# Ex-01_DS_Data_Cleansing

## AIM

To read the given data and perform data cleaning and save the cleaned data to a file.

## Explanation

Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect ,incompleted , irrelevant , duplicated or improperly formatted. Data cleaning is not simply about erasing data ,but rather finding a way to maximize datasets accuracy without necessarily deleting the information.

## ALGORITHM

### STEP 1

Read the given Data

### STEP 2

Get the information about the data

### STEP 3

Remove the null values from the data

### STEP 4

Save the Clean data to the file

```
# CODE FOR DATA 1:

import pandas as pd
df=pd.read_csv("Data_set.csv")
df.head(5)

df.info()

df.isnull()

df.isnull().sum()

df['show_name']=df['show_name'].fillna(df['aired_on'].mode()[0])
df['aired_on']=df['aired_on'].fillna(df['aired_on'].mode()[0])
df['original_network']=df['original_network'].fillna(df['aired_on'].mode()[0])
df.head()

df['rating']df['rating'].fillna(df['rating'].mean())
df['current_overall _rank']=df['current_overall_rank'].fillna(df['current_overall_rank'].mean())
df.head()

df['watchers']=df['watchers'].fillna(df['watchers'].median())
```

df.head()

df.info()

df.isnull().sum()

**OUPUT FOR DATA 1:**

DATA:

[5] df.head(5)

| | show_name | country | num_episodes | aired_on | original_network | rating | current_overall_rank | lifetime_popularity_rank | watchers |
|---|---|---|---|---|---|---|---|---|---|
| | NaN | South Korea | 16 | Friday, Saturday | tvN | 8.9 | 33.0 | 1 | 111706.0 |
| | NaN | South Korea | 16 | Friday, Saturday | jTBC | 8.7 | 89.0 | 2 | 100950.0 |
| | Descendants of the Sun | South Korea | 16 | Wednesday, Thursday | KBS2 | 8.7 | 77.0 | 3 | 96318.0 |
| | Boys Over Flowers | South Korea | 25 | Monday, Tuesday | KBS2 | 7.7 | 2249.0 | 4 | 94228.0 |
| | W | South Korea | 16 | Wednesday, Thursday | MBC | 8.5 | 201.0 | 5 | 92121.0 |

```
[6]  df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 100 entries, 0 to 99
     Data columns (total 9 columns):
      #   Column                    Non-Null Count   Dtype
     ---  ------                    --------------   -----
      0   show_name                 96 non-null      object
      1   country                   100 non-null     object
      2   num_episodes              100 non-null     int64
      3   aired_on                  99 non-null      object
      4   original_network          99 non-null      object
      5   rating                    96 non-null      float64
      6   current_overall_rank      97 non-null      float64
      7   lifetime_popularity_rank  100 non-null     int64
      8   watchers                  97 non-null      float64
     dtypes: float64(3), int64(2), object(4)
     memory usage: 7.2+ KB
```

[7] df.isnull()

| | show_name | country | num_episodes | aired_on | original_network | rating | current_overall_rank | lifetime_popularity_rank | watchers |
|---|---|---|---|---|---|---|---|---|---|
| 0 | True | False | False | False | False | False | False | False | False |
| 1 | True | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False |
| 4 | False | False | False | False | False | False | False | False | False |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | False | False | False | False | False | False | False | False | False |
| 96 | False | False | False | False | False | False | False | False | False |
| 97 | False | False | False | False | False | False | False | False | True |
| 98 | False | False | False | False | False | False | False | False | False |
| 99 | False | False | False | False | False | False | False | False | False |

100 rows × 9 columns

NON NULL BEFORE:

```
[8]  df.isnull().sum()

     show_name                   4
     country                     0
     num_episodes                0
     aired_on                    1
     original_network            1
     rating                      4
     current_overall_rank        3
     lifetime_popularity_rank    0
     watchers                    3
     dtype: int64
```

MODE:

```
[ ] df['show_name']=df['show_name'].fillna(df['aired_on'].mode()[0])
    df['aired_on']=df['aired_on'].fillna(df['aired_on'].mode()[0])
    df['original_network']=df['original_network'].fillna(df['aired_on'].mode()[0])
    df.head()
```

| show_name | country | num_episodes | aired_on | original_network | rating | current_overall_rank | lifetime_popularity_rank | watchers |
|---|---|---|---|---|---|---|---|---|
| Wednesday, Thursday | South Korea | 16 | Friday, Saturday | tvN | 8.9 | 33.0 | 1 | 111706.0 |
| Wednesday, Thursday | South Korea | 16 | Friday, Saturday | jTBC | 8.7 | 89.0 | 2 | 100950.0 |
| Descendants of the Sun | South Korea | 16 | Wednesday, Thursday | KBS2 | 8.7 | 77.0 | 3 | 96318.0 |
| Boys Over Flowers | South Korea | 25 | Monday, Tuesday | KBS2 | 7.7 | 2249.0 | 4 | 94228.0 |
| W | South Korea | 16 | Wednesday, Thursday | MBC | 8.5 | 201.0 | 5 | 92121.0 |

MEAN:

```
[ ] df['rating']=df['rating'].fillna(df['rating'].mean())
    df['current_overall_rank']=df['current_overall_rank'].fillna(df['rating'].mean())
    df.head()
```

| show_name | country | num_episodes | aired_on | original_network | rating | current_overall_rank | lifetime_popularity_rank | watchers |
|---|---|---|---|---|---|---|---|---|
| Wednesday, Thursday | South Korea | 16 | Friday, Saturday | tvN | 8.9 | 33.0 | 1 | 111706.0 |
| Wednesday, Thursday | South Korea | 16 | Friday, Saturday | jTBC | 8.7 | 89.0 | 2 | 100950.0 |
| Descendants of the Sun | South Korea | 16 | Wednesday, Thursday | KBS2 | 8.7 | 77.0 | 3 | 96318.0 |
| Boys Over Flowers | South Korea | 25 | Monday, Tuesday | KBS2 | 7.7 | 2249.0 | 4 | 94228.0 |
| W | South Korea | 16 | Wednesday, Thursday | MBC | 8.5 | 201.0 | 5 | 92121.0 |

MEDIAN:

```
[ ] df['watchers']=df['watchers'].fillna(df['watchers'].median())
    df.head()
```

| | show_name | country | num_episodes | aired_on | original_network | rating | current_overall_rank | lifetime_popularity_rank | watche |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Wednesday, Thursday | South Korea | 16 | Friday, Saturday | tvN | 8.9 | 33.0 | 1 | 111706 |
| 1 | Wednesday, Thursday | South Korea | 16 | Friday, Saturday | jTBC | 8.7 | 89.0 | 2 | 100950 |
| 2 | Descendants of the Sun | South Korea | 16 | Wednesday, Thursday | KBS2 | 8.7 | 77.0 | 3 | 96318 |
| 3 | Boys Over Flowers | South Korea | 25 | Monday, Tuesday | KBS2 | 7.7 | 2249.0 | 4 | 94228 |
| 4 | W | South Korea | 16 | Wednesday, Thursday | MBC | 8.5 | 201.0 | 5 | 92121 |

```
[ ] df.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 100 entries, 0 to 99
    Data columns (total 9 columns):
     #   Column                    Non-Null Count  Dtype
    ---  ------                    --------------  -----
     0   show_name                 100 non-null    object
     1   country                   100 non-null    object
     2   num_episodes              100 non-null    int64
     3   aired_on                  100 non-null    object
     4   original_network          100 non-null    object
     5   rating                    100 non-null    float64
     6   current_overall_rank      100 non-null    float64
     7   lifetime_popularity_rank  100 non-null    int64
     8   watchers                  100 non-null    float64
    dtypes: float64(3), int64(2), object(4)
    memory usage: 7.2+ KB
```

NON NULL AFTER:

```
[ ] df.isnull().sum()

    show_name                   0
    country                     0
    num_episodes                0
    aired_on                    0
    original_network            0
    rating                      0
    current_overall_rank        0
    lifetime_popularity_rank    0
    watchers                    0
    dtype: int64
```

```
## CODE FOR DATA 2:

import pandas as pd
import numpy as np
import seaborn as sns
d = pd.read_csv("/content/Loan_data.csv")
d
d.head()
d.describe()
d.tail()
d.isnull().sum()
d.shape
d.columns
d.duplicated

#Using mode method to fill the data in columns as Object(String)
#mode()[0] - Takes the most reccuring value and fills the empty cells
d['Gender'] = d['Gender'].fillna(d['Gender'].mode()[0])
d['Dependents'] = d['Dependents'].fillna(d['Dependents'].mode()[0])
d['Self_Employed'] = d['Self_Employed'].fillna(d['Self_Employed'].mode()[0])

#Using mean method to fill the data
d['LoanAmount'] = d['LoanAmount'].fillna(d['LoanAmount'].mean())
d['Loan_Amount_Term'] = d['Loan_Amount_Term'].fillna(d['Loan_Amount_Term'].mean())
d['Credit_History'] = d['Credit_History'].fillna(d['Credit_History'].mean())

sns.boxplot(y="LoanAmount",data=d)
#Checking the total no.of null values again
d.isnull().sum()

#Checking info of the dataset to check all the columns have entries
d.info()
```

**OUTPUT FOR DATA 2:**

DATA:

```
[ ] df.head(10)
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Area |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP001015 | Male | Yes | 0 | Graduate | No | 5720 | 0 | 110.0 | 360.0 | 1.0 | Urban |
| 1 | LP001022 | Male | Yes | 1 | Graduate | No | 3076 | 1500 | 126.0 | 360.0 | 1.0 | Urban |
| 2 | LP001031 | Male | Yes | 2 | Graduate | No | 5000 | 1800 | 208.0 | 360.0 | 1.0 | Urban |
| 3 | LP001035 | Male | Yes | 2 | Graduate | No | 2340 | 2546 | 100.0 | 360.0 | NaN | Urban |
| 4 | LP001051 | Male | No | 0 | Not Graduate | No | 3276 | 0 | 78.0 | 360.0 | 1.0 | Urban |
| 5 | LP001054 | Male | Yes | 0 | Not Graduate | Yes | 2165 | 3422 | 152.0 | 360.0 | 1.0 | Urban |
| 6 | LP001055 | Female | No | 1 | Not Graduate | No | 2226 | 0 | 59.0 | 360.0 | 1.0 | Semiurban |
| 7 | LP001056 | Male | Yes | 2 | Not Graduate | No | 3881 | 0 | 147.0 | 360.0 | 0.0 | Rural |
| 8 | LP001059 | Male | Yes | 2 | Graduate | NaN | 13633 | 0 | 280.0 | 240.0 | 1.0 | Urban |
| 9 | LP001067 | Male | No | 0 | Not Graduate | No | 2400 | 2400 | 123.0 | 360.0 | 1.0 | Semiurban |

```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             356 non-null    object
 2   Married            367 non-null    object
 3   Dependents         357 non-null    object
 4   Education          367 non-null    object
 5   Self_Employed      344 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

```
[ ] df.isnull()
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Area |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | False | False | False | False | False | False | False | False | False | False | False | False |
| 1 | False | False | False | False | False | False | False | False | False | False | False | False |
| 2 | False | False | False | False | False | False | False | False | False | False | False | False |
| 3 | False | False | False | False | False | False | False | False | False | False | True | False |
| 4 | False | False | False | False | False | False | False | False | False | False | False | False |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 362 | False | False | False | False | False | False | False | False | False | False | False | False |
| 363 | False | False | False | False | False | False | False | False | False | False | False | False |
| 364 | False | False | False | False | False | False | False | False | False | False | True | False |
| 365 | False | False | False | False | False | False | False | False | False | False | False | False |
| 366 | False | False | False | False | False | False | False | False | False | False | False | False |

367 rows × 12 columns

NULL BEFORE:

```
[ ] df.isnull().sum()

Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

MODE:

```
[ ] df['Gender']=df['Gender'].fillna(df['Gender'].mode()[0])
    df['Dependents']=df['Dependents'].fillna(df['Dependents'].mode()[0])
    df['Self_Employed']=df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
    df.head()
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Area |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP001015 | Male | Yes | 0 | Graduate | No | 5720 | 0 | 110.0 | 360.0 | 1.0 | Urban |
| 1 | LP001022 | Male | Yes | 1 | Graduate | No | 3076 | 1500 | 126.0 | 360.0 | 1.0 | Urban |
| 2 | LP001031 | Male | Yes | 2 | Graduate | No | 5000 | 1800 | 208.0 | 360.0 | 1.0 | Urban |
| 3 | LP001035 | Male | Yes | 2 | Graduate | No | 2340 | 2546 | 100.0 | 360.0 | 1.0 | Urban |
| 4 | LP001051 | Male | No | 0 | Not Graduate | No | 3276 | 0 | 78.0 | 360.0 | 1.0 | Urban |

MEDIAN:

```
[ ] df['LoanAmount']=df['LoanAmount'].fillna(df['LoanAmount'].median())
    df['Loan_Amount_Term']=df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].median())
    df['Credit_History']=df['Credit_History'].fillna(df['Credit_History'].median())
    df.head()
```

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term | Credit_History | Property_Area |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP001015 | Male | Yes | 0 | Graduate | No | 5720 | 0 | 110.0 | 360.0 | 1.0 | Urban |
| 1 | LP001022 | Male | Yes | 1 | Graduate | No | 3076 | 1500 | 126.0 | 360.0 | 1.0 | Urban |
| 2 | LP001031 | Male | Yes | 2 | Graduate | No | 5000 | 1800 | 208.0 | 360.0 | 1.0 | Urban |
| 3 | LP001035 | Male | Yes | 2 | Graduate | No | 2340 | 2546 | 100.0 | 360.0 | 1.0 | Urban |
| 4 | LP001051 | Male | No | 0 | Not Graduate | No | 3276 | 0 | 78.0 | 360.0 | 1.0 | Urban |

```
[ ]  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             367 non-null    object
 2   Married            367 non-null    object
 3   Dependents         367 non-null    object
 4   Education          367 non-null    object
 5   Self_Employed      367 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         367 non-null    float64
 9   Loan_Amount_Term   367 non-null    float64
 10  Credit_History     367 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

NON NULL AFTER:

```
[ ]  df.isnull().sum()

Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
dtype: int64
```

## RESULT:

Thus the given data is read,cleansed and cleaned data is saved into the file.

# Ex02-Outlier

## AIM:

You are given bhp.csv which contains property prices in the city of banglore, India. You need to examine price_per_sqft column and do following,

(1) Remove outliers using IQR

(2) After removing outliers in step 1, you get a new dataframe.

(3) use zscore of 3 to remove outliers. This is quite similar to IQR and you will get exact same result

(4) for the data set height_weight.csv find the following

(i) Using IQR detect weight outliers and print them

(ii) Using IQR, detect height outliers and print them

## Explanation

An Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set. An outlier is an observation of a data point that lies an abnormal distance from other values in a given population. (odd man out).Outliers badly affect mean and standard deviation of the dataset. These may statistically give erroneous results.Most machine learning algorithms do not work well in the presence of outlier. So it is desirable to detect and remove outliers.Outliers are highly useful in anomaly detection like fraud detection where the fraud transactions are very different from normal transactions.

