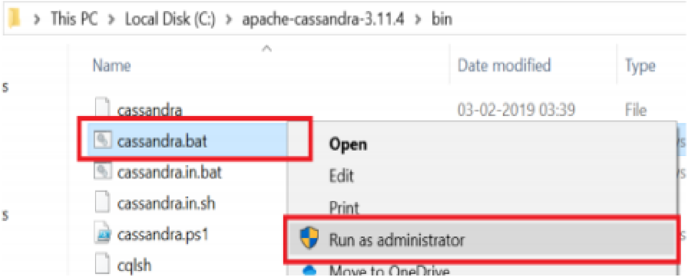
Practical 1:
Creating Data Model using Cassandra.

Go to Cassandra directory

C:\apache-cassandra-3.11.4\bin



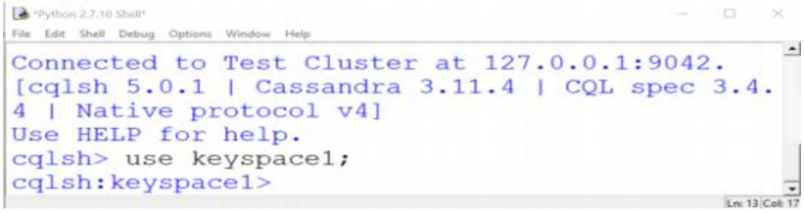
Run Cassandra.bat file

Open C:\apache-cassandra-3.11.4\bin\cqlsh.py with python 2.7 and run

Creating a Keyspace using Cqlsh

Create keyspace keyspace1 with replication = {„class‟:‟SimpleStratergy‟, „replication_factor‟: 3};

Use keyspace1;



```
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.
4 | Native protocol v4]
Use HELP for help.
cqlsh> use keyspace1;
cqlsh:keyspace1>
```

Create table dept (dept_id int PRIMARY KEY, dept_name text, dept_loc text);

Create table emp (emp_id int PRIMARY KEY, emp_name text, dept_id int, email text, phone text);

Insert into dept (dept_id, dept_name, dept_loc) values (1001, 'Accounts', 'Mumbai');

Insert into dept (dept_id, dept_name, dept_loc) values (1002, 'Marketing', 'Delhi');

Insert into dept (dept_id, dept_name, dept_loc) values (1003, 'HR', 'Chennai');

Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1001, 'ABCD', 1001,'abcd@company.com', '1122334455');

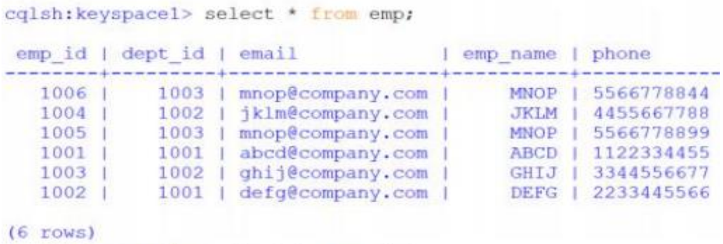
Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1002, 'DEFG', 1001,'defg@company.com', '2233445566');

Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1003, 'GHIJ', 1002,'ghij@company.com', '3344556677');

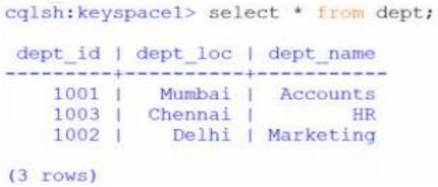
Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1004, 'JKLM', 1002,'jklm@company.com', '4455667788');

Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1005, 'MNOP', 1003,'mnop@company.com', '5566778899');

Insert into emp (emp_id, emp_name, dept_id, email, phone) values (1006, 'MNOP', 1003,'mnop@company.com', '5566778844');

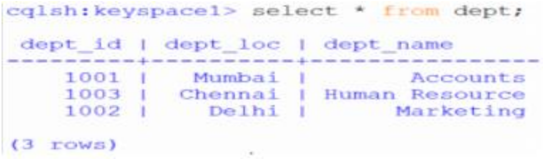


```
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email             | emp_name | phone
--------+---------+-------------------+----------+------------
   1006 |    1003 | mnop@company.com |     MNOP | 5566778844
   1004 |    1002 | jklm@company.com |     JKLM | 4455667788
   1005 |    1003 | mnop@company.com |     MNOP | 5566778899
   1001 |    1001 | abcd@company.com |     ABCD | 1122334455
   1003 |    1002 | ghij@company.com |     GHIJ | 3344556677
   1002 |    1001 | defg@company.com |     DEFG | 2233445566

(6 rows)
```

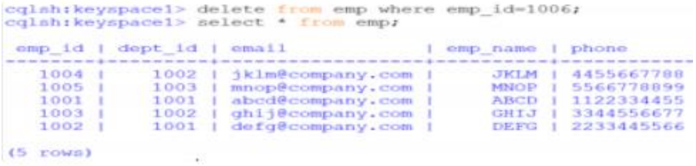


```
cqlsh:keyspace1> select * from dept;

 dept_id | dept_loc | dept_name
---------+----------+-----------
    1001 |   Mumbai |  Accounts
    1003 |  Chennai |        HR
    1002 |    Delhi | Marketing

(3 rows)
```

update dept set dept_name='Human Resource' where dept_id=1003;



```
cqlsh:keyspace1> select * from dept;

 dept_id | dept_loc | dept_name
---------+----------+----------------
    1001 |   Mumbai |       Accounts
    1003 |  Chennai | Human Resource
    1002 |    Delhi |      Marketing

(3 rows)
```



```
cqlsh:keyspace1> delete from emp where emp_id=1006;
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email             | emp_name | phone
--------+---------+-------------------+----------+------------
   1004 |    1002 | jklm@company.com |     JKLM | 4455667788
   1005 |    1003 | mnop@company.com |     MNOP | 5566778899
   1001 |    1001 | abcd@company.com |     ABCD | 1122334455
   1003 |    1002 | ghij@company.com |     GHIJ | 3344556677
   1002 |    1001 | defg@company.com |     DEFG | 2233445566

(5 rows)
```

Practical 2:

Write Python / R Program to convert from the following formats to HORUS format:

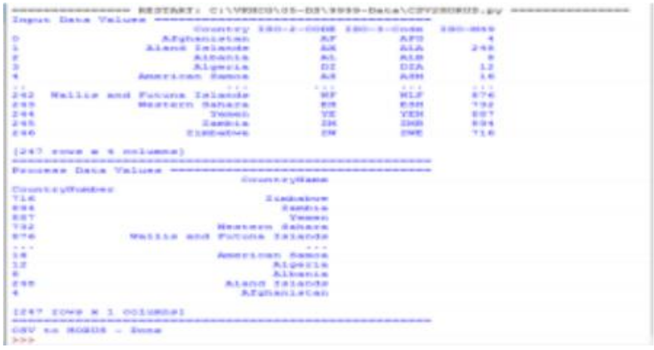
A. Text delimited CSVto HORUS format.

Code

```
# Utility Start CSV to HORUS =================================
# Standard Tools
import pandas as pd
# Input Agreement ============================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ===========================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =================================')
print(ProcessData)
print('=====================================================')
```

# Output Agreement ===

```
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
# Utility done ===============================================
```



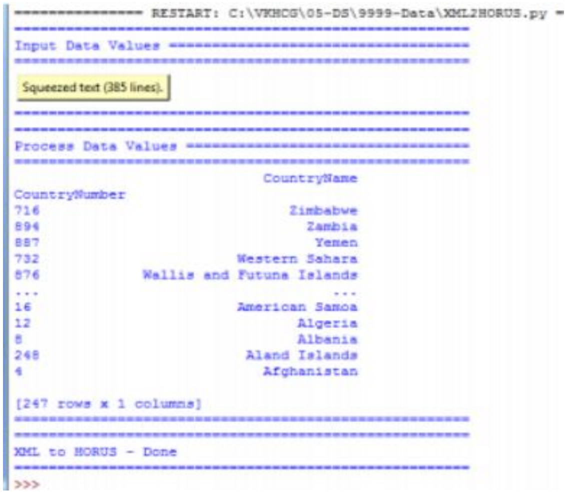
```
=============== RESTART: C:\VKHCG\05-DS\9999-Data\CSV2HORUS.py ===============
Input Data Values ===================================
                       Country ISO-2-CODE ISO-3-Code  ISO-M49
0                  Afghanistan         AF        AFG        4
1                Aland Islands         AX        ALA      248
2                      Albania         AL        ALB        8
3                      Algeria         DZ        DZA       12
4               American Samoa         AS        ASM       16
..                         ...        ...        ...      ...
242  Wallis and Futuna Islands         WF        WLF      876
243             Western Sahara         EH        ESH      732
244                      Yemen         YE        YEM      887
245                     Zambia         ZM        ZMB      894
246                   Zimbabwe         ZW        ZWE      716

[247 rows x 4 columns]
=====================================================
Process Data Values =================================
                             CountryName
CountryNumber
716                             Zimbabwe
894                               Zambia
887                                Yemen
732                       Western Sahara
876            Wallis and Futuna Islands
...                                  ...
16                        American Samoa
12                               Algeria
8                                Albania
248                        Aland Islands
4                            Afghanistan

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

B. XML to HORUS Format

Code :-

```
# Utility Start XML to HORUS ===============================
# Standard Tools
import pandas as pd
import xml.etree.ElementTree as ET
def df2xml(data):
header = data.columns
root = ET.Element('root')
for row in range(data.shape[0]):
entry = ET.SubElement(root,'entry')
for index in range(data.shape[1]):
schild=str(header[index])
child = ET.SubElement(entry, schild)
if str(data[schild][row]) != 'nan':
child.text = str(data[schild][row])
else:
child.text = 'n/a'
entry.append(child)
result = ET.tostring(root)
return result
def xml2df(xml_data):
root = ET.XML(xml_data)
all_records = []
for i, child in enumerate(root):
record = {}
for subchild in child:
record[subchild.tag] = subchild.text
all_records.append(record)
return pd.DataFrame(all_records)
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('==================================================')
print('Input Data Values =================================')
print('==================================================')
print(InputData)
print('==================================================')
#==========================================================
# Processing Rules ========================================
#==========================================================
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
```

```
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('=======================================================')
print('Process Data Values ===================================')
print('=======================================================')
print(ProcessData)
print('=======================================================')
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=======================================================')
print('XML to HORUS - Done')
print('=======================================================')
# Utility done ===============================================
```



```
============== RESTART: C:\VKHCG\05-DS\9999-Data\XML2HORUS.py =
==================================================
Input Data Values ================================
==================================================
Squeezed text (385 lines).
==================================================
==================================================
Process Data Values ==============================
==================================================
                              CountryName
CountryNumber
716                              Zimbabwe
894                                Zambia
887                                 Yemen
732                        Western Sahara
876             Wallis and Futuna Islands
...                                   ...
16                         American Samoa
12                                Algeria
8                                 Albania
248                         Aland Islands
4                             Afghanistan

[247 rows x 1 columns]
==================================================
==================================================
XML to HORUS - Done
==================================================
>>>
```

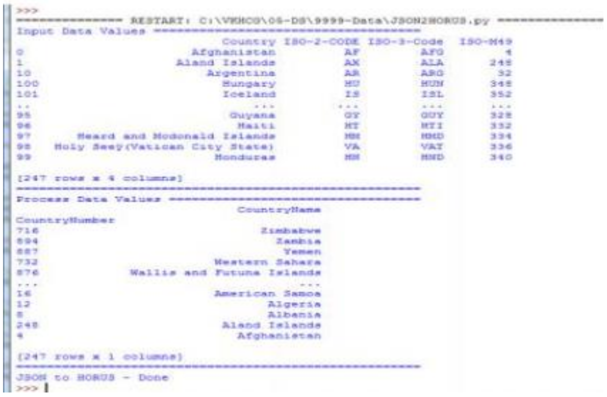
C. JSON to HORUS Format

Code:

```
# Utility Start JSON to HORUS =================================
# Standard Tools
#=============================================================
import pandas as pd
# Input Agreement ============================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.json'
InputData=pd.read_json(sInputFileName, orient='index', encoding="latin-1")
print('Input Data Values ===================================')
print(InputData)
print('=========================================================')
# Processing Rules ==============================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values ===================================')
print(ProcessData)
print('=========================================================')
```

# Output Agreement ==

```
OutputData=ProcessData
sOutputFileName='c:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
# Utility done ===============================================
```



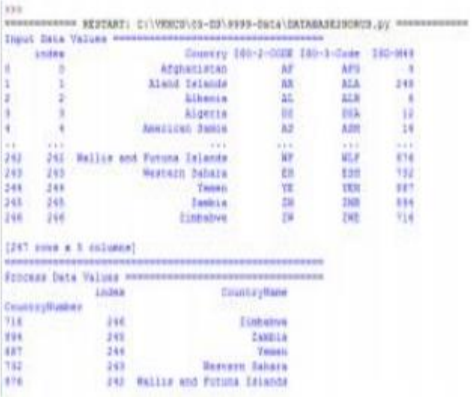
```
>>>
=============== RESTART: C:\VKHCG\05-DS\9999-Data\JSON2HORUS.py ==============
Input Data Values ======================================
                          Country ISO-2-CODE ISO-3-Code  ISO-M49
0                     Afghanistan         AF        AFG        4
1                   Aland Islands         AX        ALA      248
10                      Argentina         AR        ARG       32
100                       Hungary         HU        HUN      348
101                       Iceland         IS        ISL      352
..                            ...        ...        ...      ...
95                         Guyana         GY        GUY      328
96                          Haiti         HT        HTI      332
97     Heard and Mcdonald Islands         HM        HMD      334
98   Holy See(Vatican City State)         VA        VAT      336
99                       Honduras         HN        HND      340

