UNIT-3
DATA SCIENCE

INTRODUCTION

- The **assess super step** supports the **data** quality clean-up in the factory.

- Here we learn how to assess your data science data for invalid or erroneous data values?

- Data quality refers to the condition of a set of qualitative or quantitative variables.

- Data quality is a multidimensional measurement of the acceptability of specific data sets

In business, data quality is measured to determine whether data can be used as a basis for reliable 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 accurate and complete. The Assess superstep determines what additional processing to apply to the entries that are noncompliant.

ERROR

- Error (statistical error) describes the difference between a value obtained from a data collection process and the 'true' value for the population. The greater the error, the less representative the data are of the population.

- Error calculation is the process of isolating, observing, and diagnosing various ML predictions, thereby helping understand the highs and lows of the model.

ACTIONS FOR DIFFERENT ERRORS

- Accept the Error
- Reject the Error
- Correct the Error
- Create a Default Value

Accept the error

Take note that if you accept the error, you will affect data science techniques and algorithms that perform classification, such as binning, regression, clustering, and decision trees etc.

Reject the error

- Add a quality flag and use this flag to avoid this erroneous data being used in data science techniques and algorithms that it will negatively affect.

User	Device	OS	Transactions
A	Mobile	Android	5
B	Mobile	Windows	3
C	Tablet	NA	4
D	NA	Android	1
E	Mobile	iOS	2

Correct the error

- Spelling mistakes in customer names, addresses, and locations are a common source of errors, which are methodically corrected.

- If there are variations on a name, It is recommended that you set one data source as the “master” and keep the data consolidated and correct across all the databases using that master as your primary source.

First Name	Email	Date of Birth	Country	Height
Rakesh	rakesh@example.com	23/01/1990	India	1.49m
Samuel	samuel_329@example.com	20/09/2001		1.67m
Rob	choupipoune@supermail.eu	12 Sept. 1984	Madagascar	5'2
Mark	marco23@example.com, mc23@supermail.eu	10/02/1978	24	1.65m
Harry	helloworld@mail.example.com	04/25/1975	Germany	1.34m
Honey	honey2019@supermail.eu	01/01/1970	Canada	2.8m
Samuël	samuel_329@example.com		Benin	1.45m

Create a default value

- Most system developers assume that if the business doesn't enter the value, they should enter a default value.
- NaN is the default missing value marker for reasons of computational speed and convenience. This is a sentinel value, in the sense that it is a dummy data or flag value that can be easily detected and worked with using functions in pandas.
- It is suggested to discuss default values with your customer in detail and agree on an official "missing data" value.

Analysis of Data

- One of the causes of data quality issues is in source data that is housed in a patchwork of operational systems and enterprise applications.

- Each of these data sources can have scattered or misplaced values, outdated and duplicate records, and inconsistent (or undefined) data standards and formats across customers, products, transactions, financials and more.

Analysis of Data

To maintain the accuracy and value of the business-critical operational information that impact strategic decision-making, businesses should implement a data quality strategy that embeds data quality techniques into their business processes and into their enterprise applications and data integration.

QUALITY DIMENSIONS

Completeness

Consistency

Timeliness

Conformity

Accuracy

Integrity

COMPLETENESS

Completeness is defined as expected comprehensiveness. Data can be complete even if optional data is missing. As long as the data meets the expectations then the data is considered complete.

CONSISTENCY

- Consistency means data across all systems reflects the same information and are in synch with each other across the enterprise.

- Examples: A business unit status is closed but there are sales for that business unit. Employee status is terminated but pay status is active.

Timeliness

- Timeliness referes to whether information is available when it is expected and needed. Timeliness of data is very important.

This is reflected in:

- Companies that are required to publish their quarterly results within a given frame of time
- Customer service providing up-to date information to the customers
- Credit system checking in real-time on the credit card account activity

Conformity

- Conformity means the data is following the set of standard data definitions like data type, size and format. For example, date of birth of customer is in the format “mm/dd/yyyy” .
- BUT
- Do data values comply with the specified formats?

Accuracy

- Accuracy is the degree to which data correctly reflects the real world object OR an event being described. Examples:

- Sales of the business unit are the real value.

- Address of an employee in the employee database is the real address.

Integrity

- Integrity means validity of data across the relationships and ensures that all data in a database can be traced and connected to other data.

- If there is an address relationship data without a customer then that data is not valid and is considered an orphaned record.

PRACTICAL ACTIONS

- The package enables several automatic error-management features.