### ALGORITHM:

**STEP 1:**Read the given Data

**STEP 2:**Get the information about the data

**STEP 3:**Detect the Outliers using IQR method and Z score

**STEP 4:**Remove the outliers

**STEP 5:**Plot the datas using Box Plot

## CODE:

(1) & (2) Examine price_per_sqft column and use IQR to remove outliers and create new dataframe

```
import pandas as pd
import numpy as np
import seaborn as sns
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Semester 3/19AI403 - Data Science/bhp.csv")
df
```

```
df.head()
df.describe()
df.info()
df.isnull().sum()
df.shape
sns.boxplot(x="price_per_sqft",data=df)
q1 = df['price_per_sqft'].quantile(0.25)
q3 = df['price_Aper_sqft'].quantile(0.75)
print("First Quantile =",q1,"\nSecond Quantile =",q3)
IQR = q3-q1
ul = q3+1.5*IQR
ll = q1-1.5*IQR
df1 =df[((df['price_per_sqft']>=ll)&(df['price_per_sqft']<=ul))]
df1
df1.shape
sns.boxplot(x="price_per_sqft",data=df1)
```

(3) Examine price_per_sqft column and use zscore of 3 to remove outliers.

```
from scipy import stats
z = np.abs(stats.zscore(df['price_per_sqft']))
df2 = df[(z<3)]
df2
print(df2.shape)
sns.boxplot(x="price_per_sqft",data=df2)
```

(4)(i) For the data set height_weight.csv detect weight outliers using IQR method

```
df3 = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Semester 3/19AI403 - Data Science/height_weight.csv")
df3
df3.head()
df3.info()
df3.describe()
df3.isnull().sum()
df3.shape
sns.boxplot(x="weight",data=df3)
q1 = df3['weight'].quantile(0.25)
q3 = df3['weight'].quantile(0.75)
print("First Quantile =",q1,"\nSecond Quantile =",q3)
IQR = q3-q1
ul = q3+1.5*IQR
ll = q1-1.5*IQR
df4 =df3[((df3['weight']>=ll)&(df3['weight']<=ul))]
df4
df4.shape
sns.boxplot(x="weight",data=df4)
```

(4)(ii) For the data set height_weight.csv detect height outliers using IQR method

```
sns.boxplot(x="height",data=df3)
q1 = df3['height'].quantile(0.25)
q3 = df3['height'].quantile(0.75)
print("First Quantile =",q1,"\nSecond Quantile =",q3)
IQR = q3-q1
ul = q3+1.5*IQR
ll = q1-1.5*IQR
df5 =df3[((df3['height']>=ll)&(df3['height']<=ul))]
df5
df5.shape
sns.boxplot(x="height",data=df5)
```

## OUTPUT:

(1)(2) Examine price_per_sqft column and use IQR to remove outliers and create new dataframe.

Dataset

| | location | size | total_sqft | bath | price | bhk | price_per_sqft |
|---|---|---|---|---|---|---|---|
| 0 | Electronic City Phase II | 2 BHK | 1056.0 | 2.0 | 39.07 | 2 | 3699 |
| 1 | Chikka Tirupathi | 4 Bedroom | 2600.0 | 5.0 | 120.00 | 4 | 4615 |
| 2 | Uttarahalli | 3 BHK | 1440.0 | 2.0 | 62.00 | 3 | 4305 |
| 3 | Lingadheeranahalli | 3 BHK | 1521.0 | 3.0 | 95.00 | 3 | 6245 |
| 4 | Kothanur | 2 BHK | 1200.0 | 2.0 | 51.00 | 2 | 4250 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 13195 | Whitefield | 5 Bedroom | 3453.0 | 4.0 | 231.00 | 5 | 6689 |
| 13196 | other | 4 BHK | 3600.0 | 5.0 | 400.00 | 4 | 11111 |
| 13197 | Raja Rajeshwari Nagar | 2 BHK | 1141.0 | 2.0 | 60.00 | 2 | 5258 |
| 13198 | Padmanabhanagar | 4 BHK | 4689.0 | 4.0 | 488.00 | 4 | 10407 |
| 13199 | Doddathoguru | 1 BHK | 550.0 | 1.0 | 17.00 | 1 | 3090 |

3200 rows × 7 columns

Dataset Head

| | location | size | total_sqft | bath | price | bhk | price_per_sqft |
|---|---|---|---|---|---|---|---|
| 0 | Electronic City Phase II | 2 BHK | 1056.0 | 2.0 | 39.07 | 2 | 3699 |
| 1 | Chikka Tirupathi | 4 Bedroom | 2600.0 | 5.0 | 120.00 | 4 | 4615 |
| 2 | Uttarahalli | 3 BHK | 1440.0 | 2.0 | 62.00 | 3 | 4305 |
| 3 | Lingadheeranahalli | 3 BHK | 1521.0 | 3.0 | 95.00 | 3 | 6245 |
| 4 | Kothanur | 2 BHK | 1200.0 | 2.0 | 51.00 | 2 | 4250 |

Dataset Info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13200 entries, 0 to 13199
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   location        13200 non-null  object
 1   size            13200 non-null  object
 2   total_sqft      13200 non-null  float64
 3   bath            13200 non-null  float64
 4   price           13200 non-null  float64
 5   bhk             13200 non-null  int64
 6   price_per_sqft  13200 non-null  int64
dtypes: float64(3), int64(2), object(2)
memory usage: 722.0+ KB
```

Dataset Describe

| | total_sqft | bath | price | bhk | price_per_sqft |
|---|---|---|---|---|---|
| count | 13200.000000 | 13200.000000 | 13200.000000 | 13200.000000 | 1.320000e+04 |
| mean | 1555.302783 | 2.691136 | 112.276178 | 2.800833 | 7.920337e+03 |
| std | 1237.323445 | 1.338915 | 149.175995 | 1.292843 | 1.067272e+05 |
| min | 1.000000 | 1.000000 | 8.000000 | 1.000000 | 2.670000e+02 |
| 25% | 1100.000000 | 2.000000 | 50.000000 | 2.000000 | 4.267000e+03 |
| 50% | 1275.000000 | 2.000000 | 71.850000 | 3.000000 | 5.438000e+03 |
| 75% | 1672.000000 | 3.000000 | 120.000000 | 3.000000 | 7.317000e+03 |
| max | 52272.000000 | 40.000000 | 3600.000000 | 43.000000 | 1.200000e+07 |

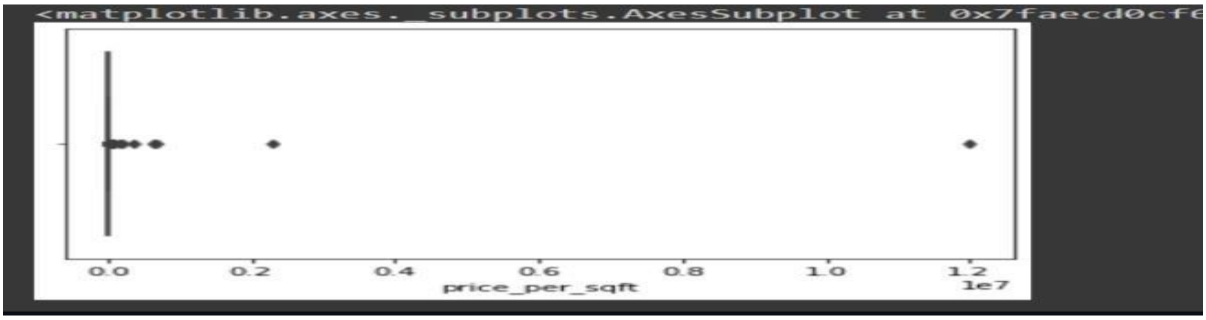
Null Values

```
location          0
size              0
total_sqft        0
bath              0
price             0
bhk               0
price_per_sqft    0
dtype: int64
```

Dataset Shape

```
(11935, 7)
```

Box plot of price_per_sqft column with outliers




price_per_sqft - Dataset after removing outliers

```
First Quantile = 4267.0
Second Quantile = 7317.0
```

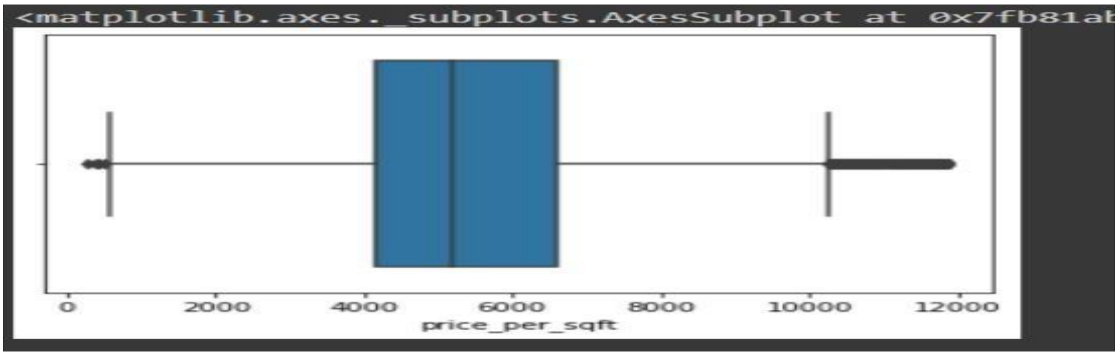
| | location | size | total_sqft | bath | price | bhk | price_per_sqft |
|---|---|---|---|---|---|---|---|
| 0 | Electronic City Phase II | 2 BHK | 1056.0 | 2.0 | 39.07 | 2 | 3699 |
| 1 | Chikka Tirupathi | 4 Bedroom | 2600.0 | 5.0 | 120.00 | 4 | 4615 |
| 2 | Uttarahalli | 3 BHK | 1440.0 | 2.0 | 62.00 | 3 | 4305 |
| 3 | Lingadheeranahalli | 3 BHK | 1521.0 | 3.0 | 95.00 | 3 | 6245 |
| 4 | Kothanur | 2 BHK | 1200.0 | 2.0 | 51.00 | 2 | 4250 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 13195 | Whitefield | 5 Bedroom | 3453.0 | 4.0 | 231.00 | 5 | 6689 |
| 13196 | other | 4 BHK | 3600.0 | 5.0 | 400.00 | 4 | 11111 |
| 13197 | Raja Rajeshwari Nagar | 2 BHK | 1141.0 | 2.0 | 60.00 | 2 | 5258 |
| 13198 | Padmanabhanagar | 4 BHK | 4689.0 | 4.0 | 488.00 | 4 | 10407 |
| 13199 | Doddathoguru | 1 BHK | 550.0 | 1.0 | 17.00 | 1 | 3090 |

11935 rows × 7 columns

price_per_sqft - Shape of Dataset after removing outliers

```
(10953, 7)
```

Box Plot of price_per_sqft column without outliers




(3) Examine price_per_sqft column and use zscore of 3 to remove outliers.

Dataset after removal of outlier using z score

| | location | size | total_sqft | bath | price | bhk | price_per_sqft |
|---|---|---|---|---|---|---|---|
| 0 | Electronic City Phase II | 2 BHK | 1056.0 | 2.0 | 39.07 | 2 | 3699 |
| 1 | Chikka Tirupathi | 4 Bedroom | 2600.0 | 5.0 | 120.00 | 4 | 4615 |
| 2 | Uttarahalli | 3 BHK | 1440.0 | 2.0 | 62.00 | 3 | 4305 |
| 3 | Lingadheeranahalli | 3 BHK | 1521.0 | 3.0 | 95.00 | 3 | 6245 |
| 4 | Kothanur | 2 BHK | 1200.0 | 2.0 | 51.00 | 2 | 4250 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 13195 | Whitefield | 5 Bedroom | 3453.0 | 4.0 | 231.00 | 5 | 6689 |
| 13196 | other | 4 BHK | 3600.0 | 5.0 | 400.00 | 4 | 11111 |
| 13197 | Raja Rajeshwari Nagar | 2 BHK | 1141.0 | 2.0 | 60.00 | 2 | 5258 |
| 13198 | Padmanabhanagar | 4 BHK | 4689.0 | 4.0 | 488.00 | 4 | 10407 |
| 13199 | Doddathoguru | 1 BHK | 550.0 | 1.0 | 17.00 | 1 | 3090 |

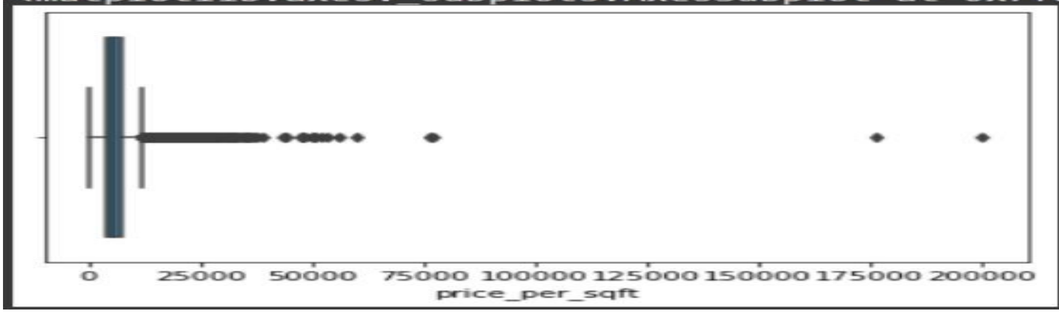
13195 rows × 7 columns

Shape of Dataset after removal of outlier using z score

```
(13195, 7)
```

price_per_sqft column after removing outliers

```
<matplotlib.axes._subplots.AxesSubplot at 0x7faecbca3
```




(4) For the data set height_weight.csv detect weight and height outliers using IQR method.

Dataset

| | gender | height | weight |
|---|---|---|---|
| 0 | Male | 73.847017 | 241.893563 |
| 1 | Male | 68.781904 | 162.310473 |
| 2 | Male | 74.110105 | 212.740856 |
| 3 | Male | 71.730978 | 220.042470 |
| 4 | Male | 69.881796 | 206.349801 |
| ... | ... | ... | ... |
| 9995 | Female | 66.172652 | 136.777454 |
| 9996 | Female | 67.067155 | 170.867906 |
| 9997 | Female | 63.867992 | 128.475319 |
| 9998 | Female | 69.034243 | 163.852461 |
| 9999 | Female | 61.944246 | 113.649103 |

10000 rows × 3 columns

Dataset Head

| | gender | height | weight |
|---|---|---|---|
| 0 | Male | 73.847017 | 241.893563 |
| 1 | Male | 68.781904 | 162.310473 |
| 2 | Male | 74.110105 | 212.740856 |
| 3 | Male | 71.730978 | 220.042470 |
| 4 | Male | 69.881796 | 206.349801 |

Dataset Info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   gender  10000 non-null  object
 1   height  10000 non-null  float64
 2   weight  10000 non-null  float64
dtypes: float64(2), object(1)
memory usage: 234.5+ KB
```

Dataset Describe

| | height | weight |
|---|---|---|
| count | 10000.000000 | 10000.000000 |
| mean | 66.367560 | 161.440357 |
| std | 3.847528 | 32.108439 |
| min | 54.263133 | 64.700127 |
| 25% | 63.505620 | 135.818051 |
| 50% | 66.318070 | 161.212928 |
| 75% | 69.174262 | 187.169525 |
| max | 78.998742 | 269.989699 |

Null Values

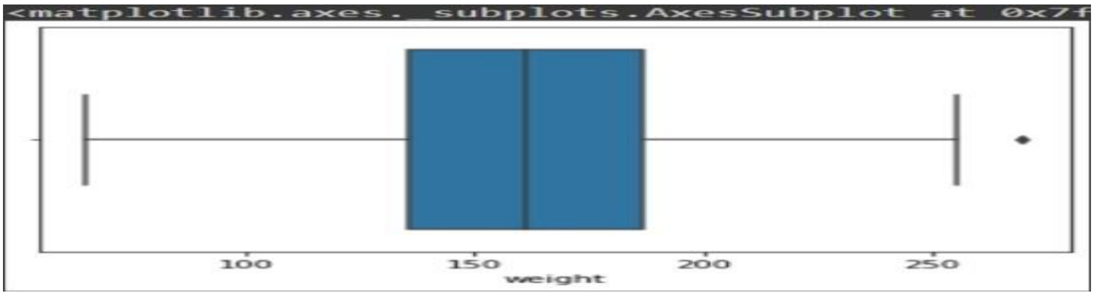
```
gender     0
height     0
weight     0
dtype: int64
```

Dataset Shape

```
(10000, 3)
```

Weight - With outliers

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
```




Weight - Dataset after removing Outliers using IQR method.