[247 rows x 4 columns]
=========================================================
Process Data Values ====================================
                                 CountryName
CountryNumber
716                                 Zimbabwe
894                                   Zambia
887                                    Yemen
732                           Western Sahara
876                Wallis and Futuna Islands
...                                      ...
16                            American Samoa
12                                   Algeria
8                                    Albania
248                            Aland Islands
4                                Afghanistan

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

D. MySql Database to HORUS Format

Code:

```
# Utility Start Database to HORUS =================================
# Standard Tools
#==============================================================
import pandas as pd
import sqlite3 as sq
# Input Agreement ============================================
sInputFileName='C:/VKHCG/05-DS/9999-Data/utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
```

```
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)
print('Input Data Values ===================================')
print(InputData)
print('=====================================================')
# Processing Rules ===========================================
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =================================')
print(ProcessData)
print('=====================================================')
# Output Agreement ===========================================
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
# Utility done ===============================================
```



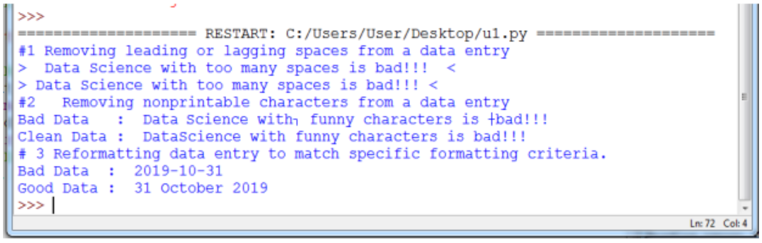
```
>>>
============ RESTART: C:\VKHCG\05-DS\9999-Data\DATABASE2HORUS.py ============
Input Data Values ===================================
     index                    Country ISO-2-CODE ISO-3-Code  ISO-M49
0        0                Afghanistan         AF        AFG        4
1        1              Aland Islands         AX        ALA      248
2        2                    Albania         AL        ALB        8
3        3                    Algeria         DZ        DZA       12
4        4             American Samoa         AS        ASM       16
..     ...                        ...        ...        ...      ...
242    242  Wallis and Futuna Islands         WF        WLF      876
243    243             Western Sahara         EH        ESH      732
244    244                      Yemen         YE        YEM      887
245    245                     Zambia         ZM        ZMB      894
246    246                   Zimbabwe         ZW        ZWE      716

[247 rows x 5 columns]
=====================================================
Process Data Values =================================
               index                CountryName
CountryNumber
716              246                   Zimbabwe
894              245                     ZAMBIA
887              244                      Yemen
732              243             Western Sahara
876              242  Wallis and Futuna Islands
```

Practical 3
Utilities and Auditing

A. Fixers Utilities:

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

```
#---------------------------- Program to Demonstrate Fixers utilities -------------------
import string
import datetime as dt
# 1 Removing leading or lagging spaces from a data entry
print('#1 Removing leading or lagging spaces from a data entry');
baddata = " Data Science with too many spaces is bad!!! "
print('>',baddata,'<')
cleandata=baddata.strip()
print('>',cleandata,'<')
# 2 Removing nonprintable characters from a data entry
print('#2 Removing nonprintable characters from a data entry')
printable = set(string.printable)
baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"
cleandata=''.join(filter(lambda x: x in string.printable,baddata))
print('Bad Data : ',baddata);
print('Clean Data : ',cleandata)
# 3 Reformatting data entry to match specific formatting criteria.
# Convert YYYY/MM/DD to DD Month YYYY
print('# 3 Reformatting data entry to match specific formatting criteria.')
baddate = dt.date(2019, 10, 31)
baddata=format(baddate,'%Y-%m-%d')
gooddate = dt.datetime.strptime(baddata,'%Y-%m-%d')
gooddata=format(gooddate,'%d %B %Y')
print('Bad Data : ',baddata)
print('Good Data : ',gooddata)
```



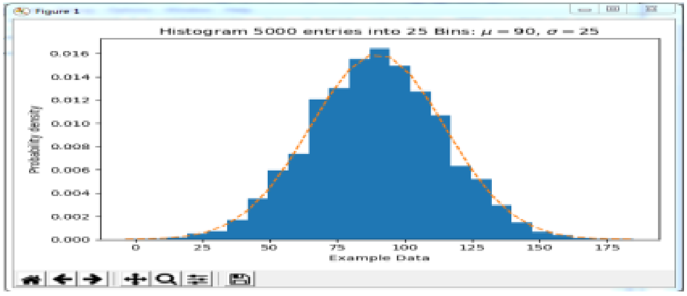
```
>>>
==================== RESTART: C:/Users/User/Desktop/u1.py ====================
#1 Removing leading or lagging spaces from a data entry
>  Data Science with too many spaces is bad!!!  <
> Data Science with too many spaces is bad!!! <
#2   Removing nonprintable characters from a data entry
Bad Data   :  Data Science with┐ funny characters is ┼bad!!!
Clean Data :  DataScience with funny characters is bad!!!
# 3 Reformatting data entry to match specific formatting criteria.
Bad Data  :  2019-10-31
Good Data :  31 October 2019
>>> |
```

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



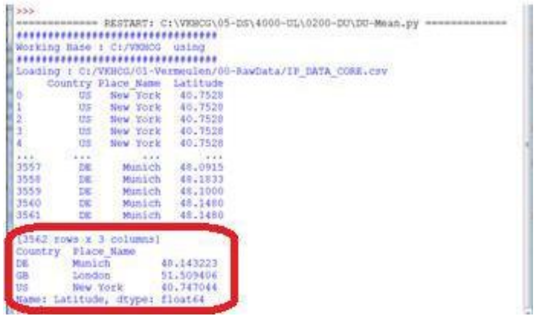
C. Averaging of Data

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

C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Mean.py

Code:

```
import pandas as pd
#######################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('###################################')
print('Working Base :',Base, ' using ')
print('##################################')
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
AllData=IP_DATA_ALL[['Country', 'Place_Name','Latitude']]
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
print(MeanData)
#######################################################################
```



```
>>>
============= RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Mean.py =============
##################################
Working Base : C:/VKHCG  using
##################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
     Country Place_Name  Latitude
0         US   New York   40.7528
1         US   New York   40.7528
2         US   New York   40.7528
3         US   New York   40.7528
4         US   New York   40.7528
...      ...        ...       ...
3557      DE     Munich   48.0915
3558      DE     Munich   48.1833
3559      DE     Munich   48.1000
3560      DE     Munich   48.1480
3561      DE     Munich   48.1480

[3562 rows x 3 columns]
Country  Place_Name
DE       Munich        48.143223
GB       London        51.509406
US       New York      40.747044
Name: Latitude, dtype: float64
```

Outlier Detection

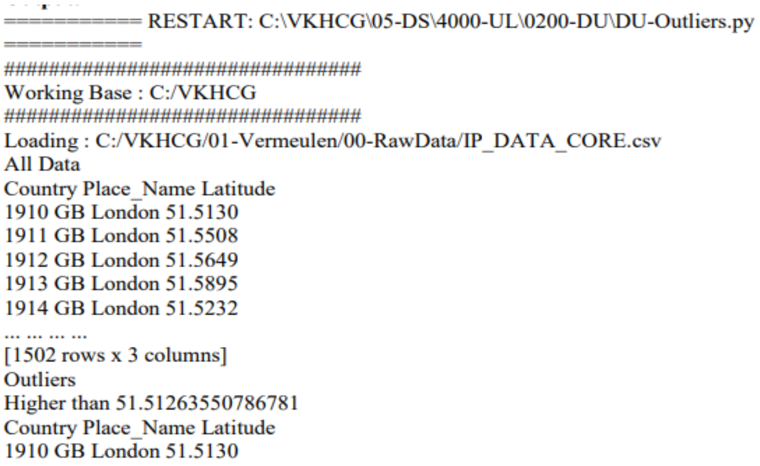
Outliers are data that is so different from the rest of the data in the data set that it may be caused by an error in the data source. There is a technique called outlier detection that, with good data science, will identify these outliers.

C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py

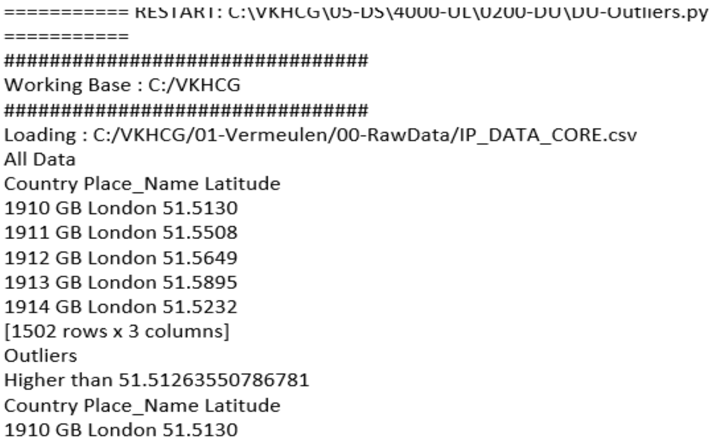
Code:

```
#######################################################################
# -*- coding: utf-8 -*-
#######################################################################
import pandas as pd
#######################################################################
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base)
print('##################################')
#######################################################################
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
```

```
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data')
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) &
(AllData.Latitude<=UpperBound)]
print(OutliersNot)
#################################################################
```



```
=========== RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py
===========
################################
Working Base : C:/VKHCG
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
Country Place_Name Latitude
1910 GB London 51.5130
1911 GB London 51.5508
1912 GB London 51.5649
1913 GB London 51.5895
1914 GB London 51.5232
... ... ... ...
[1502 rows x 3 columns]
Outliers
Higher than 51.51263550786781
Country Place_Name Latitude
1910 GB London 51.5130
```



```
=========== RESTART: C:\VKHCG\05-DS\4000-UL\0200-DU\DU-Outliers.py
===========
################################
Working Base : C:/VKHCG
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv
All Data
Country Place_Name Latitude
1910 GB London 51.5130
1911 GB London 51.5508
1912 GB London 51.5649
1913 GB London 51.5895
1914 GB London 51.5232
[1502 rows x 3 columns]
Outliers
Higher than 51.51263550786781
Country Place_Name Latitude
1910 GB London 51.5130
```

Practical 4

Retrieving Data

A. Perform the following data processing using R.

Use R-Studio for the following:

```
>library(readr)
Warning message:package „readr‟ was built under R version 3.4.4
```

Load a table named IP_DATA_ALL.csv.

```
>IP_DATA_ALL <- read_csv("C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
Parsed with column specification:
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()
)
```

This informs you that you have the following eight columns:

- ID of type integer
- Place name of type character
- Post code of type character
- Latitude of type numeric double
- Longitude of type numeric double
- First IP number of type integer
- Last IP number of type integer

```
>library(tibble)
>set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
New names:
```

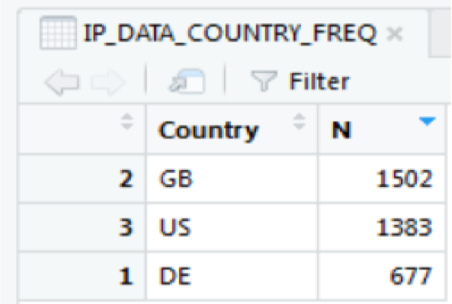
```
Place Name -> Place.Name
Post Code -> Post.Code
First IP Number -> First.IP.Number
Last IP Number -> Last.IP.Number
```

This informs you that four of the field names are not valid and suggests new field names that are valid.You can fix any detected invalid column names by executing

```
IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = TRUE)
```

By using command View(IP_DATA_ALL_FIX), you can check that you have fixed the columns. The new table IP_DATA_ALL_FIX.csv will fix the invalid column names with valid names.

```
>sapply(IP_DATA_ALL_FIX, typeof)
ID Country Place.Name Post.Code Latitude
"double" "character" "character" "double" "double"
Longitude First.IP.Number Last.IP.Number
"double" "double" "double"
>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)
```



	Country	N
2	GB	1502
3	US	1383
1	DE	677

☆ The two biggest subset volumes are from the US and GB.

☆ The US has just over four times the data as GB.

```
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)
```

☆ The two biggest data volumes are from latitudes 51.5092 and 40.6888.

☆ The spread appears to be nearly equal between the top-two latitudes.

```
>sapply(IP_DATA_ALL_FIX[,'Latitude'], min, na.rm=TRUE)
```

```
Latitude 40.6888
```

What does this tell you?

