In [1]:
```python
import pandas as pd
```

Question 1

Use the following data to create the Data Frame and store it in a variable named "Data"

In [7]:
```python
Data = pd.DataFrame({"x":[103,207,315,400,None,605,355,677,197,869],
                     "y":[18976,45789,None,78964,45683,None,78965,None,68546,20015]
                     })
Data
```

Out[7]:

	x	y
0	103.0	18976.0
1	207.0	45789.0
2	315.0	NaN
3	400.0	78964.0
4	NaN	45683.0
5	605.0	NaN
6	355.0	78965.0
7	677.0	NaN
8	197.0	68546.0
9	869.0	20015.0

Check the presence of missing values

In [10]:
```python
Data.isnull()
```

Out[10]:

	x	y
0	False	False
1	False	False
2	False	True
3	False	False
4	True	False
5	False	True
6	False	False
7	False	True
8	False	False
9	False	False

In [54]:
```python
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   x       9 non-null      float64
 1   y       7 non-null      float64
dtypes: float64(2)
memory usage: 288.0 bytes
```

Use the imputation approach to handle missing values`

- Droping the NaN values

In [11]:
```python
Data.dropna(axis=0, inplace=True)
```

In [13]:
```python
Data
```

Out[13]:

	x	y
0	103.0	18976.0
1	207.0	45789.0
3	400.0	78964.0
6	355.0	78965.0
8	197.0	68546.0
9	869.0	20015.0

In [19]:
```python
data = pd.DataFrame({"x":[103,207,315,400,None,605,355,677,197,869],
                     "y":[18976,45789,None,78964,45683,None,78965,None,68546,20015]
                     })
data
```

Out[19]:

	x	y
0	103.0	18976.0
1	207.0	45789.0
2	315.0	NaN
3	400.0	78964.0
4	NaN	45683.0
5	605.0	NaN
6	355.0	78965.0
7	677.0	NaN
8	197.0	68546.0
9	869.0	20015.0

- Filling the missing values with mean

In [20]:
```python
mean1 = int(data["x"].mean())
mean2 = int(data["y"].mean())
data["x"].fillna(mean1, inplace=True)
data["y"].fillna(mean2, inplace=True)
data
```

Out[20]:

	x	y
0	103.0	18976.0
1	207.0	45789.0
2	315.0	50991.0
3	400.0	78964.0
4	414.0	45683.0
5	605.0	50991.0
6	355.0	78965.0
7	677.0	50991.0
8	197.0	68546.0
9	869.0	20015.0

- Filling the missing values with mean

In [21]:
```python
Data_copy = pd.DataFrame({"x":[103,207,315,400,None,605,355,677,197,869],
                     "y":[18976,45789,None,78964,45683,None,78965,None,68546,20015]
                     })
Data_copy
```

Out[21]:

	x	y
0	103.0	18976.0
1	207.0	45789.0
2	315.0	NaN
3	400.0	78964.0
4	NaN	45683.0
5	605.0	NaN
6	355.0	78965.0
7	677.0	NaN
8	197.0	68546.0
9	869.0	20015.0

In [22]:
```python
median1 = int(data["x"].median())
median2 = int(data["y"].median())
Data_copy["x"].fillna(median1, inplace=True)
Data_copy["y"].fillna(median2, inplace=True)
Data_copy
```

Out[22]:

	x	y
0	103.0	18976.0
1	207.0	45789.0
2	315.0	50991.0
3	400.0	78964.0
4	377.0	45683.0
5	605.0	50991.0
6	355.0	78965.0
7	677.0	50991.0
8	197.0	68546.0
9	869.0	20015.0

Perform Normalization

In [25]:
```python
Data_copy
```

Out[25]:

	x	y
0	103.0	18976.0
1	207.0	45789.0
2	315.0	50991.0
3	400.0	78964.0
4	377.0	45683.0
5	605.0	50991.0
6	355.0	78965.0
7	677.0	50991.0
8	197.0	68546.0
9	869.0	20015.0

- Using min-max normalization with or without an inbuilt library

In [27]:
```python
Data_copying = Data_copy.copy()
for column in Data.columns:
    Data_copying[column] = (Data_copying[column] - Data_copying[column].min()) / (Data_copying[column].max() - Dat

Data_copying
```

Out[27]:

	x	y
0	0.000000	0.000000
1	0.135770	0.446965
2	0.276762	0.533681
3	0.387728	0.999983
4	0.357702	0.445198
5	0.655352	0.533681
6	0.328982	1.000000
7	0.749347	0.533681
8	0.122715	0.826318
9	1.000000	0.017320

- Using Z score standardization with or without an inbuilt library

In [28]:
```python
z_data = Data_copy
for column in z_data:
    z_data[column] = (z_data[column] - z_data[column].mean()) / z_data[column].std()
z_data
```

Out[28]:

	x	y
0	-1.286649	-1.535415
1	-0.851490	-0.249488
2	-0.399593	-0.000005
3	-0.043934	1.341554
4	-0.140172	-0.254572
5	0.813832	-0.000005
6	-0.232224	1.341602
7	1.115096	-0.000005
8	-0.893332	0.841917
9	1.918467	-1.485585