```
First Quantile = 135.8180513055015
Second Quantile = 187.16952486868348
```

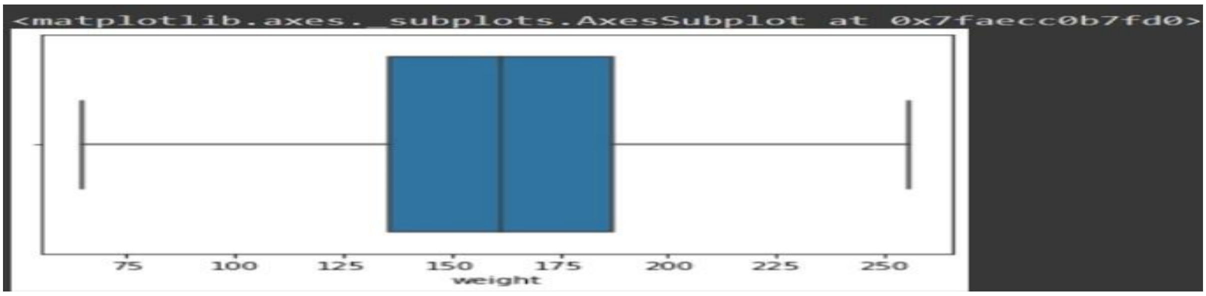
| | gender | height | weight |
|---|---|---|---|
| 0 | Male | 73.847017 | 241.893563 |
| 1 | Male | 68.781904 | 162.310473 |
| 2 | Male | 74.110105 | 212.740856 |
| 3 | Male | 71.730978 | 220.042470 |
| 4 | Male | 69.881796 | 206.349801 |
| ... | ... | ... | ... |
| 9995 | Female | 66.172652 | 136.777454 |
| 9996 | Female | 67.067155 | 170.867906 |
| 9997 | Female | 63.867992 | 128.475319 |
| 9998 | Female | 69.034243 | 163.852461 |
| 9999 | Female | 61.944246 | 113.649103 |

9999 rows × 3 columns

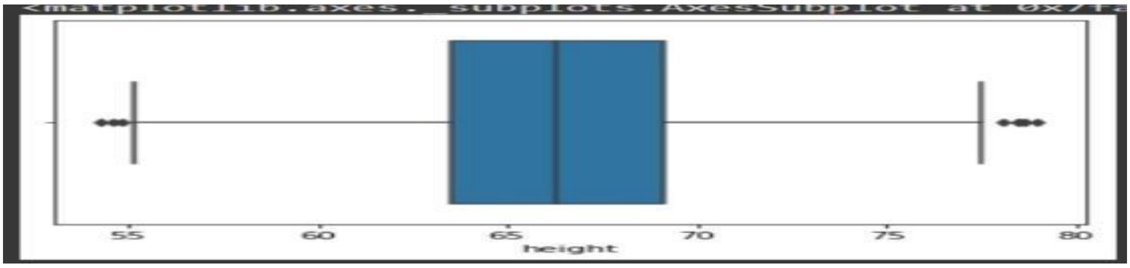
Weight - Shape of Dataset after removing Outliers using IQR method

```
(9999, 3)
```

Weight - Without Outliers using IQR method




Height - With outliers




Height - Dataset after removing Outliers using IQR method

```
First Quantile = 63.505620481218955
Second Quantile = 69.1742617268347
```

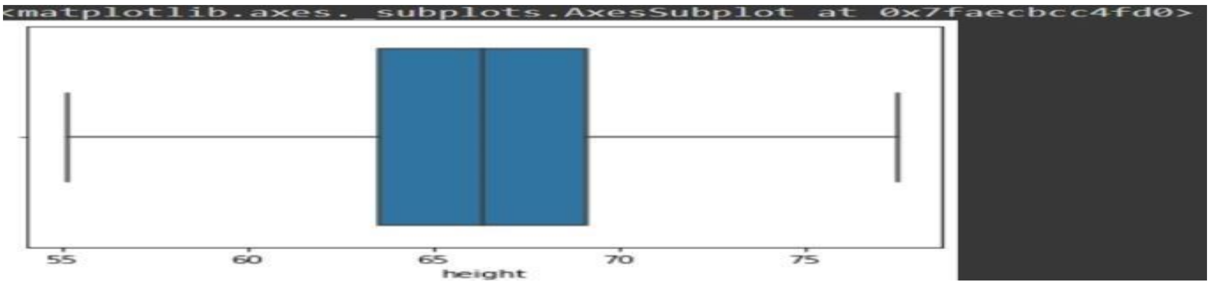
| | gender | height | weight |
|---|---|---|---|
| 0 | Male | 73.847017 | 241.893563 |
| 1 | Male | 68.781904 | 162.310473 |
| 2 | Male | 74.110105 | 212.740856 |
| 3 | Male | 71.730978 | 220.042470 |
| 4 | Male | 69.881796 | 206.349801 |
| ... | ... | ... | ... |
| 9995 | Female | 66.172652 | 136.777454 |
| 9996 | Female | 67.067155 | 170.867906 |
| 9997 | Female | 63.867992 | 128.475319 |
| 9998 | Female | 69.034243 | 163.852461 |
| 9999 | Female | 61.944246 | 113.649103 |

9992 rows × 3 columns

Height - Shape of Dataset after removing Outliers using IQR method

```
(9992, 3)
```

Height - Without Outliers using IQR method




**RESULT:**

The given datasets are read and outliers are detected and are removed using IQR and z-score methods.

# Ex03-Univariate-Analysis

## Aim

To read the given data and perform the univariate analysis with different types of plots.

## Explanation

Univariate analysis is basically the simplest form to analyze data. Uni means one and this means that the data has only one kind of variable. The major reason for univariate analysis is to use the data to describe. The analysis will take data, summarise it, and then find some pattern in the data.

## Algorithm

Step1:Read the given data.

Step2:Get the information about the data.

Step3:Remove the null values from the data.

Step4:Mention the datatypes from the data.

Step5:Count the values from the data.

Step6:Do plots like boxplots,countplot,distribution plot,histogram plot.

## Program

Developed by : M Vignesh
Registration Number : 212220233002

```
import pandas as pd
import numpy as np
import seaborn as sns
df=pd.read_csv('superstore.csv')
df
df.head()
df.info()
df.describe()
df.isnull().sum()
df.dtypes
df['Postal Code'].value_counts()
sns.boxplot(x='Postal Code', data=df)
sns.countplot(x='Postal Code',data=df)
sns.distplot(df["Postal Code"])
sns.histplot(x='Postal Code',data=df)
```

**OUTPUT:**

DATA:

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region | Product ID | Category | Sub-Category | Product Name | Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2017-152156 | 08-11-2017 | 11-11-2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-BO-10001798 | Furniture | Bookcases | Bush Somerset Collection Bookcase | 261.9600 |
| 1 | 2 | CA-2017-152156 | 08-11-2017 | 11-11-2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-CH-10000454 | Furniture | Chairs | Hon Deluxe Fabric Upholstered Stacking Chairs,... | 731.9400 |
| 2 | 3 | CA-2017-138688 | 12-06-2017 | 16-06-2017 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | California | 90036.0 | West | OFF-LA-10000240 | Office Supplies | Labels | Self-Adhesive Address Labels for Typewriters b... | 14.6200 |
| 3 | 4 | US-2016-108966 | 11-10-2016 | 18-10-2016 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | Florida | 33311.0 | South | FUR-TA-10000577 | Furniture | Tables | Bretford CR4500 Series Slim Rectangular Table | 957.5775 |
| 4 | 5 | US-2016-108966 | 11-10-2016 | 18-10-2016 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | Florida | 33311.0 | South | OFF-ST-10000760 | Office Supplies | Storage | Eldon Fold 'N Roll Cart System | 22.3680 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9795 | 9796 | CA-2017-125920 | 21-05-2017 | 28-05-2017 | Standard Class | SH-19975 | Sally Hughsby | Corporate | United States | Chicago | Illinois | 60610.0 | Central | OFF-BI-10003429 | Office Supplies | Binders | Cardinal HOLDit! Binder Insert Strips,Extra St... | 3.7980 |
| 9796 | 9797 | CA-2016-128608 | 12-01-2016 | 17-01-2016 | Standard Class | CS-12490 | Cindy Schnelling | Corporate | United States | Toledo | Ohio | 43615.0 | East | OFF-AR-10001374 | Office Supplies | Art | BIC Brite Liner Highlighters, Chisel Tip | 10.3680 |
| 9797 | 9798 | CA-2016-128608 | 12-01-2016 | 17-01-2016 | Standard Class | CS-12490 | Cindy Schnelling | Corporate | United States | Toledo | Ohio | 43615.0 | East | TEC-PH-10004977 | Technology | Phones | GE 30524EE4 | 235.1880 |
| 9798 | 9799 | CA-2016-128608 | 12-01-2016 | 17-01-2016 | Standard Class | CS-12490 | Cindy Schnelling | Corporate | United States | Toledo | Ohio | 43615.0 | East | TEC-PH-10000912 | Technology | Phones | Anker 24W Portable Micro USB Car Charger | 26.3760 |
| 9799 | 9800 | CA-2016-128608 | 12-01-2016 | 17-01-2016 | Standard Class | CS-12490 | Cindy Schnelling | Corporate | United States | Toledo | Ohio | 43615.0 | East | TEC-AC-10000487 | Technology | Accessories | SanDisk Cruzer 4 GB USB Flash Drive | 10.3840 |

9800 rows × 18 columns

DATA HEAD

```
[6] df.head()
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region | Product ID | Category | Sub-Category | Product Name | Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2017-152156 | 08-11-2017 | 11-11-2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-BO-10001798 | Furniture | Bookcases | Bush Somerset Collection Bookcase | 261.9600 |
| 1 | 2 | CA-2017-152156 | 08-11-2017 | 11-11-2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-CH-10000454 | Furniture | Chairs | Hon Deluxe Fabric Upholstered Stacking Chairs,... | 731.9400 |
| 2 | 3 | CA-2017-138688 | 12-06-2017 | 16-06-2017 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | California | 90036.0 | West | OFF-LA-10000240 | Office Supplies | Labels | Self-Adhesive Address Labels for Typewriters b... | 14.6200 |
| 3 | 4 | US-2016-108966 | 11-10-2016 | 18-10-2016 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | Florida | 33311.0 | South | FUR-TA-10000577 | Furniture | Tables | Bretford CR4500 Series Slim Rectangular Table | 957.5775 |
| 4 | 5 | US-2016-108966 | 11-10-2016 | 18-10-2016 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | Florida | 33311.0 | South | OFF-ST-10000760 | Office Supplies | Storage | Eldon Fold 'N Roll Cart System | 22.3680 |

DATA INFORMATION

```
[4]  df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 9800 entries, 0 to 9799
     Data columns (total 18 columns):
      #   Column          Non-Null Count  Dtype
     ---  ------          --------------  -----
      0   Row ID          9800 non-null   int64
      1   Order ID        9800 non-null   object
      2   Order Date      9800 non-null   object
      3   Ship Date       9800 non-null   object
      4   Ship Mode       9800 non-null   object
      5   Customer ID     9800 non-null   object
      6   Customer Name   9800 non-null   object
      7   Segment         9800 non-null   object
      8   Country         9800 non-null   object
      9   City            9800 non-null   object
      10  State           9800 non-null   object
      11  Postal Code     9789 non-null   float64
      12  Region          9800 non-null   object
      13  Product ID      9800 non-null   object
      14  Category        9800 non-null   object
      15  Sub-Category    9800 non-null   object
      16  Product Name    9800 non-null   object
      17  Sales           9800 non-null   float64
     dtypes: float64(2), int64(1), object(15)
     memory usage: 1.3+ MB
```

DATA DESCRIBE

```
[7]  df.describe()
```

| | Row ID | Postal Code | Sales |
|---|---|---|---|
| count | 9800.000000 | 9789.000000 | 9800.000000 |
| mean | 4900.500000 | 55273.322403 | 230.769059 |
| std | 2829.160653 | 32041.223413 | 626.651875 |
| min | 1.000000 | 1040.000000 | 0.444000 |
| 25% | 2450.750000 | 23223.000000 | 17.248000 |
| 50% | 4900.500000 | 58103.000000 | 54.490000 |
| 75% | 7350.250000 | 90008.000000 | 210.605000 |
| max | 9800.000000 | 99301.000000 | 22638.480000 |

DATA NULL VALUES

```
[5]  df.isnull().sum()

     Row ID            0
     Order ID          0
     Order Date        0
     Ship Date         0
     Ship Mode         0
     Customer ID       0
     Customer Name     0
     Segment           0
     Country           0
     City              0
     State             0
     Postal Code      11
     Region            0
     Product ID        0
     Category          0
     Sub-Category      0
     Product Name      0
     Sales             0
     dtype: int64
```

DATA'S DATATYPES

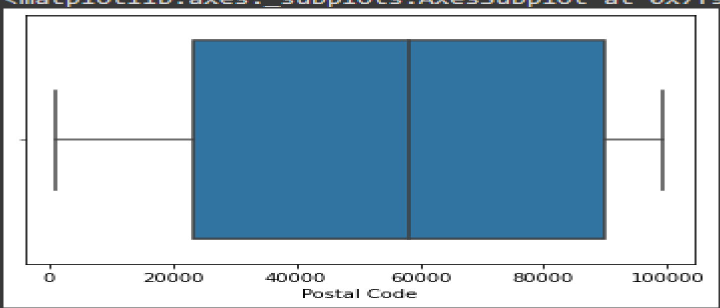
```
[ ]  df.dtypes

     Row ID              int64
     Order ID           object
     Order Date         object
     Ship Date          object
     Ship Mode          object
     Customer ID        object
     Customer Name      object
     Segment            object
     Country            object
     City               object
     State              object
     Postal Code       float64
     Region             object
     Product ID         object
     Category           object
     Sub-Category       object
     Product Name       object
     Sales             float64
     dtype: object
```

DATA'S VALUECOUNT

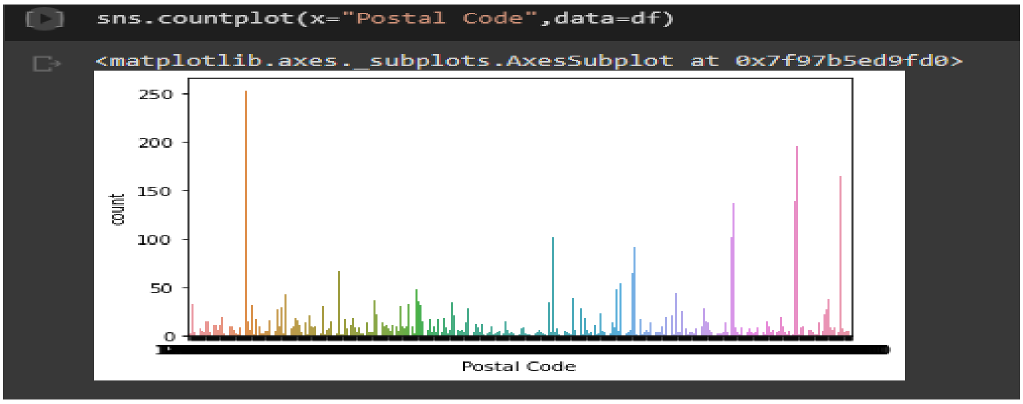
```
[8]  df['Postal Code'].value_counts()

     10035.0    253
     10024.0    225
     10009.0    220
     94122.0    195
     10011.0    193
               ...
     32935.0      1
     76248.0      1
     32127.0      1
     32503.0      1
     72762.0      1
     Name: Postal Code, Length: 626, dtype: int64
```

