data as per input agreement.

Apply processing rules of utility.

Save data as per output agreement.

There are three types of utilities

Data processing utilities

Maintenance utilities

Processing utilities

A]. Fixers Utilities

import string

#1 Removing leading or lagging spaces from a data entry

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

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

Cleandata = baddata.strip()

print('>', Cleandata,'<')



#2 Removing nonprintable characters from a data entry

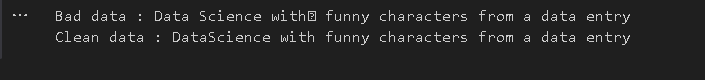
printable = set(string.printable)

baddata = "Data\x00Science with\x02 funny characters from a data entry"

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

import datetime as dt

baddate = dt.date(2019,10,21)

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

print("The date is: ", baddata)

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

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

print("Reformatted Date:", gooddata)

A black background with white numbers

Description automatically generated

B]. Data Binning or Bucketing

Binning is a data preprocessing technique used to reduce the effects of minor observation errors. Statistical data binning is a way to group a number of more or less continuous values into a smaller number of “bins.”

Code: -

import numpy as np

import matplotlib.mlab as mlab

import matplotlib.pyplot as plt

import scipy.stats as stats

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, density=1)

# add a 'best fit' line

y = stats.norm.pdf(bins, mu, sigma)

#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='B:/1MscIT material/Data science/practical\_neelam/DU-Histogram.png'

fig.savefig(sPathFig)

plt.show()

print("Bhavik")

A screenshot of a graph

Description automatically generated

C]. Averaging of Data

The use of averaging of features value enables the reduction of data volumes in a control fashion to improve effective data processing.

Code: -

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)

print("Bhavik")

A screenshot of a computer

Description automatically generated

D]. Outlier Detection

Outliers are data that is so different from the rest of the data in the data set that it may be caused by an error in the data source. There is a technique called outlier detection that, with good data science, will identify these outliers

Code: -

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

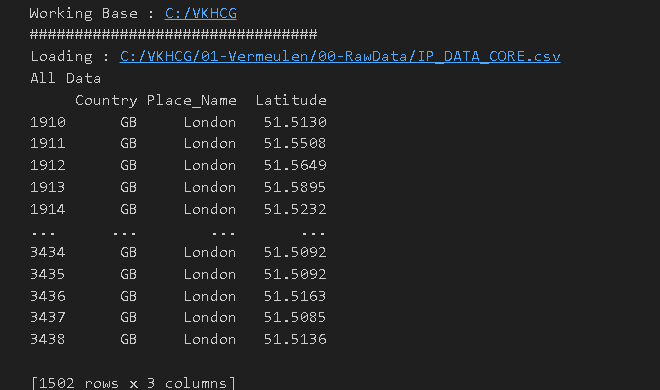
IP\_DATA\_ALL.rename(columns={'Place Name': 'Place\_Name'}, inplace=True)

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)

A screenshot of a computer

Description automatically generated

LowerBound=float(MeanData-StdData)

print('Lower than ', LowerBound)

OutliersLower=AllData[AllData.Latitude<LowerBound]

print(OutliersLower)

A black screen with white text

Description automatically generated

print('Not Outliers')

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

print(OutliersNot)

print("Bhavik")

A screenshot of a computer screen

Description automatically generated

E]. Logging

Code: -

import sys

import os

import logging

import uuid

import shutil

import time

Base='C:/VKHCG'

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

sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','06-Report']

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

for sCompany in sCompanies:

sFileDir=Base + '/' + sCompany

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

for sLayer in sLayers:

log = logging.getLogger() # root logger

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

log.removeHandler(hdlr)

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

if os.path.exists(sFileDir):

shutil.rmtree(sFileDir)

time.sleep(2)

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

skey=str(uuid.uuid4())

sLogFile=Base + '/' + sCompany + '/' + sLayer + '/Logging/Logging\_'+skey+'.log'

print('Set up:',sLogFile)

logging.basicConfig(level=logging.DEBUG,

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

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

filename=sLogFile,

filemode='w')

console = logging.StreamHandler()

console.setLevel(logging.INFO)

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

console.setFormatter(formatter)

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

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

for sLevel in sLevels:

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

logger = logging.getLogger(sApp)

if sLevel == 'debug':

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

if sLevel == 'info':

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

if sLevel == 'warning':

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

if sLevel == 'error':

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

A computer screen with white text

Description automatically generated

Practical 3: Retrieve Superstep

The Retrieve superstep is a practical method for importing completely into the processing ecosystem a data lake consisting of various external data sources. The Retrieve superstep is the first contact between your data science and the source systems. I will guide you through a methodology of how to handle this discovery of the data up to the point you have all the data you need to evaluate the system you are working with, by deploying your data science skills. The successful retrieval of the data is a major stepping-stone to ensuring that you are performing good data science. Data lineage delivers the audit trail of the data elements at the lowest granular level, to ensure full data governance.

A]. Perform the following data processing using R.

Code:

library(readr)

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

cols(

  ID = col\_double(), Country = col\_character(),

  `Place Name` = col\_character(),

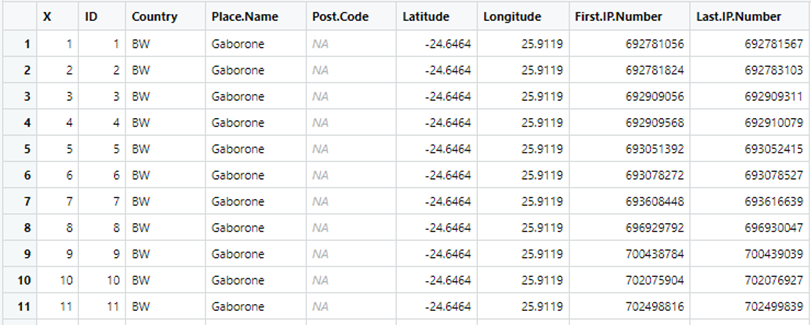
  `Post Code` = col\_double(), Latitude = col\_double(), Longitude = col\_double(),

  `First IP Number` = col\_double(),

  `Last IP Number` = col\_double()

)

View(IP\_DATA\_ALL)



spec(IP\_DATA\_ALL)

cols(

  ID = col\_double(), Country = col\_character(),

  `Place Name` = col\_character(),

  `Post Code` = col\_double(), Latitude = col\_double(), Longitude = col\_double(),

  `First IP Number` = col\_double(),

  `Last IP Number` = col\_double()

)

library(tibble)

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

New names:

  Place Name -> Place.Name,

  Post Code -> Post.Code,

  First IP Number -> First.IP.Number,

  Last IP Number -> Last.IP.Number,

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)

hist\_latitude =data.table(Latitude=unique(IP\_DATA\_ALL\_FIX [is.na(IP\_DATA\_ALL\_with\_ID ['Latitude']) == 0, ]$Latitude))

setkeyv(hist\_latitude, 'Latitude') setorder(hist\_latitude)

hist\_latitude\_with\_id=rowid\_to\_column(hist\_latitude, var = "RowID")

View(hist\_latitude\_with\_id) IP\_DATA\_Latitude\_FREQ=data.table(with(IP\_DATA\_ALL\_FIX,table(Latitude))) View(IP\_DATA\_Latitude\_FREQ)

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

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

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

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

#for mean,median and range

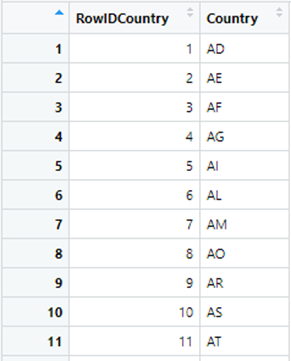
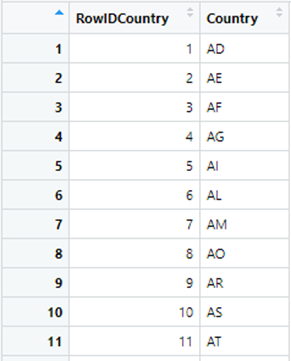
sapply(IP\_DATA\_ALL\_FIX [,'Latitude'], mean, na.rm=TRUE)

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)

Output:

B]. Program to retrieve different attributes of data.

Code:

import pandas as pd

import os

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

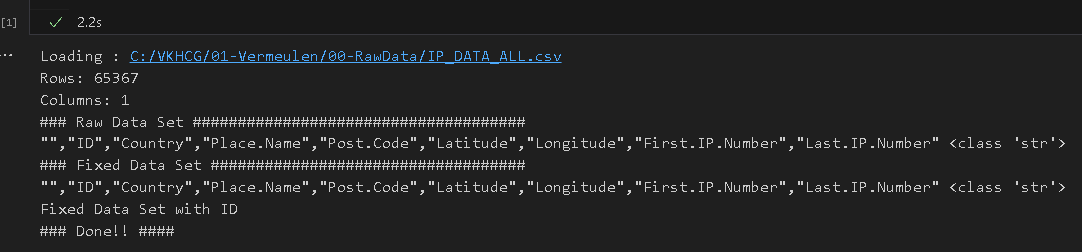
IP\_DATA\_ALL\_with\_ID.index.names = ['RowID']

#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!! ####')



C] Data Pattern

To determine a pattern of the data values, Replace all alphabet values with an uppercase case *A*, all numbers with an uppercase *N*, and replace any spaces with a lowercase letter *b* and all other unknown characters with a lowercase *u*. As a result, “Good Book 101” becomes “AAAAbAAAAbNNNu.”This pattern creation is beneficial for designing any specific assess rules. This pattern view of data is a quick way to identify common patterns or determine standard layouts.

Code: Example1

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)

};

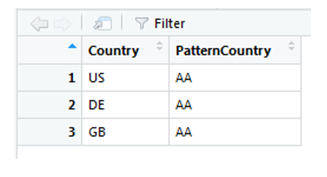
s=chartr(" ","b",s)

s=chartr(".","u",s) pattern\_country[r,]$PatternCountry=s;

};

View(pattern\_country)

Outpt:



Code: Example 2

library(readr) library(data.table) Base='C:/VKHCG'

FileName=paste0(Base,'/01-Vermeulen/00-RawData/IP\_DATA\_ALL.csv') IP\_DATA\_ALL <- read\_csv(FileName) hist\_latitude=data.table(Latitude=unique(IP\_DATA\_ALL$Latitude)) pattern\_latitude=data.table(latitude=hist\_latitude$Latitude,

Patternlatitude=as.character(hist\_latitude$Latitude)) oldchar=c(letters,LETTERS)

newchar=replicate(length(oldchar),"A") for (r in seq(nrow(pattern\_latitude))){ s=pattern\_latitude[r,]$Patternlatitude; 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)

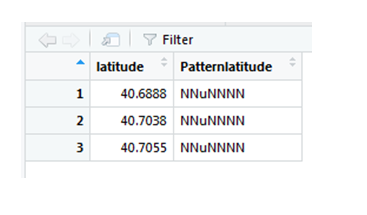
s=chartr("-","u",s)

s=chartr(".","u",s) pattern\_latitude[r,]$Patternlatitude=s;

};

setorder(pattern\_latitude,latitude) View(pattern\_latitude[1:3])

Output:



D.]Loading IP\_DATA\_ALL:

This data set contains all the IP address allocations in the world. It will help you to locateyour customers when interacting with them online.

Create a new Python script file and save it as Retrieve-IP\_DATA\_ALL.py in directory C:\VKHCG\01-Vermeulen\01-Retrieve.

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')

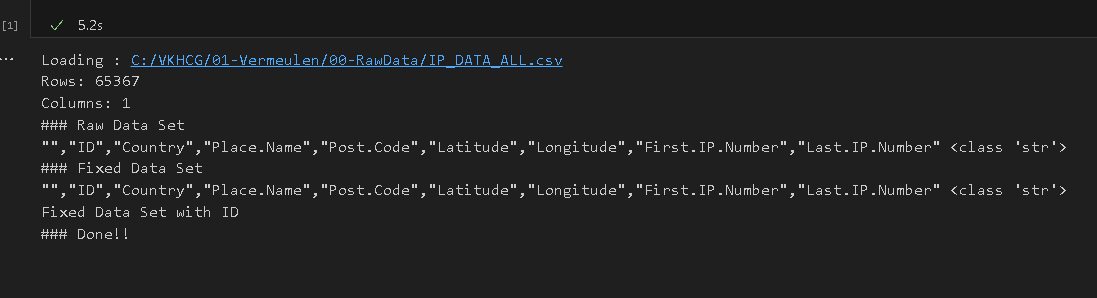
IP\_DATA\_ALL\_with\_ID=IP\_DATA\_ALL\_FIX

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!! ')



E]. Building a Diagram for the Scheduling of Jobs

import sys

import os

import pandas as pd

InputFileName='IP\_DATA\_CORE.csv'

OutputFileName='Retrieve\_Router\_Location.csv'

Base='C:/VKHCG'

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)

sFileDir=Base + '/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!! ')



F]. Picking Content for Billboards

import sys

import os

import pandas as pd

InputFileName='DE\_Billboard\_Locations.csv'

OutputFileName='Retrieve\_DE\_Billboard\_Locations.csv'

Company='02-Krennwallner'

Base='C:/VKHCG'

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

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

Base='C:/VKHCG'

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

print('Loading :',sFileName)

IP\_DATA\_ALL=pd.read\_csv(sFileName,header=0,low\_memory=False, usecols=['Country','PlaceName','Latitude','Longitude'])

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

sFileDir=Base + '/' + Company + '/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)

print('### Done!! ')

Output:

G]. 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='C:/VKHCG'

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

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

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

Base='C:/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','First 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:

H.] XML processing

I]. Connecting to other Data Sources

A. Program to connect to different data sources.

SQLite:

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

print('## Data Values')

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

print(TestData)

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

print('## Data Profile')

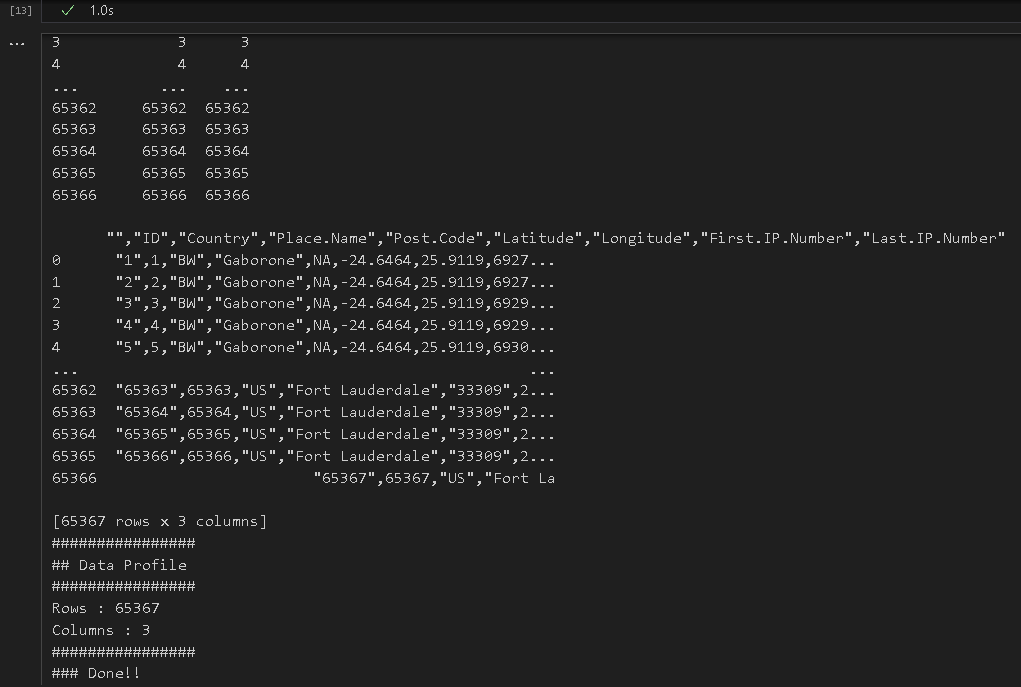
print('################')

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

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

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

print('### Done!! ')



Practical 4: Assessing Data

Assess supersteps

A]. Perform error management on the given data using pandas’ package

Missing Values in Pandas:

i) Drop the Columns Where All Elements Are Missing Values

import sys

import os

import pandas as pd

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)

#importing warehouse

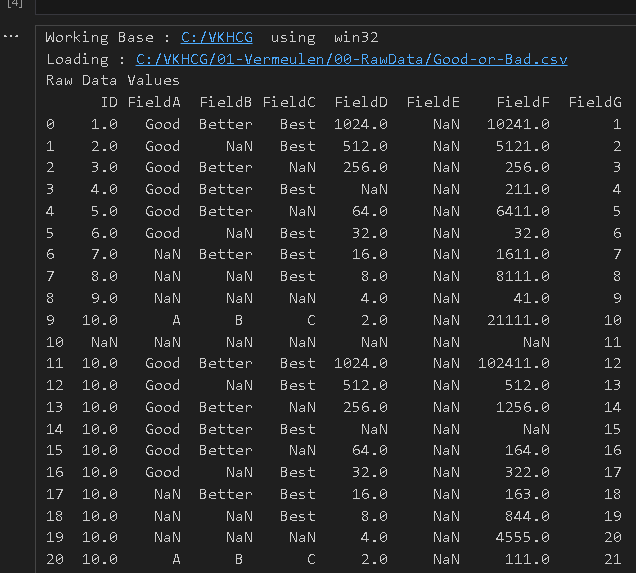
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName

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')

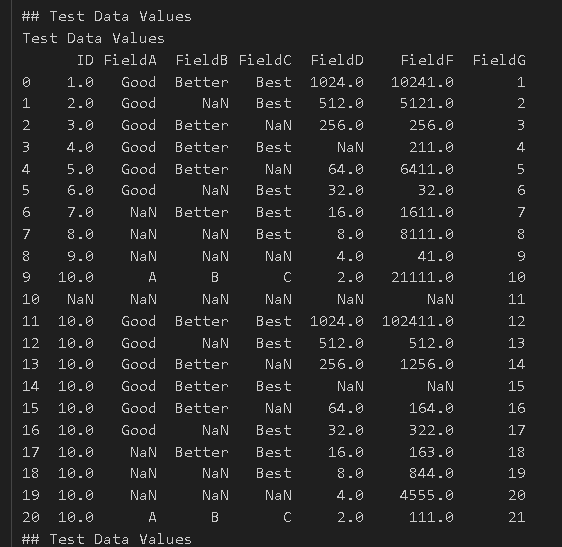
sFileName=sFileDir + '/' + sInputFileName

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

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

print('Test Data Values')

print(TestData)



sFileName=sFileDir + '/' + sInputFileName

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

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

print('## Test Data Values')

ii)Drop the Columns Where Any of the Elements Is Missing Values

import pandas as pd

Base='C:/VKHCG'

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

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

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

sInputFileName='Good-or-Bad.csv'

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

Company='01-Vermeulen'

Base='C:/VKHCG'

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

if not os.path.exists(sFileDir):

os.makedirs(sFileDir

### Import Warehouse

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

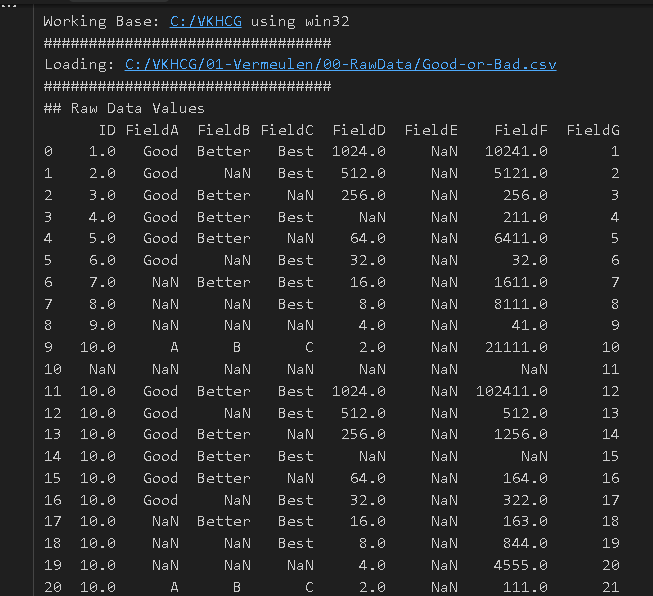
print('Loading:',sFileName)