Missing Values in Pandas

Following are four basic processing concepts.

1. Drop the Columns Where All Elements Are Missing Values
2. Drop the Columns Where Any of the Elements Is Missing Values
3. Keep Only the Rows That Contain a Maximum of Two Missing Values
4. Fill All Missing Values with the Mean, Median, Mode, Minimum

Drop the Columns Where All Elements Are Missing Values

- Importing data
- Step 1: Importing necessary libraries import os import pandas as pd
- Step 2: Changing the working directory os.chdir("D:\Pandas“)

How to represent missing data in pandas?

- None: None is a Python singleton object that is often used for missing data in Python code.

- NaN: NaN (an acronym for Not a Number), is a special floating-point value recognized by all systems that use the standard IEEE floating-point representation.

dropna() function

In order to drop a null values from a dataframe, we used dropna() function this function drop Rows/Columns of datasets with Null values in different ways.

- Syntax:

- DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

Parameters Used

Parameters:

- *axis*: axis takes int or string value for rows/columns. Input can be 0 or 1 for Integer and 'index' or 'columns' for String.
- *how*: how takes string value of two kinds only ('any' or 'all'). 'any' drops the row/column if ANY value is Null and 'all' drops only if ALL values are null.
- *thresh*: thresh takes integer value which tells minimum amount of na values to drop.
- *subset*: It's an array which limits the dropping process to passed rows/columns through list.
- *inplace*: It is a boolean which makes the changes in data frame itself if True.

Example

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

Here , we drop those columns that has all NaN values.

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
                   [np.nan, np.nan, np.nan, 5]],
                  columns=list('ABCD'))
df # it will print the data frame

df.dropna(axis=1, how='all')  # this code will delete the columns with all null values.
```

Here, axis=1 means columns and how='all' means drop the columns with all NaN values.

Output:

	A	B	D
0	NaN	2.0	0
1	3.0	4.0	1
2	NaN	NaN	5

Drop the Columns Where Any of the Elements Is Missing Values

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

Here, column A, B and C are having all NaN values.

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
                   [np.nan, np.nan, np.nan, 5]],
                  columns=list('ABCD'))
df # it will print the data frame
```

Output:

	D
0	0
1	1
2	5

```
df.dropna(axis=1, how='any')  # this code will delete the columns with all null values.
```

Here, axis=1 means columns and how='any' means drop the columns with one or nore NaN values.

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

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5

```
# importing pandas as pd
import pandas as pd
import numpy as np

df = pd.DataFrame([[np.nan, 2, np.nan, 0], [3, 4, np.nan, 1],
                   [np.nan, np.nan, np.nan, 5]],
                  columns=list('ABCD'))
df


df.dropna(thresh=2)
  # this code will delete the rows with more than two null values.
```

Here, thresh=2 means maximum two NaN will be allowed per row.

Output:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1

Fill All Missing Values with the Mean, Median, Mode, Minimum

- In pandas, .fillna can be used to replace NA's with a specified value.

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	21	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	8
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                         'Basket5', 'Basket6'])

df

df.fillna(df.mean())
```

Here, the mean of Apple Column = (10 + 7 + 55 + 15 + 7)/5 = 18.8. So, Nan value is replaced by 18.8.

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	8.25	30	40
Basket 2	7	14	21	28
Basket 3	55	8.25	8	12
Basket 4	15	14	13.8	8
Basket 5	7	1	1	18
Basket 6	18.8	4	9	2

Replacing Nan values with median:

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	21	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	8
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                         'Basket5', 'Basket6'])

df

df.fillna(df.median())
```

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	9.0	30	40
Basket 2	7	14	21	28
Basket 3	55	9.0	8	12
Basket 4	15	14	9.0	8
Basket 5	7	1	1	12.0
Basket 6	10.0	4	9	2

Here, the median of Apple Column = (7, 7, 10, 15, 55) = 10. So, Nan value is replaced by 10.