BOXPLOT

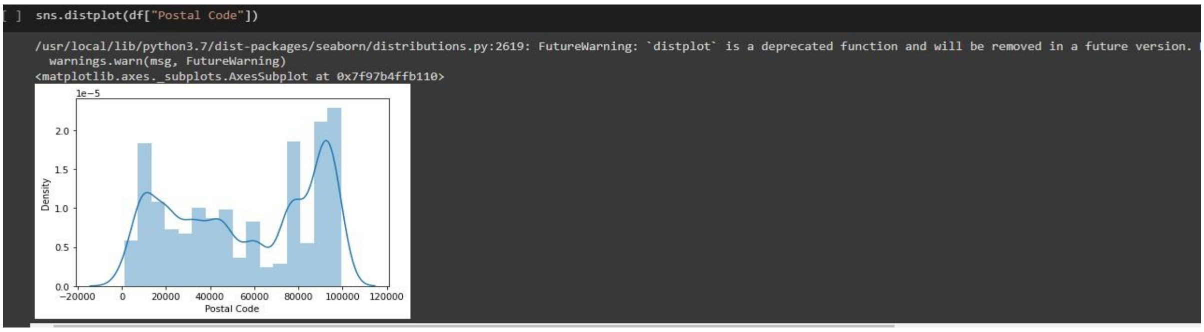
```
[ ]  sns.boxplot(x="Postal Code",data=df)

     <matplotlib.axes._subplots.AxesSubplot at 0x7f97b62d4510>
```



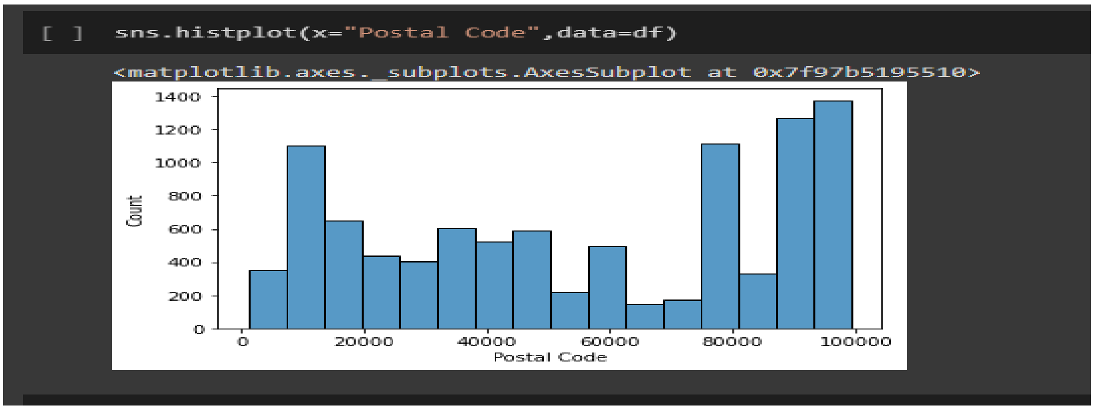
COUNTPLOT




DISTRIBUTION PLOT




HISTOGRAM PLOT




## Result

Thus we have read the given data and performed the univariate analysis with different types of plots.

# Ex-04-Multivariate-Analysis

### AIM

To perform Multivariate EDA on the given data set.

### Explanation

Exploratory data analysis is used to understand the messages within a dataset. This technique involves many iterative processes to ensure that the cleaned data is further sorted to better understand the useful meaning.The primary aim with exploratory analysis is to examine the data for distribution, outliers and anomalies to direct specific testing of your hypothesis.

### ALGORITHM

STEP 1 Import the built libraries required to perform EDA and outlier removal.

STEP 2 Read the given csv file

STEP 3 Convert the file into a dataframe and get information of the data.

STEP 4 Return the objects containing counts of unique values using (value_counts()).

STEP 5 Plot the counts in the form of Histogram or Bar Graph.

STEP 6 Use seaborn the bar graph comparison of data can be viewed.

STEP 7 Find the pairwise correlation of all columns in the dataframe.corr()

STEP 8 Save the final data set into the file

### CODE

Developed by : M Vignesh
Registration Number : 212220233002

```
import pandas as pd
import numpy as np
import seaborn as sbn
import matplotlib.pyplot as plt
df = pd.read_csv("/content/SuperStore.csv")
df.head(10)
df.info()
df.describe()
df.isnull().sum()
df['Postal Code'] = df["Postal Code"].fillna(df['Postal Code'].mode()[0])
df.isnull().sum()
df.dtypes
sbn.scatterplot(df['Postal Code'],df['Sales'])
states=df.loc[:,["State","Sales"]]
states=states.groupby(by=["State"]).sum().sort_values(by="Sales")
plt.figure(figsize=(17,7))
sbn.barplot(x=states.index,y="Sales",data=states)
```

```
plt.xticks(rotation = 90)
plt.xlabel=("STATES")
plt.ylabel=("SALES")
plt.show()
states=df.loc[:,["State","Postal Code"]]
states=states.groupby(by=["State"]).sum().sort_values(by="Postal Code")
plt.figure(figsize=(17,7))
sbn.barplot(x=states.index,y="Postal Code",data=states)
plt.xticks(rotation = 90)
plt.xlabel=("STATES")
plt.ylabel=("Postal Code")
plt.show()
states=df.loc[:,["Segment","Sales"]]
states=states.groupby(by=["Segment"]).sum().sort_values(by="Sales")
#plt.figure(figsize=(10,7))
sbn.barplot(x=states.index,y="Sales",data=states)
plt.xticks(rotation = 90)
plt.xlabel=("SEGMENT")
plt.ylabel=("SALES")
plt.show()
sbn.barplot(df['Postal Code'],df['Ship Mode'],hue=df['Region'])
df.corr()
sbn.heatmap(df.corr(),annot=True)
```

# OUTPUT

EDA - SuperStore.csv

```
import pandas as pd
import numpy as np
import seaborn as sbn
import matplotlib.pyplot as plt
df = pd.read_csv("/content/SuperStore.csv")
df.head(10)
```

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | State | Postal Code | Region | Product ID | Category | Sub-Category | Product Name | Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2017-152156 | 08-11-2017 | 11-11-2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-BO-10001798 | Furniture | Bookcases | Bush Somerset Collection Bookcase | 261.9600 |
| 1 | 2 | CA-2017-152156 | 08-11-2017 | 11-11-2017 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | Kentucky | 42420.0 | South | FUR-CH-10000454 | Furniture | Chairs | Hon Deluxe Fabric Upholstered Stacking Chairs,... | 731.9400 |
| 2 | 3 | CA-2017-138688 | 12-06-2017 | 16-06-2017 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | California | 90036.0 | West | OFF-LA-10000240 | Office Supplies | Labels | Self-Adhesive Address Labels for Typewriters b... | 14.6200 |
| 3 | 4 | US-2016-108966 | 11-10-2016 | 18-10-2016 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | Florida | 33311.0 | South | FUR-TA-10000577 | Furniture | Tables | Bretford CR4500 Series Slim Rectangular Table | 957.5775 |
| 4 | 5 | US-2016-108966 | 11-10-2016 | 18-10-2016 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | Florida | 33311.0 | South | OFF-ST-10000760 | Office Supplies | Storage | Eldon Fold 'N Roll Cart System | 22.3680 |
| 5 | 6 | CA-2015-115812 | 09-06-2015 | 14-06-2015 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | California | 90032.0 | West | FUR-FU-10001487 | Furniture | Furnishings | Eldon Expressions Wood and Plastic Desk Access... | 48.8600 |
| 6 | 7 | CA-2015-115812 | 09-06-2015 | 14-06-2015 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | California | 90032.0 | West | OFF-AR-10002833 | Office Supplies | Art | Newell 322 | 7.2800 |
| 7 | 8 | CA-2015-115812 | 09-06-2015 | 14-06-2015 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | California | 90032.0 | West | TEC-PH-10002275 | Technology | Phones | Mitel 5320 IP Phone VoIP phone | 907.1520 |
| 8 | 9 | CA-2015-115812 | 09-06-2015 | 14-06-2015 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | California | 90032.0 | West | OFF-BI-10003910 | Office Supplies | Binders | DXL Angle-View Binders with Locking Rings by S... | 18.5040 |
| 9 | 10 | CA-2015-115812 | 09-06-2015 | 14-06-2015 | Standard Class | BH-11710 | Brosina Hoffman | Consumer | United States | Los Angeles | California | 90032.0 | West | OFF-AP-10002892 | Office Supplies | Appliances | Belkin F5C206VTEL 6 Outlet Surge | 114.9000 |

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9800 entries, 0 to 9799
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9800 non-null   int64
 1   Order ID       9800 non-null   object
 2   Order Date     9800 non-null   object
 3   Ship Date      9800 non-null   object
 4   Ship Mode      9800 non-null   object
 5   Customer ID    9800 non-null   object
 6   Customer Name  9800 non-null   object
 7   Segment        9800 non-null   object
 8   Country        9800 non-null   object
 9   City           9800 non-null   object
 10  State          9800 non-null   object
 11  Postal Code    9789 non-null   float64
 12  Region         9800 non-null   object
 13  Product ID     9800 non-null   object
 14  Category       9800 non-null   object
 15  Sub-Category   9800 non-null   object
 16  Product Name   9800 non-null   object
 17  Sales          9800 non-null   float64
dtypes: float64(2), int64(1), object(15)
memory usage: 1.3+ MB
```

df.describe()

| | Row ID | Postal Code | Sales |
|---|---|---|---|
| count | 9800.000000 | 9789.000000 | 9800.000000 |
| mean | 4900.500000 | 55273.322403 | 230.769059 |
| std | 2829.160653 | 32041.223413 | 626.651875 |
| min | 1.000000 | 1040.000000 | 0.444000 |
| 25% | 2450.750000 | 23223.000000 | 17.248000 |
| 50% | 4900.500000 | 58103.000000 | 54.490000 |
| 75% | 7350.250000 | 90008.000000 | 210.605000 |
| max | 9800.000000 | 99301.000000 | 22638.480000 |

Checking the null values and Cleaning it

```
df.isnull().sum()
```

```
Row ID            0
Order ID          0
Order Date        0
Ship Date         0
Ship Mode         0
Customer ID       0
Customer Name     0
Segment           0
Country           0
City              0
State             0
Postal Code      11
Region            0
Product ID        0
Category          0
Sub-Category      0
Product Name      0
Sales             0
dtype: int64
```

```
df['Postal Code'] = df["Postal Code"].fillna(df['Postal Code'].mode()[0])
df.isnull().sum()
```

```
Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
dtype: int64
```

Displaying datatypes of each features

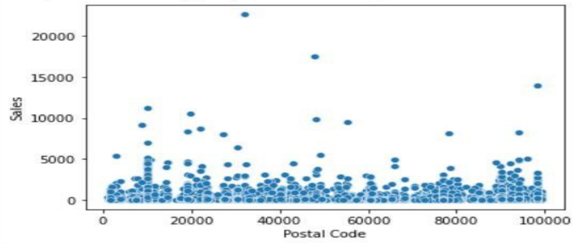
```
df.dtypes
```

```
Row ID             int64
Order ID          object
Order Date        object
Ship Date         object
Ship Mode         object
Customer ID       object
Customer Name     object
Segment           object
Country           object
City              object
State             object
Postal Code      float64
Region            object
Product ID        object
Category          object
Sub-Category      object
Product Name      object
Sales            float64
dtype: object
```

Multivariate Analysis - Scatterplot

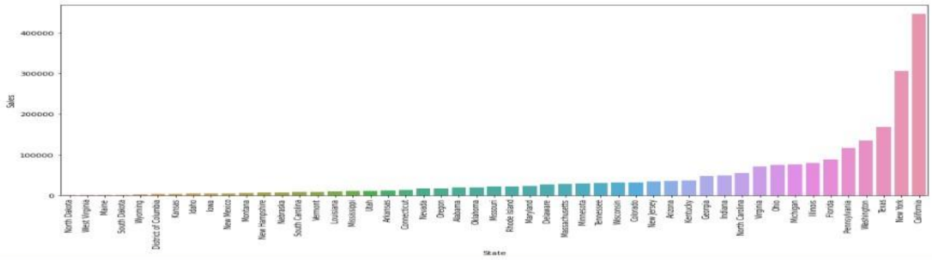
```
sbn.scatterplot(df['Postal Code'],df['Sales'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e1305abd0>
```

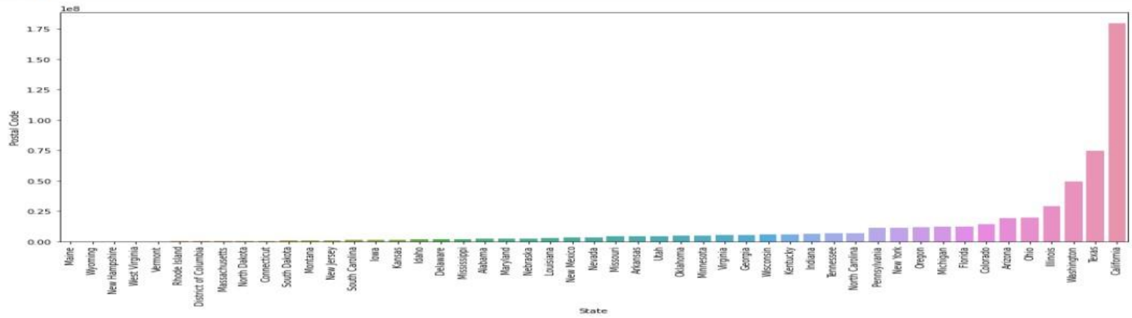



# Multivariate Analysis - Barplot

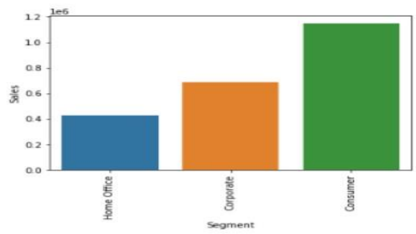
```
states=df.loc[:,["State","Sales"]]
states=states.groupby(by=["State"]).sum().sort_values(by="Sales")
plt.figure(figsize=(17,7))
sbn.barplot(x=states.index,y="Sales",data=states)
plt.xticks(rotation = 90)
plt.xlabel=("STATES")
plt.ylabel=("SALES")
plt.show()
```




```
states=df.loc[:,["State","Postal Code"]]
states=states.groupby(by=["State"]).sum().sort_values(by="Postal Code")
plt.figure(figsize=(17,7))
sbn.barplot(x=states.index,y="Postal Code",data=states)
plt.xticks(rotation = 90)
plt.xlabel=("STATES")
plt.ylabel=("Postal Code")
plt.show()
```

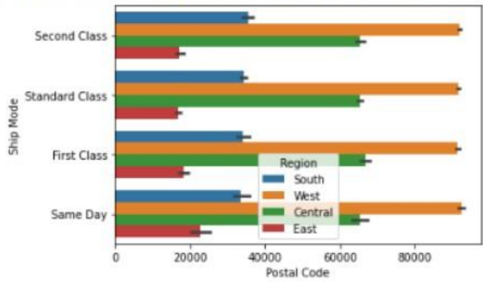