Fact: The range of latitude for the Northern Hemisphere is from 0 to 90. So, if you do not have any latitudes farther south than 40.6888, you can improve your retrieve routine.

```
>sapply(IP_DATA_ALL_FIX[,'Country'], min, na.rm=TRUE)
Country "DE"
```

Minimum business frequency is from DE – Denmark.

```
>sapply(IP_DATA_ALL_FIX[,'Latitude'], max, na.rm=TRUE)
Latitude
51.5895
>sapply(IP_DATA_ALL_FIX[,'Country'], max, na.rm=TRUE)
Country
"US"
```

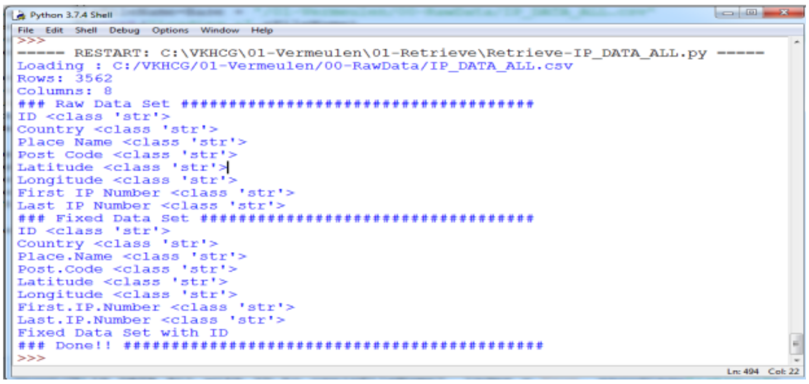
The result is 51.5895. What does this tell you?

Fact: The range in latitude for the Northern Hemisphere is from 0 to 90. So, if you do not have any latitudes more northerly than 51.5895, you can improve your retrieve routine.

```
>sapply(IP_DATA_ALL_FIX [,'Latitude'], mean, na.rm=TRUE)
Latitude
46.69097
>sapply(IP_DATA_ALL_FIX [,'Latitude'], median, na.rm=TRUE)
Latitude
48.15
>sapply(IP_DATA_ALL_FIX [,'Latitude'], range, na.rm=TRUE)
Latitude
[1,] 40.6888
[2,] 51.5895
>sapply(IP_DATA_ALL_FIX [,'Latitude'], quantile, na.rm=TRUE)
Latitude
0% 40.6888
25% 40.7588
50% 48.1500
75% 51.5092
100% 51.5895
>sapply(IP_DATA_ALL_FIX [,'Latitude'], sd, na.rm=TRUE)
Latitude
4.890387
>sapply(IP_DATA_ALL_FIX [,'Longitude'], sd, na.rm=TRUE)
Longitude
38.01702
```

B. Program to retrieve different attributes of data.

```
##### C:\ VKHCG\01-Vermeulen\01-Retrieve\Retrive_IP_DATA_ALL.py###
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!! ############################################')
################################################################
```

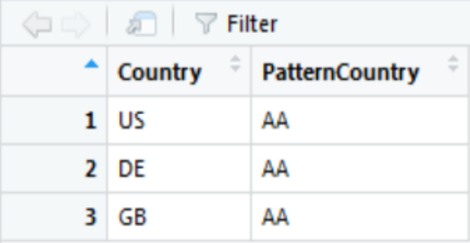


```
>>>
===== RESTART: C:\VKHCG\01-Vermeulen\01-Retrieve\Retrieve-IP_DATA_ALL.py =====
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv
Rows: 3562
Columns: 8
### Raw Data Set #####################################
ID <class 'str'>
Country <class 'str'>
Place Name <class 'str'>
Post Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First IP Number <class 'str'>
Last IP Number <class 'str'>
### Fixed Data Set ###################################
ID <class 'str'>
Country <class 'str'>
Place.Name <class 'str'>
Post.Code <class 'str'>
Latitude <class 'str'>
Longitude <class 'str'>
First.IP.Number <class 'str'>
Last.IP.Number <class 'str'>
Fixed Data Set with ID
### 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.

```
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)
```



	Country	PatternCountry
1	US	AA
2	DE	AA
3	GB	AA

PRACTICAL 05

Assessing Data

Assess Superstep

Data quality refers to the condition of a set of qualitative or quantitative variables. Dataquality is a multidimensional measurement of the acceptability of specific data sets. Inbusiness, data quality is measured to determine whether data can be used as a basis forreliable intelligence extraction for supporting organizational decisions. Data profiling involves observing in your data sources all the viewpoints that the information offers. The main goal is to determine if individual viewpoints are accurateand complete. The Assess superstep determines what additional processing to apply tothe entries that are noncompliant.

Errors

Typically, one of four things can be done with an error to the data.

1. Accept the Error
2. Reject the Error
3. Correct the Error
4. Create a Default Value

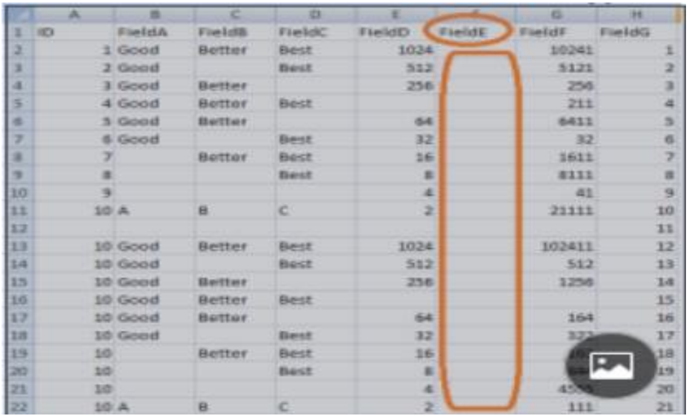
A. Perform error management on the given data using pandas package.

Python pandas package enables several automatic error-management features.

File Location: C:\VKHCG\01-Vermeulen\02-Assess

Missing Values in Pandas:

i. Drop the Columns Where All Elements Are Missing Values

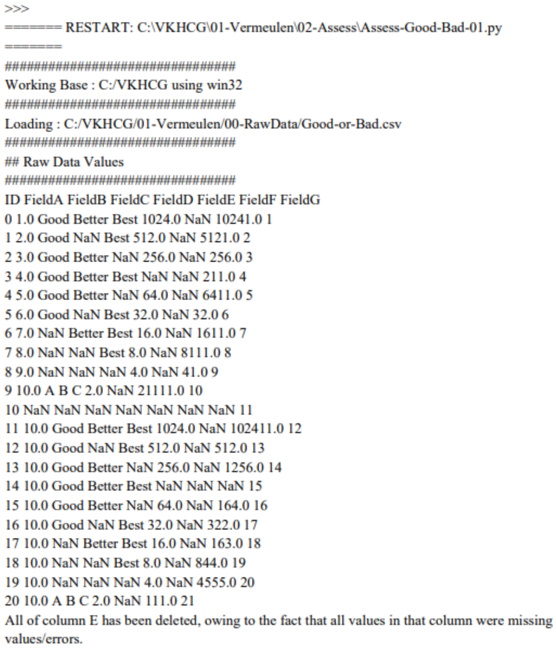


	A	B	C	D	E	F	G	H
1	ID	FieldA	FieldB	FieldC	FieldD	FieldE	FieldF	FieldG
2	1	Good	Better	Best	1024		10241	1
3	2	Good		Best	512		5121	2
4	3	Good	Better		256		256	3
5	4	Good	Better	Best			211	4
6	5	Good	Better		64		6411	5
7	6	Good		Best	32		32	6
8	7		Better	Best	16		1611	7
9	8			Best	8		8111	8
10	9				4		41	9
11	10	A	B	C	2		21111	10
12								11
13	10	Good	Better	Best	1024		102411	12
14	10	Good		Best	512		512	13
15	10	Good	Better		256		1256	14
16	10	Good	Better	Best				15
17	10	Good	Better		64		164	16
18	10	Good		Best	32		322	17
19	10		Better	Best	16		[illegible]	18
20	10			Best	8		[illegible]	19
21	10				4		[illegible]	20
22	10	A	B	C	2		111	21

Code :

```
#################### Assess-Good-Bad-01.py#########################
# -*- coding: utf-8 -*-
############################################################################
import sys
import os
import pandas as pd
############################################################################
Base='C:/VKHCG'
############################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-01.csv'Company='01-Vermeulen'
####################################################################
Base='C:/VKHCG'
####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
####################################################################
### Import Warehouse
####################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('##################################')
print('## Raw Data Values')
print('##################################')
print(RawData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('##################################')
####################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
####################################################################
TestData=RawData.dropna(axis=1, how='all')
####################################################################
print('##################################')
print('## Test Data Values')
print('##################################')
print(TestData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('##################################')
```

```
#######################################################################
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#######################################################################
print('###################################')
print('### Done!! #######################')
print('###################################')
#######################################################################
```



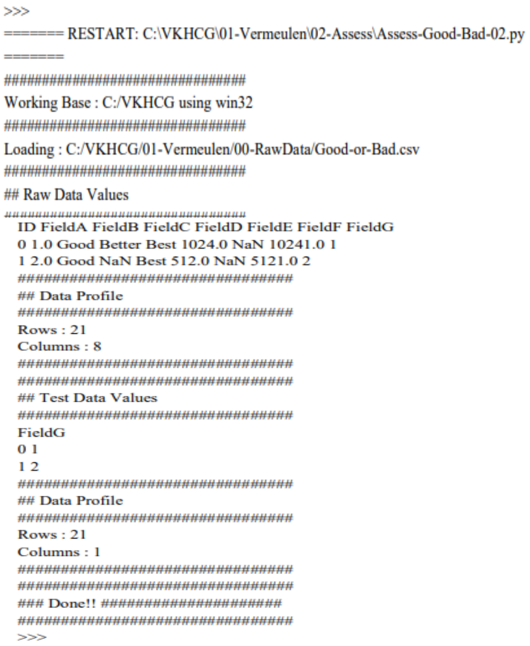
```
>>>
======= RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-01.py
=======
###################################
Working Base : C:/VKHCG using win32
###################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
###################################
## Raw Data Values
###################################
ID FieldA FieldB FieldC FieldD FieldE FieldF FieldG
0 1.0 Good Better Best 1024.0 NaN 10241.0 1
1 2.0 Good NaN Best 512.0 NaN 5121.0 2
2 3.0 Good Better NaN 256.0 NaN 256.0 3
3 4.0 Good Better Best NaN NaN 211.0 4
4 5.0 Good Better NaN 64.0 NaN 6411.0 5
5 6.0 Good NaN Best 32.0 NaN 32.0 6
6 7.0 NaN Better Best 16.0 NaN 1611.0 7
7 8.0 NaN NaN Best 8.0 NaN 8111.0 8
8 9.0 NaN NaN NaN 4.0 NaN 41.0 9
9 10.0 A B C 2.0 NaN 21111.0 10
10 NaN NaN NaN NaN NaN NaN NaN 11
11 10.0 Good Better Best 1024.0 NaN 102411.0 12
12 10.0 Good NaN Best 512.0 NaN 512.0 13
13 10.0 Good Better NaN 256.0 NaN 1256.0 14
14 10.0 Good Better Best NaN NaN NaN 15
15 10.0 Good Better NaN 64.0 NaN 164.0 16
16 10.0 Good NaN Best 32.0 NaN 322.0 17
17 10.0 NaN Better Best 16.0 NaN 163.0 18
18 10.0 NaN NaN Best 8.0 NaN 844.0 19
19 10.0 NaN NaN NaN 4.0 NaN 4555.0 20
20 10.0 A B C 2.0 NaN 111.0 21
```

All of column E has been deleted, owing to the fact that all values in that column were missing values/errors.

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

```
#################### Assess-Good-Bad-02.py##############################
import sys
import os
import pandas as pd
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-02.csv'
Company='01-Vermeulen'
#######################################################################
Base='C:/VKHCG'
#######################################################################
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
######################################################################sFil
eDir=Base + '/' + Company
+ '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
```

```
############################################################################
### Import Warehouse
############################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('##################################')
print('## Raw Data Values')
print('##################################')
print(RawData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('##################################')
############################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
############################################################################
TestData=RawData.dropna(axis=1, how='any')
############################################################################
print('##################################')
print('## Test Data Values')
print('##################################')
print(TestData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('##################################')
############################################################################
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
############################################################################
print('##################################')
print('### Done!! ######################')
print('##################################')
############################################################################
```

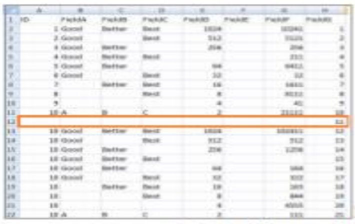


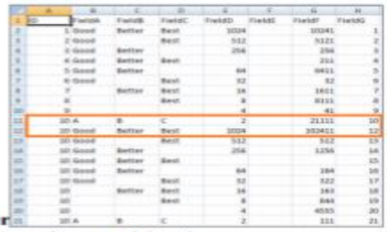
```
>>>
====== RESTART: C:\VKHCG\01-Vermeulen\02-Assess\Assess-Good-Bad-02.py
======
################################
Working Base : C:/VKHCG using win32
################################
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
################################
## Raw Data Values
################################
  ID FieldA FieldB FieldC FieldD FieldE FieldF FieldG
  0 1.0 Good Better Best 1024.0 NaN 10241.0 1
  1 2.0 Good NaN Best 512.0 NaN 5121.0 2
  ####################################
  ## Data Profile
  ####################################
  Rows : 21
  Columns : 8
  ####################################
  ####################################
  ## Test Data Values
  ####################################
  FieldG
  0 1
  1 2
  ####################################
  ## Data Profile
  ####################################
  Rows : 21
  Columns : 1
  ####################################
  ####################################
  ### Done!! #########################
  ####################################
  >>>
```

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

```
####################### Assess-Good-Bad-03.py ##################
# -*- coding: utf-8 -*-
########################################################################
import sys
import os
import pandas as pd
########################################################################
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-03.csv'
Company='01-Vermeulen'
Base='C:/VKHCG'
########################################################################
print('####################################')
print('Working Base :',Base, ' using Windows ~~~~')
print('####################################')
########################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
########################################################################
### Import Warehouse
########################################################################
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('####################################')
print('## Raw Data Values')
```

```
print('##################################')
print(RawData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('##################################')
####################################################################
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
####################################################################
TestData=RawData.dropna(thresh=2)
print('##################################')
print('## Test Data Values')
print('##################################')
print(TestData)
print('##################################')
print('## Data Profile')
print('##################################')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('##################################')
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
####################################################################
print('##################################')
print('### Done!! ######################')
print('##################################')
####################################################################
```





Before After

Row with more than two missing values got deleted.

The next step along the route is to generate a full network routing solution for the company, to resolve the data issues in the retrieve data.

Practical 6:

Processing Data

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

Open your Python editor and create a file named Process_Time.py. Save it into directory C:\VKHCG\01-Vermeulen\03-Process.