Replacing Nan values with mode

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	8	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	12
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 8, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 12], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                       'Basket5', 'Basket6'])

df

for column in df.columns:
    df[column].fillna(df[column].mode()[0], inplace=True)

df
```

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	14	30	40
Basket 2	7	14	8	28
Basket 3	55	14	8	12
Basket 4	15	14	8	12
Basket 5	7	1	1	12
Basket 6	7.0	4	9	2

Here, the mode of Apple Column = (10, 7, 55, 15, 7) = 7. So, Nan value is replaced by 7.

Replacing Nan values with min:

	Apple	Orange	Banana	Pear
Basket 1	10	NaN	30	40
Basket 2	7	14	21	28
Basket 3	55	NaN	8	12
Basket 4	15	14	NaN	8
Basket 5	7	1	1	NaN
Basket 6	NaN	4	9	2

```
import pandas as pd
import numpy as np

df = pd.DataFrame([[10, np.nan, 30, 40], [7, 14, 21, 28], [55, np.nan, 8, 12],
                   [15, 14, np.nan, 8], [7, 1, 1, np.nan], [np.nan, 4, 9, 2]],
                  columns=['Apple', 'Orange', 'Banana', 'Pear'],
                  index=['Basket1', 'Basket2', 'Basket3', 'Basket4',
                         'Basket5', 'Basket6'])

df

df.fillna(df.min())
```

Output:

	Apple	Orange	Banana	Pear
Basket 1	10	1	30	40
Basket 2	7	14	21	28
Basket 3	55	1	8	12
Basket 4	15	14	1	8
Basket 5	7	1	1	2
Basket 6	7	4	9	2

Here, the minimum of Apple Column = (10, 7, 55, 15, 7) = 7. So, Nan value is replaced by 7.

ENGINEERING A PRACTICAL ASSESS SUPERSTEP

- **Vermeulen PLC**

- Vermeulen PLC has two primary processes **network routing** and **job scheduling.**

Creating a Network Routing Diagram

- **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.**

```
import sys
import os
import pandas as pd
##################################################################
pd.options.mode.chained_assignment = None
##################################################################
##################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#################################')
print('Working Base:',Base,'using',sys.platform)
print('#################################') pri
nt('#################################')
print('Working Base:',Base,'using',sys.platform)
print('#################################')
##################################################################
sInputFileName1='01-Retrieve/01-EDS/01-R/Retrieve_Country_Code.csv'
sInputFileName2='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
```

pandas.options.mode.chained_assignment which can be set to

None, ignoring the warning

"warn", printing a warning message

"raise", raising an exception

```
sOutputFileName='Assess-Network-Routing-Company.csv'
Company='01-Vermeulen'
##################################################################
##################################################################
### Import Country Data
##################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName1
print('#################################')
print('Loading:',sFileName)
print('#################################')
CountryData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('Loaded Country:',CountryData.columns.values)
print('#################################')
##################################################################
## Assess Country Data
##################################################################
print('#################################')
print('Changed:',CountryData.columns.values)
```

```
CountryData.rename(columns={'Country': 'Country_Name'}, inplace=True)
CountryData.rename(columns={'ISO-2-CODE': 'Country_Code'}, inplace=True)
CountryData.drop('ISO-M49', axis=1, inplace=True)
CountryData.drop('ISO-3-Code', axis=1, inplace=True)
CountryData.drop('RowID', axis=1, inplace=True)
print('To:',CountryData.columns.values)
print('###################################')
#############################################################
```

In pandas axis = 0 refers to horizontal axis or rows and axis = 1 refers to vertical axis or columns.

```
### Import Company Data
#############################################################
sFileName=Base + '/' + Company + '/' + sInputFileName2
print('####################################')
print('Loading:',sFileName)
print('####################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('Loaded Company:',CompanyData.columns.values)
print('####################################')
#############################################################
## Assess Company Data
#############################################################
print('####################################')
print('Changed:',CompanyData.columns.values)
CompanyData.rename(columns={'Country':'Country_Code'}, inplace=True)
print('To:',CompanyData.columns.values)
print('####################################')
```

```
### Import Customer Data
#################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName3
print('###################################')
print('Loading:',sFileName)
print('###################################')
CustomerRawData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('###################################')
print('Loaded Customer:',CustomerRawData.columns.values)
print('###################################')
#################################################################
CustomerData=CustomerRawData.dropna(axis=0, how='any')
```

```
print('####################################')
print('Remove Blank Country Code')
print('Reduce Rows from',CustomerRawData.shape[0],'to',CustomerData.
shape[0])
print('####################################')
############################################################
print('####################################')
print('Changed:',CustomerData.columns.values)
CustomerData.rename(columns={'Country':'Country_Code'},inplace=True)
print('To:',CustomerData.columns.values)
print('####################################')
############################################################
print('####################################')
print('Merge Company and Country Data')
print('####################################')
```

```
CompanyNetworkData=pd.merge(
        CompanyData,
        CountryData,
        how='inner',
        on='Country_Code'
        )
#####################################################################
print('###################################')
print('Change',CompanyNetworkData.columns.values)
for i in CompanyNetworkData.columns.values:
    j='Company_'+i
    CompanyNetworkData.rename(columns={i:j},inplace=True)
print('To',CompanyNetworkData.columns.values)
print('###################################')
#####################################################################
#####################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
```

Pandas uses “inner” merge by default. This keeps only the common values in both the left and right dataframes for the merged data.