```
states=df.loc[:,["Segment","Sales"]]
states=states.groupby(by=["Segment"]).sum().sort_values(by="Sales")
#plt.figure(figsize=(10,7))
sbn.barplot(x=states.index,y="Sales",data=states)
plt.xticks(rotation = 90)
plt.xlabel=("SEGMENT")
plt.ylabel=("SALES")
plt.show()
```



```
sbn.barplot(df['Postal Code'],df['Ship Mode'],hue=df['Region'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From versi
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e0f98cc50>
```




Correlation Coefficient Interpretation using HeatMap

```
df.corr()
```

| | Row ID | Postal Code | Sales |
|---|---|---|---|
| **Row ID** | 1.000000 | 0.011723 | 0.001151 |
| **Postal Code** | 0.011723 | 1.000000 | -0.025444 |
| **Sales** | 0.001151 | -0.025444 | 1.000000 |

```
sbn.heatmap(df.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7e0f6cc8d0>
```


## RESULT

Thus the program to perform EDA on the given data set is successfully executed

# EX-05-Feature-Generation

## AIM

To read the given data and perform Feature Generation process and save the data to a file.

## Explanation

Feature Generation (also known as feature construction, feature extraction or feature engineering) is the process of transforming features into new features that better relate to the target.

## ALGORITHM

**STEP 1:**Read the given Data

**STEP 2:**Clean the Data Set using Data Cleaning Process

**STEP 3:**Apply Feature Generation techniques to all the feature of the data set

**STEP 4:**Save the data to the file

## CODE

```
import pandas as pd
df=pd.read_csv("data.csv")
df

feature generation:
import category_encoders as ce
be=ce.BinaryEncoder()
ndf=be.fit_transform(df["bin_1"])
df["bin_1"] = be.fit_transform(df["bin_1"])
ndf
ndf2=be.fit_transform(df["bin_2"])
df["bin_2"] = be.fit_transform(df["bin_2"])
ndf2
df1=df.copy()
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder,OneHotEncoder
import category_encoders as ce
be=ce.BinaryEncoder()
ohe=OneHotEncoder(sparse=False)
le=LabelEncoder()
oe=OrdinalEncoder()
df1["City"] = ohe.fit_transform(df1[["City"]])
temp=['Cold','Warm','Hot','Very Hot']
oe1=OrdinalEncoder(categories=[temp])
df1['Ord_1'] = oe1.fit_transform(df1[["Ord_1"]])
edu=['High School','Diploma','Bachelors','Masters','PhD']
oe2=OrdinalEncoder(categories=[edu])
df1['Ord_2']= oe2.fit_transform(df1[["Ord_2"]])
df1
```

feature scaling:

```
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
df2=pd.DataFrame(sc.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'City', 'Ord_1','Ord_2','Target'])
df2
from sklearn.preprocessing import StandardScaler
sc1=StandardScaler()
df3=pd.DataFrame(sc1.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'City', 'Ord_1','Ord_2','Target'])
df3
from sklearn.preprocessing import MaxAbsScaler
sc2=MaxAbsScaler()
df4=pd.DataFrame(sc2.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'City', 'Ord_1','Ord_2','Target'])
df4
from sklearn.preprocessing import RobustScaler
sc3=RobustScaler()
df5=pd.DataFrame(sc3.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'City', 'Ord_1','Ord_2','Target'])
df5
```

Encoading.draw:

```
import pandas as pd
df=pd.read_csv("Encoding Data.csv")
df
```

feature generation:

```
import category_encoders as ce
be=ce.BinaryEncoder()
ndf=be.fit_transform(df["bin_1"])
df["bin_1"] = be.fit_transform(df["bin_1"])
ndf
ndf2=be.fit_transform(df["bin_2"])
df["bin_2"] = be.fit_transform(df["bin_2"])
ndf2
df1=df.copy()
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder
le=LabelEncoder()
oe=OrdinalEncoder()
df1["nom_0"] = oe.fit_transform(df1[["nom_0"]])
temp=['Cold','Warm','Hot']
oe2=OrdinalEncoder(categories=[temp])
df1['ord_2'] = oe2.fit_transform(df1[['ord_2']])
df1
```

feature scaling:

```
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
df0=pd.DataFrame(sc.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'nom_0','ord_2'])
df0
from sklearn.preprocessing import StandardScaler
sc1=StandardScaler()
df2=pd.DataFrame(sc1.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'nom_0','ord_2'])
df2
from sklearn.preprocessing import MaxAbsScaler
sc2=MaxAbsScaler()
df3=pd.DataFrame(sc2.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'nom_0','ord_2'])
df3
from sklearn.preprocessing import RobustScaler
sc3=RobustScaler()
df4=pd.DataFrame(sc3.fit_transform(df1),columns=['id', 'bin_1', 'bin_2', 'nom_0','ord_2'])
```

df4

Titanic.csv:

```
import pandas as pd
df=pd.read_csv("titanic_dataset.csv")
df

#removing unwanted data
df.drop("Name",axis=1,inplace=True)
df.drop("Ticket",axis=1,inplace=True)
df.drop("Cabin",axis=1,inplace=True)

#data cleaning
df.isnull().sum()
df["Age"]=df["Age"].fillna(df["Age"].median())
df["Embarked"]=df["Embarked"].fillna(df["Embarked"].mode()[0])
df.isnull().sum()
df
```

feature encoding:

```
from category_encoders import BinaryEncoder
be=BinaryEncoder()
df["Sex"]=be.fit_transform(df[["Sex"]])
ndf=be.fit_transform(df["Sex"])
ndf
df1=df.copy()
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
embark=['S','C','Q']
e1=OrdinalEncoder(categories=[embark])
df1['Embarked'] = e1.fit_transform(df[['Embarked']])
df1
```

feature scaling:

```
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
df2=pd.DataFrame(sc.fit_transform(df1),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Fare
','Embarked'])
df2
from sklearn.preprocessing import StandardScaler
sc1=StandardScaler()
df3=pd.DataFrame(sc1.fit_transform(df1),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Far
e','Embarked'])
df3
from sklearn.preprocessing import MaxAbsScaler
sc2=MaxAbsScaler()
df4=pd.DataFrame(sc2.fit_transform(df1),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Far
e','Embarked'])
df4
from sklearn.preprocessing import RobustScaler
sc3=RobustScaler()
df5=pd.DataFrame(sc3.fit_transform(df1),columns=['Passenger','Survived','Pclass','Sex','Age','SibSp','Parch','Far
e','Embarked'])
df5
```

# OUTPUT:

| | id | bin_1 | bin_2 | City | Ord_1 | Ord_2 | Target |
|---|---|---|---|---|---|---|---|
| 0 | 0 | F | N | Delhi | Hot | High School | 0 |
| 1 | 1 | F | Y | Bangalore | Warm | Masters | 1 |
| 2 | 2 | M | N | Mumbai | Very Hot | Diploma | 1 |
| 3 | 3 | M | Y | Chennai | Cold | Bachelors | 0 |
| 4 | 4 | M | Y | Delhi | Cold | Bachelors | 1 |
| 5 | 5 | F | N | Delhi | Very Hot | Masters | 0 |
| 6 | 6 | M | N | Chennai | Warm | PhD | 1 |
| 7 | 7 | F | N | Chennai | Hot | High School | 1 |
| 8 | 8 | M | N | Delhi | Very Hot | High School | 0 |
| 9 | 9 | F | Y | Delhi | Warm | PhD | 0 |

| | bin_1_0 | bin_1_1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 1 |
| 2 | 1 | 0 |
| 3 | 1 | 0 |
| 4 | 1 | 0 |
| 5 | 0 | 1 |
| 6 | 1 | 0 |
| 7 | 0 | 1 |
| 8 | 1 | 0 |
| 9 | 0 | 1 |

| | bin_2_0 | bin_2_1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | 1 | 0 |
| 4 | 1 | 0 |
| 5 | 0 | 1 |
| 6 | 0 | 1 |
| 7 | 0 | 1 |
| 8 | 0 | 1 |
| 9 | 1 | 0 |

| | id | bin_1 | bin_2 | City | Ord_1 | Ord_2 | Target |
|---|---|---|---|---|---|---|---|
| 0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.666667 | 0.00 | 0.0 |
| 1 | 0.111111 | 0.0 | 1.0 | 1.0 | 0.333333 | 0.75 | 1.0 |
| 2 | 0.222222 | 1.0 | 0.0 | 0.0 | 1.000000 | 0.25 | 1.0 |
| 3 | 0.333333 | 1.0 | 1.0 | 0.0 | 0.000000 | 0.50 | 0.0 |
| 4 | 0.444444 | 1.0 | 1.0 | 0.0 | 0.000000 | 0.50 | 1.0 |
| 5 | 0.555556 | 0.0 | 0.0 | 0.0 | 1.000000 | 0.75 | 0.0 |
| 6 | 0.666667 | 1.0 | 0.0 | 0.0 | 0.333333 | 1.00 | 1.0 |
| 7 | 0.777778 | 0.0 | 0.0 | 0.0 | 0.666667 | 0.00 | 1.0 |
| 8 | 0.888889 | 1.0 | 0.0 | 0.0 | 1.000000 | 0.00 | 0.0 |
| 9 | 1.000000 | 0.0 | 1.0 | 0.0 | 0.333333 | 1.00 | 0.0 |

| | id | bin_1 | bin_2 | City | Ord_1 | Ord_2 | Target |
|---|---|---|---|---|---|---|---|
| 0 | -1.566699 | -1.0 | -0.816497 | -0.333333 | 0.359211 | -1.255555 | -1.0 |
| 1 | -1.218544 | -1.0 | 1.224745 | 3.000000 | -0.538816 | 0.726900 | 1.0 |
| 2 | -0.870388 | 1.0 | -0.816497 | -0.333333 | 1.257237 | -0.594737 | 1.0 |
| 3 | -0.522233 | 1.0 | 1.224745 | -0.333333 | -1.436842 | 0.066082 | -1.0 |
| 4 | -0.174078 | 1.0 | 1.224745 | -0.333333 | -1.436842 | 0.066082 | 1.0 |
| 5 | 0.174078 | -1.0 | -0.816497 | -0.333333 | 1.257237 | 0.726900 | -1.0 |
| 6 | 0.522233 | 1.0 | -0.816497 | -0.333333 | -0.538816 | 1.387719 | 1.0 |
| 7 | 0.870388 | -1.0 | -0.816497 | -0.333333 | 0.359211 | -1.255555 | 1.0 |
| 8 | 1.218544 | 1.0 | -0.816497 | -0.333333 | 1.257237 | -1.255555 | -1.0 |
| 9 | 1.566699 | -1.0 | 1.224745 | -0.333333 | -0.538816 | 1.387719 | -1.0 |

| | id | bin_1 | bin_2 | City | Ord_1 | Ord_2 | Target |
|---|---|---|---|---|---|---|---|
| 0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.666667 | 0.00 | 0.0 |
| 1 | 0.111111 | 0.0 | 1.0 | 1.0 | 0.333333 | 0.75 | 1.0 |
| 2 | 0.222222 | 1.0 | 0.0 | 0.0 | 1.000000 | 0.25 | 1.0 |
| 3 | 0.333333 | 1.0 | 1.0 | 0.0 | 0.000000 | 0.50 | 0.0 |
| 4 | 0.444444 | 1.0 | 1.0 | 0.0 | 0.000000 | 0.50 | 1.0 |
| 5 | 0.555556 | 0.0 | 0.0 | 0.0 | 1.000000 | 0.75 | 0.0 |
| 6 | 0.666667 | 1.0 | 0.0 | 0.0 | 0.333333 | 1.00 | 1.0 |
| 7 | 0.777778 | 0.0 | 0.0 | 0.0 | 0.666667 | 0.00 | 1.0 |
| 8 | 0.888889 | 1.0 | 0.0 | 0.0 | 1.000000 | 0.00 | 0.0 |
| 9 | 1.000000 | 0.0 | 1.0 | 0.0 | 0.333333 | 1.00 | 0.0 |

| | id | bin_1 | bin_2 | City | Ord_1 | Ord_2 | Target |
|---|---|---|---|---|---|---|---|
| 0 | -1.000000 | -0.5 | 0.0 | 0.0 | 0.285714 | -0.727273 | -0.5 |
| 1 | -0.777778 | -0.5 | 1.0 | 1.0 | -0.285714 | 0.363636 | 0.5 |
| 2 | -0.555556 | 0.5 | 0.0 | 0.0 | 0.857143 | -0.363636 | 0.5 |
| 3 | -0.333333 | 0.5 | 1.0 | 0.0 | -0.857143 | 0.000000 | -0.5 |
| 4 | -0.111111 | 0.5 | 1.0 | 0.0 | -0.857143 | 0.000000 | 0.5 |
| 5 | 0.111111 | -0.5 | 0.0 | 0.0 | 0.857143 | 0.363636 | -0.5 |
| 6 | 0.333333 | 0.5 | 0.0 | 0.0 | -0.285714 | 0.727273 | 0.5 |
| 7 | 0.555556 | -0.5 | 0.0 | 0.0 | 0.285714 | -0.727273 | 0.5 |
| 8 | 0.777778 | 0.5 | 0.0 | 0.0 | 0.857143 | -0.727273 | -0.5 |
| 9 | 1.000000 | -0.5 | 1.0 | 0.0 | -0.285714 | 0.727273 | -0.5 |

| | id | bin_1 | bin_2 | nom_0 | ord_2 |
|---|---|---|---|---|---|
| 0 | 0 | F | N | Red | Hot |
| 1 | 1 | F | Y | Blue | Warm |
| 2 | 2 | F | N | Blue | Cold |
| 3 | 3 | F | N | Green | Warm |
| 4 | 4 | T | N | Red | Cold |
| 5 | 5 | T | N | Green | Hot |
| 6 | 6 | F | N | Red | Cold |
| 7 | 7 | T | N | Red | Cold |
| 8 | 8 | F | N | Blue | Warm |
| 9 | 9 | F | Y | Red | Hot |

| | bin_1_0 | bin_1_1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 0 | 1 |
| 2 | 0 | 1 |
| 3 | 0 | 1 |
| 4 | 1 | 0 |
| 5 | 1 | 0 |
| 6 | 0 | 1 |
| 7 | 1 | 0 |
| 8 | 0 | 1 |
| 9 | 0 | 1 |

| | bin_2_0 | bin_2_1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | 0 | 1 |
| 4 | 0 | 1 |
| 5 | 0 | 1 |
| 6 | 0 | 1 |
| 7 | 0 | 1 |
| 8 | 0 | 1 |
| 9 | 1 | 0 |

| | id | bin_1 | bin_2 | nom_0 | ord_2 |
|---|---|---|---|---|---|
| 0 | 0.000000 | 0.0 | 0.0 | 1.0 | 1.0 |
| 1 | 0.111111 | 0.0 | 1.0 | 0.0 | 0.5 |
| 2 | 0.222222 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 0.333333 | 0.0 | 0.0 | 0.5 | 0.5 |
| 4 | 0.444444 | 1.0 | 0.0 | 1.0 | 0.0 |
| 5 | 0.555556 | 1.0 | 0.0 | 0.5 | 1.0 |
| 6 | 0.666667 | 0.0 | 0.0 | 1.0 | 0.0 |
| 7 | 0.777778 | 1.0 | 0.0 | 1.0 | 0.0 |
| 8 | 0.888889 | 0.0 | 0.0 | 0.0 | 0.5 |
| 9 | 1.000000 | 0.0 | 1.0 | 1.0 | 1.0 |

| | id | bin_1 | bin_2 | nom_0 | ord_2 |
|---|---|---|---|---|---|
| 0 | -1.566699 | -0.654654 | -0.5 | 0.917663 | 1.324244 |
| 1 | -1.218544 | -0.654654 | 2.0 | -1.376494 | 0.120386 |
| 2 | -0.870388 | -0.654654 | -0.5 | -1.376494 | -1.083473 |
| 3 | -0.522233 | -0.654654 | -0.5 | -0.229416 | 0.120386 |
| 4 | -0.174078 | 1.527525 | -0.5 | 0.917663 | -1.083473 |
| 5 | 0.174078 | 1.527525 | -0.5 | -0.229416 | 1.324244 |
| 6 | 0.522233 | -0.654654 | -0.5 | 0.917663 | -1.083473 |
| 7 | 0.870388 | 1.527525 | -0.5 | 0.917663 | -1.083473 |
| 8 | 1.218544 | -0.654654 | -0.5 | -1.376494 | 0.120386 |
| 9 | 1.566699 | -0.654654 | 2.0 | 0.917663 | 1.324244 |

| | id | bin_1 | bin_2 | nom_0 | ord_2 |
|---|---|---|---|---|---|
| 0 | 0.000000 | 0.0 | 0.0 | 1.0 | 1.0 |
| 1 | 0.111111 | 0.0 | 1.0 | 0.0 | 0.5 |
| 2 | 0.222222 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 0.333333 | 0.0 | 0.0 | 0.5 | 0.5 |
| 4 | 0.444444 | 1.0 | 0.0 | 1.0 | 0.0 |
| 5 | 0.555556 | 1.0 | 0.0 | 0.5 | 1.0 |
| 6 | 0.666667 | 0.0 | 0.0 | 1.0 | 0.0 |
| 7 | 0.777778 | 1.0 | 0.0 | 1.0 | 0.0 |
| 8 | 0.888889 | 0.0 | 0.0 | 0.0 | 0.5 |
| 9 | 1.000000 | 0.0 | 1.0 | 1.0 | 1.0 |

| | id | bin_1 | bin_2 | nom_0 | ord_2 |
|---|---|---|---|---|---|
| 0 | -1.000000 | 0.000000 | 0.0 | 0.285714 | 0.571429 |
| 1 | -0.777778 | 0.000000 | 1.0 | -0.857143 | 0.000000 |
| 2 | -0.555556 | 0.000000 | 0.0 | -0.857143 | -0.571429 |
| 3 | -0.333333 | 0.000000 | 0.0 | -0.285714 | 0.000000 |
| 4 | -0.111111 | 1.333333 | 0.0 | 0.285714 | -0.571429 |
| 5 | 0.111111 | 1.333333 | 0.0 | -0.285714 | 0.571429 |
| 6 | 0.333333 | 0.000000 | 0.0 | 0.285714 | -0.571429 |
| 7 | 0.555556 | 1.333333 | 0.0 | 0.285714 | -0.571429 |
| 8 | 0.777778 | 0.000000 | 0.0 | -0.857143 | 0.000000 |
| 9 | 1.000000 | 0.000000 | 1.0 | 0.285714 | 0.571429 |

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 2 | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| 3 | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| 4 | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 887 | 0 | 2 | Montvila, Rev. Juozas | male | 27.0 | 0 | 0 | 211536 | 13.0000 | NaN | S |
| 887 | 888 | 1 | 1 | Graham, Miss. Margaret Edith | female | 19.0 | 0 | 0 | 112053 | 30.0000 | B42 | S |
| 888 | 889 | 0 | 3 | Johnston, Miss. Catherine Helen "Carrie" | female | NaN | 1 | 2 | W./C. 6607 | 23.4500 | NaN | S |
| 889 | 890 | 1 | 1 | Behr, Mr. Karl Howell | male | 26.0 | 0 | 0 | 111369 | 30.0000 | C148 | C |
| 890 | 891 | 0 | 3 | Dooley, Mr. Patrick | male | 32.0 | 0 | 0 | 370376 | 7.7500 | NaN | Q |

891 rows × 12 columns