```
import sys
import os
from datetime import datetime
from datetime import timedelta
from pytz import timezone, all_timezones
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid
pd.options.mode.chained_assignment = None
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
print('#################################')
print('Working Base :',Base, ' using ', sys.platform)
print('#################################')
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
####################################################################
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
####################################################################
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
base = datetime(2018,1,1,0,0,0)
numUnits=10*365*24
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)]
t=0
for i in date_list:
now_utc=i.replace(tzinfo=timezone('UTC'))
```

```
sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")
print(sDateTime)
sDateTimeKey=sDateTime.replace(' ','-').replace(':','-')
t+=1
IDNumber=str(uuid.uuid4())
TimeLine=[('ZoneBaseKey', ['UTC']),
('IDNumber', [IDNumber]),
('nDateTimeValue', [now_utc])
('DateTimeValue', [sDateTime]),
('DateTimeKey', [sDateTimeKey])]
if t==1:
TimeFrame = pd.DataFrame.from_items(TimeLine)
else:
TimeRow = pd.DataFrame.from_items(TimeLine)
TimeFrame = TimeFrame.append(TimeRow)
################################################################
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue]
]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
TimeFrame.set_index(['IDNumber'],inplace=True)
sTable = 'Process-Time'
print('Storing :',sDatabaseName,' Table:',sTable)
TimeHubIndex.to_sql(sTable, conn1, if_exists="replace")
sTable = 'Hub-Time'
print('Storing :',sDatabaseName,' Table:',sTable)
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
active_timezones=all_timezones
z=0
for zone in active_timezones:
t=0
for j in range(TimeFrame.shape[0]):
now_date=TimeFrame['nDateTimeValue'][j]
DateTimeKey=TimeFrame['DateTimeKey'][j]
now_utc=now_date.replace(tzinfo=timezone('UTC'))
sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")
now_zone = now_utc.astimezone(timezone(zone))
sZoneDateTime=now_zone.strftime("%Y-%m-%d %H:%M:%S")
print(sZoneDateTime)
t+=1
z+=1
IDZoneNumber=str(uuid.uuid4())
TimeZoneLine=[('ZoneBaseKey', ['UTC']),
('IDZoneNumber', [IDZoneNumber]),
('DateTimeKey', [DateTimeKey]),
('UTCDateTimeValue', [sDateTime]),
('Zone', [zone]),
('DateTimeValue', [sZoneDateTime])]
if t==1:
TimeZoneFrame = pd.DataFrame.from_items(TimeZoneLine)
else:
TimeZoneRow = pd.DataFrame.from_items(TimeZoneLine)
TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)
TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber'],inplace=False)
sZone=zone.replace('/','-').replace(' ','')
sTable = 'Process-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn1, if_exists="replace")
sTable = 'Satellite-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")
print('################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('################')
print('### Done!! ############################################')
```

You have built your first hub and satellites for time in the data vault.
The data vault has been built in directory ..\ VKHCG\88-DV\datavault.db. You can access it with your SQLite tools

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 proram will generate a golden nominal using our sample data set.
Open your Python editor and create a file called Process-People.py in the .. C:\VKHCG\04-Clark\03-Process directory.

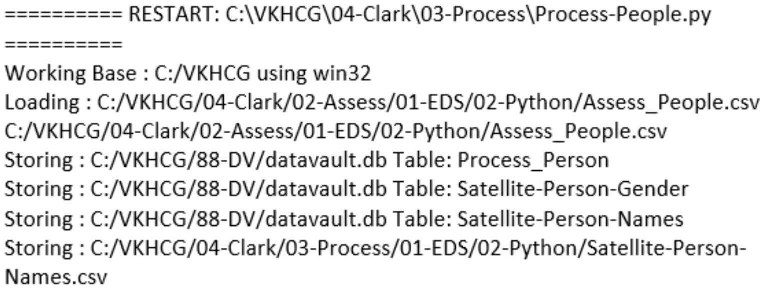
```
################################################################
import sys
import os
import sqlite3 as sq
import pandas as pd
from pandas.io import sql
from datetime import datetime, timedelta
from pytz import timezone, all_timezones
from random import randint
import uuid
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
print('###################################')
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('###################################')
print('Loading :',sFileName)
print('###################################')
print(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('################################')
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','BirthDat
eKey']]
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','LastNam
e'\,'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','LastNa
me',\'BirthDateKey','TimeZone','BirthDate']]
PersonSatelliteBirthdayRaw['PersonSatelliteID']=RawData.apply(lambda row:
str(uuid.uuid4()),axis=1)
PersonSatelliteBirthday=PersonSatelliteBirthdayRaw.drop_duplicates(subset=Non
e, \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('#################')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#################')
print('### Done!! ################################################')
```

Output :

It will apply golden nominal rules by assuming nobody born before January 1, 1900, droping to two ISO complex date time structures, as the code does not translate into SQLite‟s data types and saves your new golden nominal to a CSV file.Load the person into the data vault



```
========== RESTART: C:\VKHCG\04-Clark\03-Process\Process-People.py
==========
Working Base : C:/VKHCG using win32
Loading : C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
C:/VKHCG/04-Clark/02-Assess/01-EDS/02-Python/Assess_People.csv
Storing : C:/VKHCG/88-DV/datavault.db Table: Process_Person
Storing : C:/VKHCG/88-DV/datavault.db Table: Satellite-Person-Gender
Storing : C:/VKHCG/88-DV/datavault.db Table: Satellite-Person-Names
Storing : C:/VKHCG/04-Clark/03-Process/01-EDS/02-Python/Satellite-Person-
Names.csv
```

Practical 7
Transforming Data

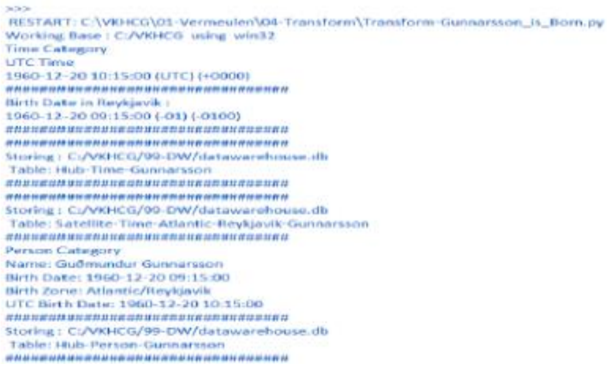
Transform Superstep

C: \VKHCG\01-Vermeulen\04-Transform.

```
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")
####################################################################
```

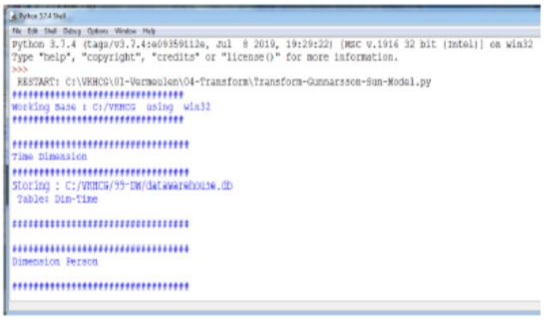


```
>>>
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson_is_Born.py
Working Base : C:/VKHCG  using  win32
Time Category
UTC Time
1960-12-20 10:15:00 (UTC) (+0000)
###################################
Birth Date in Reykjavik :
1960-12-20 09:15:00 (-01) (-0100)
###################################
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Time-Gunnarsson
###################################
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Satellite-Time-Atlantic-Reykjavik-Gunnarsson
###################################
Person Category
Name: Guðmundur Gunnarsson
Birth Date: 1960-12-20 09:15:00
Birth Zone: Atlantic/Reykjavik
UTC Birth Date: 1960-12-20 10:15:00
###################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Hub-Person-Gunnarsson
###################################
```

You must build three items: dimension Person, dimension Time, and factPersonBornAtTime. Open your Python editor and create a file named Transform-Gunnarsson-Sun-Model.py in directory C:\VKHCG\01-Vermeulen\04-Transform.

```
######################################################################
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
######################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + '/VKHCG'
else:
Base='C:/VKHCG'
print('###################################')
print('Working Base :',Base, ' using ', sys.platform)
print('###################################')
######################################################################
Company='01-Vermeulen'
######################################################################
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
os.makedirs(sDataBaseDir)
######################################################################
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
######################################################################
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
os.makedirs(sDataWarehousetDir)
######################################################################
```

```
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
########################################################################
print('\n####################################')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z)
(%z)")
BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
########################################################################
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]),
('UTCDate', [BirthDateZoneStr]),
('LocalTime', [BirthDateLocal]),
('TimeZone', [BirthZone])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
########################################################################
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
########################################################################
sTable = 'Dim-Time'
print('\n####################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n####################################')
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")
########################################################################
print('\n####################################')
print('Dimension Person')
print('\n####################################')
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
#######################################################################
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]),
('FirstName', [FirstName]),
('LastName', [LastName]),
('Zone', ['UTC']),
('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
####################################################################
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
####################################################################
sTable = 'Dim-Person'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
####################################################################
print('\n###################################')
print('Fact - Person - time')
print('\n###################################')
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
('IDPersonNumber', [IDPersonNumber]),
('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)
####################################################################
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
####################################################################
sTable = 'Fact-Person-Time'
print('\n###################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n###################################')
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
####################################################################
```



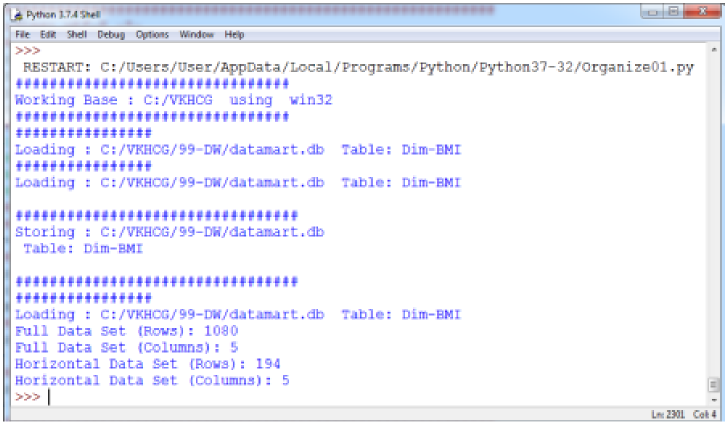
```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
 RESTART: C:\VKHCG\01-Vermeulen\04-Transform\Transform-Gunnarsson-Sun-Model.py
#################################
Working Base : C:/VKHCG  using  win32
#################################

##################################
Time Dimension
##################################
Storing : C:/VKHCG/99-DW/datawarehouse.db
 Table: Dim-Time

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

##################################
Dimension Person

##################################
```

Practical 8:

Organizing Data

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Horizontal.py

```
####################################################################
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 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'
print('\n#################################')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#################################')
#DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
######################################################################
print('################')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
```



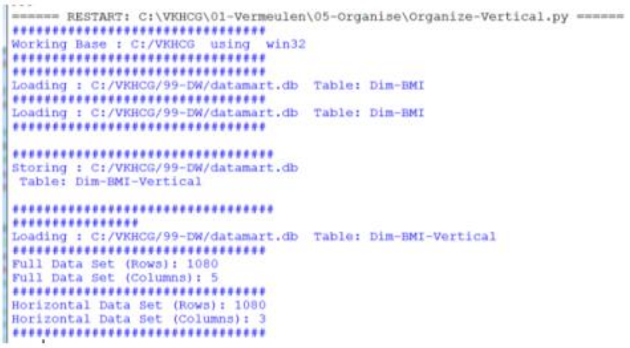
```
>>>
 RESTART: C:/Users/User/AppData/Local/Programs/Python/Python37-32/Organize01.py
################################
Working Base : C:/VKHCG  using  win32
################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI

#################################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI

#################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
>>> |
```

Vertical Style

C:\VKHCG\01-Vermeulen\05-Organise\ Organize-Vertical.py

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



```
====== RESTART: C:\VKHCG\01-Vermeulen\05-Organise\Organize-Vertical.py ======
################################
Working Base : C:/VKHCG  using  win32
################################
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI
################################

#################################
Storing : C:/VKHCG/99-DW/datamart.db
 Table: Dim-BMI-Vertical