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

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

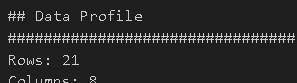
print('## Raw Data Values')

print(RawData)



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

print('## Data Profile')



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

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

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

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

sFileName=sFileDir + '/' + sInputFileName

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

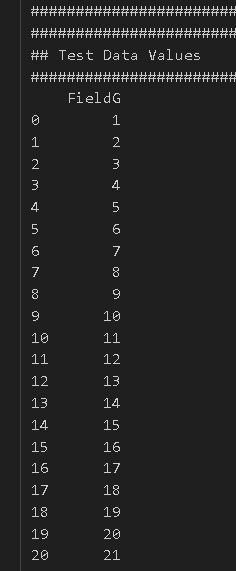
TestData=RawData.dropna(axis=1, how='any')

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

print('## Test Data Values')

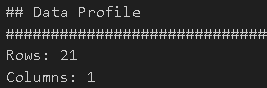
print('################################')

print(TestData)



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

print('## Data Profile')



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

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

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

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

sFileName=sFileDir + '/' + sOutputFileName

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

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

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

iii)Keep Only the Rows That Contain a Maximum of Two Missing Values

import pandas as pd

sInputFileName='Good-or-Bad.csv'

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

Company='01-Vermeulen'

Base='C:/VKHCG'

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

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

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

### Import Warehouse

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

print('Loading :',sFileName)

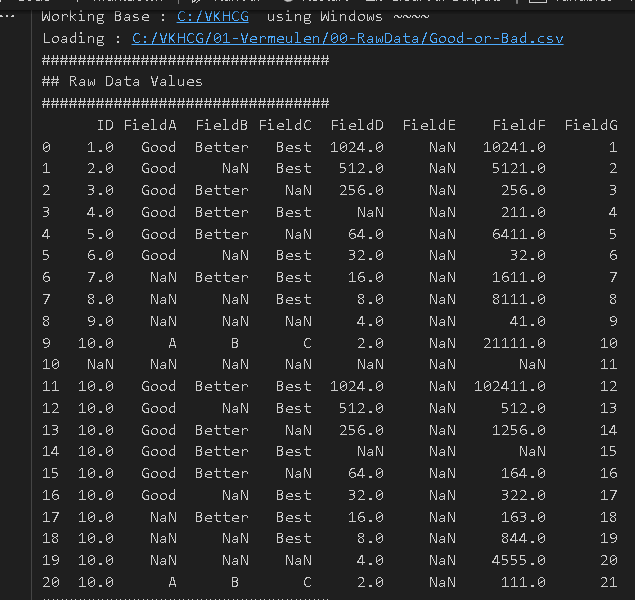
RawData=pd.read\_csv(sFileName,header=0)

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

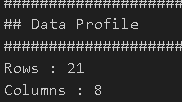
print('## Raw Data Values')

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

print(RawData)

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

print('## Data Profile')



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

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

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

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

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

sFileName=sFileDir + '/' + sInputFileName

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

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

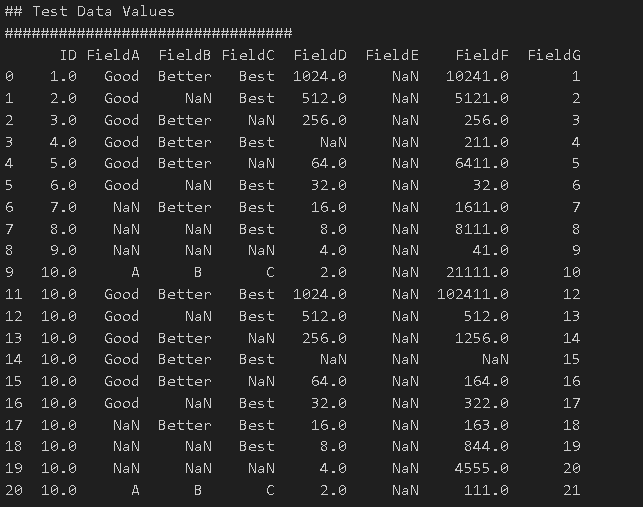
TestData=RawData.dropna(thresh=2)

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

print('## Test Data Values')

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

print(TestData)



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

print('## Data Profile')

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

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

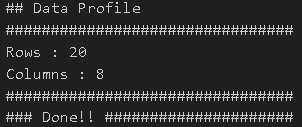
print('Columns :',TestData.shape[1])

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

sFileName=sFileDir + '/' + sOutputFileName

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

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



B]. Write Python / R program to create the network routing diagram from the given data on routers.

Code: -

# Assess-Network-Routing-Company.py#

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

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

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

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

print('Loading :',sFileName)

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

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

print('Loaded Country:',CountryData.columns.values)

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

## Assess Country Data

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

print('Changed :',CountryData.columns.values)

CountryData.rename(columns={'Country': 'Country\_Name'}, inplace=True)

CountryData.rename(columns={'ISO-2-CODE': 'Country\_Code'}, inplace=True)

CountryData.drop('ISO-M49', axis=1, inplace=True)

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

CountryData.drop('RowID', axis=1, inplace=True)

print('To :',CountryData.columns.values)

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

### Import Company Data

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

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

print('Loading :',sFileName)

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

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

print('Loaded Company :',CompanyData.columns.values)

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

## Assess Company Data

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

print('Changed :',CompanyData.columns.values)

CompanyData.rename(columns={'Country': 'Country\_Code'}, inplace=True)

print('To :',CompanyData.columns.values)

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

### Import Customer Data

sFileName=Base + '/' + Company + '/' + sInputFileName3

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

print('Loading :',sFileName)

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

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

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

print('Loaded Customer :',CustomerRawData.columns.values)

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

CustomerData=CustomerRawData.dropna(axis=0, how='any')

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

print('Remove Blank Country Code')

print('Reduce Rows from', CustomerRawData.shape[0],' to ', CustomerData.shape[0])

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

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

print('Changed :',CustomerData.columns.values)

CustomerData.rename(columns={'Country': 'Country\_Code'}, inplace=True)

print('To :',CustomerData.columns.values)

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

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

print('Merge Company and Country Data')

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

CompanyNetworkData=pd.merge(

CompanyData,

CountryData,

how='inner',

on='Country\_Code'

)

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

print('Change ',CompanyNetworkData.columns.values)

for i in CompanyNetworkData.columns.values:

j='Company\_'+i

CompanyNetworkData.rename(columns={i: j}, inplace=True)

print('To ', CompanyNetworkData.columns.values)

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

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

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName

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

print('Storing :', sFileName)

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

CompanyNetworkData.to\_csv(sFileName, index = False, encoding="latin-1")

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

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

print(“Bhavik”)



A screenshot of a computer

Description automatically generated

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

import sys

import os

import pandas as pd

pd.options.mode.chained\_assignment = None

Base='C:/VKHCG'

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

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

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

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

print('Loading :',sFileName)

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

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

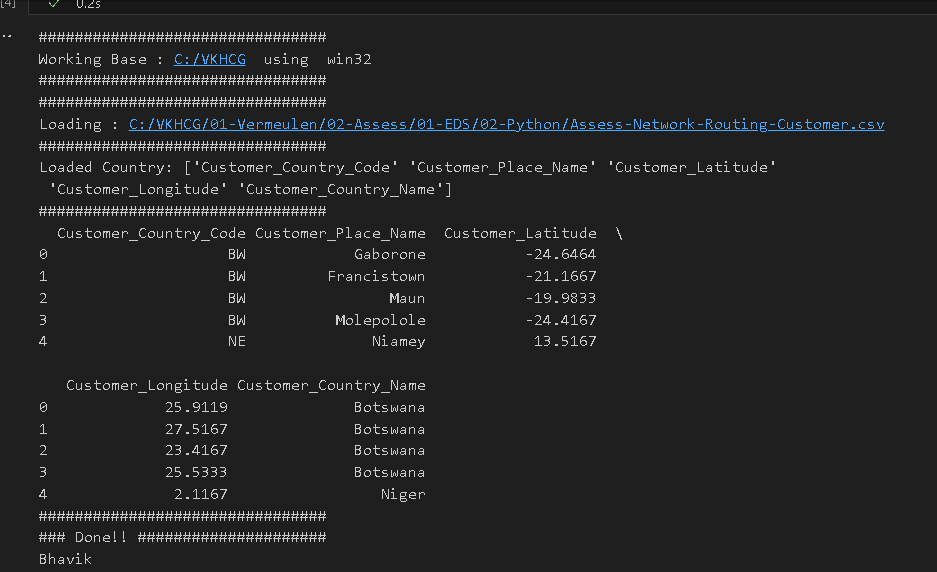
print('Loaded Country:',CustomerData.columns.values)

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

print(CustomerData.head())

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

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



##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)

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

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

print('Loading :',sFileName)

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

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

print('Loaded IP :', IPData.columns.values)

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

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

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)

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

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

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName

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

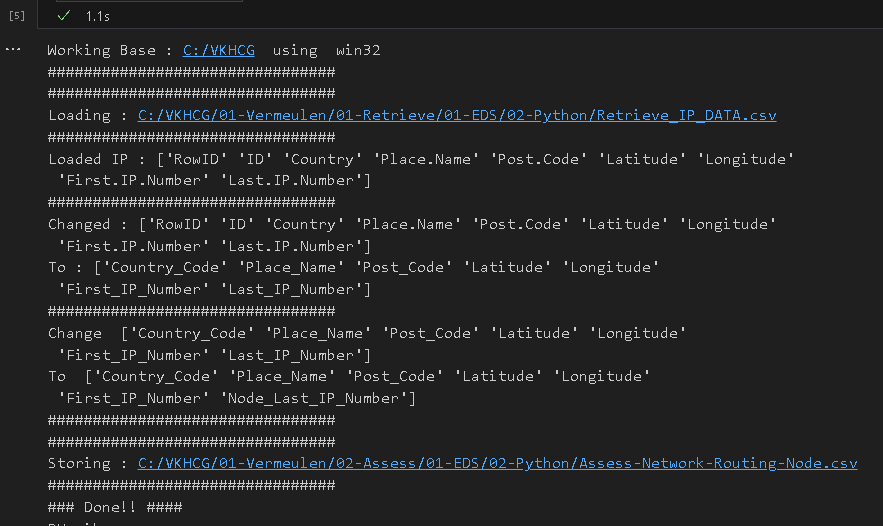
print('Storing :', sFileName)

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

IPData.to\_csv(sFileName, index = False, encoding="latin-1")

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

print("BHavik")



C]. Write a Python to build directed acyclic graph.

import networkx as nx

import matplotlib.pyplot as plt

import sys

import os

import pandas as pd

Base='C:/VKHCG'

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

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

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

sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve\_Router\_Location.csv'

sOutputFileName1='Assess-DAG-Company-Country.png'

sOutputFileName2='Assess-DAG-Company-Country-Place.png'

Company='01-Vermeulen'

### Import Company Data

sFileName=Base + '/' + Company + '/' + sInputFileName

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

print('Loading :',sFileName)

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

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

print('Loaded Company :',CompanyData.columns.values)

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

print(CompanyData)

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

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

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

G1=nx.DiGraph()

G2=nx.DiGraph()

for i in range(CompanyData.shape[0]): G1.add\_node(CompanyData['Country'][i])

sPlaceName= CompanyData['Place\_Name'][i] + '-' + CompanyData['Country'][i]

G2.add\_node(sPlaceName)

for n1 in G1.nodes():

for n2 in G1.nodes():

if n1 != n2:

print('Link :',n1,' to ', n2)

G1.add\_edge(n1,n2)

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

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

print("Nodes of graph: ")

print(G1.nodes())

print("Edges of graph: ")

print(G1.edges())

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

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

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName1

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

print('Storing :', sFileName)

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

nx.draw(G1,pos=nx.spectral\_layout(G1),

nodecolor='r',edge\_color='g',

with\_labels=True,

node\_size=8000,

font\_size=12)

plt.savefig(sFileName)

plt.show() # display

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

for n1 in G2.nodes():

for n2 in G2.nodes():

if n1 != n2:

print('Link :',n1,' to ', n2)

G2.add\_edge(n1,n2)

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

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

print("Nodes of graph: ")

print(G2.nodes())

print("Edges of graph: ")

print(G2.edges())

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

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

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName=sFileDir + '/' + sOutputFileName2

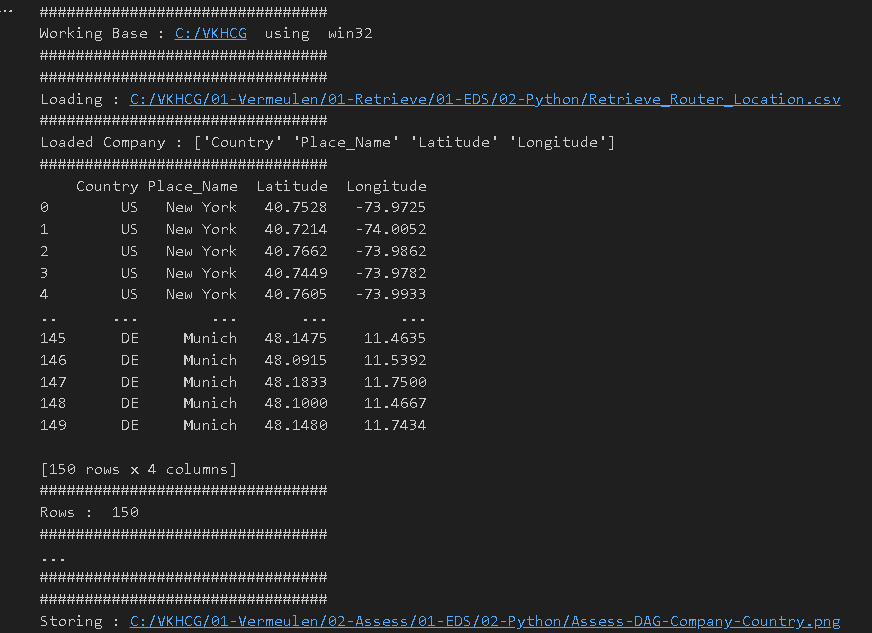
print('################################')

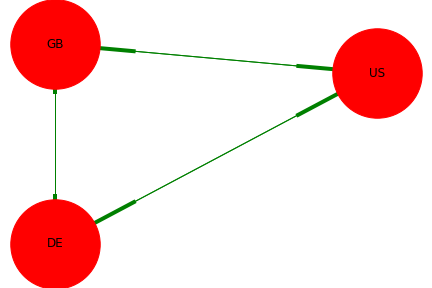
print('Storing :', sFileName)

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

nx.draw(G2,pos=nx.spectral\_layout(G2), nodecolor='r',edge\_color='b', with\_labels=True,node\_size=8000, font\_size=12)

plt.savefig(sFileName) # save as png plt.show() # display





F]. Write a Python / R program to plan the locations of the warehouses from the given data.

Planning the Locations of the Warehouses

Planning the location of the warehouses requires the assessment of the GPS locations of these warehouses against the requirements for Hillman’s logistics needs.

Code:

#4F

import os

import pandas as pd

from geopy.geocoders import Nominatim

geolocator = Nominatim()

InputDir='01-Retrieve/01-EDS/01-R'

InputFileName='Retrieve\_GB\_Postcode\_Warehouse.csv'

EDSDir='02-Assess/01-EDS'

OutputDir=EDSDir + '/02-Python'

OutputFileName='Assess\_GB\_Warehouse\_Address.csv'

Company='03-Hillman'

Base='C:/VKHCG'

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

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

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

sFileDir=Base + '/' + Company + '/' + EDSDir

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileDir=Base + '/' + Company + '/' + OutputDir

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName

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

print('Loading :',sFileName)

Warehouse=pd.read\_csv(sFileName,header=0,low\_memory=False)

Warehouse.sort\_values(by='postcode', ascending=1)

## Limited to 10 due to service limit on Address Service.

WarehouseGoodHead=Warehouse[Warehouse.latitude != 0].head(5)

WarehouseGoodTail=Warehouse[Warehouse.latitude != 0].tail(5)

WarehouseGoodHead['Warehouse\_Point']=WarehouseGoodHead.apply(lambda row: (str(row['latitude'])+','+str(row['longitude'])),axis=1)

WarehouseGoodHead['Warehouse\_Address']=WarehouseGoodHead.apply(lambda row:geolocator.reverse(row['Warehouse\_Point']).address,axis=1)

WarehouseGoodHead.drop('Warehouse\_Point', axis=1, inplace=True)

WarehouseGoodHead.drop('id', axis=1, inplace=True)

WarehouseGoodHead.drop('postcode', axis=1, inplace=True)

WarehouseGoodTail['Warehouse\_Point']=WarehouseGoodTail.apply(lambda row: (str(row['latitude'])+','+str(row['longitude'])),axis=1)

WarehouseGoodTail['Warehouse\_Address']=WarehouseGoodTail.apply(lambda row:geolocator.reverse(row['Warehouse\_Point']).address,axis=1)

WarehouseGoodTail.drop('Warehouse\_Point', axis=1, inplace=True)

WarehouseGoodTail.drop('id', axis=1, inplace=True)

WarehouseGoodTail.drop('postcode', axis=1, inplace=True)

WarehouseGood=WarehouseGoodHead.append(WarehouseGoodTail, ignore\_index=True)

print(WarehouseGood)

sFileName=sFileDir + '/' + OutputFileName

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

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

Output:

Practical 5: Processing Data

A]. Build the time hub, links, and satellites.

import sys

import os

from datetime import datetime

from datetime import timedelta

from pytz import timezone, all\_timezones

import pandas as pd

import sqlite3 as sq

from pandas.io import sql

import uuid

pd.options.mode.chained\_assignment = None

Base='C:/VKHCG'

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

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

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

Company='03-Hillman'

InputDir='00-RawData'

InputFileName='VehicleData.csv'

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

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

sDatabaseName=sDataBaseDir + '/Hillman.db'

conn1 = sq.connect(sDatabaseName)

sDataVaultDir=Base + '/88-DV'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

sDatabaseName=sDataVaultDir + '/datavault.db'

conn2 = sq.connect(sDatabaseName)

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

numUnits=10\*365\*24

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

t=0

for i in date\_list: now\_utc=i.replace(tzinfo=timezone('UTC'))

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

print(sDateTime)

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

t+=1

IDNumber=str(uuid.uuid4())

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

('IDNumber', [IDNumber]), ('nDateTimeValue', [now\_utc]), ('DateTimeValue', [sDateTime]), ('DateTimeKey', [sDateTimeKey])]

if t==1:

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

else:

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

TimeFrame = TimeFrame.append(TimeRow)

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

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

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

sTable = 'Process-Time'

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

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

sTable = 'Hub-Time'

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

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

active\_timezones=all\_timezones

z=0

for zone in active\_timezones: t=0

for j in range(TimeFrame.shape[0]): now\_date=TimeFrame['nDateTimeValue'][j]

DateTimeKey=TimeFrame['DateTimeKey'][j]

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

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

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

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

print(sZoneDateTime)

t+=1

z+=1

IDZoneNumber=str(uuid.uuid4())

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

('IDZoneNumber', [IDZoneNumber]), ('DateTimeKey', [DateTimeKey]), ('UTCDateTimeValue', [sDateTime]), ('Zone', [zone]),

('DateTimeValue', [sZoneDateTime])]

if t==1:

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

else:

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

TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)

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

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

sTable = 'Process-Time-'+sZone

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

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

sTable = 'Satellite-Time-'+sZone

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

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

print('Vacuum Databases')

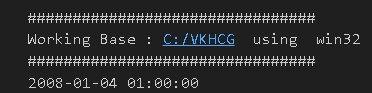
sSQL="VACUUM;"

sql.execute(sSQL,conn1)

sql.execute(sSQL,conn2)

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

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



Golden Nominal

A golden nominal record is a single person’s record, with distinctive references for use by all systems. This gives the system a single view of the person. I use first name, other names, last name, and birth date as my golden nominal. The data we have in the assess directory requires a birth date to become a golden nominal. The program will generate a golden nominal using our sample data set.