```
sFileName=sFileDir + '/' + sOutputFileName
print('################################')
print('Storing:',sFileName)
print('################################')
CompanyNetworkData.to_csv(sFileName, index = False, encoding="latin-1")
#########################################################
#########################################################
print('################################')
print('### Done!! ######################')
```

Company_ Country_Code	Company_ Place_Name	Company_Latitude	Company_ Longitude	Company_Country_ Name
US	New York	40.7528	-73.9725	United States of America
US	New York	40.7214	-74.0052	United States of America
US	New York	40.7662	-73.9862	United States of America

Directed Acyclic Graph (DAG)

- A directed acyclic graph is a specific graph that only has one path through the graph.

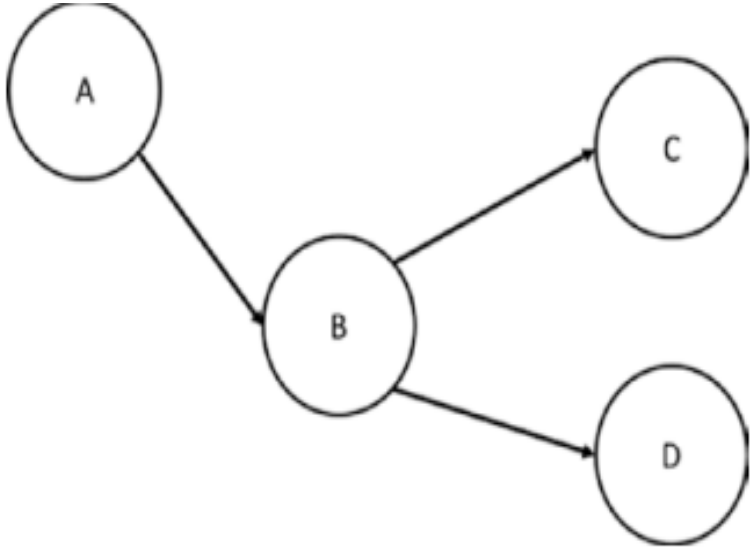


Figure 8-5. *This graph is a DAG*

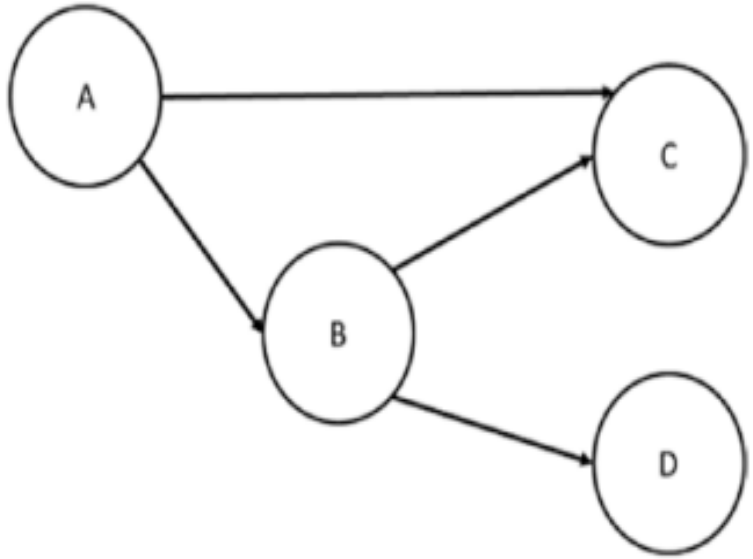


Figure 8-6. *This graph is not a DAG*

Building a DAG for Scheduling Jobs

- DAG is also used for recording task scheduling and process interdependencies in processing data. Here , we use a new Python library called networkx.

```
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
#################################################################
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
print('##################################')
print('Working Base:',Base,'using',sys.platform)
print('##################################')
#################################################################
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName1='Assess-DAG-Company-Country.png'
sOutputFileName2='Assess-DAG-Company-Country-Place.png'
Company='01-Vermeulen'
```

```
### Import Company Data
#########################################################################
sFileName=Base + '/' + Company + '/' + sInputFileName
print('####################################')
print('Loading:',sFileName)
print('####################################')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False,
encoding="latin-1")
print('Loaded Company:',CompanyData.columns.values)
print('####################################')
#########################################################################
print(CompanyData)
print('####################################')
```

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

We will now create two directed graphs: G1 and G2 using the G=DiGraph(), if you just wanted a graph you would use G=Graph().