#################################
################
Loading : C:/VKHCG/99-DW/datamart.db  Table: Dim-BMI-Vertical
################################
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
################################
Horizontal Data Set (Rows): 1080
Horizontal Data Set (Columns): 3
################################
```

Practical 9

Generating Data

Report Superstep

The Report superstep is the step in the ecosystem that enhances the data science findings with the art of storytelling and data visualization. You can perform the best data science, but if you cannot execute a respectable and trustworthy Report step by turning your data science into actionable business insights, you have achieved no advantage for your business.

Vermeulen PLC

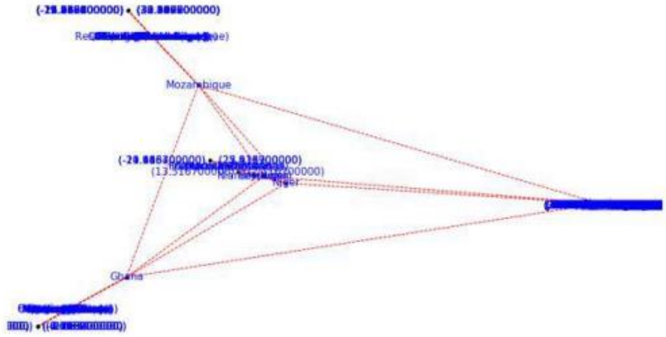
Vermeulen requires a map of all their customers‟ data links. Can you provide a report to deliver this? I will guide you through an example that delivers this requirement.

C:\VKHCG\01-Vermeulen\06-Report\Raport-Network-Routing-Customer.py

```
####################################################################
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
####################################################################
pd.options.mode.chained_assignment = None
####################################################################
if sys.platform == 'linux':
Base=os.path.expanduser('~') + 'VKHCG'
else:
Base='C:/VKHCG'
####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-
Customer.csv'
####################################################################
sOutputFileName1='06-Report/01-EDS/02-Python/Report-Network-Routing-
Customer.gml'
sOutputFileName2='06-Report/01-EDS/02-Python/Report-Network-Routing-
Customer.png'
Company='01-Vermeulen'
####################################################################
####################################################################
### Import Country Data
####################################################################
```

```
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
CustomerDataRaw=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
CustomerData=CustomerDataRaw.head(100)
print('Loaded Country:',CustomerData.columns.values)
print('##################################')
####################################################################
print(CustomerData.head())
print(CustomerData.shape)
####################################################################
G=nx.Graph()
for i in range(CustomerData.shape[0]):
for j in range(CustomerData.shape[0]):
Node0=CustomerData['Customer_Country_Name'][i]
Node1=CustomerData['Customer_Country_Name'][j]
if Node0 != Node1:
G.add_edge(Node0,Node1)
for i in range(CustomerData.shape[0]):
Node0=CustomerData['Customer_Country_Name'][i]
Node1=CustomerData['Customer_Place_Name'][i] + '('+
CustomerData['Customer_Country_Name'][i] + ')'
Node2='('+ "{:.9f}".format(CustomerData['Customer_Latitude'][i]) + ')\
('+ "{:.9f}".format(CustomerData['Customer_Longitude'][i]) + ')'
if Node0 != Node1:
G.add_edge(Node0,Node1)
if Node1 != Node2:
G.add_edge(Node1,Node2)
print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
####################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName1
print('##################################')
print('Storing :',sFileName)
print('##################################')
nx.write_gml(G, sFileName)
####################################################################
sFileName=Base + '/' + Company + '/' + sOutputFileName2
print('##################################')
print('Storing Graph Image:',sFileName)
print('##################################')
```

```
plt.figure(figsize=(25, 25))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
print('##################################')
print('### Done!! #####################')
print('##################################')
```



Krennwallner AG

The Krennwallner marketing department wants to deploy the locations of the billboards onto the company web server. Can you prepare three versions of the locations‟ web pages?

- ☆ Locations clustered into bubbles when you zoom out
- ☆ Locations as pins
- ☆ Locations as heat map

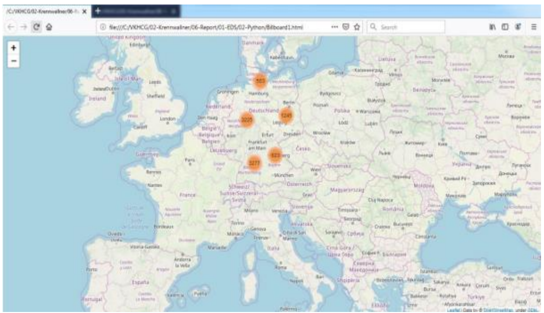
Picking Content for Billboards

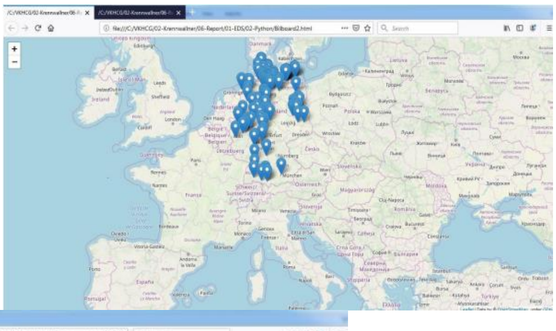
C:\VKHCG\02-Krennwallner\06-Report\Report_Billboard.py

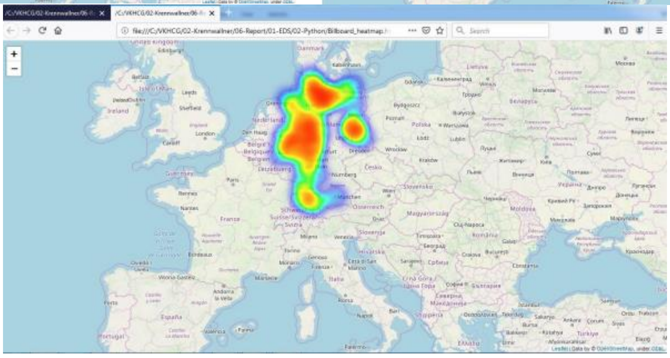
```
import sys
import os
import pandas as pd
from folium.plugins import FastMarkerCluster, HeatMap
from folium import Marker, Map
import webbrowser
################################################################
Base='C:/VKHCG'
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
################################################################
sFileName=Base+'/02-Krennwallner/01-Retrieve/01-EDS/02-Python/Retrieve_DE_Billboard_Locations.csv'
```

```
df = pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
df.fillna(value=0, inplace=True)
print(df.shape)
################################################################
t=0
for i in range(df.shape[0]):
try:
sLongitude=df["Longitude"][i]
sLongitude=float(sLongitude)
except Exception:
sLongitude=float(0.0)
try:
sLatitude=df["Latitude"][i]
sLatitude=float(sLatitude)
except Exception:
sLatitude=float(0.0)
try:
sDescription=df["Place_Name"][i] + ' (' + df["Country"][i]+')'
except Exception:
sDescription='VKHCG'
if sLongitude != 0.0 and sLatitude != 0.0:
DataClusterList=list([sLatitude, sLongitude])
DataPointList=list([sLatitude, sLongitude, sDescription])
t+=1
if t==1:
DataCluster=[DataClusterList]
DataPoint=[DataPointList]
else:
DataCluster.append(DataClusterList)
DataPoint.append(DataPointList)
data=DataCluster
pins=pd.DataFrame(DataPoint)
pins.columns = [ 'Latitude','Longitude','Description']
################################################################
stops_map1 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
marker_cluster = FastMarkerCluster(data).add_to(stops_map1)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-
Python/Billboard1.html'
stops_map1.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
################################################################
stops_map2 = Map(location=[48.1459806, 11.4985484], zoom_start=5)
for name, row in pins.iloc[:100].iterrows():
```