```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Fare             0
Embarked         2
dtype: int64
```

```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age            177
SibSp            0
Parch            0
Fare             0
Embarked         2
dtype: int64
```

```
PassengerId      0
Survived         0
Pclass           0
Sex              0
Age              0
SibSp            0
Parch            0
Fare             0
Embarked         0
dtype: int64
```

| | PassengerId | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S |
| 1 | 2 | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C |
| 2 | 3 | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S |
| 3 | 4 | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S |
| 4 | 5 | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 887 | 0 | 2 | male | 27.0 | 0 | 0 | 13.0000 | S |
| 887 | 888 | 1 | 1 | female | 19.0 | 0 | 0 | 30.0000 | S |
| 888 | 889 | 0 | 3 | female | 28.0 | 1 | 2 | 23.4500 | S |
| 889 | 890 | 1 | 1 | male | 26.0 | 0 | 0 | 30.0000 | C |
| 890 | 891 | 0 | 3 | male | 32.0 | 0 | 0 | 7.7500 | Q |

891 rows × 9 columns

| | Sex_0 | Sex_1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 0 |
| 4 | 0 | 1 |
| ... | ... | ... |
| 886 | 0 | 1 |
| 887 | 1 | 0 |
| 888 | 1 | 0 |
| 889 | 0 | 1 |
| 890 | 0 | 1 |

891 rows × 2 columns

| | PassengerId | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | 0 | 22.0 | 1 | 0 | 7.2500 | 0.0 |
| 1 | 2 | 1 | 1 | 1 | 38.0 | 1 | 0 | 71.2833 | 1.0 |
| 2 | 3 | 1 | 3 | 1 | 26.0 | 0 | 0 | 7.9250 | 0.0 |
| 3 | 4 | 1 | 1 | 1 | 35.0 | 1 | 0 | 53.1000 | 0.0 |
| 4 | 5 | 0 | 3 | 0 | 35.0 | 0 | 0 | 8.0500 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 887 | 0 | 2 | 0 | 27.0 | 0 | 0 | 13.0000 | 0.0 |
| 887 | 888 | 1 | 1 | 1 | 19.0 | 0 | 0 | 30.0000 | 0.0 |
| 888 | 889 | 0 | 3 | 1 | 28.0 | 1 | 2 | 23.4500 | 0.0 |
| 889 | 890 | 1 | 1 | 0 | 26.0 | 0 | 0 | 30.0000 | 1.0 |
| 890 | 891 | 0 | 3 | 0 | 32.0 | 0 | 0 | 7.7500 | 2.0 |

891 rows × 9 columns

| | Passenger | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.000000 | 0.0 | 1.0 | 0.0 | 0.271174 | 0.125 | 0.000000 | 0.014151 | 0.0 |
| 1 | 0.001124 | 1.0 | 0.0 | 1.0 | 0.472229 | 0.125 | 0.000000 | 0.139136 | 0.5 |
| 2 | 0.002247 | 1.0 | 1.0 | 1.0 | 0.321438 | 0.000 | 0.000000 | 0.015469 | 0.0 |
| 3 | 0.003371 | 1.0 | 0.0 | 1.0 | 0.434531 | 0.125 | 0.000000 | 0.103644 | 0.0 |
| 4 | 0.004494 | 0.0 | 1.0 | 0.0 | 0.434531 | 0.000 | 0.000000 | 0.015713 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 0.995506 | 0.0 | 0.5 | 0.0 | 0.334004 | 0.000 | 0.000000 | 0.025374 | 0.0 |
| 887 | 0.996629 | 1.0 | 0.0 | 1.0 | 0.233476 | 0.000 | 0.000000 | 0.058556 | 0.0 |
| 888 | 0.997753 | 0.0 | 1.0 | 1.0 | 0.346569 | 0.125 | 0.333333 | 0.045771 | 0.0 |
| 889 | 0.998876 | 1.0 | 0.0 | 0.0 | 0.321438 | 0.000 | 0.000000 | 0.058556 | 0.5 |
| 890 | 1.000000 | 0.0 | 1.0 | 0.0 | 0.396833 | 0.000 | 0.000000 | 0.015127 | 1.0 |

891 rows × 9 columns

| | Passenger | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|---|
| 0 | -1.730108 | -0.789272 | 0.827377 | -0.737695 | -0.565736 | 0.432793 | -0.473674 | -0.502445 | -0.568837 |
| 1 | -1.726220 | 1.266990 | -1.566107 | 1.355574 | 0.663861 | 0.432793 | -0.473674 | 0.786845 | 1.005181 |
| 2 | -1.722332 | 1.266990 | 0.827377 | 1.355574 | -0.258337 | -0.474545 | -0.473674 | -0.488854 | -0.568837 |
| 3 | -1.718444 | 1.266990 | -1.566107 | 1.355574 | 0.433312 | 0.432793 | -0.473674 | 0.420730 | -0.568837 |
| 4 | -1.714556 | -0.789272 | 0.827377 | -0.737695 | 0.433312 | -0.474545 | -0.473674 | -0.486337 | -0.568837 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 1.714556 | -0.789272 | -0.369365 | -0.737695 | -0.181487 | -0.474545 | -0.473674 | -0.386671 | -0.568837 |
| 887 | 1.718444 | 1.266990 | -1.566107 | 1.355574 | -0.796286 | -0.474545 | -0.473674 | -0.044381 | -0.568837 |
| 888 | 1.722332 | -0.789272 | 0.827377 | 1.355574 | -0.104637 | 0.432793 | 2.008933 | -0.176263 | -0.568837 |
| 889 | 1.726220 | 1.266990 | -1.566107 | -0.737695 | -0.258337 | -0.474545 | -0.473674 | -0.044381 | 1.005181 |
| 890 | 1.730108 | -0.789272 | 0.827377 | -0.737695 | 0.202762 | -0.474545 | -0.473674 | -0.492378 | 2.579199 |

891 rows × 9 columns

| | Passenger | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.001122 | 0.0 | 1.000000 | 0.0 | 0.2750 | 0.125 | 0.000000 | 0.014151 | 0.0 |
| 1 | 0.002245 | 1.0 | 0.333333 | 1.0 | 0.4750 | 0.125 | 0.000000 | 0.139136 | 0.5 |
| 2 | 0.003367 | 1.0 | 1.000000 | 1.0 | 0.3250 | 0.000 | 0.000000 | 0.015469 | 0.0 |
| 3 | 0.004489 | 1.0 | 0.333333 | 1.0 | 0.4375 | 0.125 | 0.000000 | 0.103644 | 0.0 |
| 4 | 0.005612 | 0.0 | 1.000000 | 0.0 | 0.4375 | 0.000 | 0.000000 | 0.015713 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 0.995511 | 0.0 | 0.666667 | 0.0 | 0.3375 | 0.000 | 0.000000 | 0.025374 | 0.0 |
| 887 | 0.996633 | 1.0 | 0.333333 | 1.0 | 0.2375 | 0.000 | 0.000000 | 0.058556 | 0.0 |
| 888 | 0.997755 | 0.0 | 1.000000 | 1.0 | 0.3500 | 0.125 | 0.333333 | 0.045771 | 0.0 |
| 889 | 0.998878 | 1.0 | 0.333333 | 0.0 | 0.3250 | 0.000 | 0.000000 | 0.058556 | 0.5 |
| 890 | 1.000000 | 0.0 | 1.000000 | 0.0 | 0.4000 | 0.000 | 0.000000 | 0.015127 | 1.0 |

891 rows × 9 columns

| | Passenger | Survived | Pclass | Sex | Age | SibSp | Parch | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|---|
| 0 | -1.000000 | 0.0 | 0.0 | 0.0 | -0.461538 | 1.0 | 0.0 | -0.312011 | 0.0 |
| 1 | -0.997753 | 1.0 | -2.0 | 1.0 | 0.769231 | 1.0 | 0.0 | 2.461242 | 1.0 |
| 2 | -0.995506 | 1.0 | 0.0 | 1.0 | -0.153846 | 0.0 | 0.0 | -0.282777 | 0.0 |
| 3 | -0.993258 | 1.0 | -2.0 | 1.0 | 0.538462 | 1.0 | 0.0 | 1.673732 | 0.0 |
| 4 | -0.991011 | 0.0 | 0.0 | 0.0 | 0.538462 | 0.0 | 0.0 | -0.277363 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 886 | 0.991011 | 0.0 | -1.0 | 0.0 | -0.076923 | 0.0 | 0.0 | -0.062981 | 0.0 |
| 887 | 0.993258 | 1.0 | -2.0 | 1.0 | -0.692308 | 0.0 | 0.0 | 0.673281 | 0.0 |
| 888 | 0.995506 | 0.0 | 0.0 | 1.0 | 0.000000 | 1.0 | 2.0 | 0.389604 | 0.0 |
| 889 | 0.997753 | 1.0 | -2.0 | 0.0 | -0.153846 | 0.0 | 0.0 | 0.673281 | 1.0 |
| 890 | 1.000000 | 0.0 | 0.0 | 0.0 | 0.307692 | 0.0 | 0.0 | -0.290356 | 2.0 |

891 rows × 9 columns

**RESULT:**

Feature Generation process and Feature Scaling process is applied to the given data frames sucessfully.

# Ex-06-Feature-Transformation

## AIM:

To read the given data and perform Feature Transformation process and save the data to a file.

## EXPLANATION:

Feature Transformation is a technique by which we can boost our model performance. Feature transformation is a mathematical transformation in which we apply a mathematical formula to a particular column(feature) and transform the values which are useful for our further analysis.

## ALGORITHM:

STEP 1: Read the given Data

STEP 2: Clean the Data Set using Data Cleaning Process

STEP 3: Apply Feature Transformation techniques to all the features of the data set

STEP 4: Save the data to the file

## CODE:

Developed by : M Vignesh
Registration Number : 212220233002

```
Feature Transformation - Data_to_Transform.csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
df = pd.read_csv("/content/Data_to_Transform.csv")
print(df)
df.head()
df.isnull().sum()
df.info()
df.describe()
df1 = df.copy()
sm.qqplot(df1.HighlyPositiveSkew,fit=True,line='45')
plt.show()
sm.qqplot(df1.HighlyNegativeSkew,fit=True,line='45')
plt.show()
sm.qqplot(df1.ModeratePositiveSkew,fit=True,line='45')
plt.show()
sm.qqplot(df1.ModerateNegativeSkew,fit=True,line='45')
plt.show()
df1['HighlyPositiveSkew'] = np.log(df1.HighlyPositiveSkew)
sm.qqplot(df1.HighlyPositiveSkew,fit=True,line='45')
plt.show()
df2 = df.copy()
df2['HighlyPositiveSkew'] = 1/df2.HighlyPositiveSkew
sm.qqplot(df2.HighlyPositiveSkew,fit=True,line='45')
plt.show()
df3 = df.copy()
```