Code:

import sys

import os

import sqlite3 as sq

import pandas as pd

from pandas.io import sql

from datetime import datetime, timedelta

from pytz import timezone, all\_timezones

from random import randint

import uuid

Base='C:/VKHCG'

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

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

Company='04-Clark'

sInputFileName='02-Assess/01-EDS/02-Python/Assess\_People.csv'

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

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

sDatabaseName=sDataBaseDir + '/clark.db'

conn1 = sq.connect(sDatabaseName)

sDataVaultDir=Base + '/88-DV'

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

sDatabaseName=sDataVaultDir + '/datavault.db'

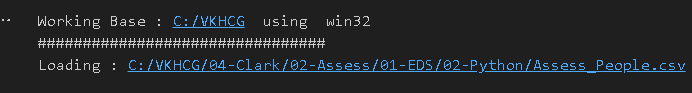
conn2 = sq.connect(sDatabaseName)

### Import Female Data

sFileName=Base + '/' + Company + '/' + sInputFileName

print('Loading :',sFileName)

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



RawData.drop\_duplicates(subset=None, keep='first', inplace=True)

start\_date = datetime(1900,1,1,0,0,0)

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

HoursBirth=100\*365\*24

RawData['BirthDateUTC']=RawData.apply(lambda row:

(start\_date\_utc + timedelta(hours=randint(0, HoursBirth)))

,axis=1)

zonemax=len(all\_timezones)-1

RawData['TimeZone']=RawData.apply(lambda row:

(all\_timezones[randint(0, zonemax)]),axis=1)

RawData['BirthDateISO']=RawData.apply(lambda row:

row["BirthDateUTC"].astimezone(timezone(row['TimeZone']))

,axis=1)

RawData['BirthDateKey']=RawData.apply(lambda row:

row["BirthDateUTC"].strftime("%Y-%m-%d %H:%M:%S")

,axis=1)

RawData['BirthDate']=RawData.apply(lambda row:

row["BirthDateISO"].strftime("%Y-%m-%d %H:%M:%S")

,axis=1)

RawData['PersonID']=RawData.apply(lambda row:

str(uuid.uuid4())

,axis=1)

Data=RawData.copy()

Data.drop('BirthDateUTC', axis=1,inplace=True)

Data.drop('BirthDateISO', axis=1,inplace=True)

indexed\_data = Data.set\_index(['PersonID'])

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

sTable='Process\_Person'

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

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

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

PersonHubRaw=Data[['PersonID','FirstName','SecondName','LastName','BirthDateKey']]

PersonHubRaw['PersonHubID']=RawData.apply(lambda row:

str(uuid.uuid4())

,axis=1)

PersonHub=PersonHubRaw.drop\_duplicates(subset=None,keep='first',inplace=False)

indexed\_PersonHub = PersonHub.set\_index(['PersonHubID'])

sTable = 'Hub-Person'

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

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

PersonSatelliteGenderRaw=Data[['PersonID','FirstName','SecondName','LastName'\

,'BirthDateKey','Gender']]

PersonSatelliteGenderRaw['PersonSatelliteID']=RawData.apply(lambda row:str(uuid.uuid4()),axis=1)

PersonSatelliteGender=PersonSatelliteGenderRaw.drop\_duplicates(subset=None,keep='first',inplace=False)

indexed\_PersonSatelliteGender = PersonSatelliteGender.set\_index(['PersonSatelliteID'])

sTable = 'Satellite-Person-Gender'

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

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

PersonSatelliteBirthdayRaw=Data[['PersonID','FirstName','SecondName','LastName', 'BirthDateKey','TimeZone','BirthDate']]

PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row: str(uuid.uuid4()),axis=1)

PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop\_duplicates(subset=None, keep='first', inplace=False)

indexed\_PersonSatelliteBirthday = PersonSatelliteBirthday.set\_index(['PersonSatelliteID'])

sTable = 'Satellite-Person-Names'

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

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

sFileDir=Base + '/' + Company + '/03-Process/01-EDS/02-Python'

if not os.path.exists(sFileDir):

os.makedirs(sFileDir)

sOutputFileName = sTable + '.csv'

sFileName=sFileDir + '/' + sOutputFileName

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

print('Storing :', sFileName)

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

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

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

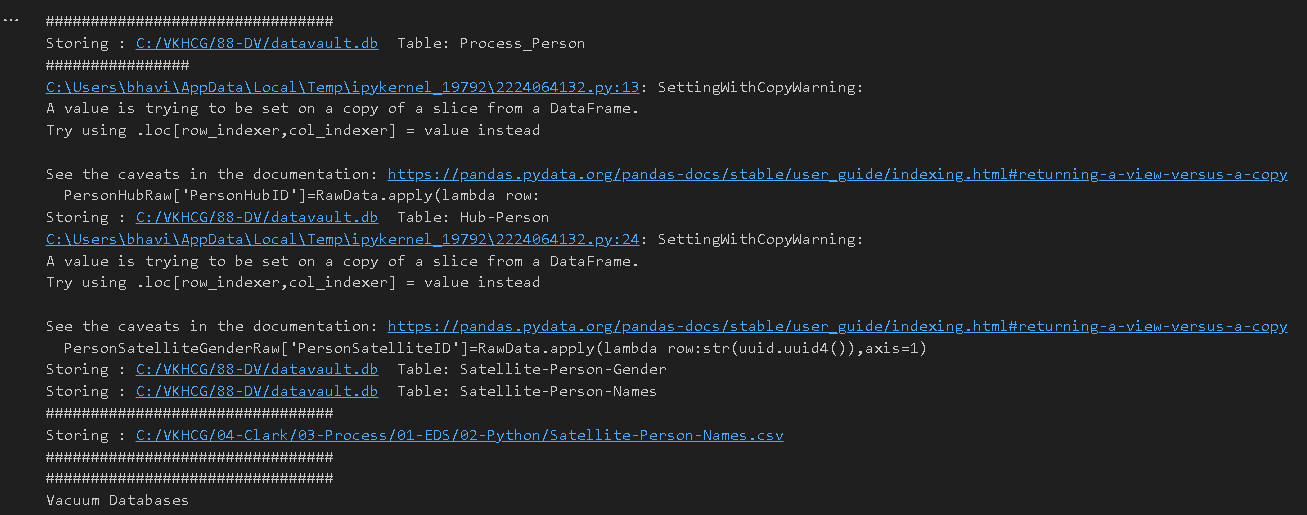
print('Vacuum Databases')

sSQL="VACUUM;"

sql.execute(sSQL,conn1)

sql.execute(sSQL,conn2)

print("Bhavik")



Vehicles

import sys

import os

import pandas as pd

import sqlite3 as sq

from pandas.io import sql

import uuid

pd.options.mode.chained\_assignment = None

Base='C:/VKHCG'

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

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

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

Company='03-Hillman'

InputDir='00-RawData'

InputFileName='VehicleData.csv'

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

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

sDatabaseName=sDataBaseDir + '/Hillman.db'

conn1 = sq.connect(sDatabaseName)

sDataVaultDir=Base + '/88-DV'

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

sDatabaseName=sDataVaultDir + '/datavault.db'

conn2 = sq.connect(sDatabaseName)

sFileName=Base + '/' + Company + '/' + InputDir + '/' + InputFileName

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

print('Loading :',sFileName)

VehicleRaw=pd.read\_csv(sFileName,header=0,low\_memory=False,encoding='latin-1')

sTable='Process\_Vehicles'

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

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

VehicleRawKey=VehicleRaw[['Make','Model']].copy()

VehicleKey=VehicleRawKey.drop\_duplicates()

VehicleKey['ObjectKey']=VehicleKey.apply(lambda row:

str('('+ str(row['Make']).strip().replace(' ', '-').replace('/', '-').lower() +

')-(' + (str(row['Model']).strip().replace(' ', '-').replace(' ', '-').lower())

+')'),axis=1)

VehicleKey['ObjectType']=VehicleKey.apply(lambda row: 'vehicle',axis=1)

VehicleKey['ObjectUUID']=VehicleKey.apply(lambda row: str(uuid.uuid4())

,axis=1)

### Vehicle Hub

VehicleHub=VehicleKey[['ObjectType','ObjectKey','ObjectUUID']].copy()

VehicleHub.index.name='ObjectHubID'

sTable = 'Hub-Object-Vehicle'

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

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

### Vehicle Satellite #

VehicleSatellite=VehicleKey[['ObjectType','ObjectKey','ObjectUUID','Make','Model']].copy()

VehicleSatellite.index.name='ObjectSatelliteID'

sTable = 'Satellite-Object-Make-Model'

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

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

### Vehicle Dimension

sView='Dim-Object'

print('Storing :',sDatabaseName,' View:',sView)

sSQL="CREATE VIEW IF NOT EXISTS [" + sView + "] AS"

sSQL=sSQL+ " SELECT DISTINCT"

sSQL=sSQL+ " H.ObjectType,"

sSQL=sSQL+ " H.ObjectKey AS VehicalKey,"

sSQL=sSQL+ " TRIM(S.Make) AS VehicalMake,"

sSQL=sSQL+ " TRIM(S.Model) AS VehicalModel,"

sSQL=sSQL+ " FROM"

sSQL=sSQL+ " [Hub-Object-Vehicle] AS H"

sSQL=sSQL+ " JOIN"

sSQL=sSQL+ " [Satellite-Object-Make-Model] AS S"

sSQL=sSQL+ " ON"

sSQL=sSQL+ " H.ObjectType=S.ObjectType"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " H.ObjectUUID=S.ObjectUUID;"

sql.execute(sSQL,conn2)

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

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

sSQL=" SELECT DISTINCT"

sSQL=sSQL+ " VehicleMake,"

sSQL=sSQL+ " VehicleModel"

sSQL=sSQL+ " FROM"

sSQL=sSQL+ " [" + sView + "]"

sSQL=sSQL+ " ORDER BY"

sSQL=sSQL+ " VehicleMake"

sSQL=sSQL+ " AND"

sSQL=sSQL+ " VehicleMake;"

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

DimObjectData.index.name='ObjectDimID'

DimObjectData.sort\_values(['VehicleMake','VehicleModel'],inplace=True, ascending=True)

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

print(DimObjectData)

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

print('Vacuum Databases')

sSQL="VACUUM;"

sql.execute(sSQL,conn1)

sql.execute(sSQL,conn2)

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

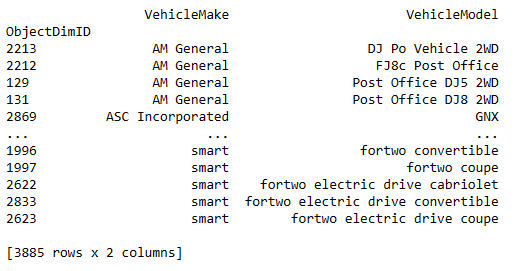
conn1.close()

conn2.close()

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

print("Bhavik")

Output:



Practical 6: Transforming Data

Transform Superstep

The Transform superstep allows you, as a data scientist, to take data from the data vault and formulate answers to questions raised by your investigations. The transformation step is the data science process that converts results into insights. It takes standard data science techniques and methods to attain insight and knowledge about the data that then can be transformed into actionable decisions, which, through storytelling, you can explain to non-data scientists what you have discovered in the data lake.

A] person being borne##

Code:

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

Base='C:/VKHCG'

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

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

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

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

print('Time Category')

print('UTC Time')

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

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

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

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

print(BirthDateZoneUTCStr)

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

print('Birth Date in Reykjavik :')

BirthZone = 'Atlantic/Reykjavik'

BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))

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

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

print(BirthDateStr)

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

IDZoneNumber=str(uuid.uuid4())

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

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

('IDNumber', [IDZoneNumber]),

('DateTimeKey', [sDateTimeKey]),

('UTCDateTimeValue', [BirthDateZoneUTC]),

('Zone', [BirthZone]),

('DateTimeValue', [BirthDateStr])]

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

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

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

sTable = 'Hub-Time-Gunnarsson'

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

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

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

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

sTable = 'Dim-Time-Gunnarsson'

TimeHubIndex.to\_sql(sTable, conn3, if\_exists="replace")

TimeSatellite=TimeFrame[['IDNumber','DateTimeKey','Zone','DateTimeValue']]

TimeSatelliteIndex=TimeSatellite.set\_index(['IDNumber'],inplace=False)

BirthZoneFix=BirthZone.replace(' ','-').replace('/','-')

sTable = 'Satellite-Time-' + BirthZoneFix + '-Gunnarsson'

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

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

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

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

sTable = 'Dim-Time-' + BirthZoneFix + '-Gunnarsson'

TimeSatelliteIndex.to\_sql(sTable, conn3, if\_exists="replace")

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

print('Person Category')

FirstName = 'Guðmundur'

LastName = 'Gunnarsson'

print('Name:',FirstName,LastName)

print('Birth Date:',BirthDateLocal)

print('Birth Zone:',BirthZone)

print('UTC Birth Date:',BirthDateZoneStr)

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

IDPersonNumber=str(uuid.uuid4())

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

('FirstName', [FirstName]),

('LastName', [LastName]),

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

('DateTimeValue', [BirthDateZoneStr])]

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

TimeHub=PersonFrame

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

sTable = 'Hub-Person-Gunnarsson'

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

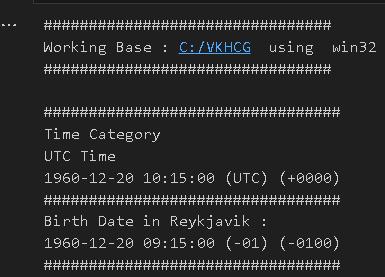
print('Storing :',sDatabaseName,'\n Table:',sTable)

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

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

sTable = 'Dim-Person-Gunnarsson'

TimeHubIndex.to\_sql(sTable, conn3, if\_exists="replace")



**B] Build three items: dimension Person, dimension Time, and factPersonBornAtTime**

Code:

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

Base='C:/VKHCG'

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

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

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

Company='01-Vermeulen'

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

if not os.path.exists(sDataBaseDir):

os.makedirs(sDataBaseDir)

sDatabaseName=sDataBaseDir + '/Vermeulen.db'

conn1 = sq.connect(sDatabaseName)sDataWarehousetDir=Base + '/99-DW'

if not os.path.exists(sDataWarehousetDir):

os.makedirs(sDataWarehousetDir)

sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'

conn2 = sq.connect(sDatabaseName)

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

print('Time Dimension')

BirthZone = 'Atlantic/Reykjavik'

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

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

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

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

BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))

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

BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")IDTimeNumber=str(uuid.uuid4())

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

('UTCDate', [BirthDateZoneStr]),

('LocalTime', [BirthDateLocal]),

('TimeZone', [BirthZone])]

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

DimTime=TimeFrame

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

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

sTable = 'Dim-Time'

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

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

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

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

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

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

print('Dimension Person')

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

FirstName = 'Guðmundur'

LastName = 'Gunnarsson'

IDPersonNumber=str(uuid.uuid4())

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

('FirstName', [FirstName]),

('LastName', [LastName]),

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

('DateTimeValue', [BirthDateZoneStr])]

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

DimPerson=PersonFrame

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

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

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

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

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

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

print('\n')

print('Fact - Person - time')

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

IDFactNumber=str(uuid.uuid4())

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

('IDPersonNumber', [IDPersonNumber]),

('IDTimeNumber', [IDTimeNumber])]

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

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

sTable = 'Fact-Person-Time'

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

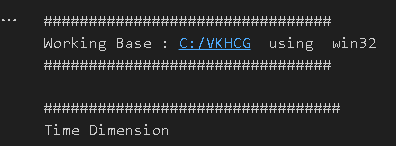
print('Storing :',sDatabaseName,'\n Table:',sTable)

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

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

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

Output:



## **C] Write python program to demonstrate Simple Linear Regression.**

Simple Linear Regression

Linear regression is used if there is a relationship or significant association between the variables. This can be checked by scatterplots. If no linear

association appears between the variables, fitting a linear regression model to the data will not provide a useful model.

A linear regression line has equations in the following form:

Y = a + bX,

Where, X = explanatory variable and

Y = dependent variable

b = slope of the line

a = intercept (the value of y when x = 0)

Code:

import os

import sqlite3 as sq

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets,linear\_model

from sklearn.metrics import mean\_squared\_error,r2\_score

# Load Diabetes Data

diabetes=datasets.load\_diabetes()

# Use only one feature

diabetes\_X=diabetes.data[:,np.newaxis,2]

diabetes\_X\_train= diabetes\_X[:-30]

diabetes\_X\_test= diabetes\_X[-50:]

diabetes\_Y\_train= diabetes.target[:-30]

diabetes\_Y\_test= diabetes.target[-50:]

regr= linear\_model.LinearRegression()

regr.fit(diabetes\_X\_train,diabetes\_Y\_train)

diabetes\_Y\_pred= regr.predict(diabetes\_X\_test)

print('Coefficients: \n',regr.coef\_)

print('Mean Squared Error: %.2f'%mean\_squared\_error(diabetes\_Y\_test,diabetes\_Y\_pred))

print('Variance Score: %.2f'%r2\_score(diabetes\_Y\_test,diabetes\_Y\_pred))

plt.scatter(diabetes\_X\_test,diabetes\_Y\_test,color='black')

plt.plot(diabetes\_X\_test,diabetes\_Y\_pred,color='red',linewidth=3)

plt.xticks(())

plt.yticks(())

plt.axis('tight')

plt.title('Diabetes')

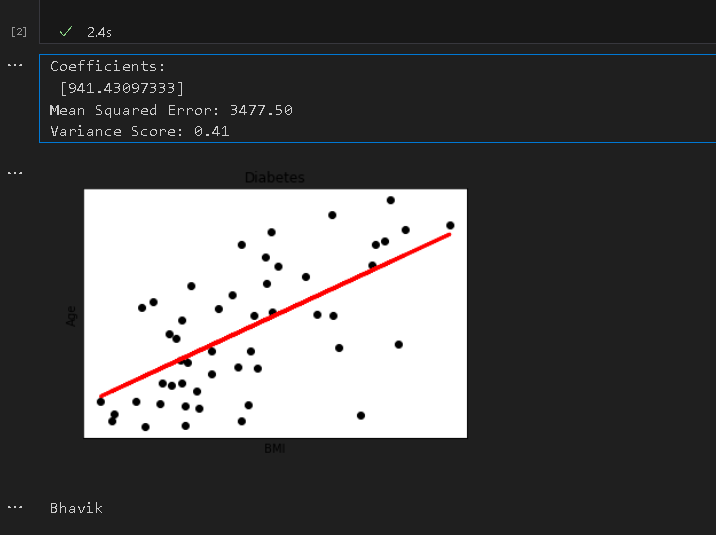
plt.xlabel('BMI')

plt.ylabel('Age')

plt.show()

print("Bhavik")

Output:



Practical 7: Organizing Data

Organize Superstep

The Organize superstep takes the complete data warehouse you built at the end of the Transform superstep and subsectionsit intobusiness-specific datamarts. A datamart is the access layer of the datawarehouseenvironment built to expose data to the users. The data mart is a subset of the data warehouse and is generally oriented to a specific business group.

1) Horizontal Style

Performing horizontal-style slicing or subsetting of the data warehouse is achieved by applying a filter technique that forces the data warehouse to show only the data for a specific preselected set of filtered outcomes against the data population.The horizontal-style slicing selects the subset of rows from the population while preserving the columns. That is, the data science tool can see the complete record for the records in the subset of records.

Code:

import sys

import os

import pandas as pd

import sqlite3 as sqif sys.platform == 'linux':

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

else:

Base='C:/VKHCG'

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

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)

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

sTable = 'Dim-BMI'

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

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

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

print(')

sTable = 'Dim-BMI'

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

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

sSQL="SELECT PersonID,\

Height,\

Weight,\

bmi,\

Indicator\

FROM [Dim-BMI]\

WHERE \

Height > 1.5 \

and Indicator = 1\

ORDER BY \

Height,\

Weight;"

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

DimPerson=PersonFrame1

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

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

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

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

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

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

sTable = 'Dim-BMI-Horizontal'

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

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

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

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

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

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

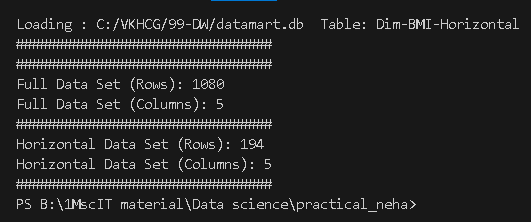
print('Full Data Set (Columns):', PersonFrame0.shape[1])

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

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

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

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



2) Vertical Style

Performing vertical-style slicing or subsetting of the data warehouse is achieved by applying a filter technique that forces the data warehouse to show only the data for specific preselected filtered outcomes against the data population. The vertical-style slicing selects the subset of columns from the population, while preserving the rows. That is, the data science tool can see only the preselected columns from a record for all the records in the population.