Stored the pre-processed data in the external file “preprocessed123.csv”

In [58]:
```python
z_data.to_csv("preprocessed123(Z-Score).csv")
Data_copying.to_csv("preprocessed123(Mim-Max).csv")
```

Question 2

Download the dataset from the link

In [44]:
```python
social_network = pd.read_csv("Social_Network_Ads.csv")
social_network
```

Out[44]:

	Age	EstimatedSalary
0	19.0	19000.0
1	35.0	20000.0
2	26.0	43000.0
3	27.0	57000.0
4	19.0	76000.0
...	...	...
395	46.0	41000.0
396	51.0	23000.0
397	50.0	NaN
398	36.0	33000.0
399	49.0	36000.0

400 rows × 2 columns

Check the presence of missing value

In [45]:
```python
social_network.isnull()
```

Out[45]:

	Age	EstimatedSalary
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
395	False	False
396	False	False
397	False	True
398	False	False
399	False	False

400 rows × 2 columns

In [46]:
```python
social_network.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Age              396 non-null    float64
 1   EstimatedSalary  396 non-null    float64
dtypes: float64(2)
memory usage: 6.4 KB
```

Use the imputation approach to handle missing values

- Dropping the NaN values

In [47]:
```python
social_network_drop = social_network.copy()
social_network_drop.dropna(axis=0, inplace=True)
social_network_drop
```

Out[47]:

	Age	EstimatedSalary
0	19.0	19000.0
1	35.0	20000.0
2	26.0	43000.0
3	27.0	57000.0
4	19.0	76000.0
...	...	...
394	39.0	59000.0
395	46.0	41000.0
396	51.0	23000.0
398	36.0	33000.0
399	49.0	36000.0

392 rows × 2 columns

- Filling the NaN values with Mean

In [50]:
```python
social_network_mean = social_network.copy()
mean1 = int(social_network_mean["Age"].mean())
meen2 = int(social_network_mean["EstimatedSalary"].mean())
social_network_mean["Age"].fillna(mean1, inplace=True)
social_network_mean["EstimatedSalary"].fillna(mean2, inplace=True)
social_network_mean
```

Out[50]:

	Age	EstimatedSalary
0	19.0	19000.0
1	35.0	20000.0
2	26.0	43000.0
3	27.0	57000.0
4	19.0	76000.0
...	...	...
395	46.0	41000.0
396	51.0	23000.0
397	50.0	50991.0
398	36.0	33000.0
399	49.0	36000.0

400 rows × 2 columns

- Filling the NaN Values with Median

In [52]:
```python
social_network_median = social_network.copy()
maedian1 = int(social_network_mean["Age"].median())
median2 = int(social_network_mean["EstimatedSalary"].median())
social_network_median["Age"].fillna(median1, inplace=True)
social_network_median["EstimatedSalary"].fillna(median2, inplace=True)
social_network_median
```

Out[52]:

	Age	EstimatedSalary
0	19.0	19000.0
1	35.0	20000.0
2	26.0	43000.0
3	27.0	57000.0
4	19.0	76000.0
...	...	...
395	46.0	41000.0
396	51.0	23000.0
397	50.0	69500.0
398	36.0	33000.0
399	49.0	36000.0

400 rows × 2 columns

Perform Normalization

- Using min-max normalization with or without an inbuilt library

In [55]:
```python
social_network_minmax = social_network.copy()
for column in social_network_minmax:
    social_network_minmax[column] = (social_network_minmax[column] - social_network_minmax[column].min()) / (socia

social_network_minmax
```

Out[55]:

	Age	EstimatedSalary
0	0.023810	0.029630
1	0.404762	0.037037
2	0.190476	0.207407
3	0.214286	0.311111
4	0.023810	0.451852
...	...	...
395	0.666667	0.192593
396	0.785714	0.059259
397	0.761905	NaN
398	0.428571	0.133333
399	0.738095	0.155556

400 rows × 2 columns

- Using Z score standardization with or without an inbuilt library

In [56]:
```python
z_social_network = social_network.copy()
for column in z_social_network:
    z_social_network[column] = (z_social_network[column] - z_social_network[column].mean()) / z_social_network[col
z_social_network
```

Out[56]:

	Age	EstimatedSalary
0	-1.779591	-1.490638
1	-0.257111	-1.461305
2	-1.113506	-0.786651
3	-1.018351	-0.375993
4	-1.779591	0.181330
...	...	...
395	0.789594	-0.845317
396	1.265369	-1.373307
397	1.170214	NaN
398	-0.161956	-1.079979
399	1.075059	-0.991981

400 rows × 2 columns

Stored the pre-processed data in the external file “preprocessed123.csv”

In [59]:
```python
z_social_network.to_csv("preprocessed123(Z-scoree).csv")
social_network_minmax.to_csv("preprocessed123(mim-maxx).csv")
```