```
df3['HighlyPositiveSkew'] = df3.HighlyPositiveSkew**(1/1.2)
sm.qqplot(df2.HighlyPositiveSkew,fit=True,line='45')
plt.show()
df4 = df.copy()
df4['ModeratePositiveSkew_1'],parameters =stats.yeojohnson(df4.ModeratePositiveSkew)
sm.qqplot(df4.ModeratePositiveSkew_1,fit=True,line='45')
plt.show()
from sklearn.preprocessing import PowerTransformer
trans = PowerTransformer("yeo-johnson")
df5 = df.copy()
df5['ModerateNegativeSkew_1'] = pd.DataFrame(trans.fit_transform(df5[['ModerateNegativeSkew']]))
sm.qqplot(df5['ModerateNegativeSkew_1'],line='45')
plt.show()
from sklearn.preprocessing import QuantileTransformer
qt = QuantileTransformer(output_distribution = 'normal')
df5['ModerateNegativeSkew_2'] = pd.DataFrame(qt.fit_transform(df5[['ModerateNegativeSkew']]))
sm.qqplot(df5['ModerateNegativeSkew_2'],line='45')
plt.show()
```

## OUTPUT:

Feature Transformation - Data_to_Transform.csv

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
from sklearn.preprocessing import QuantileTransformer
```

```
df = pd.read_csv("/content/Data_to_Transform.csv")
print(df)
```

```
      ModeratePositiveSkew  HighlyPositiveSkew  ModerateNegativeSkew  \
0                 0.899990            2.895074             11.180748
1                 1.113554            2.962385             10.842938
2                 1.156830            2.966378             10.817934
3                 1.264131            3.000324             10.764570
4                 1.323914            3.012109             10.753117
...                    ...                 ...                   ...
9995             14.749050           16.289513             -2.980821
9996             14.854474           16.396252             -3.147526
9997             15.262103           17.102991             -3.517256
9998             15.269983           17.628467             -4.689833
9999             16.204517           18.052331             -6.335679

      HighlyNegativeSkew
0               9.027485
1               9.009762
2               9.006134
3               9.000125
4               8.981296
...                  ...
9995           -3.254882
9996           -3.772332
9997           -4.717950
9998           -5.670496
9999           -7.036091

[10000 rows x 4 columns]
```

```
df.head()
```

| | ModeratePositiveSkew | HighlyPositiveSkew | ModerateNegativeSkew | HighlyNegativeSkew |
|---|---|---|---|---|
| **0** | 0.899990 | 2.895074 | 11.180748 | 9.027485 |
| **1** | 1.113554 | 2.962385 | 10.842938 | 9.009762 |
| **2** | 1.156830 | 2.966378 | 10.817934 | 9.006134 |
| **3** | 1.264131 | 3.000324 | 10.764570 | 9.000125 |
| **4** | 1.323914 | 3.012109 | 10.753117 | 8.981296 |

```
df.isnull().sum()
```

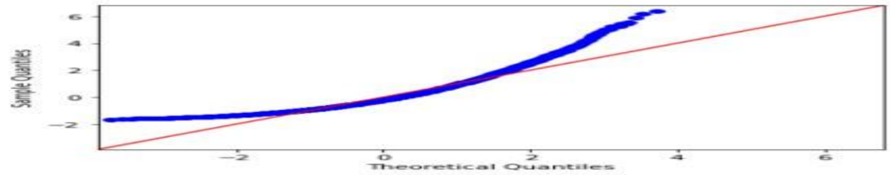
```
ModeratePositiveSkew    0
HighlyPositiveSkew      0
ModerateNegativeSkew    0
HighlyNegativeSkew      0
dtype: int64
```

```
df.info()
df.describe()
```

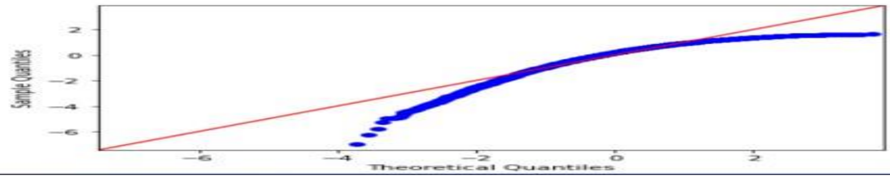
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ModeratePositiveSkew    10000 non-null  float64
 1   HighlyPositiveSkew      10000 non-null  float64
 2   ModerateNegativeSkew    10000 non-null  float64
 3   HighlyNegativeSkew      10000 non-null  float64
dtypes: float64(4)
memory usage: 312.6 KB
```

| | ModeratePositiveSkew | HighlyPositiveSkew | ModerateNegativeSkew | HighlyNegativeSkew |
|---|---|---|---|---|
| count | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 |
| mean | 6.000000 | 6.000000 | 6.000000 | 6.000000 |
| std | 2.047237 | 1.882685 | 2.045060 | 1.860556 |
| min | 0.899990 | 2.895074 | -6.335679 | -7.036091 |
| 25% | 4.518431 | 4.614818 | 4.779235 | 5.049961 |
| 50% | 5.751642 | 5.634956 | 6.233932 | 6.360402 |
| 75% | 7.258956 | 6.924035 | 7.465319 | 7.366753 |
| max | 16.204517 | 18.052331 | 11.180748 | 9.027485 |

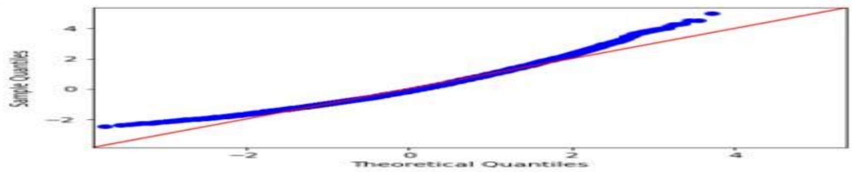
```
df1 = df.copy()
sm.qqplot(df1.HighlyPositiveSkew,fit=True,line='45')
plt.show()
```



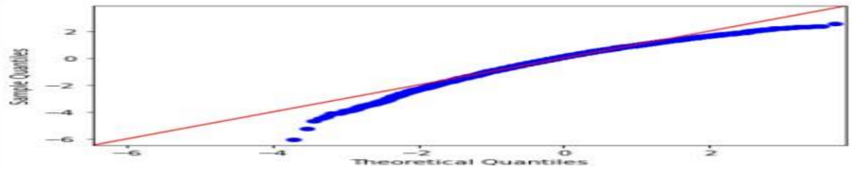

```
sm.qqplot(df1.HighlyNegativeSkew,fit=True,line='45')
plt.show()
```




```
sm.qqplot(df1.ModeratePositiveSkew,fit=True,line='45')
plt.show()
```

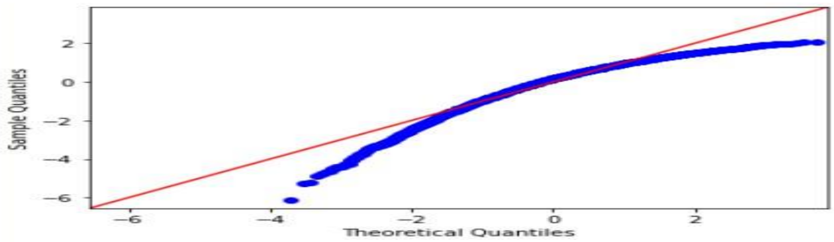



```
sm.qqplot(df1.ModerateNegativeSkew,fit=True,line='45')
plt.show()
```



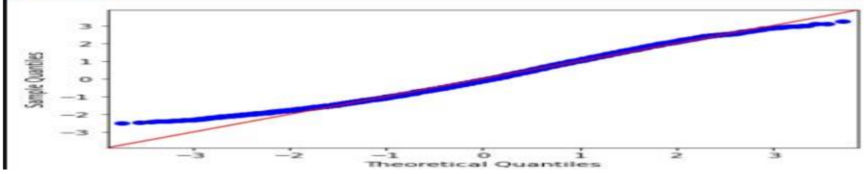

Log Transformation

```
df1['HighlyPositiveSkew'] = np.log(df1.HighlyPositiveSkew)
sm.qqplot(df1.HighlyPositiveSkew,fit=True,line='45')
plt.show()
```



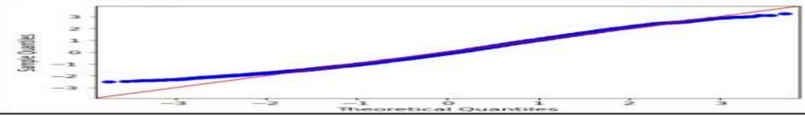
Reciprocal Transformation

```
df2 = df.copy()
df2['HighlyPositiveSkew'] = 1/df2.HighlyPositiveSkew
sm.qqplot(df2.HighlyPositiveSkew,fit=True,line='45')
plt.show()
```



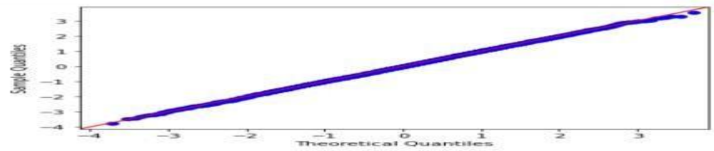

SquareRoot Transformation

```
df3 = df.copy()
df3['HighlyPositiveSkew'] = df3.HighlyPositiveSkew**(1/1.2)
sm.qqplot(df2.HighlyPositiveSkew,fit=True,line='45')
plt.show()
```

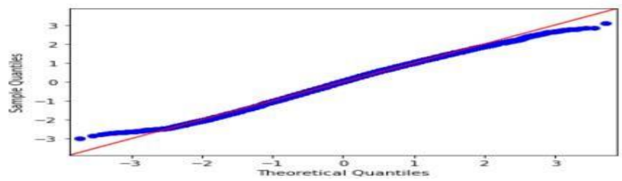



Power Transformation

```
df4 = df.copy()
df4['ModeratePositiveSkew_1'],parameters =stats.yeojohnson(df4.ModeratePositiveSkew
sm.qqplot(df4.ModeratePositiveSkew_1,fit=True,line='45')
plt.show()
```

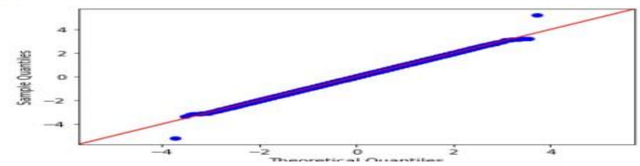



```
from sklearn.preprocessing import PowerTransformer
trans = PowerTransformer("yeo-johnson")
df5 = df.copy()
df5['ModerateNegativeSkew_1'] = pd.DataFrame(trans.fit_transform(df5[['ModerateNegativeSkew']]))
sm.qqplot(df5['ModerateNegativeSkew_1'],line='45')
plt.show()
```




Quantile Transformation

```
from sklearn.preprocessing import QuantileTransformer
qt = QuantileTransformer(output_distribution = 'normal')
df5['ModerateNegativeSkew_2'] = pd.DataFrame(qt.fit_transform(df5[['ModerateNegativeSkew']]))
sm.qqplot(df5['ModerateNegativeSkew_2'],line='45')
plt.show()
```




**RESULT:**

Thus the Feature Transformation for the given datasets had been executed successfully

# Ex-07-Data-Visualization

### AIM

To Perform Data Visualization on a complex dataset and save the data to a file.

### Explanation

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

### ALGORITHM

**STEP 1:** Read the given Data

**STEP 2:** Clean the Data Set using Data Cleaning Process

**STEP 3:** Apply Feature generation and selection techniques to all the features of the data set

**STEP 4:** Apply data visualization techniques to identify the patterns of the data.

### CODE

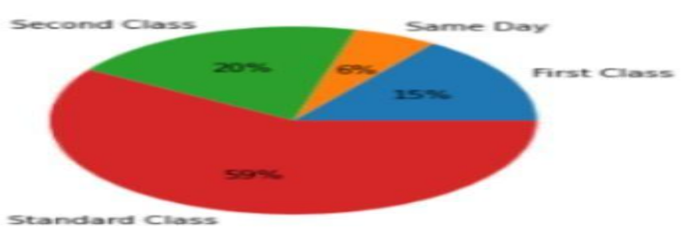
Developed by : M Vignesh
Registration Number : 212220233002

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv("Superstore.csv")
df.head()
df1=df.loc[:,["Ship Mode","Sales"]]
df1=df.groupby(by=["Ship Mode"]).sum()
labels=[]
for i in df1.index:
    labels.append(i)
colors = sns.color_palette('bright')
plt.pie(df1["Sales"],labels=labels,autopct = '%0.0f%%')
plt.show
df.head()
df1
df1.info()
df1=df1.groupby(by=["Category"]).sum()
labels=[]
for i in df1.index:
    plt.pie(df1["Profit"],colors = colors,labels=labels,autopct='0.0f%%')
sates=df.loc[:,["State","Sales"]]
plt.figure(figsize=(10,10))
sns.barplot(x="State",y="Sales",data=states)
plt.xticks(rotation=90)
```

```
plt.xlabel=("STATE")
plt.ylabel=("SALES")
plt.show()
sns.set_style('whitegrid')
sns.countplot(x='Segment',data= df, palette='rainbow')
sns.set_style('whitegrid')
sns.countplot(x='Category',data=df, palette='rainbow')
sns.set_style('whitegrid')
sns.countplot(x='Sub-Category',data=df, palette='rainbow')
sns.set_style('whitegrid')
sns.countplot(x='Region',data=df, palette='rainbow')
sns.set_style('whitegrid')
sns.countplot(x='Ship Mode',data=df, palette='rainbow')
category_hist = sns.FacetGrid(df, col='Ship Mode', palette='rainbow')
category_hist.map(plt.hist, 'Category')
category_hist.set_ylabels('Number')
subcategory_hist = sns.FacetGrid(df, col='Segment', height=10.5, aspect=4.6)
subcategory_hist.map(plt.hist, 'Sub-Category')
subcategory_hist.set_ylabels('Number')
grid = sns.FacetGrid(df, row='Category', col='Sub-Category', height=2.2, aspect=1.6)
grid.map(sns.barplot, 'Profit', 'Segment', alpha=.5, ci=None)
grid.add_legend()
```

# OUTPUT

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | ... | Postal Code | Region | Product ID | Category | Sub-Category |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2016-152156 | 11/8/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | 42420 | South | FUR-BO-10001798 | Furniture | Bookcases |
| 1 | 2 | CA-2016-152156 | 11/8/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | 42420 | South | FUR-CH-10000454 | Furniture | Chairs |
| 2 | 3 | CA-2016-138688 | 6/12/2016 | 6/16/2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... | 90036 | West | OFF-LA-10000240 | Office Supplies | Labels |
| 3 | 4 | US-2015-108966 | 10/11/2015 | 10/18/2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | 33311 | South | FUR-TA-10000577 | Furniture | Tables |
| 4 | 5 | US-2015- | 10/11/2015 | 10/18/2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | 33311 | South | OFF-ST-10000760 | Office Supplies | Storage |