Code:

import sys

import os

import pandas as pd

import sqlite3 as sq

if sys.platform == 'linux':

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

else:

Base='C:/VKHCG'

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

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

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

Company='01-Vermeulen'

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn1 = sq.connect(sDatabaseName)

sDatabaseName=sDataWarehouseDir + '/datamart.db'

conn2 = sq.connect(sDatabaseName)

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

sTable = 'Dim-BMI'

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

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

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

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

sTable = 'Dim-BMI'

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

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

sSQL="SELECT \

Height,\

Weight,\

Indicator\

FROM [Dim-BMI];"

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

DimPerson=PersonFrame1

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

sTable = 'Dim-BMI-Vertical'

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

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

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

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

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

sTable = 'Dim-BMI-Vertical'

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

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

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

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

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

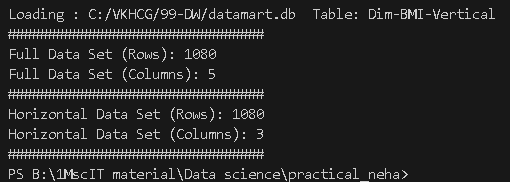
print('Full Data Set (Columns):', PersonFrame0.shape[1])

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

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

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

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



3) Island Style

Performing island-style slicing or subsetting of the data warehouse is achieved by applying a combination of horizontal- and vertical-style slicing. This generates a subset of specific rows and specific columns reduced at the same time.

Code:

import sys

import os

import pandas as pd

import sqlite3 as sq

:Base='C:/VKHCG'

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

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

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

Company='01-Vermeulen'

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn1 = sq.connect(sDatabaseName)

sDatabaseName=sDataWarehouseDir + '/datamart.db'

conn2 = sq.connect(sDatabaseName)

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

sTable = 'Dim-BMI'

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

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

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

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

sTable = 'Dim-BMI'

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

sSQL="SELECT \

Height,\

Weight,\

Indicator\

FROM [Dim-BMI]\

WHERE Indicator > 2\

ORDER BY \

Height,\

Weight;"

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

DimPerson=PersonFrame1

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

sTable = 'Dim-BMI-Vertical'

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

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

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

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

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

sTable = 'Dim-BMI-Vertical'

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

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

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

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

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

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

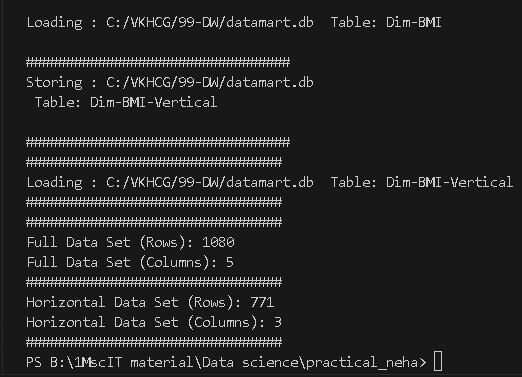
print('Full Data Set (Columns):', PersonFrame0.shape[1])

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

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

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

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



4) Secure Vault Style

The secure vault is a version of one of the horizontal, vertical, or island slicing techniques, but the outcome is also attached to the person who performs the query. This is common in multi-security environments, where different users are allowed to see different data sets.

This process works well, if you use a role-based access control (RBAC) approach to restricting system access to authorized users. The security is applied against the “role,” and a person can then, by the security system, simply be added or removed from the role, to enable or disable access.

Code:

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

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

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

import sys

import os

import pandas as pd

import sqlite3 as sq

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

if sys.platform == 'linux':

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

else:

Base='C:/VKHCG'

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

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

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

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

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

Company='01-Vermeulen'

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

sDataWarehouseDir=Base + '/99-DW'

if not os.path.exists(sDataWarehouseDir):

os.makedirs(sDataWarehouseDir)

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

sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'

conn1 = sq.connect(sDatabaseName)

sDatabaseName=sDataWarehouseDir + '/datamart.db'

conn2 = sq.connect(sDatabaseName)

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

sTable = 'Dim-BMI'

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

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

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

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

sTable = 'Dim-BMI'

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

sSQL="SELECT \

Height,\

Weight,\

Indicator,\

CASE Indicator\

WHEN 1 THEN 'Pip'\

WHEN 2 THEN 'Norman'\

WHEN 3 THEN 'Grant'\

ELSE 'Sam'\

END AS Name\

FROM [Dim-BMI]\

WHERE Indicator > 2\

ORDER BY \

Height,\

Weight;"

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

DimPerson=PersonFrame1

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

sTable = 'Dim-BMI-Secure'

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

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

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

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

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

sTable = 'Dim-BMI-Secure'

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

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

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

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

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

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

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

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

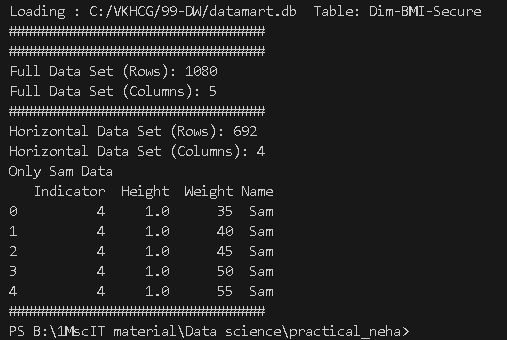
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])

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

print('Only Sam Data')

print(PersonFrame2.head())

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



B]. Create a Network Routing Diagram

Code:

import sys

import os

import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

pd.options.mode.chained\_assignment = None

Base='C:/VKHCG'

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

sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv'

sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company1.gml'

sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company1.png'

Company='01-Vermeulen'

### Import Country Data

sFileName=Base + '/' + Company + '/' + sInputFileName

print('Loading :',sFileName)

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

print(CompanyData.head())

print(CompanyData.shape)

G=nx.Graph()

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

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

Node0=CompanyData['Company\_Country\_Name'][i]

Node1=CompanyData['Company\_Country\_Name'][j]

if Node0 != Node1:

G.add\_edge(Node0,Node1)

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

Node1=CompanyData['Company\_Place\_Name'][i] + '('+ CompanyData['Company\_Country\_Name'][i] + ')'

if Node0 != Node1:

G.add\_edge(Node0,Node1)

print('Nodes:', G.number\_of\_nodes())

print('Edges:', G.number\_of\_edges())

sFileName=Base + '/' + Company + '/' + sOutputFileName1

print('Storing :',sFileName)

nx.write\_gml(G, sFileName)

sFileName=Base + '/' + Company + '/' + sOutputFileName2

print('Storing Graph Image:',sFileName)

plt.figure(figsize=(15, 15))

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

nx.draw\_networkx\_nodes(G,pos, node\_color='k', node\_size=10, alpha=0.8)

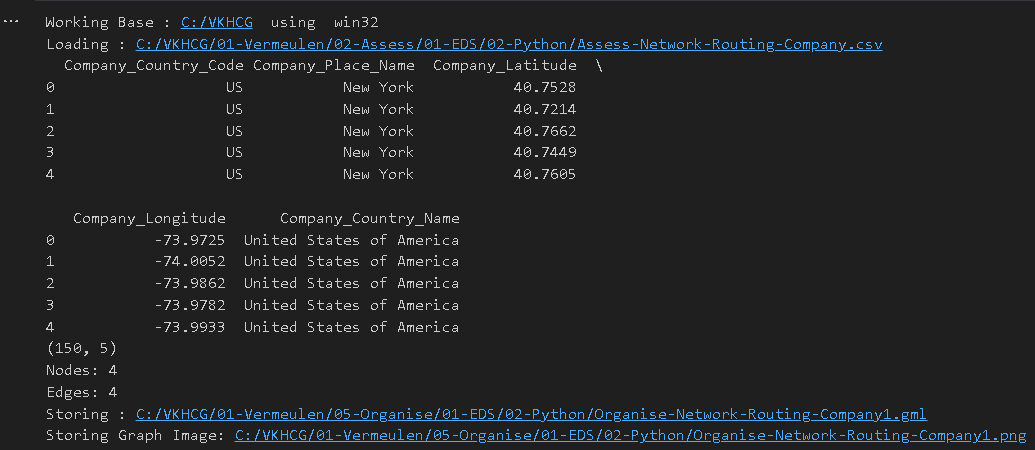
nx.draw\_networkx\_edges(G, pos,edge\_color='r', arrows=False, style='dashed')

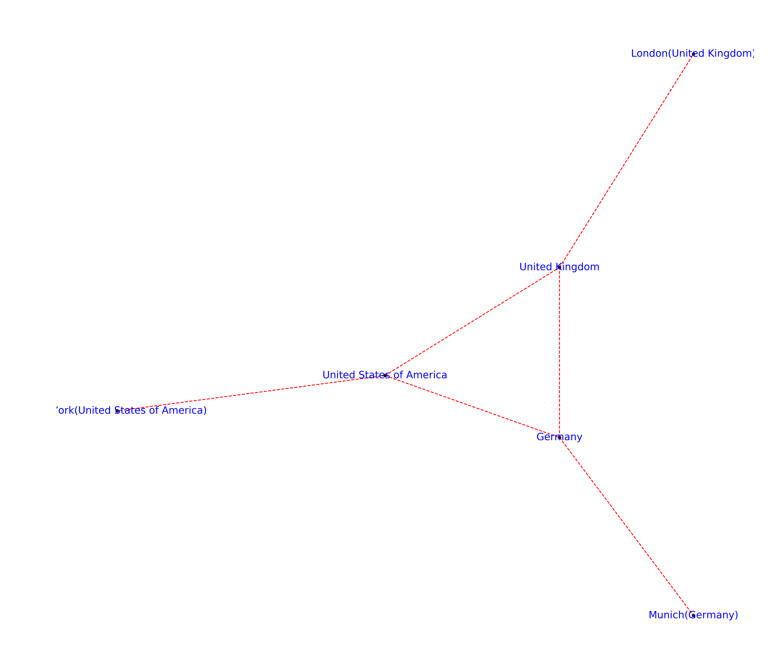
nx.draw\_networkx\_labels(G,pos,font\_size=12,font\_family='sans-serif',font\_color='b')

plt.axis('off')

plt.savefig(sFileName,dpi=600)

plt.show()





Practical 8: Generating Data

Graphics

This section will now guide you through a number of visualizations that particularly useful in presenting data to my customers.

Code:

import sys

import os

import pandas as pd

import matplotlib as ml

from matplotlib import pyplot as plt

Base='C:/VKHCG'

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

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

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

GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'

ml.style.use('ggplot')

1)Pie graph

#pie Graph

data=[

['London', 29.2, 17.4],

['Glasgow', 18.8, 11.3],

['Cape Town', 15.3, 9.0],

['Houston', 22.0, 7.8],

['Perth', 18.0, 23.7],

['San Francisco', 11.4, 33.3]

]

os\_new=pd.DataFrame(data)

pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'Conversion Percent',

'URL', 'Stats URL'],

dtype='object')

os\_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)

os\_new.rename(columns = {1 : "Profit 2016"}, inplace=True)

os\_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

explode = (0, 0, 0, 0, 0, 0.1)

labels=os\_new['Warehouse Location']

colors\_mine = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral',

'lightcyan','lightblue']

os\_new.plot(figsize=(10, 10),kind="pie", y="Profit 2017",autopct='%.2f%%', \

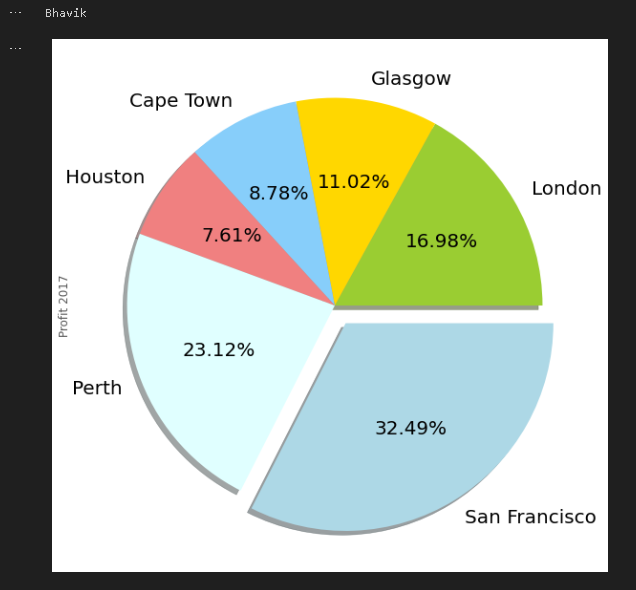
shadow=True, explode=explode, legend = False, colors = colors\_mine,\

labels=labels, fontsize=20)

sPicNameOut1=GBase+'pie\_explode.png'

plt.savefig(sPicNameOut1,dpi=600)

print("Bhavik")



2}Double pie

#Double pie graph

explode = (0, 0, 0, 0, 0, 0)

colors\_mine = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral',

'lightcyan','lightblue']

os\_new.plot(figsize=(10, 5),kind="pie", y=['Profit 2016','Profit 2017'],autopct='%.2f%%', \

shadow=True, explode=explode, legend = False, colors = colors\_mine,\

subplots=True, labels=labels, fontsize=10)

sPicNameOut2=GBase+'pie.png'

plt.savefig(sPicNameOut2,dpi=600)

print("Bhavik")

3)Line graph

#Line graph

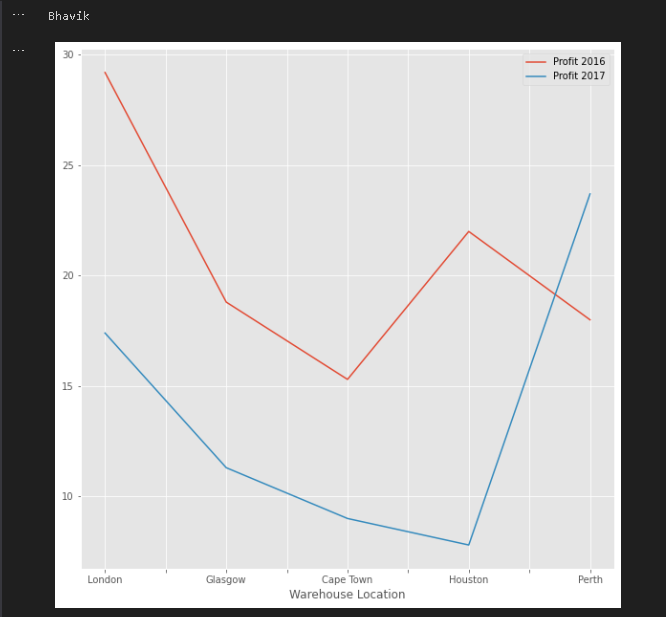
os\_new.iloc[:5].plot(figsize=(10, 10),kind='line',x='Warehouse Location',\

y=['Profit 2016','Profit 2017']);

sPicNameOut3=GBase+'line.png'

plt.savefig(sPicNameOut3,dpi=600)

print("Bhavik")



4)Bar graph

#Bar graph

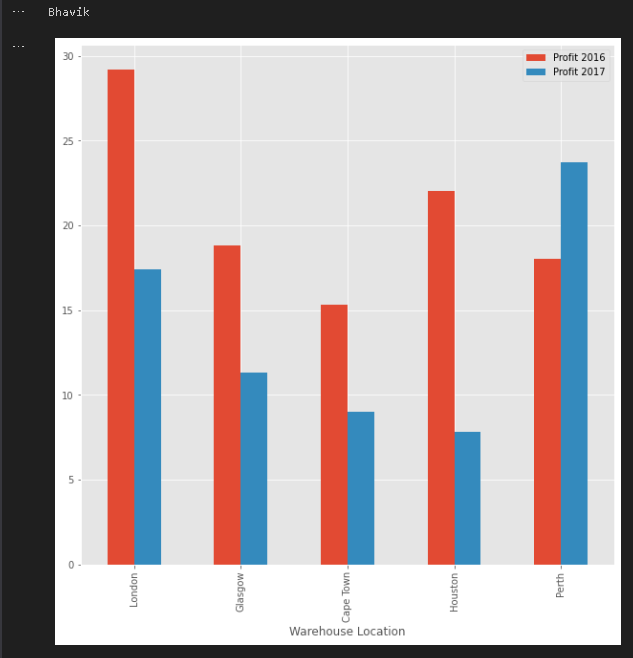
os\_new.iloc[:5].plot(figsize=(10, 10),kind='bar',x='Warehouse Location',\

y=['Profit 2016','Profit 2017']);

sPicNameOut4=GBase+'bar.png'

plt.savefig(sPicNameOut4,dpi=600)

print("Bhavik")



5)Horizontal bar graph

#Horizontal bar

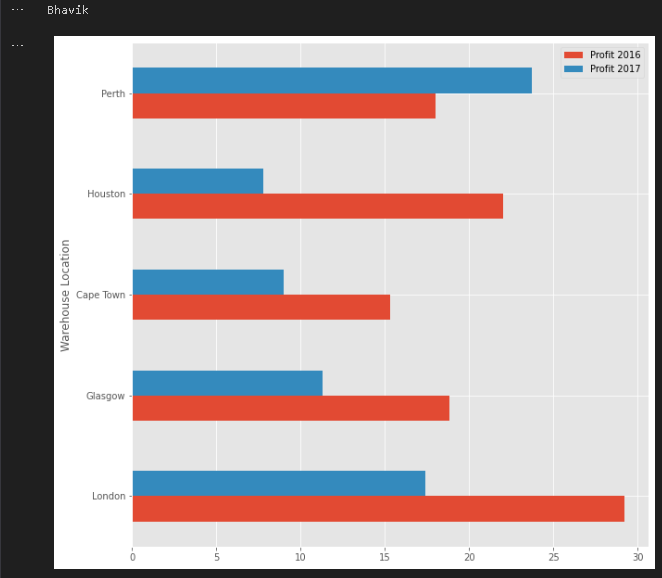
os\_new.iloc[:5].plot(figsize=(10, 10),kind='barh',x='Warehouse Location',\

y=['Profit 2016','Profit 2017']);

sPicNameOut4=GBase+'bar.png'

plt.savefig(sPicNameOut4,dpi=600)

print("Bhavik")



6)Area graph

#Area graph

os\_new.iloc[:5].plot(figsize=(10, 10),kind='area',x='Warehouse Location',\

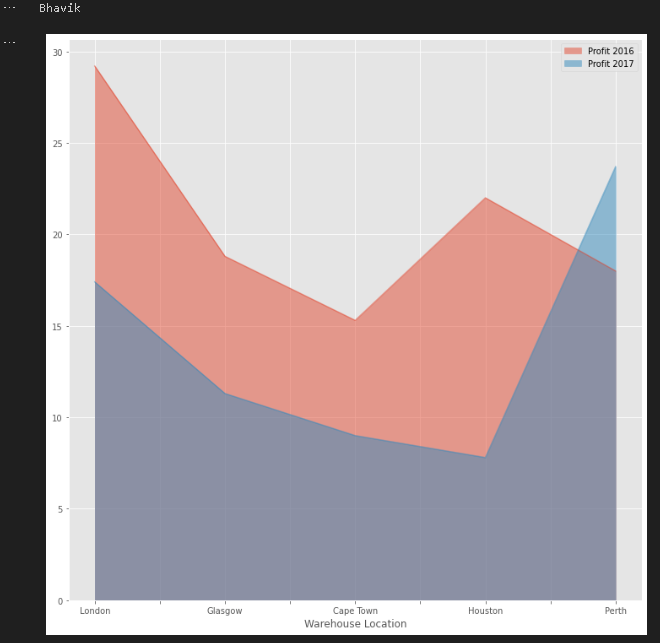
y=['Profit 2016','Profit 2017'],stacked=False);

sPicNameOut6=GBase+'area.png'

plt.tight\_layout()

plt.savefig(sPicNameOut6,dpi=600)

print("Bhavik")