```
#################################################################
G1=nx.DiGraph()
G2=nx.DiGraph()
#################################################################
```

We will now create a node for each of the places in our data. For this, use the G.add_node() function.

Note If you execute G.add_node("A") and then G.add_node("A") again, you only get one node: "A". The graph automatically resolves any duplicate node requests.

We will now loop through all the country and place-name-country records.

```
for i in range(CompanyData.shape[0]):
    G1.add_node(CompanyData['Country'][i])
    sPlaceName= CompanyData['Place_Name'][i] + '-' + CompanyData['Country'][i]
    G2.add_node(sPlaceName)
```

We will now interconnect all the nodes (G.nodes()), by looping though them in pairs.

```
print('###################################')
for n1 in G1.nodes():
    for n2 in G1.nodes():
        if n1 != n2:
            print('Link:',n1,' to ', n2)
            G1.add_edge(n1,n2)
print('###################################')
```

Now, we can see the nodes and edges you created.

```
print('###################################')
print("Nodes of graph:")
print(G1.nodes())
```

```
print("Edges of graph:")
print(G1.edges())
print('#################################')
```

We can now save our graph, as follows:

```
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
sFileName=sFileDir + '/' + sOutputFileName1
print('#################################')
print('Storing:', sFileName)
print('#################################')
```

Now, you can display the graph as a picture.