```
Marker([row["Latitude"],row["Longitude"]],
popup=row["Description"]).add_to(stops_map2)
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-
Python/Billboard2.html'
stops_map2.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
################################################################
stops_heatmap = Map(location=[48.1459806, 11.4985484], zoom_start=5)
stops_heatmap.add_child(HeatMap([[row["Latitude"], row["Longitude"]] for
name, row in
pins.iloc[:100].iterrows()]))
sFileNameHtml=Base+'/02-Krennwallner/06-Report/01-EDS/02-
Python/Billboard_heatmap.html'
stops_heatmap.save(sFileNameHtml)
webbrowser.open('file://' + os.path.realpath(sFileNameHtml))
################################################################
print('### Done!! ############################################')
################################################################
```







Hillman Ltd

Dr. Hillman Sr. has just installed a camera system that enables the company to capture video and, therefore, indirectly, images of all containers that enter or leave the warehouse. Can you convert the number on the side of the containers into digits?

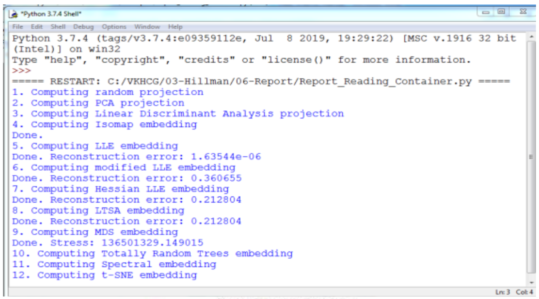
Reading the Containers

C:\VKHCG\03-Hillman\06-Report\

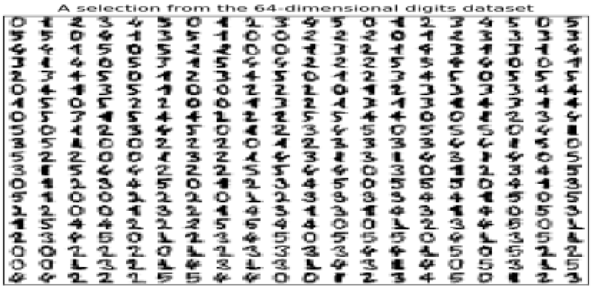
Report_Reading_Container.py

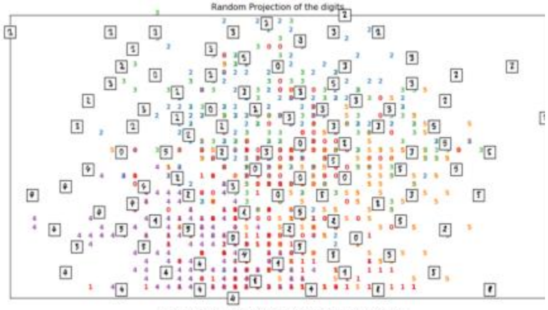
```
from time import time
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import offsetbox
from sklearn import (manifold, datasets, decomposition, ensemble,
discriminant_analysis,
random_projection)
digits = datasets.load_digits(n_class=6)
X = digits.data
y = digits.target
n_samples, n_features = X.shape
n_neighbors = 30
def plot_embedding(X, title=None):
x_min, x_max = np.min(X, 0), np.max(X, 0)
X = (X - x_min) / (x_max - x_min)
plt.figure(figsize=(10, 10))
ax = plt.subplot(111)
for i in range(X.shape[0]):
plt.text(X[i, 0], X[i, 1], str(digits.target[i]),
color=plt.cm.Set1(y[i] / 10.),
fontdict={'weight': 'bold', 'size': 9})
if hasattr(offsetbox, 'AnnotationBbox'):
# only print thumbnails with matplotlib > 1.0
shown_images = np.array([[1., 1.]]) # just something big
for i in range(digits.data.shape[0]):
dist = np.sum((X[i] - shown_images) ** 2, 1)
if np.min(dist) < 4e-3:
# don't show points that are too close continue
shown_images = np.r_[shown_images, [X[i]]]
imagebox = offsetbox.AnnotationBbox(offsetbox.OffsetImage(digits.images[i],
cmap=plt.cm.gray_r),X[i])
ax.add_artist(imagebox)
plt.xticks([]), plt.yticks([])
if title is not None:
plt.title(title)
n_img_per_row = 20
img = np.zeros((10 * n_img_per_row, 10 * n_img_per_row))
for i in range(n_img_per_row):
ix = 10 * i + 1
for j in range(n_img_per_row):
iy = 10 * j + 1
img[ix:ix + 8, iy:iy + 8] = X[i * n_img_per_row + j].reshape((8, 8))
plt.figure(figsize=(10, 10))
plt.imshow(img, cmap=plt.cm.binary)
plt.xticks([])
plt.yticks([])
plt.title('A selection from the 64-dimensional digits dataset')
print("Computing random projection")
rp = random_projection.SparseRandomProjection(n_components=2,
random_state=42)
X_projected = rp.fit_transform(X)
plot_embedding(X_projected, "Random Projection of the digits")
print("Computing PCA projection") t0 = time()
X_pca = decomposition.TruncatedSVD(n_components=2).fit_transform(X)
plot_embedding(X_pca,"Principal Components projection of the digits (time
%.2fs)" %(time() - t0))
print("Computing Linear Discriminant Analysis projection")
X2 = X.copy()
X2.flat[::X.shape[1] + 1] += 0.01 # Make X invertible
t0 = time()
X_lda =
discriminant_analysis.LinearDiscriminantAnalysis(n_components=2).fit_transform
(X2, y)
plot_embedding(X_lda,"Linear Discriminant projection of the digits (time %.2fs)"
%(time() -t0))
print("Computing Isomap embedding")
t0 = time()
X_iso = manifold.Isomap(n_neighbors, n_components=2).fit_transform(X)
print("Done.")
plot_embedding(X_iso,"Isomap projection of the digits (time %.2fs)" %(time() -
t0))
print("Computing LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors,
n_components=2,method='standard')
t0 = time()
X_lle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_lle,"Locally Linear Embedding of the digits (time %.2fs)"
%(time() - t0))
print("Computing modified LLE embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors, n_components=2,
method='modified') t0 = time()
X_mlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_mlle,"Modified Locally Linear Embedding of the digits (time
%.2fs)" %(time() - t0))
print("Computing Hessian LLE embedding")
```

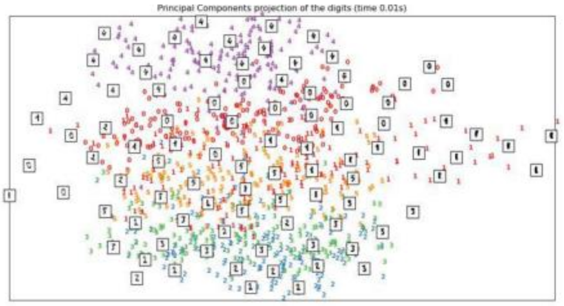
```
clf = manifold.LocallyLinearEmbedding(n_neighbors,
n_components=2,method='hessian') t0 = time()
X_hlle = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_hlle,"Hessian Locally Linear Embedding of the digits (time
%.2fs)" %(time() - t0))
print("Computing LTSA embedding")
clf = manifold.LocallyLinearEmbedding(n_neighbors,
n_components=2,method='ltsa')
t0 = time()
X_ltsa = clf.fit_transform(X)
print("Done. Reconstruction error: %g" % clf.reconstruction_error_)
plot_embedding(X_ltsa,"Local Tangent Space Alignment of the digits (time %.2fs)"
%(time() - t0))
print("Computing MDS embedding")
clf = manifold.MDS(n_components=2, n_init=1, max_iter=100)
t0 = time()
X_mds = clf.fit_transform(X)
print("Done. Stress: %f" % clf.stress_)
plot_embedding(X_mds,"MDS embedding of the digits (time %.2fs)" %(time() -
t0))
print("Computing Totally Random Trees embedding")
hasher = ensemble.RandomTreesEmbedding(n_estimators=200, random_state=0,
max_depth=5)
t0 = time()
X_transformed = hasher.fit_transform(X)
pca = decomposition.TruncatedSVD(n_components=2)
X_reduced = pca.fit_transform(X_transformed)
plot_embedding(X_reduced,"Random forest embedding of the digits (time
%.2fs)" %(time() -t0))
print("Computing Spectral embedding")
embedder = manifold.SpectralEmbedding(n_components=2, random_state=0,
eigen_solver="arpack") t0 = time()
X_se = embedder.fit_transform(X)
plot_embedding(X_se,"Spectral embedding of the digits (time %.2fs)" %(time() -
t0))
print("Computing t-SNE embedding")
tsne = manifold.TSNE(n_components=2, init='pca', random_state=0)
t0 = time()
X_tsne = tsne.fit_transform(X)
plot_embedding(X_tsne,"t-SNE embedding of the digits (time %.2fs)" %(time() -
t0))
plt.show()
```

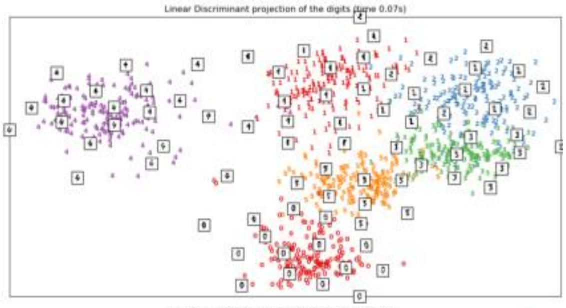


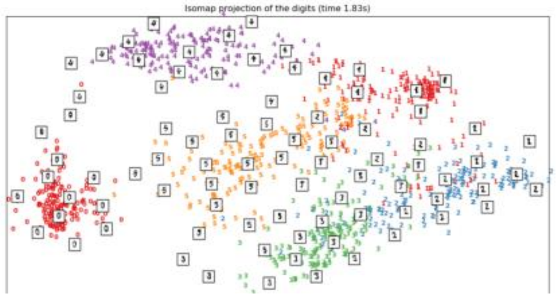
```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/VKHCG/03-Hillman/06-Report/Report_Reading_Container.py =====
1. Computing random projection
2. Computing PCA projection
3. Computing Linear Discriminant Analysis projection
4. Computing Isomap embedding
Done.
5. Computing LLE embedding
Done. Reconstruction error: 1.63544e-06
6. Computing modified LLE embedding
Done. Reconstruction error: 0.360655
7. Computing Hessian LLE embedding
Done. Reconstruction error: 0.212804
8. Computing LTSA embedding
Done. Reconstruction error: 0.212804
9. Computing MDS embedding
Done. Stress: 136501329.149015
10. Computing Totally Random Trees embedding
11. Computing Spectral embedding
12. Computing t-SNE embedding
```

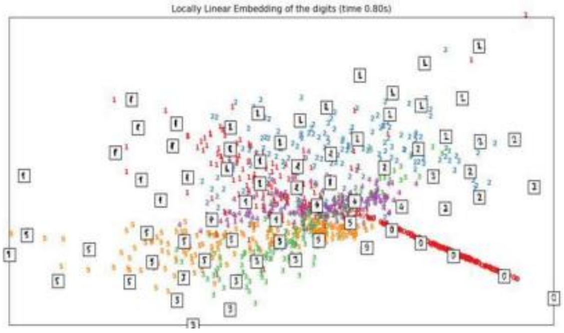


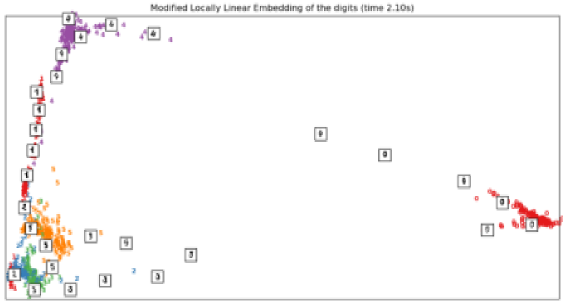


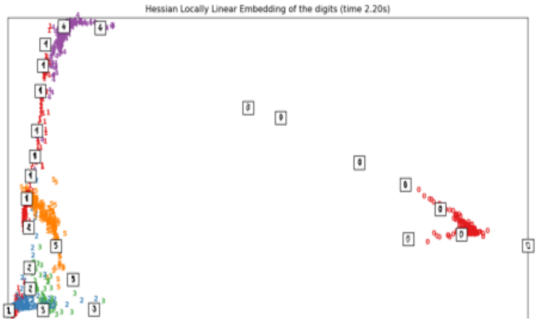


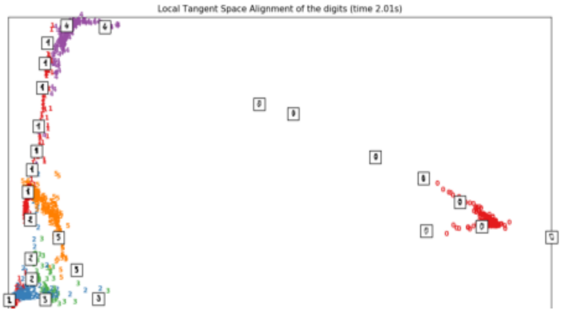












Clark Ltd

The financial company in VKHCG is the Clark accounting firm that VKHCG owns with a 60% stake. The accountants are the financial advisers to the group and handle everything to do with the complex work of international accounting.

Financials

The VKHCG companies did well last year, and the teams at Clark must prepare a balance sheet for each company in the group. The companies require a balance sheet for each company, to be produced using the template (Balance-Sheet-Template.xlsx) that can be found in the example directory (..\VKHCG\04-Clark\00-RawData).
The Program will guide you through a process that will enable you to merge the data science with preformatted Microsoft Excel template, to produce a balance sheet for each of the VKHCG companies.

C:\VKHCG\04-Clark\06-Report\Report-Balance-Sheet.py

```
import sys
import os
import pandas as pd
import sqlite3 as sq
import re
from openpyxl import load_workbook
####################################################################
Base='C:/VKHCG'
####################################################################
print('##################################')
print('Working Base :',Base, ' using ', sys.platform)
print('##################################')
####################################################################
sInputTemplateName='00-RawData/Balance-Sheet-Template.xlsx'
####################################################################
sOutputFileName='06-Report/01-EDS/02-Python/Report-Balance-Sheet'
Company='04-Clark'
####################################################################
sDatabaseName=Base + '/' + Company + '/06-Report/SQLite/clark.db'
conn = sq.connect(sDatabaseName)
#conn = sq.connect(':memory:')
####################################################################
### Import Balance Sheet Data
####################################################################
for y in range(1,13):
sInputFileName='00-RawData/BalanceSheets' + str(y).zfill(2) + '.csv'
sFileName=Base + '/' + Company + '/' + sInputFileName
print('##################################')
print('Loading :',sFileName)
print('##################################')
ForexDataRaw=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('##################################')
####################################################################
```

```
ForexDataRaw.index.names = ['RowID']
sTable='BalanceSheets'
print('Storing :',sDatabaseName,' Table:',sTable)
if y == 1:
print('Load Data')
ForexDataRaw.to_sql(sTable, conn, if_exists="replace")
else:
print('Append Data')
ForexDataRaw.to_sql(sTable, conn, if_exists="append")
###########################################################################
sSQL="SELECT \
Year, \
Quarter, \
Country, \
Company, \
CAST(Year AS INT) || 'Q' || CAST(Quarter AS INT) AS sDate, \
Company || ' (' || Country || ')' AS sCompanyName , \
CAST(Year AS INT) || 'Q' || CAST(Quarter AS INT) || '-' ||\
Company || '-' || Country AS sCompanyFile \
FROM BalanceSheets \
GROUP BY \
Year, \
Quarter, \
Country, \
Company \
HAVING Year is not null \;"
sSQL=re.sub("\s\s+", " ", sSQL)
sDatesRaw=pd.read_sql_query(sSQL, conn)
print(sDatesRaw.shape)
sDates=sDatesRaw.head(5)
###########################################################################
## Loop Dates
###########################################################################
for i in range(sDates.shape[0]):
sFileName=Base + '/' + Company + '/' + sInputTemplateName
wb = load_workbook(sFileName)
ws=wb.get_sheet_by_name("Balance-Sheet")
sYear=sDates['sDate'][i]
sCompany=sDates['sCompanyName'][i]
sCompanyFile=sDates['sCompanyFile'][i]
sCompanyFile=re.sub("\s+", "", sCompanyFile)\
ws['D3'] = sYear
ws['D5'] = sCompany
```