7)Scatter graph

#Scatter graph

os\_new.iloc[:5].plot(figsize=(10, 10),kind='scatter',x='Profit 2016',\

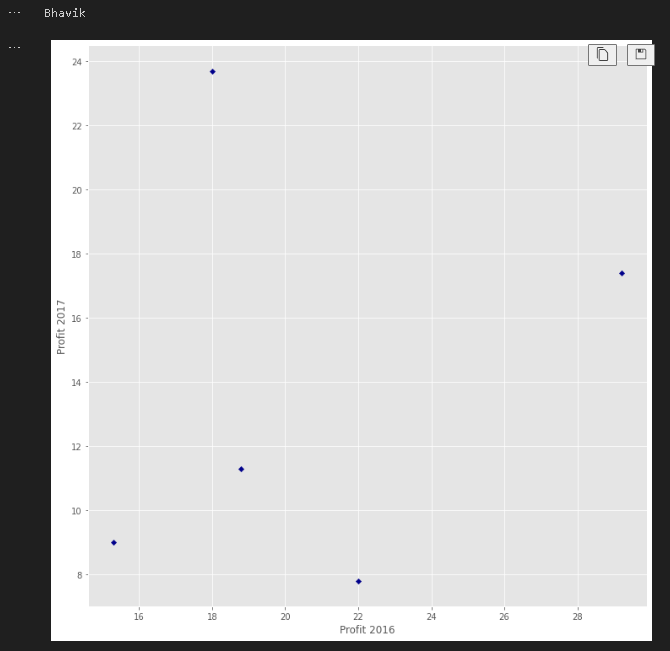
y='Profit 2017',color='DarkBlue',marker='D');

sPicNameOut7=GBase+'scatter.png'

plt.tight\_layout()

plt.savefig(sPicNameOut7,dpi=600)

print("Bhavik")



8)Hex bin graph

#Hex bin graph

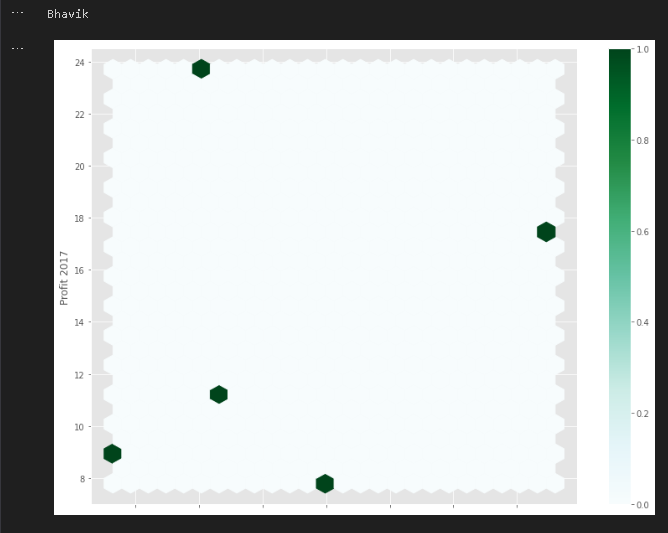
os\_new.iloc[:5].plot(figsize=(13, 10),kind='hexbin',x='Profit 2016',\

y='Profit 2017', gridsize=25);

sPicNameOut8=GBase+'hexbin.png'

plt.savefig(sPicNameOut8,dpi=600)

print("Bhavik")



9) Kernel Density Estimation (KDE) Graph

fig1=plt.figure(figsize=(10, 10))

ser = pd.Series(np.random.randn(1000))

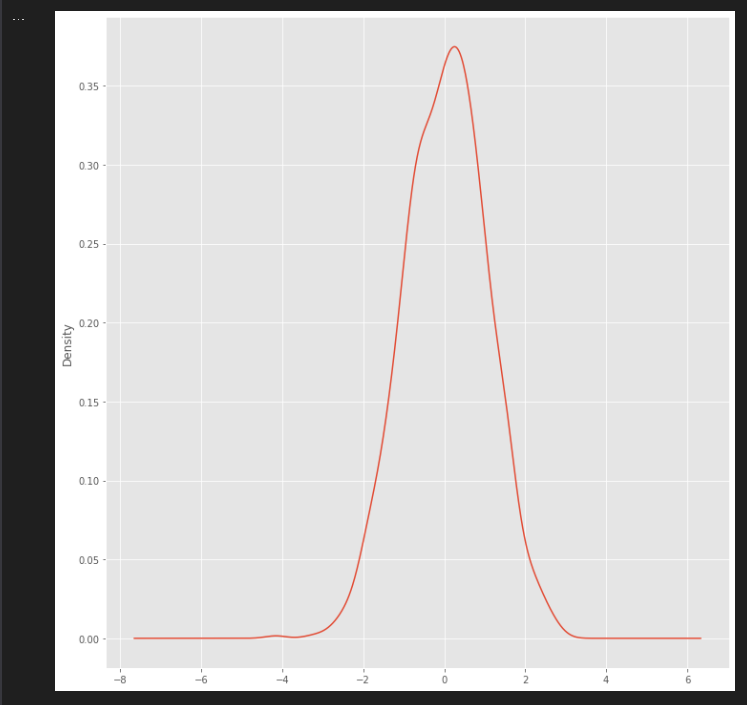
ser.plot(figsize=(10, 10),kind='kde')

sPicNameOut1=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/kde.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight\_layout()

plt.show()



10) Scatter Matrix Graph

fig2=plt.figure(figsize=(10, 10))

from pandas.plotting import scatter\_matrix

df = pd.DataFrame(np.random.randn(1000, 5), columns=['Y2014','Y2015', 'Y2016', 'Y2017', 'Y2018'])

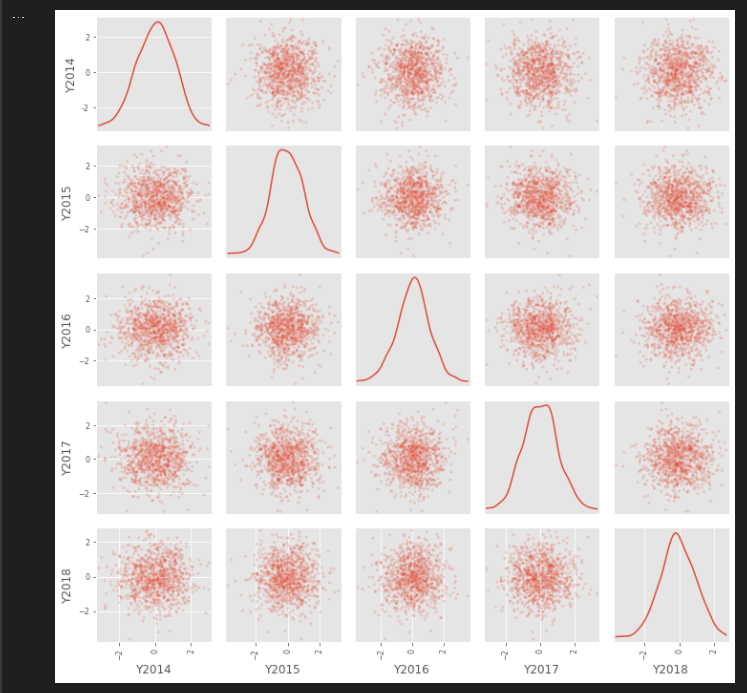
scatter\_matrix(df, alpha=0.2, figsize=(10, 10), diagonal='kde')

sPicNameOut2=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/scatter\_matrix.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight\_layout()

plt.show()



11) Andrews’ Curves

import sys

import os

import pandas as pd

from matplotlib import pyplot as plt

Base='C:/VKHCG'

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

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

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

sDataFile=Base+'/01-Vermeulen/00-RawData/irisdata.csv'

data = pd.read\_csv(sDataFile)

from pandas.plotting import andrews\_curves

plt.figure(figsize=(10, 10))

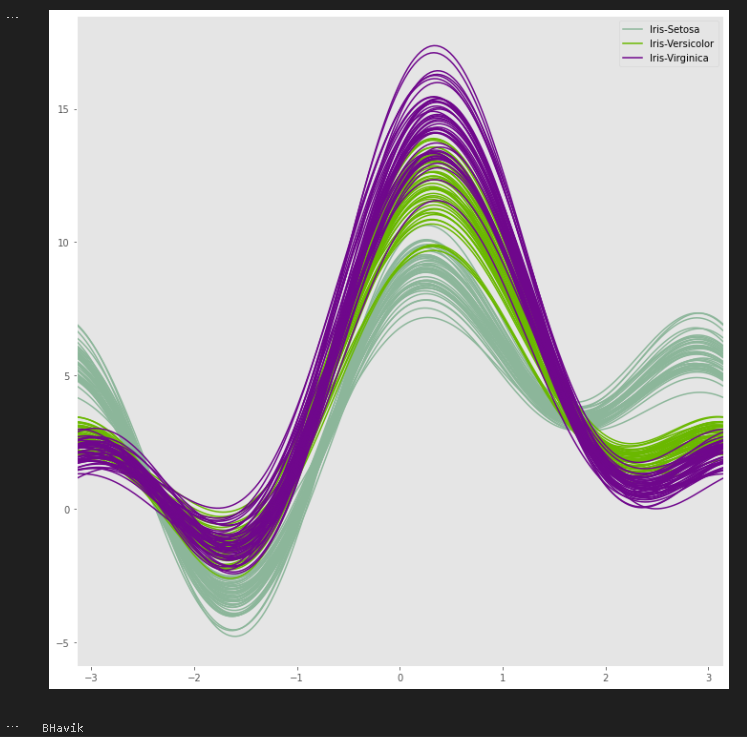
andrews\_curves(data, 'Name')

sPicNameOut1=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/andrews\_curves.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight\_layout()

plt.show()



12) Parallel Coordinates

from pandas.plotting import parallel\_coordinates

plt.figure(figsize=(10, 10))

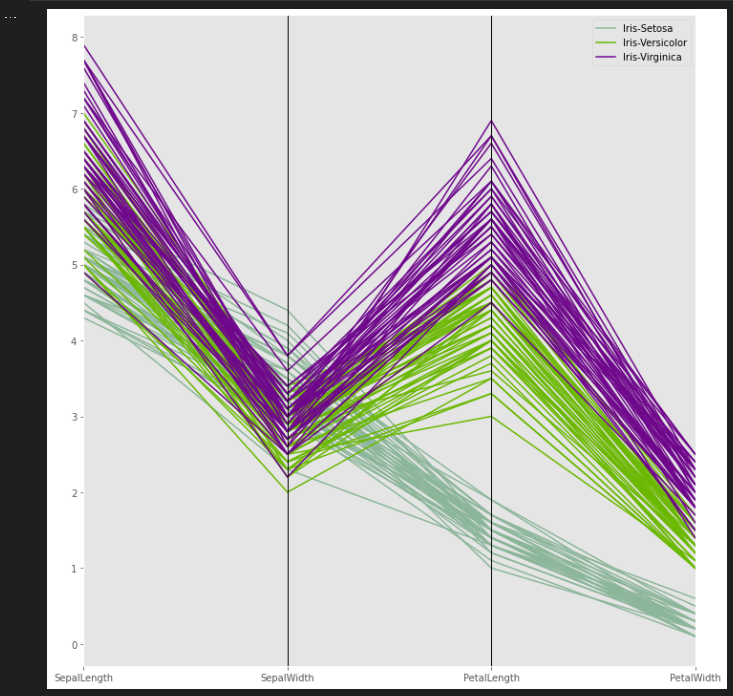
parallel\_coordinates(data, 'Name')

sPicNameOut2=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/parallel\_coordinates.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight\_layout()

plt.show()



13) Lag Plot

from pandas.plotting import lag\_plot

plt.figure(figsize=(10, 10))

data = pd.Series(0.1 \* np.random.rand(1000) + \

0.9 \* np.sin(np.linspace(-99 \* np.pi, 99 \* np.pi, num=1000)))

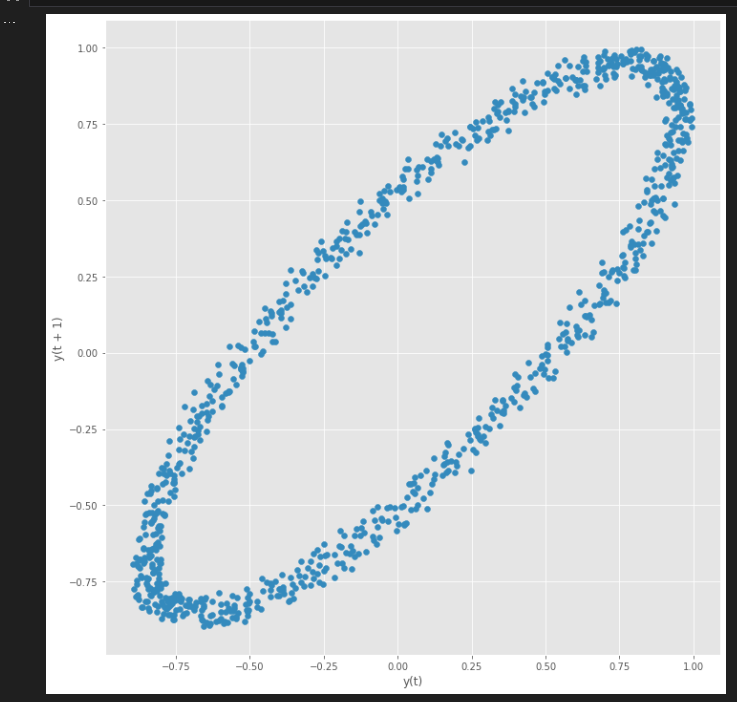
lag\_plot(data)

sPicNameOut1=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/lag\_plot.png'

plt.savefig(sPicNameOut1,dpi=600)

plt.tight\_layout()

plt.show()



14) Autocorrelation Plot

from pandas.plotting import autocorrelation\_plot

plt.figure(figsize=(10, 10))

data = pd.Series(0.7 \* np.random.rand(1000) + \

0.3 \* np.sin(np.linspace(-9 \* np.pi, 9 \* np.pi, num=1000)))

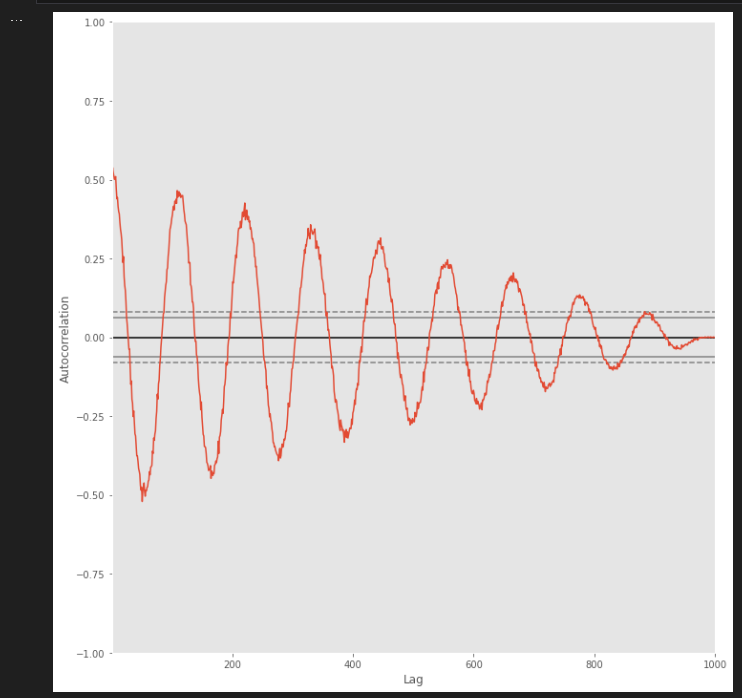
autocorrelation\_plot(data)

sPicNameOut2=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/autocorrelation\_plot.png'

plt.savefig(sPicNameOut2,dpi=600)

plt.tight\_layout()

plt.show()



15) Bootstrap Plot

from pandas.plotting import bootstrap\_plot

data = pd.Series(np.random.rand(1000))

plt.figure(figsize=(10, 10))

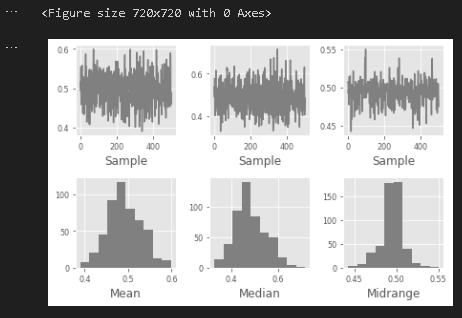
bootstrap\_plot(data, size=50, samples=500, color='grey')

sPicNameOut3=Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/bootstrap\_plot.png'

plt.savefig(sPicNameOut3,dpi=600)

plt.tight\_layout()

plt.show()



16)3DGraph

import os

import numpy as np

from matplotlib import pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

from sklearn import decomposition

from sklearn import datasets

# 3D Graphs

np.random.seed(5)

centers = [[1,1],[-1,-1],[1,-1]]

iris = datasets.load\_iris()

X = iris.data

y = iris.target

fig = plt.figure(1,figsize=(16,12))

plt.clf()

ax = Axes3D(fig,rect=[0,0,0.95,1],elev=48,azim=134)

plt.cla()

pca = decomposition.PCA(n\_components=3)

pca.fit(X)

X = pca.transform(X)

for name,label in [('Setosa',0),('Versicolor',1),('Virginica',2)]:

ax.text3D(X[y == label,0].mean(),

X[y == label,1].mean()+1.5,

X[y == label,2].mean(),name,

horizontalalignment='center',

bbox = dict(alpha=0.5, edgecolor='w', facecolor='w'))

y = np.choose(y,[1,2,0]).astype(np.float)

ax.scatter(X[:,0],X[:,1],X[:,2],c=y,edgecolor='k',marker='p',s=300)

ax.w\_xaxis.set\_ticklabels([])

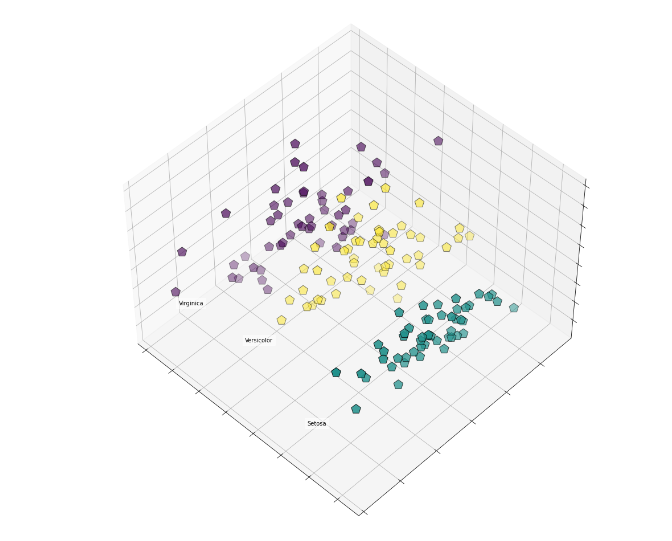
ax.w\_yaxis.set\_ticklabels([])

ax.w\_zaxis.set\_ticklabels([])

Picnameout10 ='/3Dplot.png'

plt.savefig(Picnameout10,dpi=600)

plt.show()



17) Contour Graphs

import os

import numpy as np

import matplotlib

import matplotlib.cm as cm

import matplotlib.pyplot as plt

def bivariate\_normal(X,Y,sigmax=1.0,sigmay=1.0,mux=0.0,muy=0.0,sigmaxy=0.0):

Xmu = X-mux

Ymu = Y-muy

rho = sigmaxy/(sigmax\*sigmay)

z = Xmu\*\*2/sigmax\*\*2 + Ymu\*\*2/sigmay\*\*2 - 2\*rho\*Xmu\*Ymu/(sigmax\*sigmay)

denom = 2\*np.pi\*sigmax\*sigmay\*np.sqrt(1-rho\*\*2)

return np.exp(-z/(2\*(1-rho\*\*2)))/denom

delta = 0.025

x = np.arange(-3.0,3.0,delta)

y = np.arange(-2.0,2.0,delta)

X,Y = np.meshgrid(x,y)

Z1 = bivariate\_normal(X,Y,1.0,1.0,0.0,0.0)

Z2 = bivariate\_normal(X,Y,1.5,0.5,1,1)

Z = 10.0\*(Z2-Z1)

plt.figure(figsize=(10,10))

CS = plt.contour(X, Y, Z)