```
nx.draw(G1,pos=nx.spectral_layout(G1),
        nodecolor='r',edge_color='b',
        with_labels=True,node_size=8000,
        font_size=12)
```

Spectral layout is a class of algorithm for drawing graphs. The idea of the layout is to compute the two largest (or smallest) eigenvalues and corresponding eigenvectors of the Laplacian matrix of the graph and then use those for actually placing the nodes.

draw(G[, pos, ax])	Draw the graph G with Matplotlib.

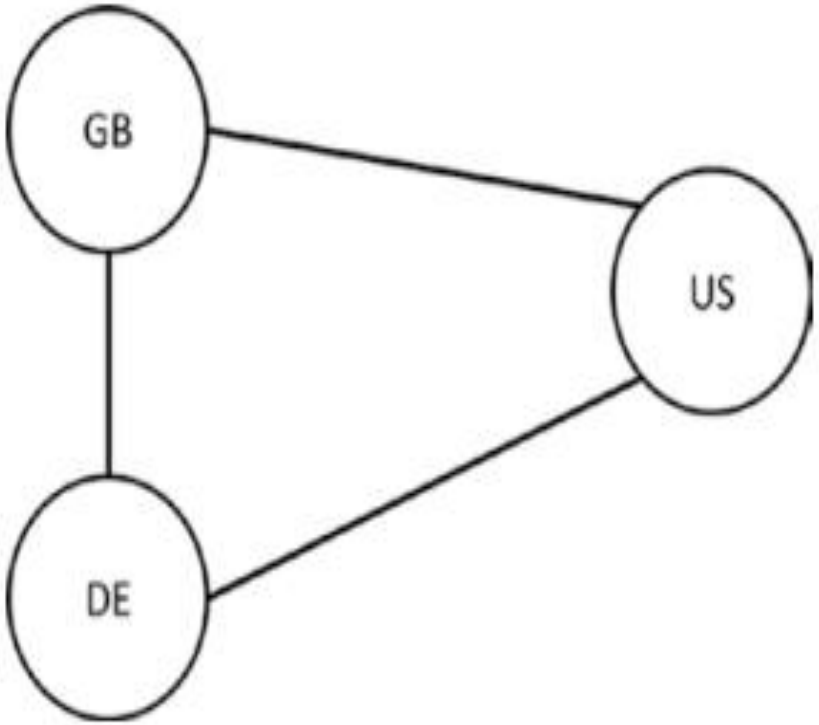
```
plt.savefig(sFileName) # save as png
plt.show() # display
#################################################################
```

Now, you can complete the second graph.

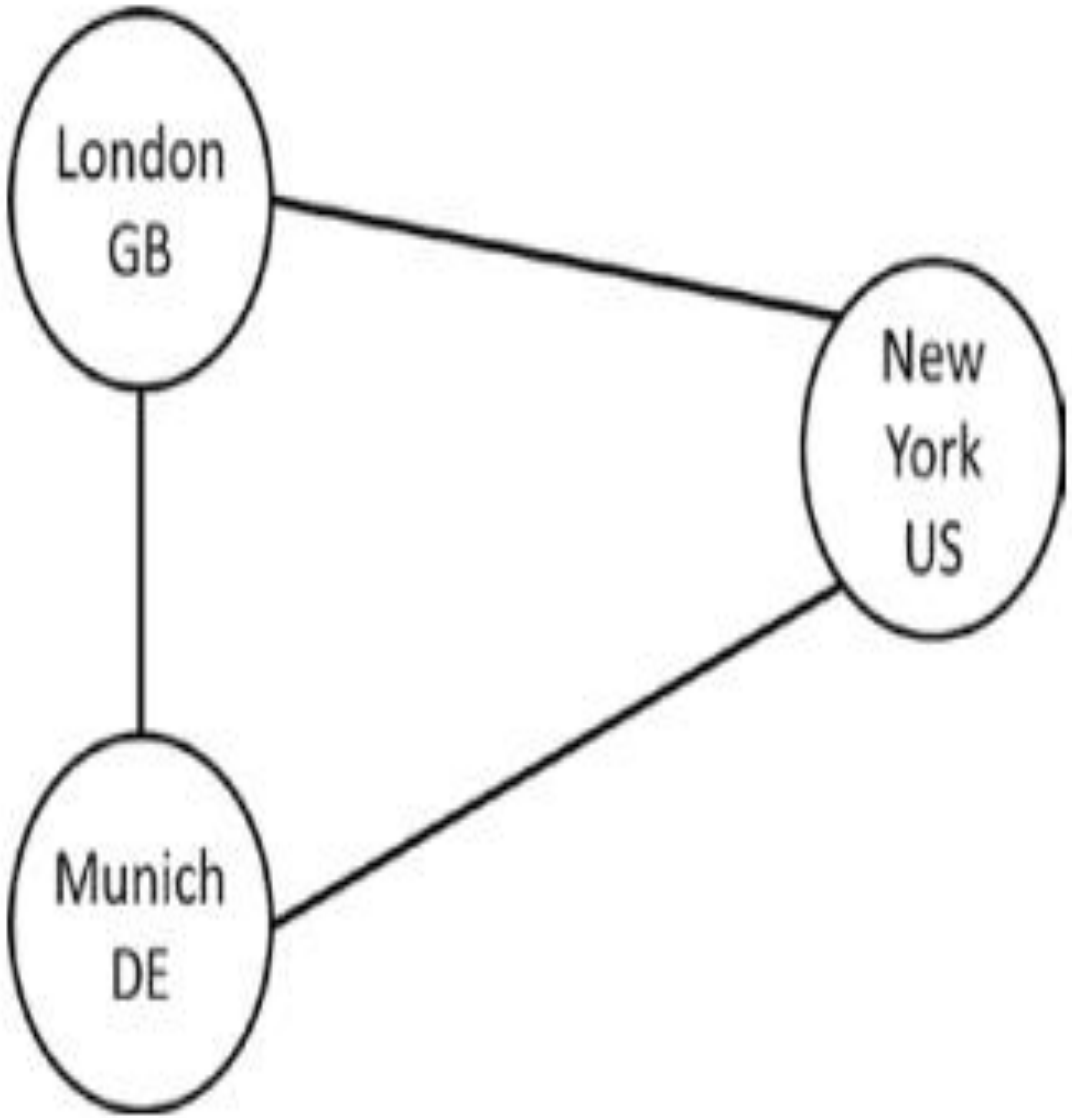
```
print('###################################')
for n1 in G2.nodes():
    for n2 in G2.nodes():
        if n1 != n2:
            print('Link:',n1,'to',n2)
            G2.add_edge(n1,n2)
print('###################################')

print('###################################')
print("Nodes of graph:")
print(G2.nodes())
```

```
print("Edges of graph:")
print(G2.edges())
print('################################')
#################################################################
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#################################################################
sFileName=sFileDir + '/' + sOutputFileName2
print('################################')
print('Storing:', sFileName)
print('################################')
nx.draw(G2,pos=nx.spectral_layout(G2),
        nodecolor='r',edge_color='b',
        with_labels=True,node_size=8000,
        font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#################################################################
```

OUTPUT



-7. Assess-DAG-Company-Country.png file



Assess-DAG-Company-Country-Place.png file

THANKS