```
sFields = pd.DataFrame(
[
['Cash','D16', 1],
['Accounts_Receivable','D17', 1],
['Doubtful_Accounts','D18', 1],
['Inventory','D19', 1],
['Temporary_Investment','D20', 1],
'Prepaid_Expenses','D21', 1],
['Long_Term_Investments','D24', 1],
['Land','D25', 1],
['Buildings','D26', 1],
['Depreciation_Buildings','D27', -1],
['Plant_Equipment','D28', 1],
['Depreciation_Plant_Equipment','D29', -1],
['Furniture_Fixtures','D30', 1],
['Depreciation_Furniture_Fixtures','D31', -1],
['Accounts_Payable','H16', 1],
['Short_Term_Notes','H17', 1],
['Current_Long_Term_Notes','H18', 1],
['Interest_Payable','H19', 1],
['Taxes_Payable','H20', 1],
['Accrued_Payroll','H21', 1],
['Mortgage','H24', 1],
['Other_Long_Term_Liabilities','H25', 1],
['Capital_Stock','H30', 1]
] )
nYear=str(int(sDates['Year'][i]))
nQuarter=str(int(sDates['Quarter'][i]))
sCountry=str(sDates['Country'][i])
sCompany=str(sDates['Company'][i])
sFileName=Base + '/' + Company + '/' + sOutputFileName + \ '-' + sCompanyFile +
'.xlsx'
print(sFileName)
for j in range(sFields.shape[0]):
sSumField=sFields[0][j]
sCellField=sFields[1][j]
nSumSign=sFields[2][j]
sSQL="SELECT \
Year, \
Quarter, \
Country, \
Company, \
SUM(" + sSumField + ") AS nSumTotal \
FROM BalanceSheets \
GROUP BY \
Year, \
Quarter, \
Country, \
Company \
HAVING \
Year=" + nYear + " \
AND \
Quarter=" + nQuarter + " \
AND \
Country='" + sCountry + "' \
AND \
Company='" + sCompany + "' \;"
sSQL=re.sub("\s\s+", " ", sSQL)
sSumRaw=pd.read_sql_query(sSQL, conn)
ws[sCellField] = sSumRaw["nSumTotal"][0] * nSumSign
print('Set cell',sCellField,' to ', sSumField,'Total')
wb.save(sFileName)
```

Output:

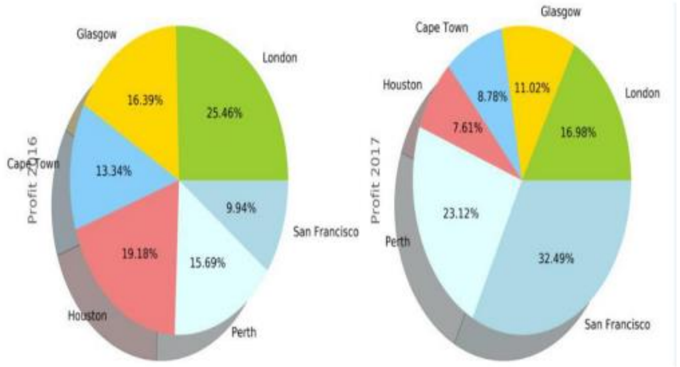
Graphics

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

Pie Graph

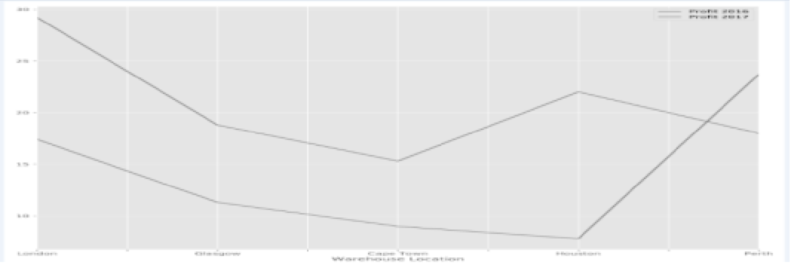
Double Pie

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



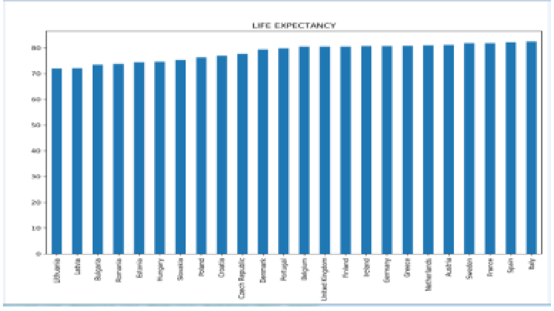
Line Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



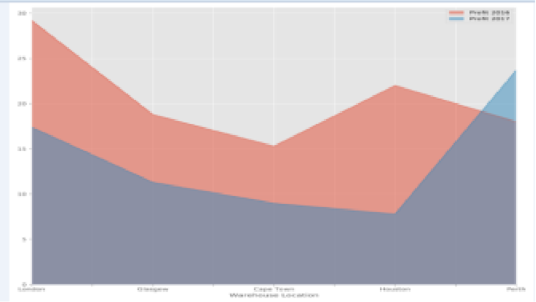
Bar Graph / Horizontal Bar Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



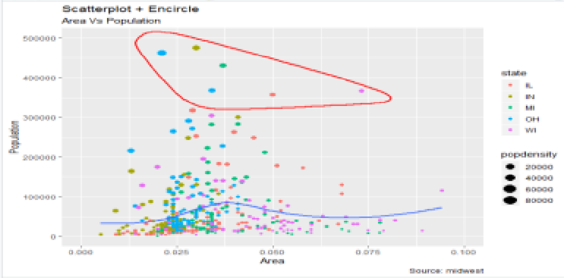
Area Graph

C:/VKHCG/01-Vermeulen/06-Report/Report_Graph_A.py



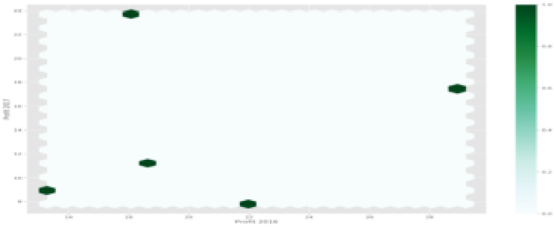
SCATTER GRAPH

C:/ VKHCG/03-HILLMAN/06-REPORT/REPORT-SCATTERPLOT-WITH-ENCIRCLING.R



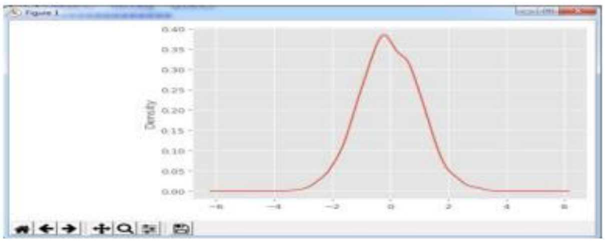
Hexbin:

Program : C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_A.py



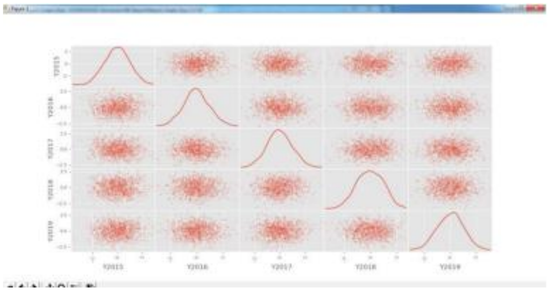
Kernel Density Estimation (KDE) Graph

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py



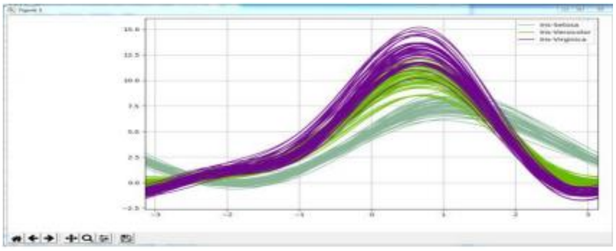
Scatter Matrix Graph

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_B.py



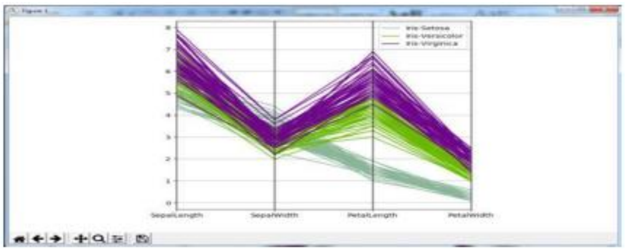
Andrews' Curves

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py



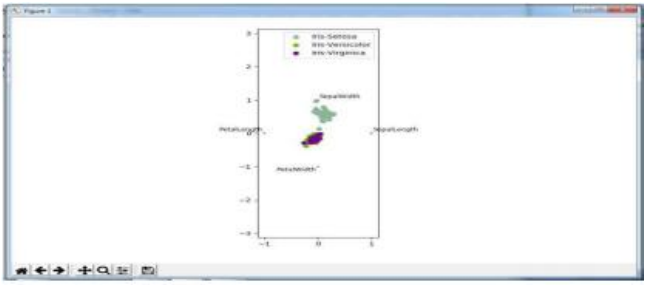
Parallel Coordinates

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py



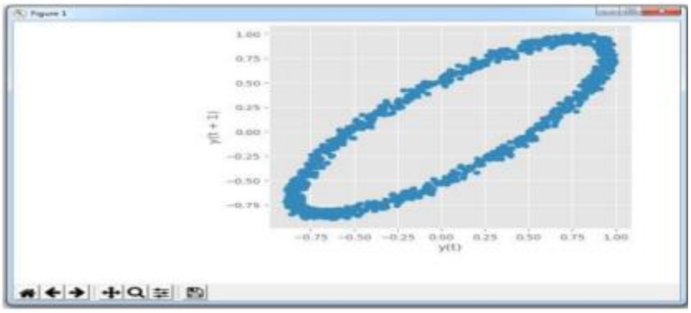
RADVIZ Method

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_C.py



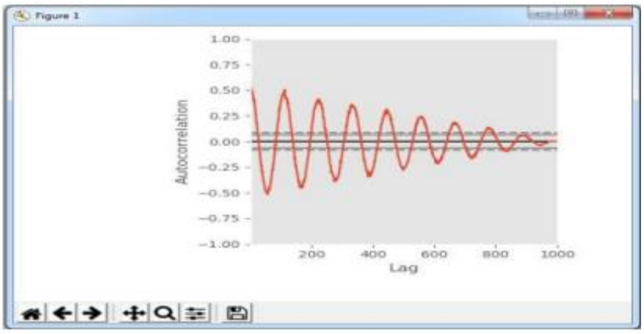
Lag Plot

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py



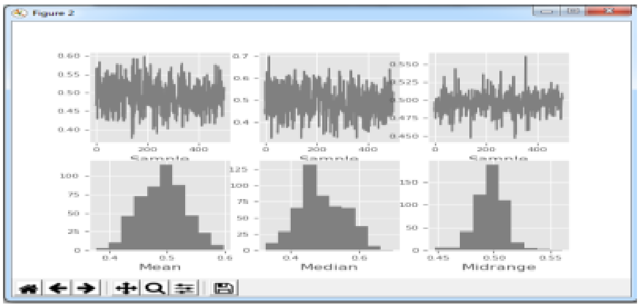
Autocorrelation Plot

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py



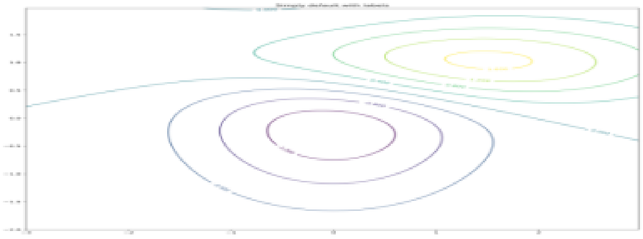
Bootstrap Plot

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_D.py



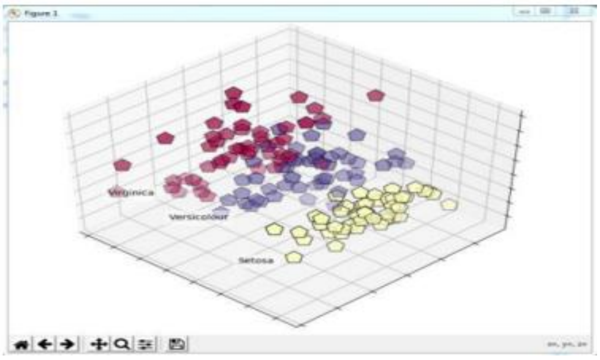
Contour Graphs

C:\VKHCG\01-Vermeulen\06-Report\Report_Graph_G.py



3D Graphs

C:\VKHCG\01-Vermeulen\06-Report\Report_PCA_IRIS.py

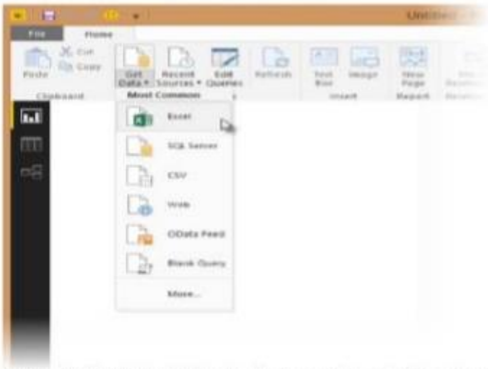


Practical 10
Data Visualization with Power BI

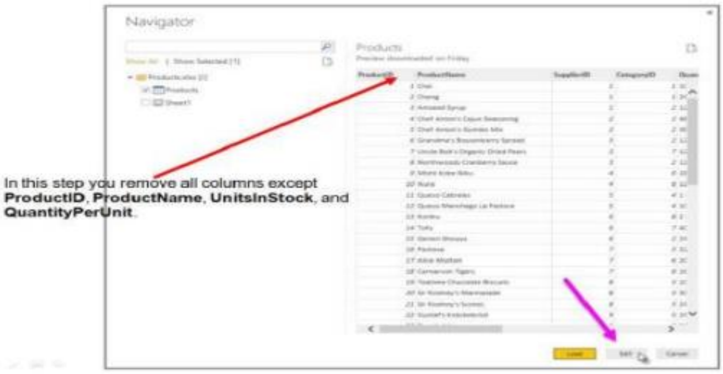
Case Study : Sales Data

Step 1: Connect to an Excel workbook

1. Launch Power BI Desktop.
2. From the Home ribbon, select **Get Data**. Excel is one of the **Most Common** data connections, so you can select it directly from the **Get Data** menu.



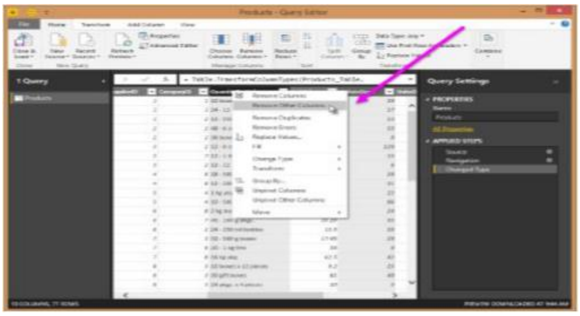
3. If you select the Get Data button directly, you can also select **File > Excel** and select **Connect**.
4. In the **Open File** dialog box, select the **Products.xlsx** file.

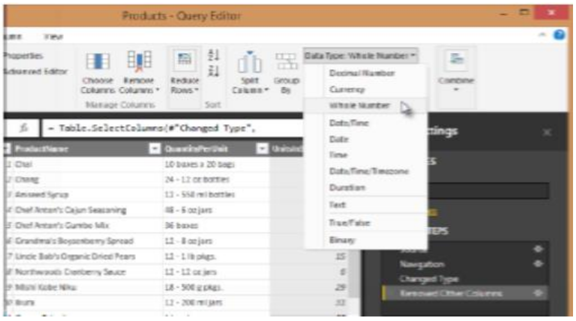


In this step you remove all columns except **ProductID**, **ProductName**, **UnitsInStock**, and **QuantityPerUnit**.

You can also open the Query Editor by selecting Edit Queries from the Home ribbon in Power BI Desktop. The following steps are performed in Query Editor.

1. In Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns (use Ctrl+Click to select more than one column, or Shift+Click to select columns that are beside each other)
2. Select Remove Columns Remove Other Columns from the ribbon, or right-click on a column header and click Remove Other Columns.





Step 3: Change the data type of the UnitsInStock column

For the Excel workbook, products in stock will always be a whole number, so in this step you confirm the UnitsInStock column‟s datatype is Whole Number.

1. Select the UnitsInStock column.
2. Select the Data Type drop-down button in the Home ribbon.

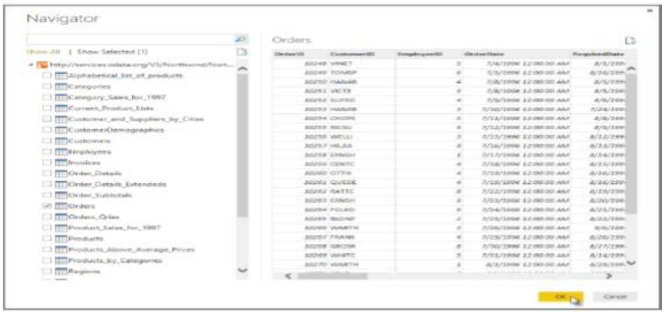
3. If not already a Whole Number, select Whole Number for data type from the drop down (the Data Type:button also displays the data type for the current selection).