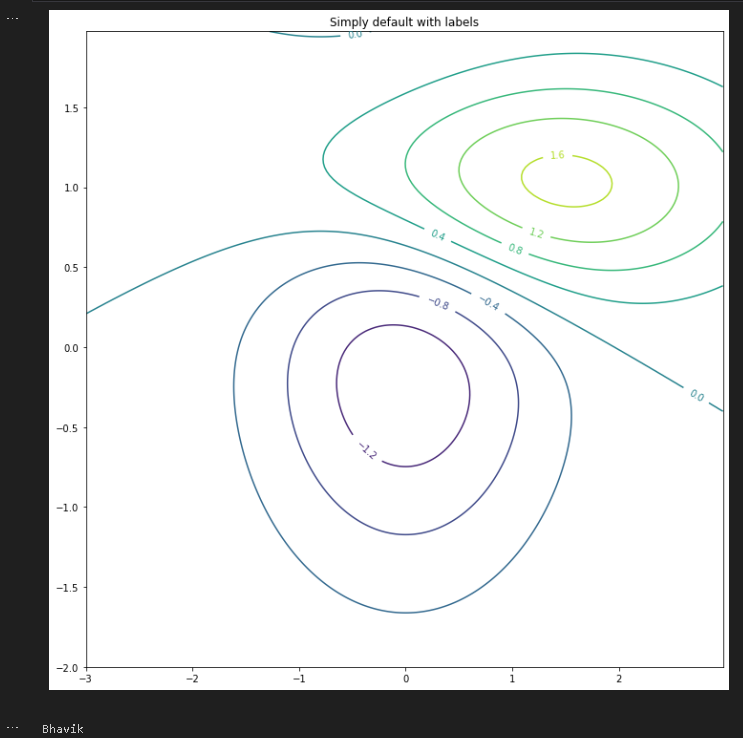
plt.clabel(CS, inline=1, fontsize=10)

plt.title('Simply default with labels')

plt.tight\_layout()

plt.show()

print("Bhavik")

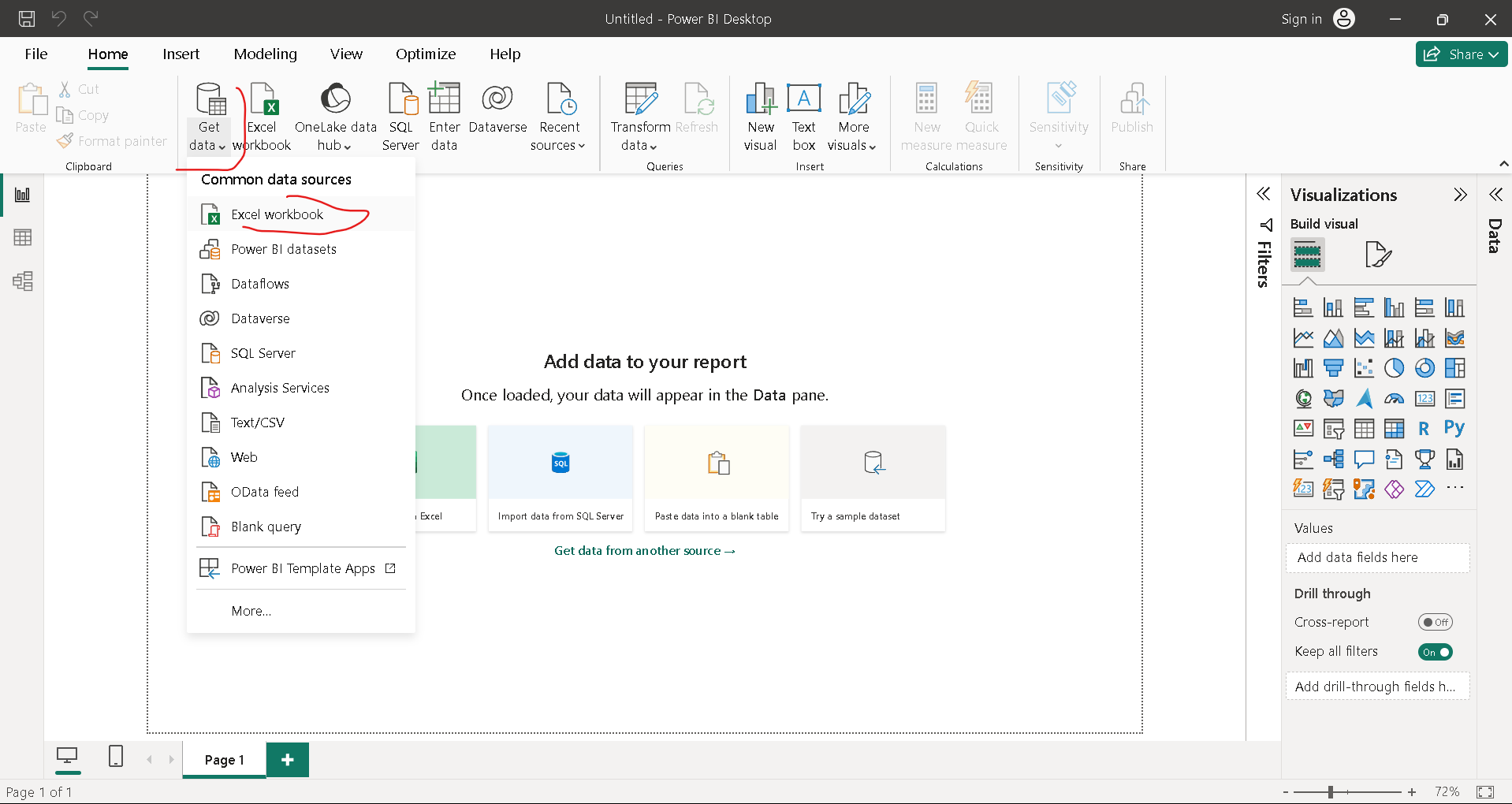


Practical 9: Data Visualization with Power BI

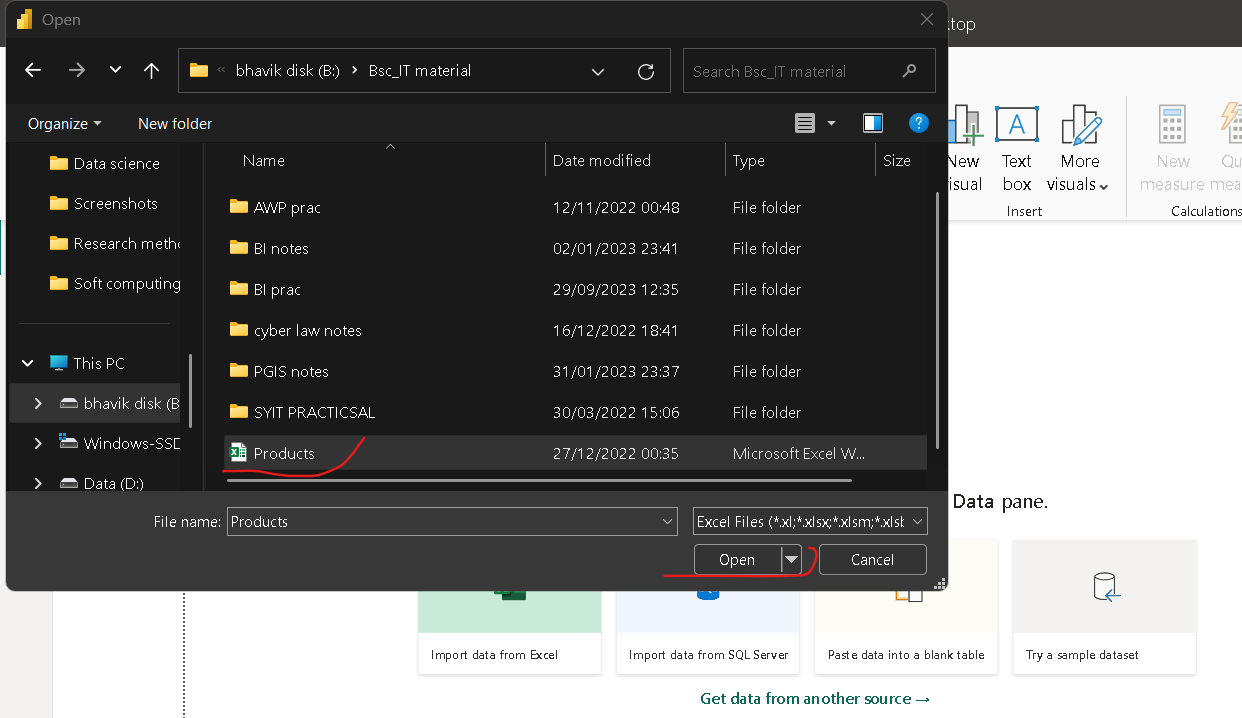
power BI is a Data Visualization and Business Intelligence tool that converts data from different data sources to interactive dashboards and BI reports. Power BI suite provides multiple software, connector, and services - Power BI desktop, Power BI service based on Saas, and mobile Power BI apps available for different platforms. These set of services are used by business users to consume data and build BI reports. Power BI desktop app is used to create reports, while Power BI Services (Software as a Service - SaaS) is used to publish the reports, and Power BI mobile app is used to view the reports and dashboards. Visualizations are used to effectively present your data and are the basic building blocks of any Business Intelligence tool. Power BI contains various default data visualization components that include simple bar charts to pie charts to maps, and also complex models such as waterfalls, funnels, gauges, and many other components. In Power BI, you can create visualization in two ways. First is by adding from the right side pane to Report Canvas. By default, it is the table type visualization, which is selected in Power BI. Another way is to drag the fields from right side bar to the axis and value axis under Visualization. You can add multiple fields to each axis as per the requirement.

A. Data Visualization with Power BI

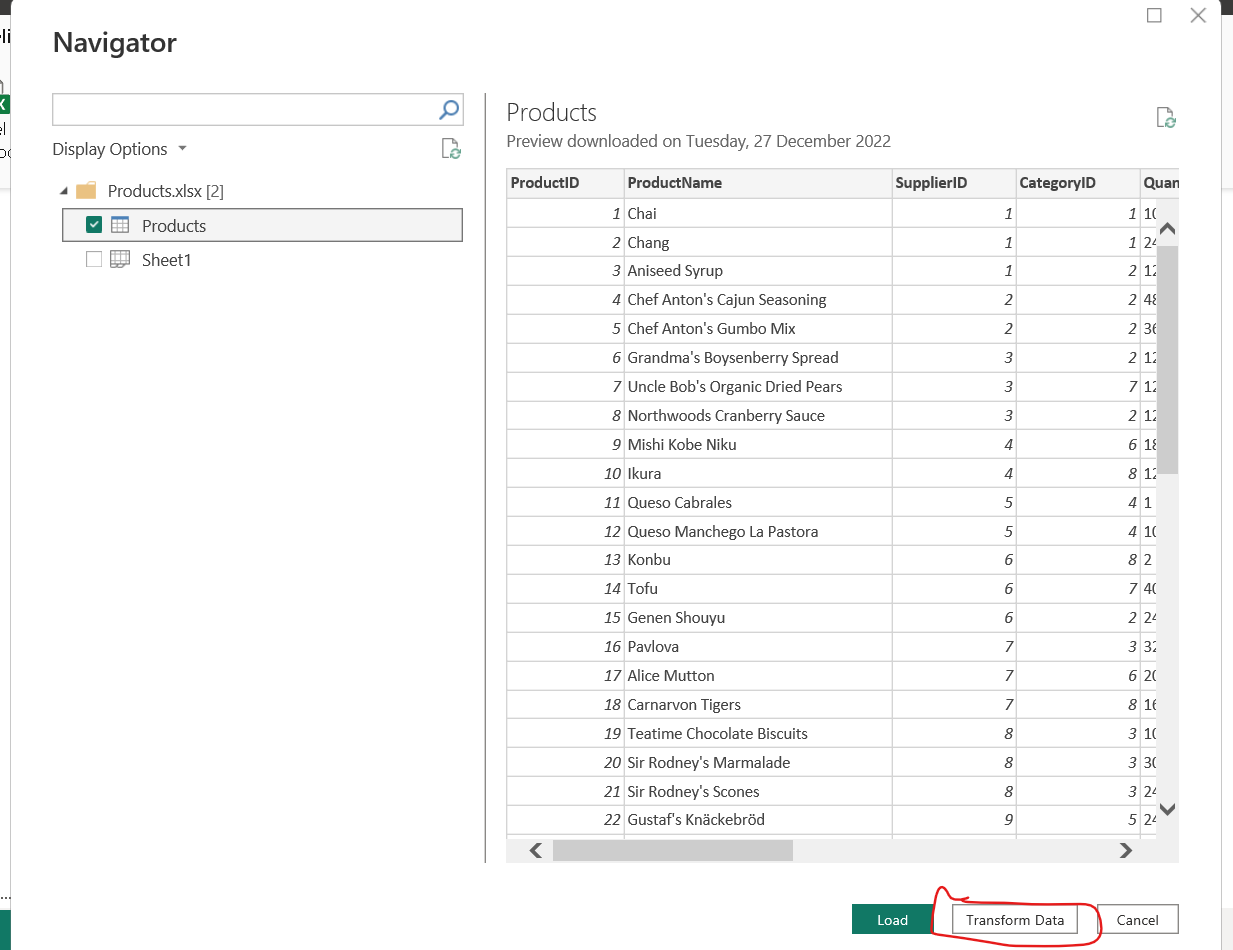
Task 1: Import Products Data from Excel Workbook Step 1: Connect to an Excel workbook 1. Launch Power BI Desktop 2. From the Home ribbon, select Get Data. Click on Excel Workbook.



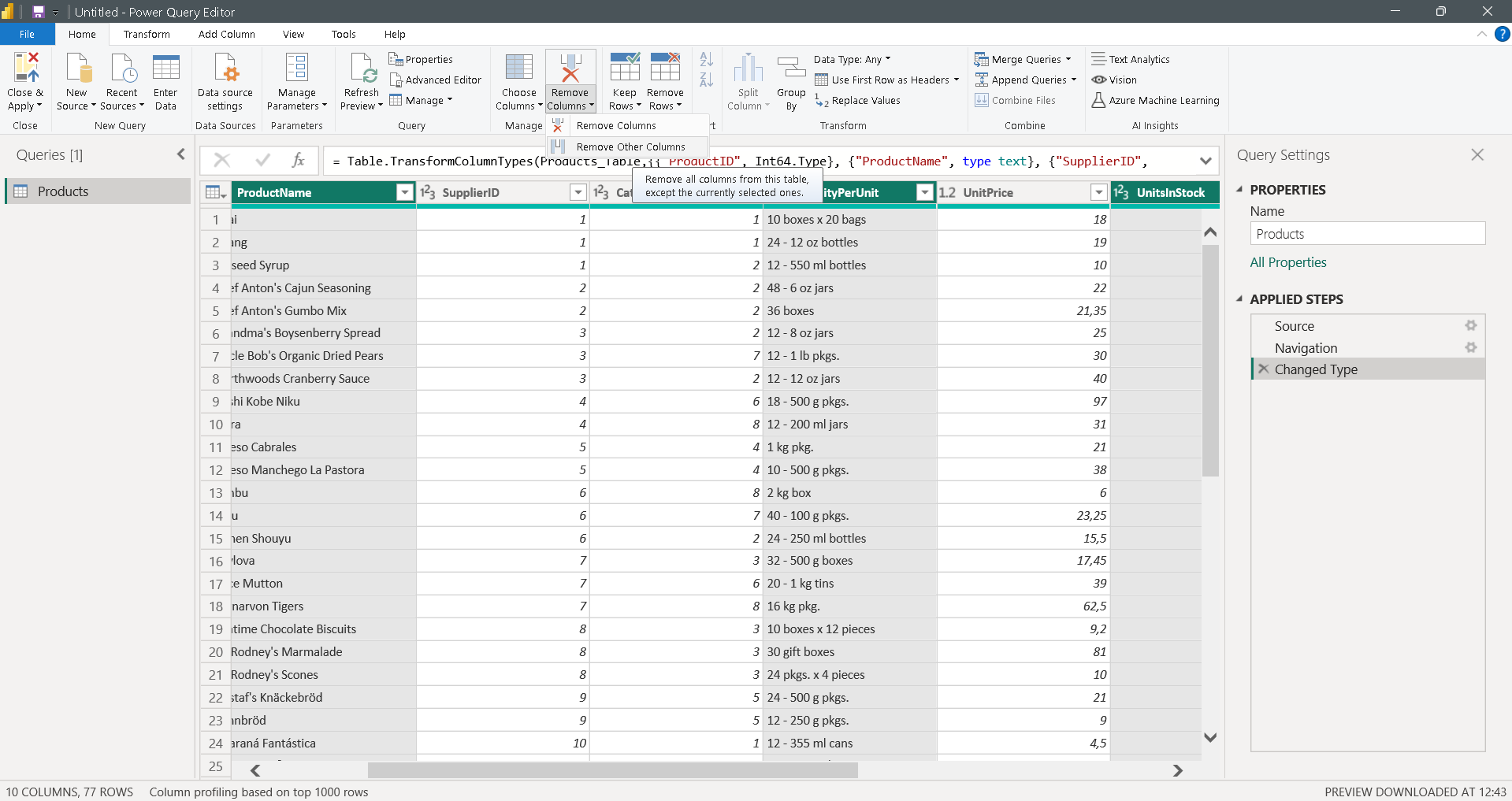
3. In the Open File dialog box, select the Products.xlsx file click on open

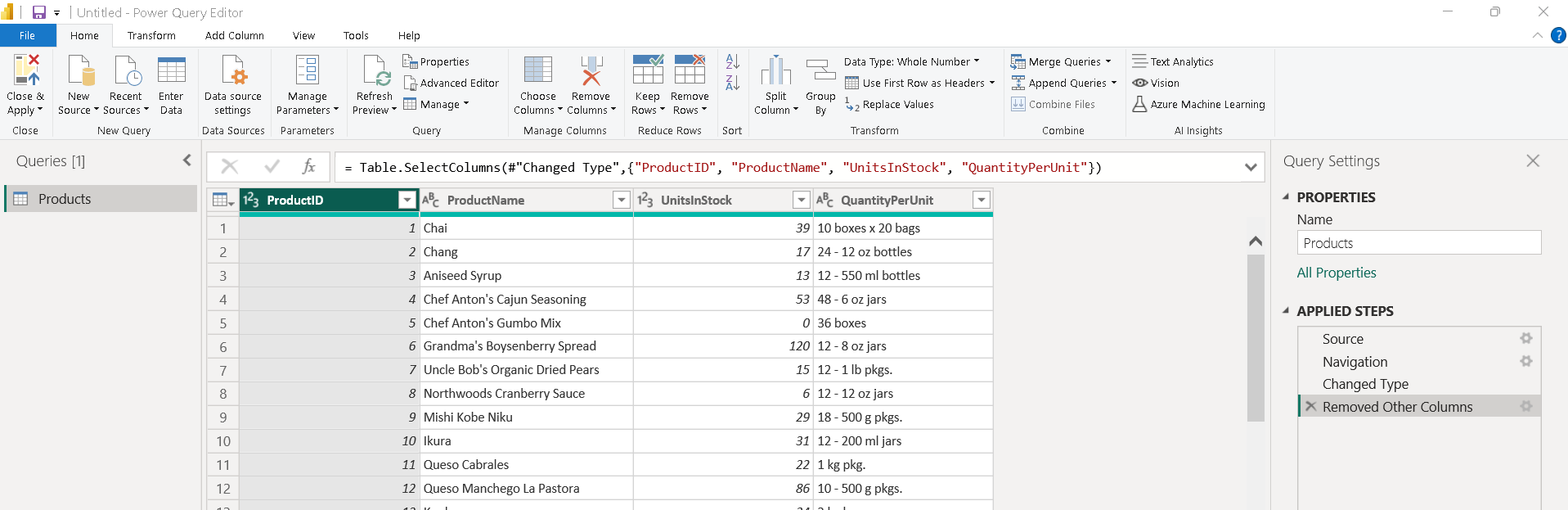


4. Navigator Dialog box will open, Select Product table and Click on Transform Data

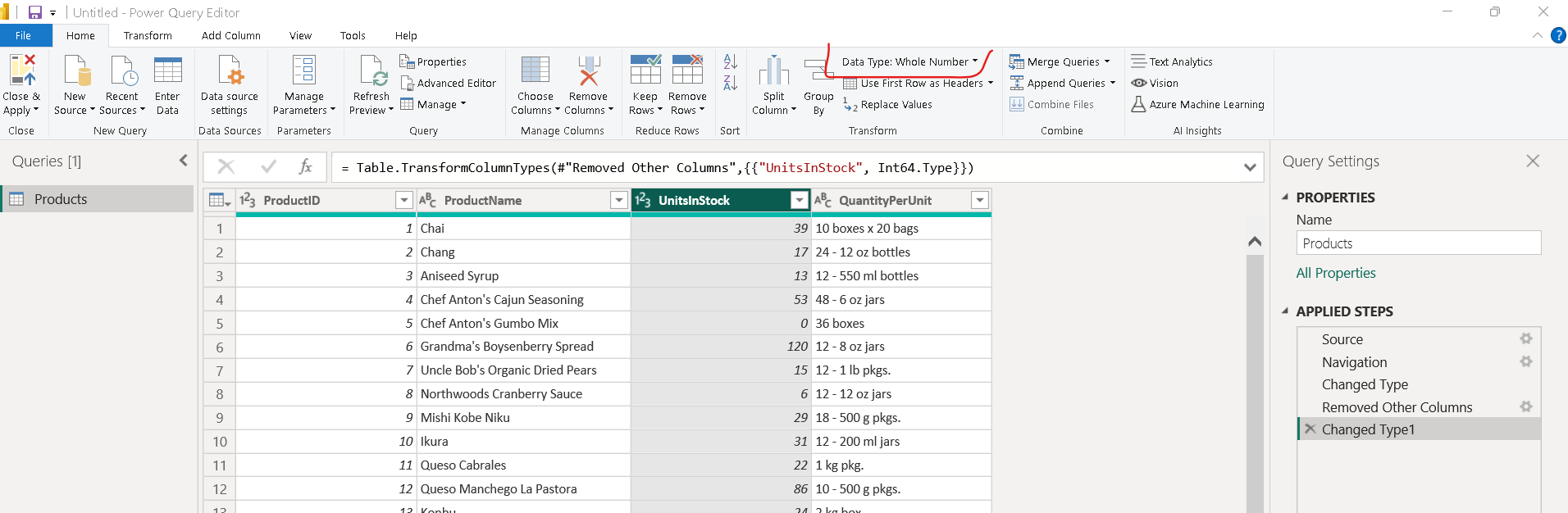


5. In Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns. (Use Ctrl+ Click to select more than one columns , or Shift+Click to select columns that are beside each other). . Select Remove Columns -> Remove Other Columns from the ribbon, or right-click on a columns header and click Remove Other Columns.

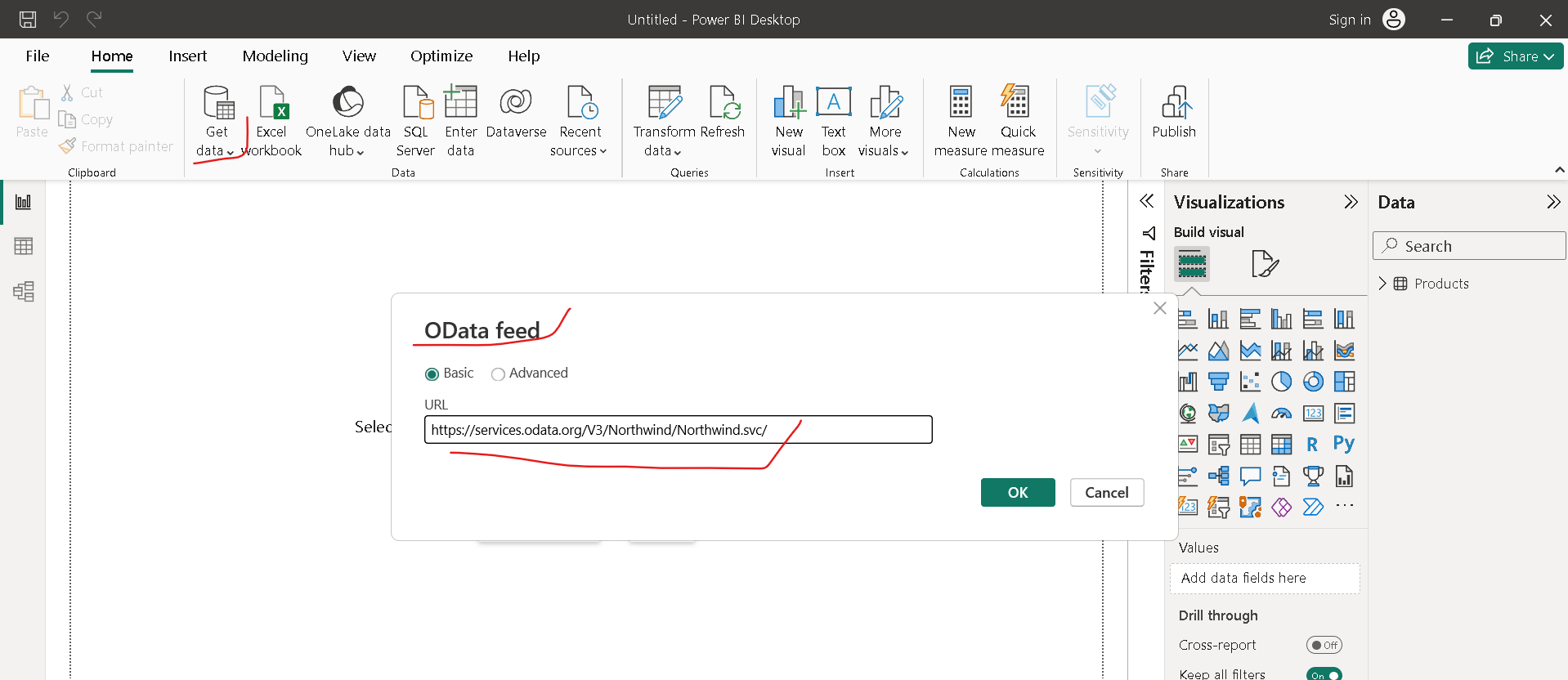




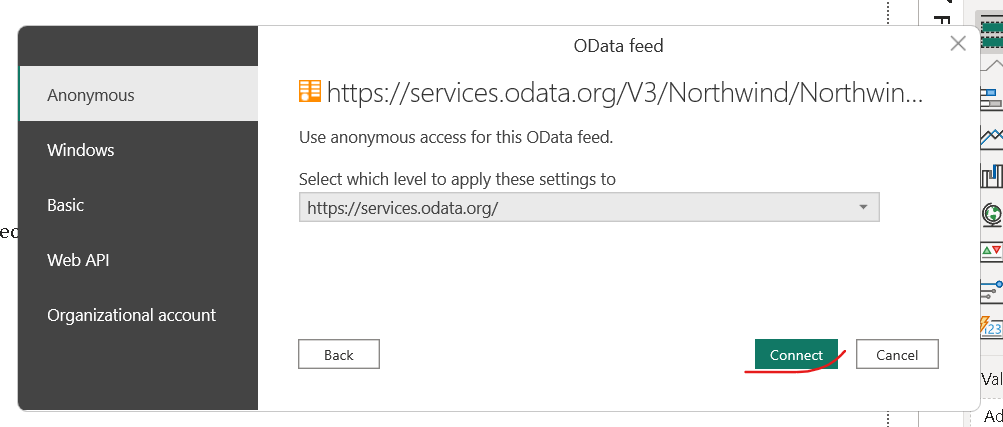
6. Change the data type of the UnitsInStock column. Select UnitsInStock Column > select Data Type drop-down button > Select Whole Number if not already selected. Click on Close and Apply button.



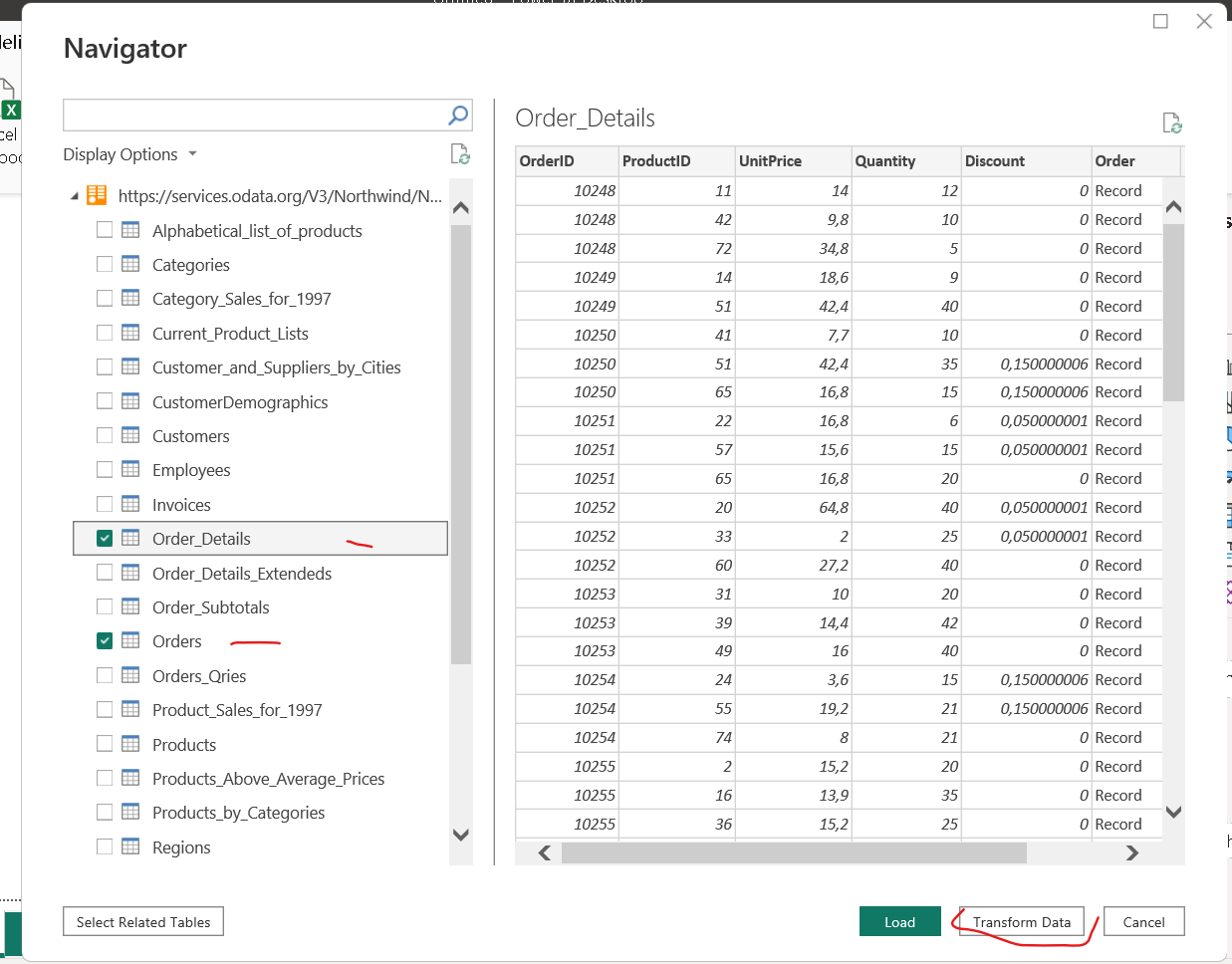
Task 2: Import order data from an OData feed 1. Click on Get Data > Select OData Feed > In OData Feed dialog box paste the link of Northwind OData Feed – ‘http://services.odata.org/V3/Northwind/Northwind.svc/’



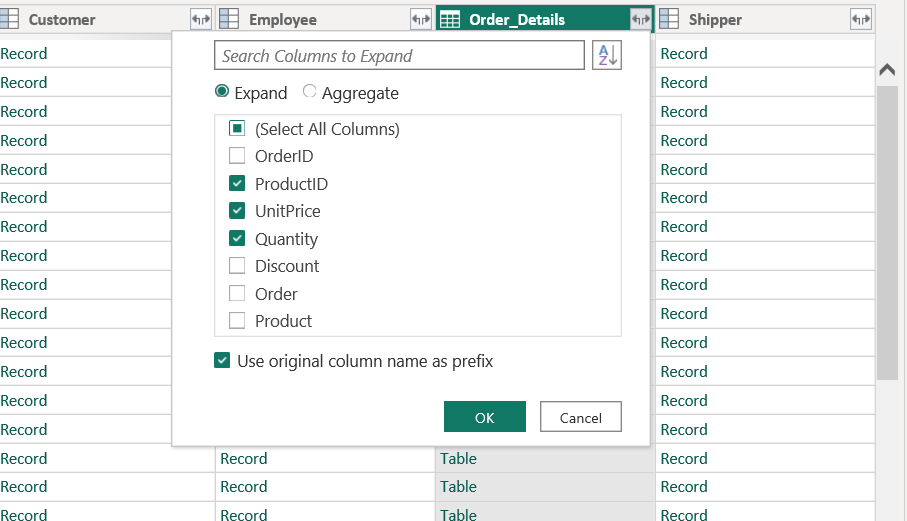
2. Select Anonymous and click on Connect



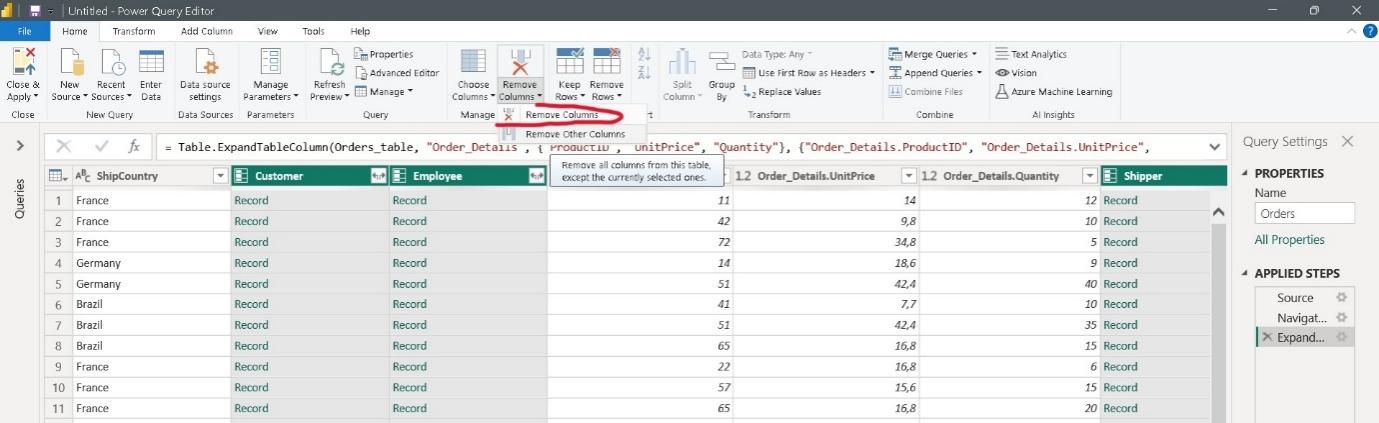
3. Select Order table and Click on Transform Data

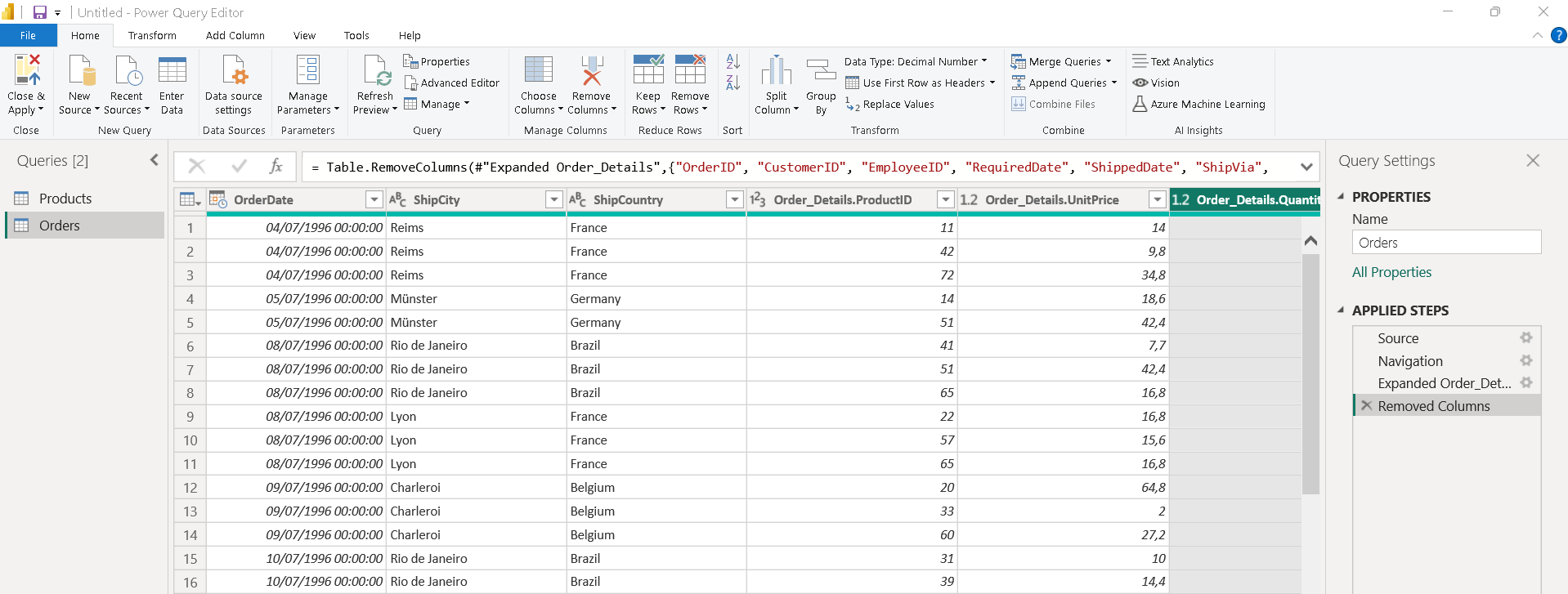


4. In the Query View, scroll to the Order\_Details column. In the Order\_Details column, select the expand icon. In the Expand drop-down, DeSelect (Select All Columns) to clear all columns. Select ProductID, UnitPrice, and Quantity. Click OK. The Expand operation combines columns from a related table into a subject table. When the query runs, rows from the related table (Order\_Details) are combined into rows of Order table.

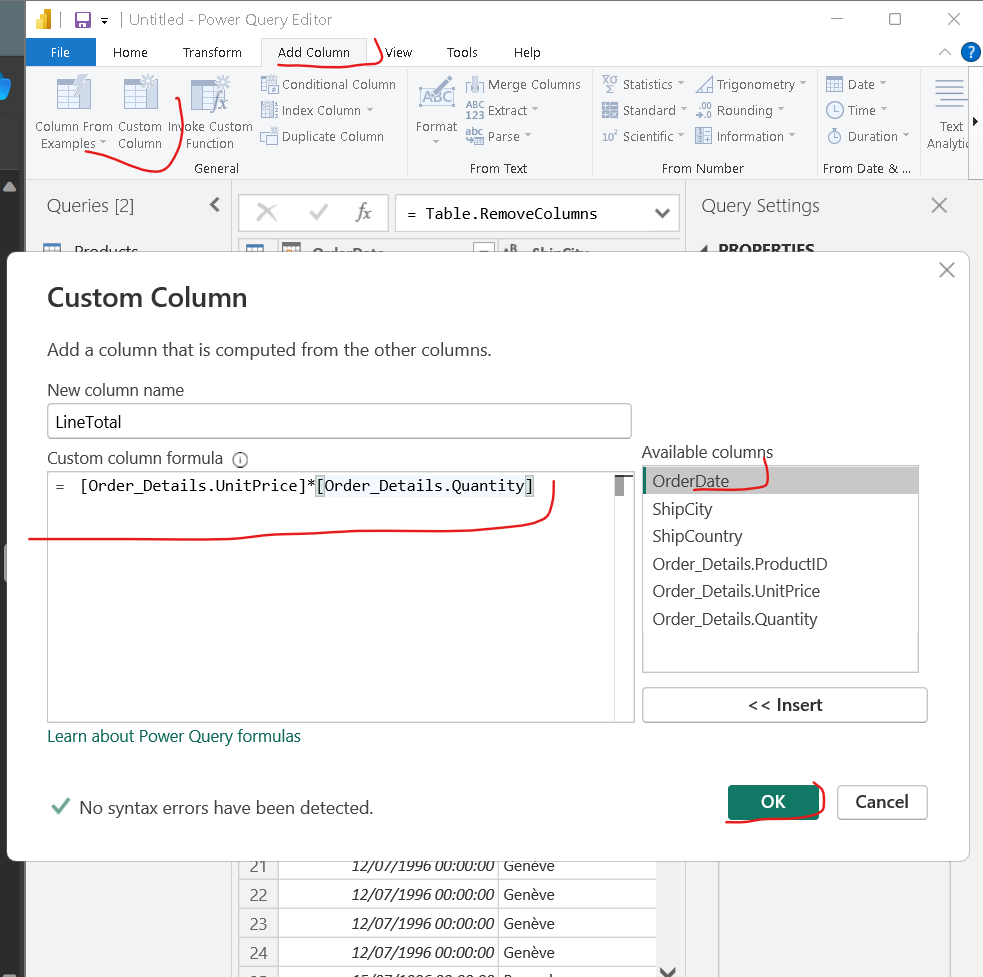


5. Click the first column (OrderID) and Shift+Click the last column (Shipper). Now that all columns are selected, use Ctrl+Click to unselect the following columns: OrderDate, ShipCity, ShipCountry, Order\_Details.ProductID, Order\_Details.UnitPrice, and Order\_Details.Quantity. Slect Remove Column and click on Remove Columns.