Task 2: Import order data from an OData feed

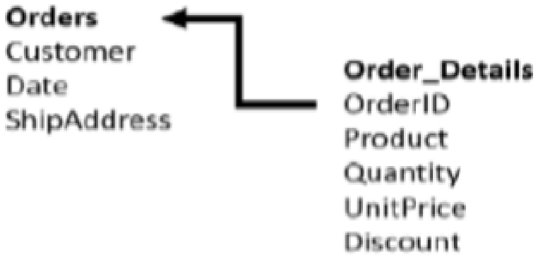
You import data into Power BI Desktop from the sample Northwind OData feed at the following URL, which you can copy (and then paste) in the steps below: http://services.odata.org/V3/Northwind/Northwind.svc/

Step 1: Connect to an OData feed

1. From the Home ribbon tab in Query Editor, select Get Data.
2. Browse to the OData Feed data source.
3. In the OData Feed dialog box, paste the URL for the Northwind OData feed.
4. Select OK



Step 2: Expand the Order_Details table



Expand the Order_Details table that is related to the Orders table, to combine the ProductID, UnitPrice, and Quantity columns from Order_Details into the Orders table.

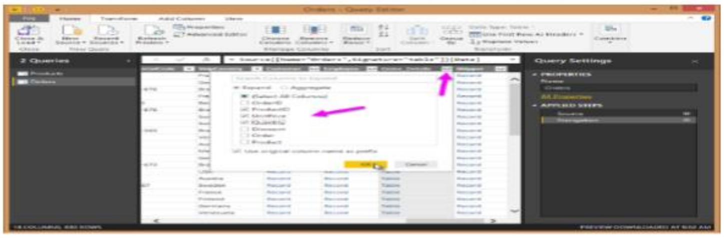
The Expand operation combines columns from a related table into a subject table. When the query runs, rows from the related table (Order_Details) are combined into rows from the subject table (Orders).

After you expand the Order_Details table, three new columns and additional rows are added to the Orders table, one for each row in the nested or related table.

1. In the Query View, scroll to the Order_Details column.
2. In the Order_Details column, select the expand icon ().
3. In the Expand drop-down: a. Select (Select All Columns) to clear all columns.

Select ProductID, UnitPrice, and Quantity.

click OK



Step 3: Remove other columns to only display columns of interest

In this step you remove all columns except OrderDate, ShipCity, ShipCountry, Order_Details.ProductID, Order_Details.UnitPrice, and Order_Details.Quantity columns. In the previous task, you used Remove Other Columns. For this task, you remove selected columns.

In the Query View, select all columns by completing a.

a. Click the first column (OrderID).
b. Shift+Click the last column (Shipper).
c. 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.

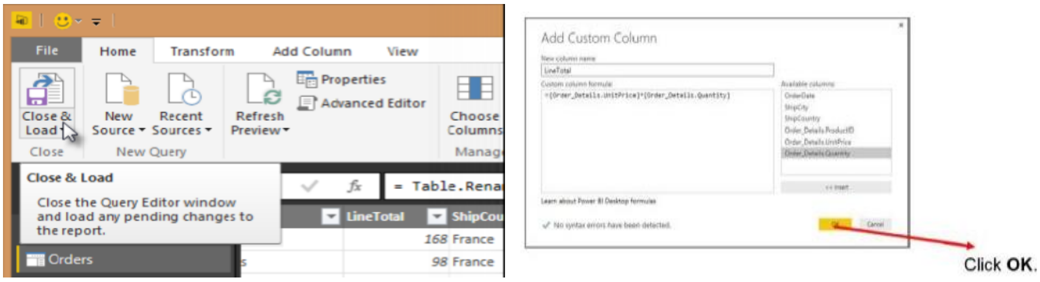
Now that only the columns we want to remove are selected, right-click on any selected column header and click Remove Columns.

Step 4: Calculate the line total for each Order_Details row

Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order_Details row.

Calculate the line total for each Order_Details row:

1. In the Add Column ribbon tab, click Add Custom Column.
2. In the Add Custom Column dialog box, in the Custom Column Formula textbox, enter [Order_Details.UnitPrice] * [Order_Details.Quantity].
3. In the New column name textbox, enter LineTotal.



Close & Load

Close the Query Editor window and load any pending changes to the report.

LineTotal

=[Order_Details.UnitPrice]*[Order_Details.Quantity]

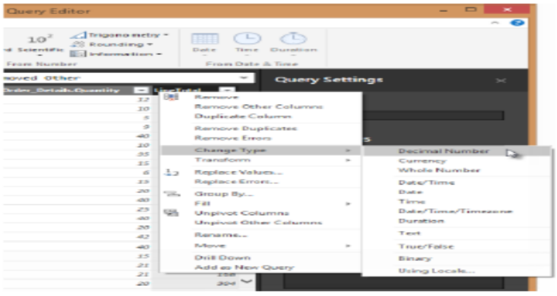
Learn about Power BI Desktop formulas

✓ No syntax errors have been detected.

Click **OK**.

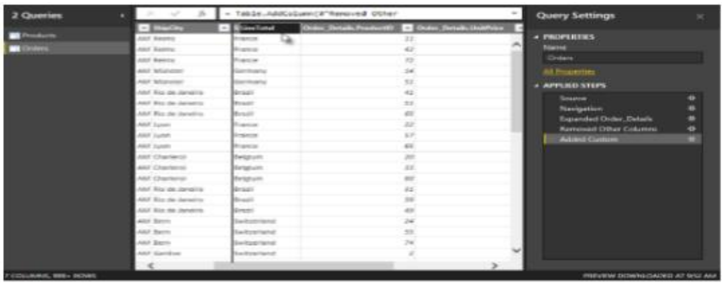
Step 5: Set the datatype of the LineTotal field

1. Right click the LineTotal column.
2. Select Change Type and choose Decimal Number.

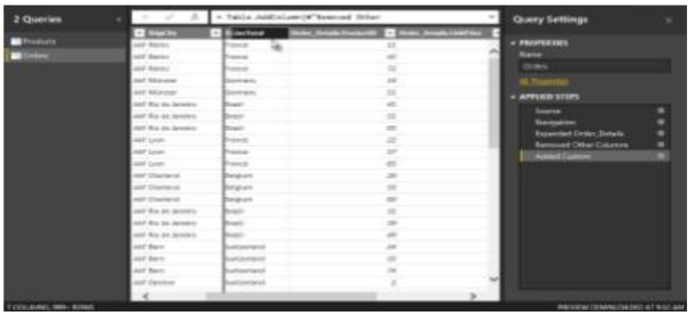


Step 6: Rename and reorder columns in the query

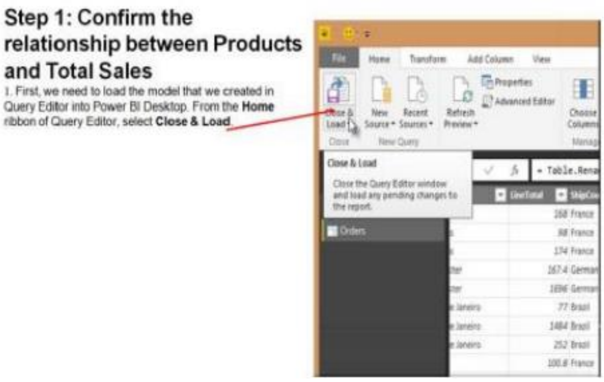
1. In Query Editor, drag the LineTotal column to the left, after ShipCountry.
2. Remove



3. Remove the Order_Details. prefix from the Order_Details.ProductID, Order_Details.UnitPrice and Order_Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.



Task 3: Combine the Products and Total Sales queries



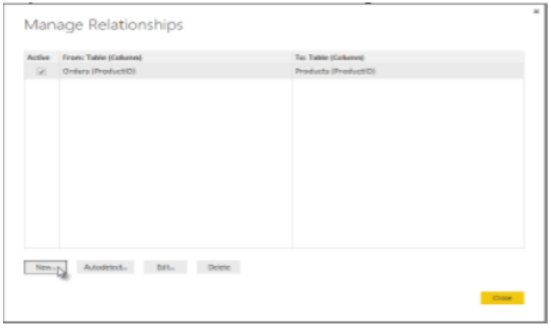
Step 1: Confirm the relationship between Products and Total Sales

1. First, we need to load the model that we created in Query Editor into Power BI Desktop. From the **Home** ribbon of Query Editor, select **Close & Load**.

Close & Load

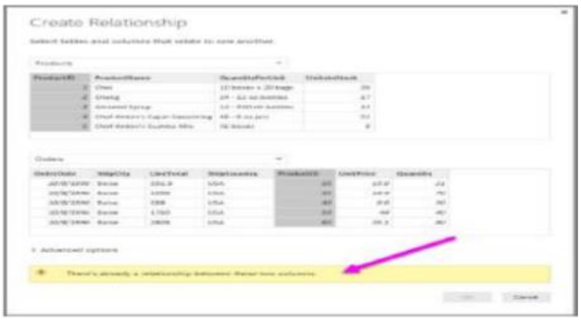
Close the Query Editor window and load any pending changes to the report.

1. Power BI Desktop loads the data from the two queries
2. Once the data is loaded, select the Manage Relationships button Home ribbon
3. Select the New… button
4. When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.



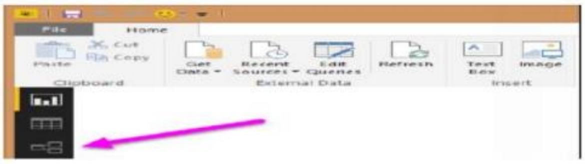
Manage Relationships

Active	From: Table (Column)	To: Table (Column)
✓	Orders (ProductID)	Products (ProductID)



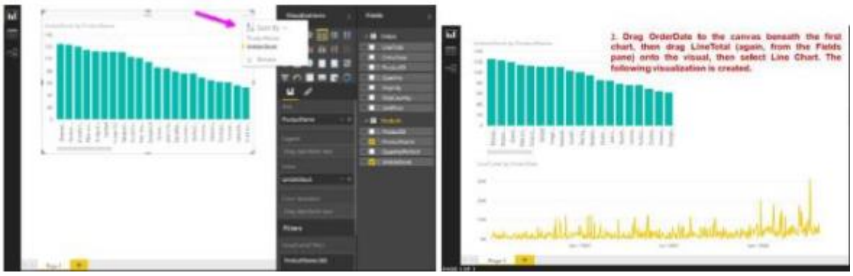
Create Relationship

5. Select Cancel, and then select Relationship view in Power BI Desktop.



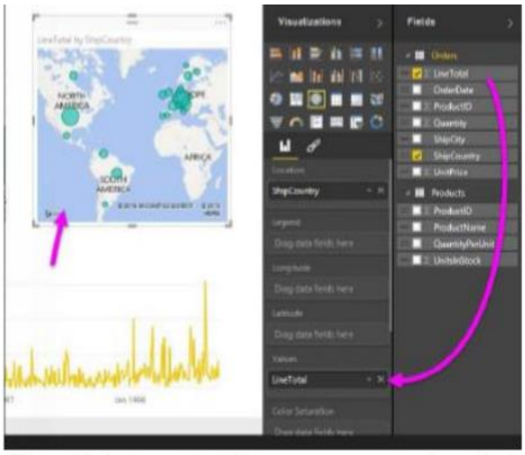
Task 4: Build visuals using your data

Step 1: Create charts showing Units in Stock by Product and Total Sales by Year

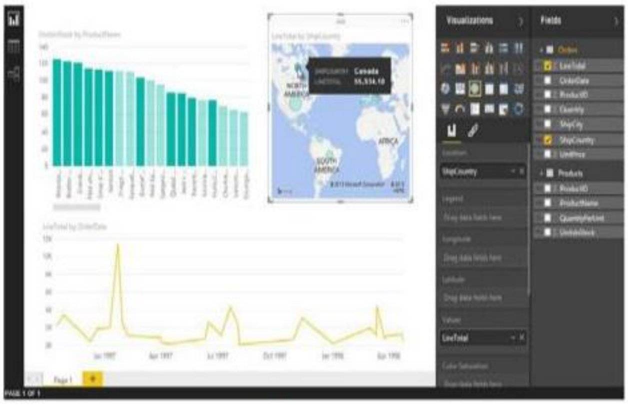


1. Drag OrderDate to the canvas beneath the first chart, then drag LineTotal (again, from the Fields pane) onto the visual, then select Line Chart. The following visualization is created.

- ☆ Next, drag ShipCountry to a space on the canvas in the top right. Because you selected a geographic field, a map was created automatically. Now drag LineTotal to the Values field; the circles on the map for each country are now relative in size to the LineTotal for orders shipped to that country.



Step 2: Interact with your report visuals to analyze further


Visualizations
Fields
Canada
Page 1
PAGE 1 OF 1