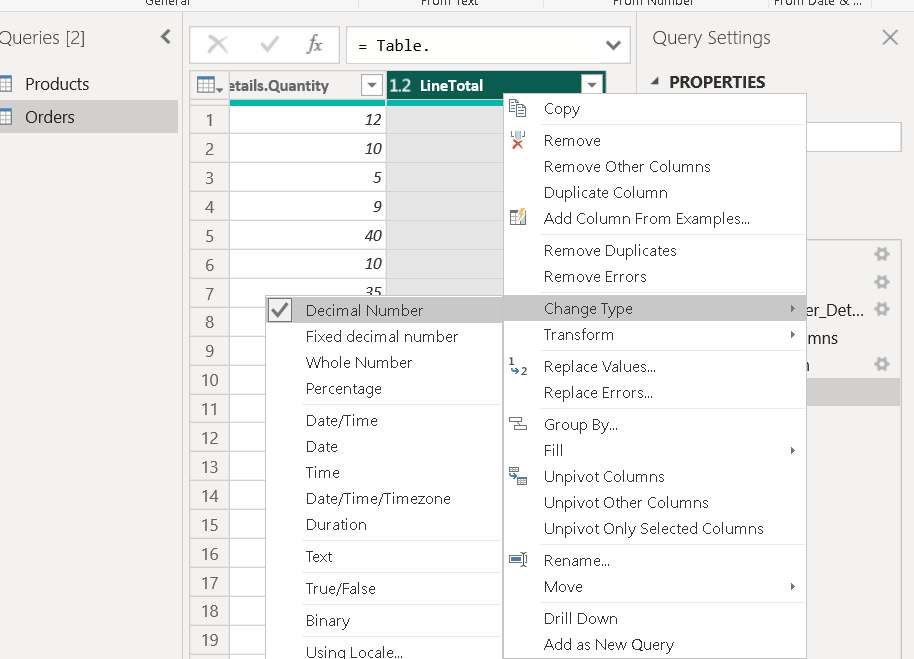




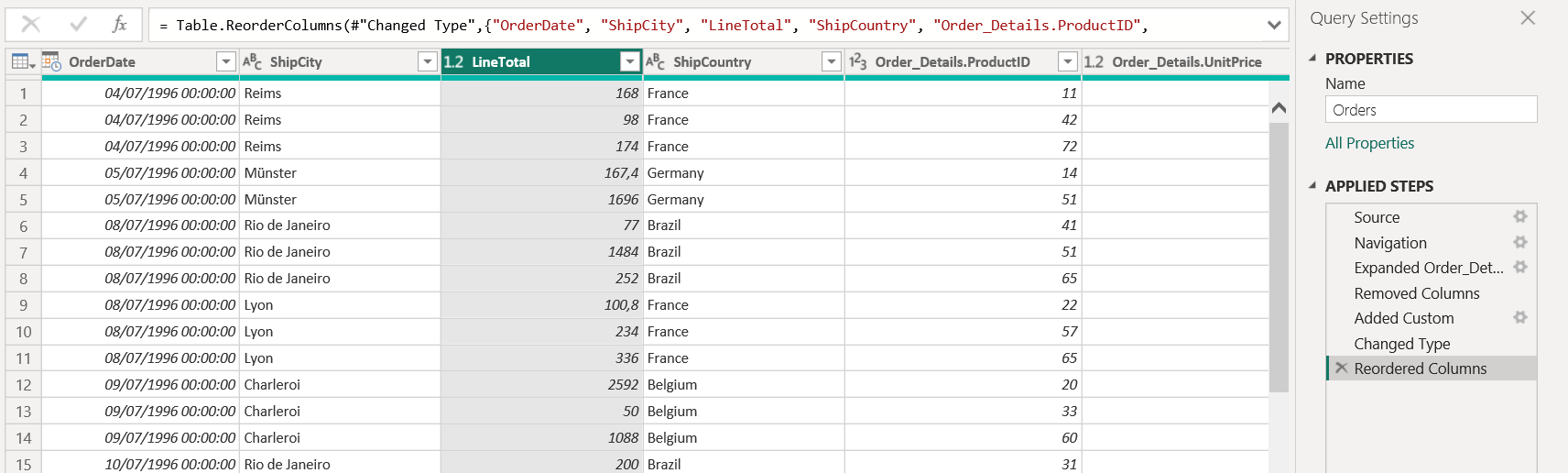
6. In the Add Column ribbon tab, click Add Custom Column. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order\_Details.UnitPrice] \* [Order\_Details.Quantity]. In the New column name textbox, enter LineTotal.

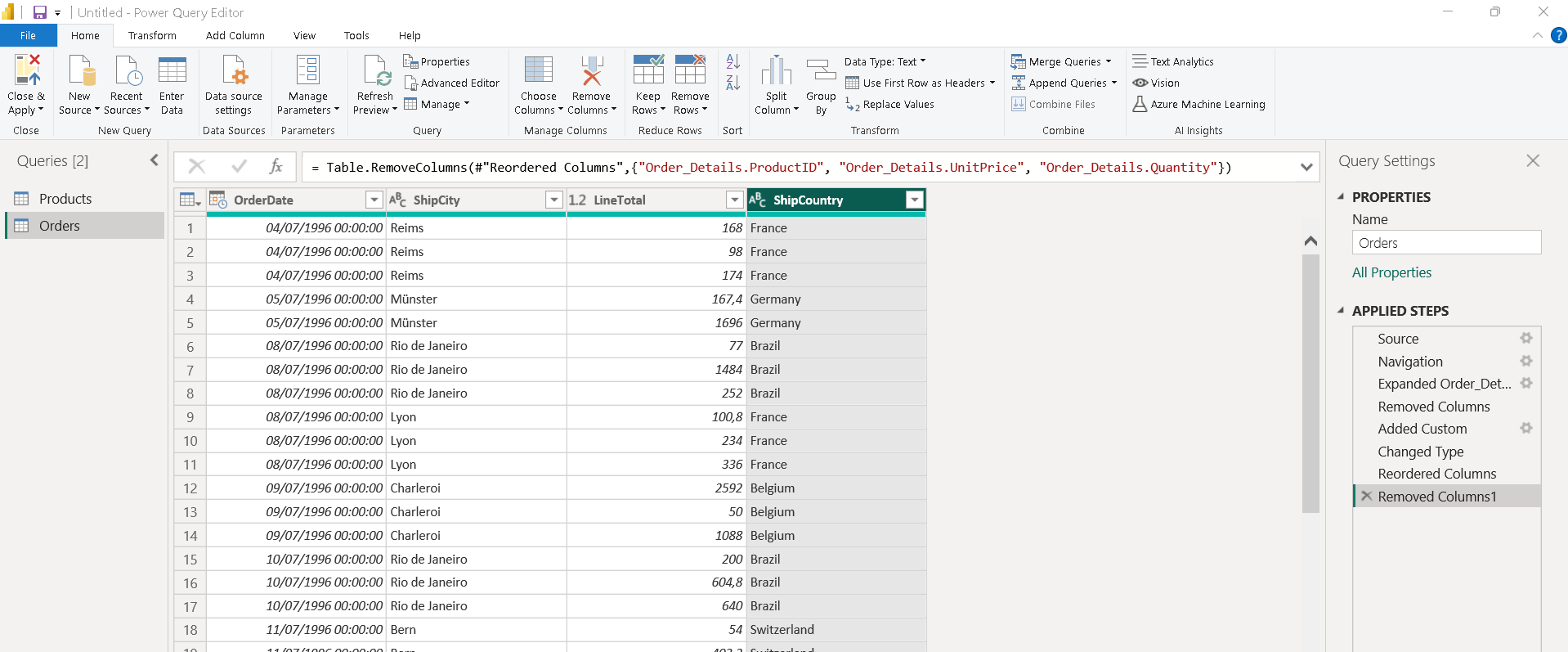


7. Change Data Type of LineTotal Column to Decimal Number.

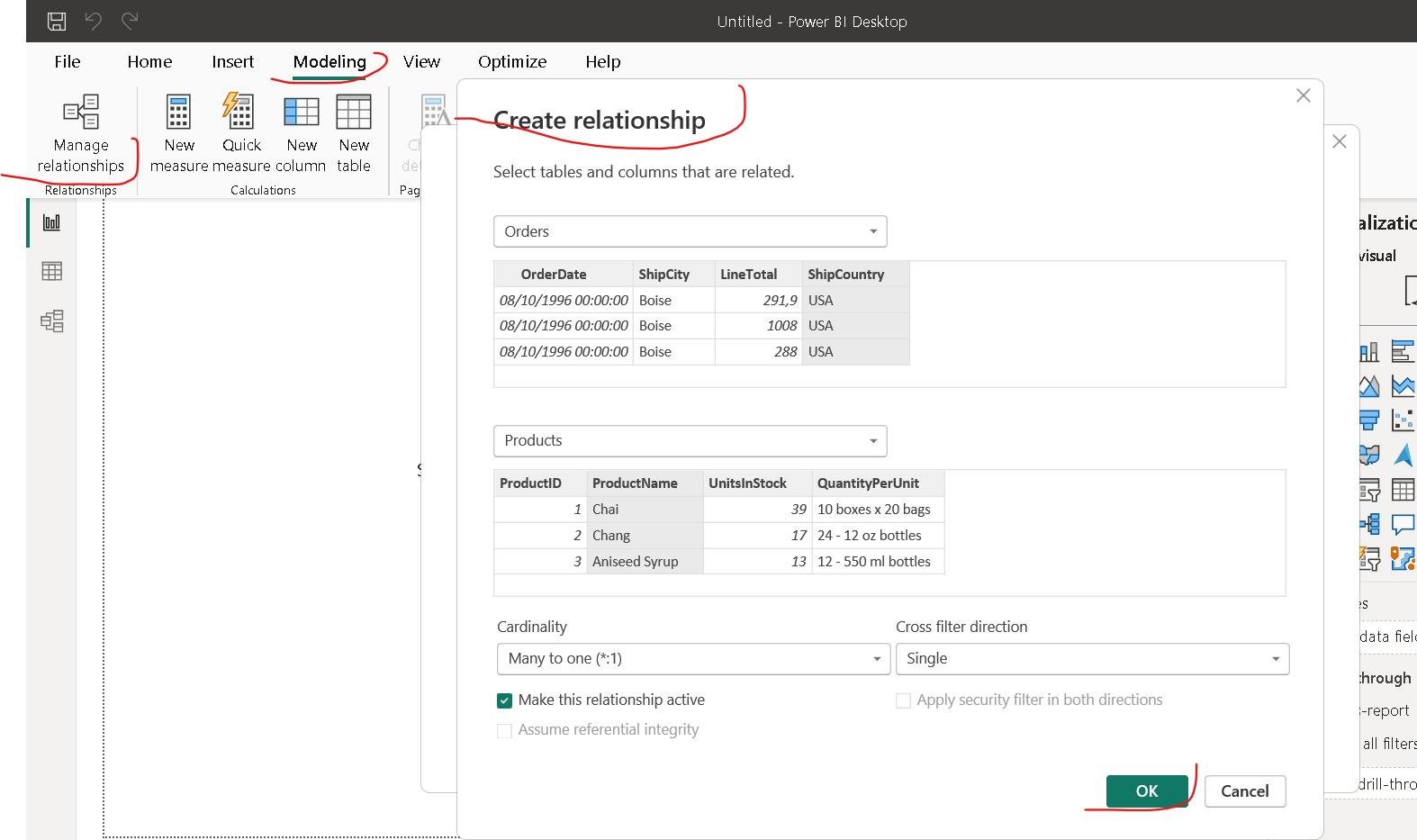


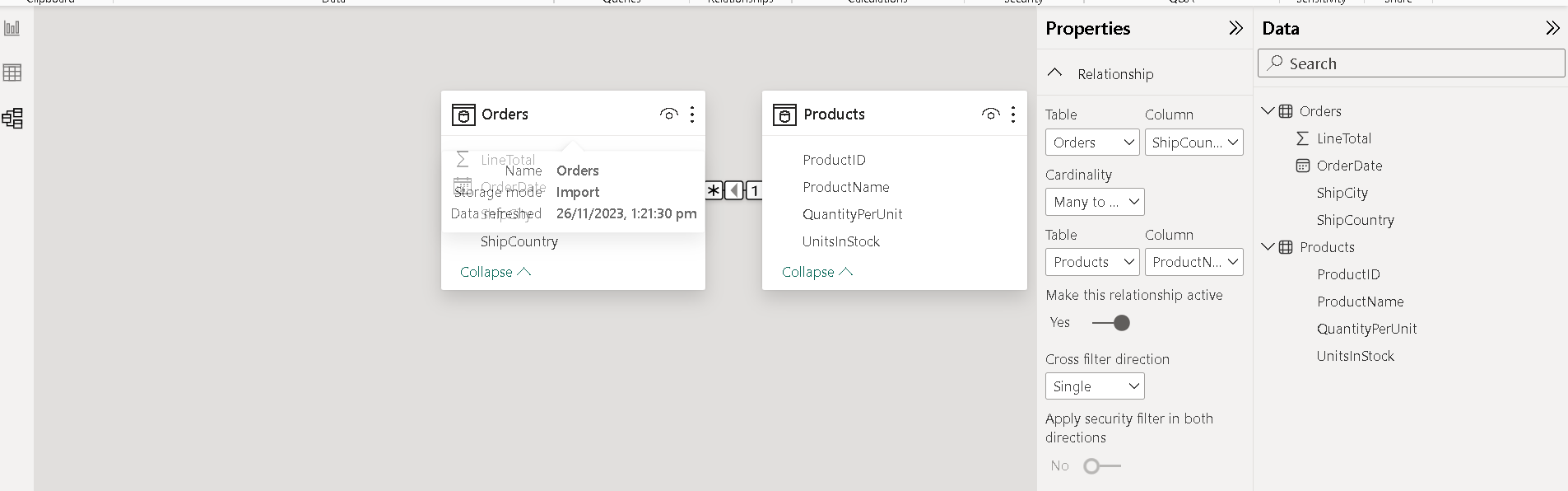
8.In Query Editor, drag the LineTotal column to the left, after ShipCountry. Remove the Order\_Details from ProductID, UnitPrice and Quantity columns. Click on Close and Apply button.



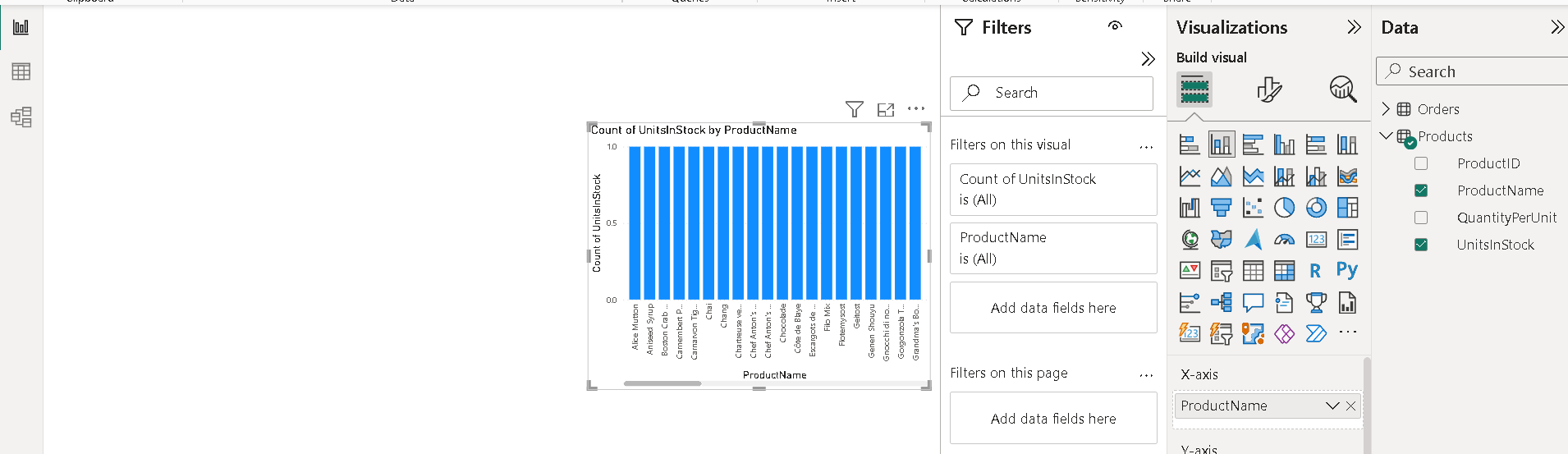


Task 3: Combine the Products and Total Sales queries 1. Power BI Desktop loads the data from the two queries. Once the data is loaded, select the Manage Relationships button in Home ribbon. Click on new and select Order and Products tables and we see that one already exists!

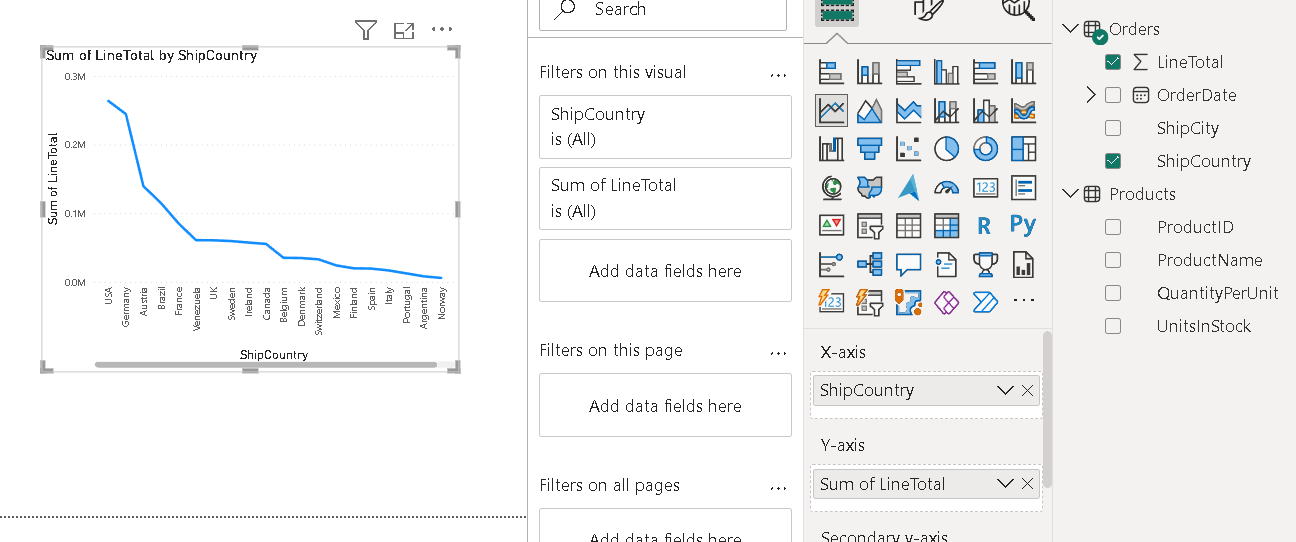




Task 4: Build visuals using your data 1. Click on Report, Drag UnitInStock to canva and Select Stacked Column Chart. Create charts showing Units in Stock by Product and Total Sales by Year. Change the X-axis and Y-axis as shown below.



2. Drag OrderDate to the canvas beneath the first chart, then drag LineTotal (again, the Fields pane) onto the visual, then select Line Chart.

****