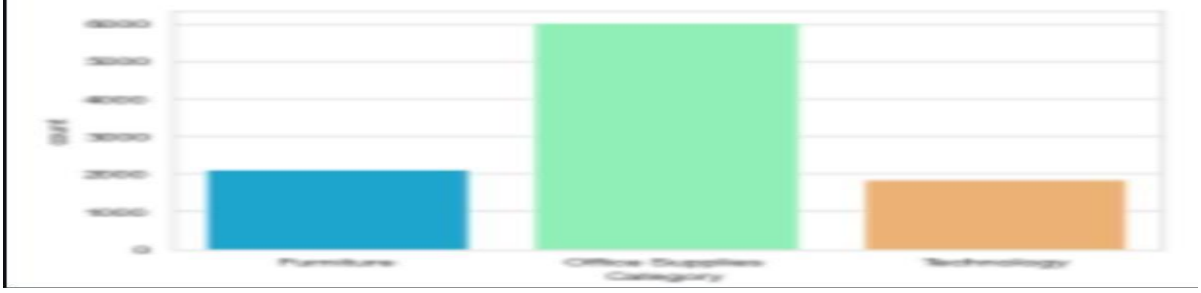

| | Row ID | Order ID | Order Date | Ship Date | Ship Mode | Customer ID | Customer Name | Segment | Country | City | ... | Postal Code | Region | Product ID | Category | Sub-Category |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CA-2016-152156 | 11/8/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | 42420 | South | FUR-BO-10001798 | Furniture | Bookcases |
| 1 | 2 | CA-2016-152156 | 11/8/2016 | 11/11/2016 | Second Class | CG-12520 | Claire Gute | Consumer | United States | Henderson | ... | 42420 | South | FUR-CH-10000454 | Furniture | Chairs |
| 2 | 3 | CA-2016-138688 | 6/12/2016 | 6/16/2016 | Second Class | DV-13045 | Darrin Van Huff | Corporate | United States | Los Angeles | ... | 90036 | West | OFF-LA-10000240 | Office Supplies | Labels |
| 3 | 4 | US-2015-108966 | 10/11/2015 | 10/18/2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | 33311 | South | FUR-TA-10000577 | Furniture | Tables |
| 4 | 5 | US-2015-108966 | 10/11/2015 | 10/18/2015 | Standard Class | SO-20335 | Sean O'Donnell | Consumer | United States | Fort Lauderdale | ... | 33311 | South | OFF-ST-10000760 | Office Supplies | Storage |

5 rows × 21 columns

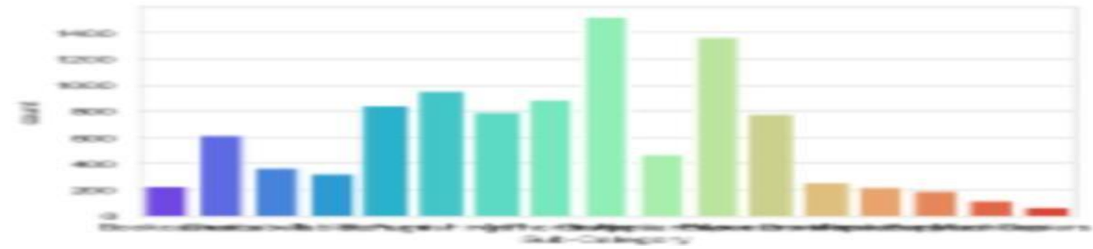
| Ship Mode | Row ID | Postal Code | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|
| First Class | 7498535 | 84229511 | 3.514284e+05 | 5693 | 253.17 | 48969.8399 |
| Same Day | 2784998 | 31242093 | 1.283631e+05 | 1960 | 82.75 | 15891.7589 |
| Second Class | 9601997 | 108192588 | 4.591936e+05 | 7423 | 270.15 | 57446.6354 |
| Standard Class | 30059485 | 327908460 | 1.358216e+06 | 22797 | 955.02 | 164088.7875 |

```
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, First Class to Standard Class
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Row ID        4 non-null      int64
 1   Postal Code   4 non-null      int64
 2   Sales         4 non-null      float64
 3   Quantity      4 non-null      int64
 4   Discount      4 non-null      float64
 5   Profit        4 non-null      float64
dtypes: float64(3), int64(3)
memory usage: 224.0+ bytes
```

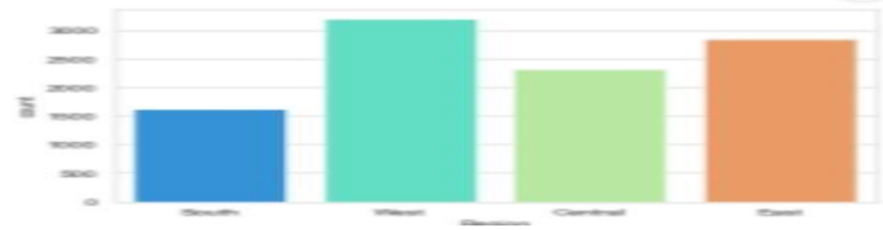
<AxesSubplot:xlabel='Category', ylabel='count'>



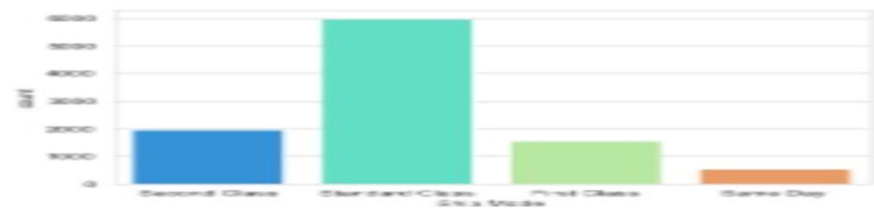
<AxesSubplot:xlabel='Sub-Category', ylabel='count'>



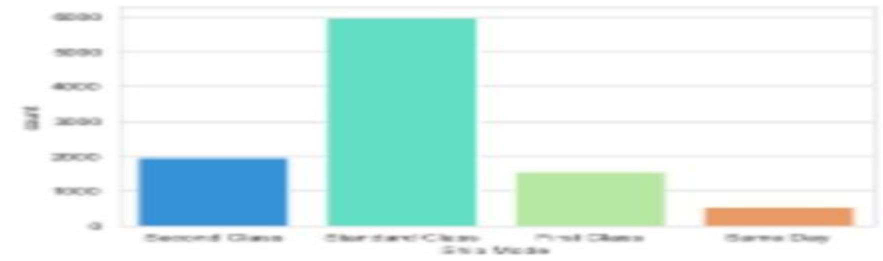
<AxesSubplot:xlabel='Region', ylabel='count'>



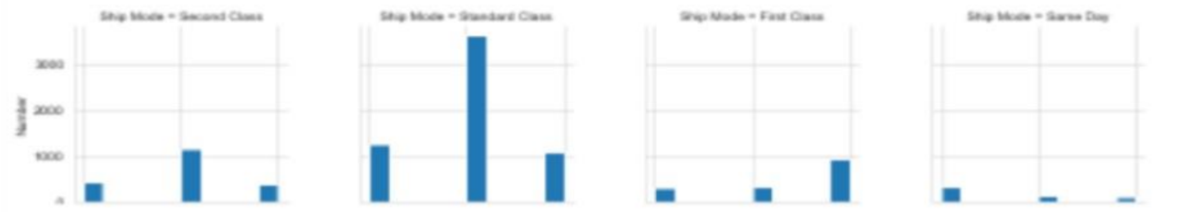
<AxesSubplot:xlabel='Ship Mode', ylabel='count'>



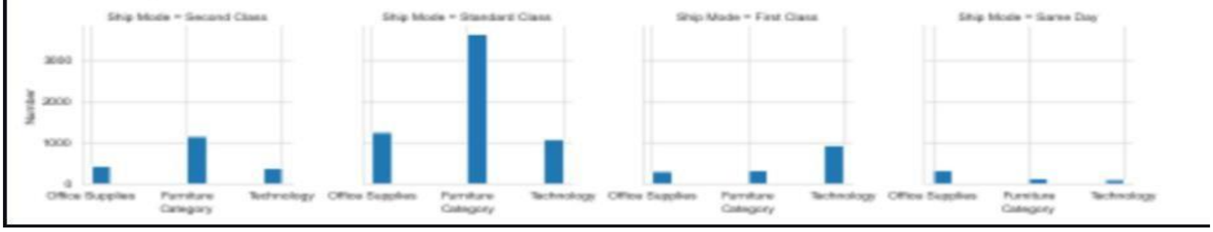
<AxesSubplot:xlabel='Ship Mode', ylabel='count'>




<seaborn.axisgrid.FacetGrid at 0x1b993ee8850>




<seaborn.axisgrid.FacetGrid at 0x1b993ee8850>




<seaborn.axisgrid.FacetGrid at 0x1b993ff90a0>

<seaborn.axisgrid.FacetGrid at 0x1b995f465e0>

**RESULT:**

Data Visualization on a complex dataset and save the data to a file has been performed.

# Ex-08-Data-Visualization

### AIM

To Perform Data Visualization on a complex dataset and save the data to a file.

### Explanation

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

### ALGORITHM

**STEP 1:** Read the given Data

**STEP 2:** Clean the Data Set using Data Cleaning Process

**STEP 3:** Apply Feature generation and selection techniques to all the features of the data set

**STEP 4:** Apply data visualization techniques to identify the patterns of the data.

### CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("Superstore.csv")
df

removing unnecessary data variables:
df.drop('Row ID',axis=1,inplace=True)
df.drop('Order ID',axis=1,inplace=True)
df.drop('Customer ID',axis=1,inplace=True)
df.drop('Customer Name',axis=1,inplace=True)
df.drop('Country',axis=1,inplace=True)
df.drop('Postal Code',axis=1,inplace=True)
df.drop('Product ID',axis=1,inplace=True)
df.drop('Product Name',axis=1,inplace=True)
df.drop('Order Date',axis=1,inplace=True)
df.drop('Ship Date',axis=1,inplace=True)
print("Updated dataset")
df

df.isnull().sum()

detecting and removing outliers in current numeric data:
plt.figure(figsize=(12,10))
plt.title("Data with outliers")
df.boxplot()
plt.show()
plt.figure(figsize=(12,10))
```

```
cols = ['Sales','Quantity','Discount','Profit']
Q1 = df[cols].quantile(0.25)
Q3 = df[cols].quantile(0.75)
IQR = Q3 - Q1
df = df[~((df[cols] < (Q1 - 1.5 * IQR)) |(df[cols] > (Q3 + 1.5 * IQR))).any(axis=1)]
plt.title("Dataset after removing outliers")
df.boxplot()
plt.show()
```

data visualization

line plots:

```
import seaborn as sns
sns.lineplot(x="Sub-Category",y="Sales",data=df,marker='o')
plt.title("Sub Categories vs Sales")
plt.xticks(rotation = 90)
plt.show()
sns.lineplot(x="Category",y="Profit",data=df,marker='o')
plt.xticks(rotation = 90)
plt.title("Categories vs Profit")
plt.show()
sns.lineplot(x="Region",y="Sales",data=df,marker='o')
plt.xticks(rotation = 90)
plt.title("Region area vs Sales")
plt.show()
sns.lineplot(x="Category",y="Discount",data=df,marker='o')
plt.title("Categories vs Discount")
plt.show()
sns.lineplot(x="Sub-Category",y="Quantity",data=df,marker='o')
plt.xticks(rotation = 90)
plt.title("Sub Categories vs Quantity")
plt.show()
```

bar plots:

```
sns.barplot(x="Sub-Category",y="Sales",data=df)
plt.title("Sub Categories vs Sales")
plt.xticks(rotation = 90)
plt.show()
sns.barplot(x="Category",y="Profit",data=df)
plt.title("Categories vs Profit")
plt.show()

sns.barplot(x="Sub-Category",y="Quantity",data=df)
plt.title("Sub Categories vs Quantity")
plt.xticks(rotation = 90)
plt.show()
sns.barplot(x="Category",y="Discount",data=df)
plt.title("Categories vs Discount")
plt.show()
plt.figure(figsize=(12,7))
sns.barplot(x="State",y="Sales",data=df)
plt.title("States vs Sales")
plt.xticks(rotation = 90)
plt.show()
plt.figure(figsize=(25,8))
sns.barplot(x="State",y="Sales",hue="Region",data=df)
plt.title("State vs Sales based on Region")
plt.xticks(rotation = 90)
plt.show()
```

Histogram:

```
sns.histplot(data = df,x = 'Region',hue='Ship Mode')
sns.histplot(data = df,x = 'Category',hue='Quantity')
sns.histplot(data = df,x = 'Sub-Category',hue='Category')
plt.xticks(rotation = 90)
plt.show()
sns.histplot(data = df,x = 'Quantity',hue='Segment')
plt.hist(data = df,x = 'Profit')
plt.show()
```

count plot:

```
plt.figure(figsize=(10,7))
sns.countplot(x ='Segment', data = df,hue = 'Sub-Category')
sns.countplot(x ='Region', data = df,hue = 'Segment')
sns.countplot(x ='Category', data = df,hue='Discount')
sns.countplot(x ='Ship Mode', data = df,hue = 'Quantity')
```

Barplot :

```
sns.boxplot(x="Sub-Category",y="Discount",data=df)
plt.xticks(rotation = 90)
plt.show()
sns.boxplot( x="Profit", y="Category",data=df)
plt.xticks(rotation = 90)
plt.show()
plt.figure(figsize=(10,7))
sns.boxplot(x="Sub-Category",y="Sales",data=df)
plt.xticks(rotation = 90)
plt.show()
sns.boxplot(x="Category",y="Profit",data=df)
sns.boxplot(x="Region",y="Sales",data=df)
plt.figure(figsize=(10,7))
sns.boxplot(x="Sub-Category",y="Quantity",data=df)
plt.xticks(rotation = 90)
plt.show()
sns.boxplot(x="Category",y="Discount",data=df)
plt.figure(figsize=(15,7))
sns.boxplot(x="State",y="Sales",data=df)
plt.xticks(rotation = 90)
plt.show()
```

KDE plot:

```
sns.kdeplot(x="Profit", data = df,hue='Category')
sns.kdeplot(x="Sales", data = df,hue='Region')
sns.kdeplot(x="Quantity", data = df,hue='Segment')
sns.kdeplot(x="Discount", data = df,hue='Segment')
```

violin plot:

```
sns.violinplot(x="Profit",data=df)
sns.violinplot(x="Discount",y="Ship Mode",data=df)
sns.violinplot(x="Quantity",y="Ship Mode",data=df)
```

point plot:

```
sns.pointplot(x=df["Quantity"],y=df["Discount"])
sns.pointplot(x=df["Quantity"],y=df["Category"])
sns.pointplot(x=df["Sales"],y=df["Sub-Category"])
```

Pie Chart:

```
df.groupby(['Category']).sum().plot(kind='pie', y='Discount',figsize=(6,10),pctdistance=1.7,labeldistance=1.2)
df.groupby(['Sub-Category']).sum().plot(kind='pie',  y='Sales',figsize=(10,10),pctdistance=1.7,labeldistance=1.2)
```

```
df.groupby(['Region']).sum().plot(kind='pie', y='Profit',figsize=(6,9),pctdistance=1.7,labeldistance=1.2)
df.groupby(['Ship Mode']).sum().plot(kind='pie', y='Quantity',figsize=(8,11),pctdistance=1.7,labeldistance=1.2)
df1=df.groupby(by=["Category"]).sum()
labels=[]
for i in df1.index:
    labels.append(i)
plt.figure(figsize=(8,8))
colors = sns.color_palette('pastel')
plt.pie(df1["Profit"],colors = colors,labels=labels, autopct = '%0.0f%%')
plt.show()
df1=df.groupby(by=["Ship Mode"]).sum()
labels=[]
for i in df1.index:
    labels.append(i)
colors=sns.color_palette("bright")
plt.pie(df1["Sales"],labels=labels,autopct="%0.0f%%")
plt.show()

HeatMap:
df4=df.copy()

encoding:
from sklearn.preprocessing import LabelEncoder,OrdinalEncoder,OneHotEncoder
le=LabelEncoder()
ohe=OneHotEncoder
oe=OrdinalEncoder()

df4["Ship Mode"]=oe.fit_transform(df[["Ship Mode"]])
df4["Segment"]=oe.fit_transform(df[["Segment"]])
df4["City"]=le.fit_transform(df[["City"]])
df4["State"]=le.fit_transform(df[["State"]])
df4['Region'] = oe.fit_transform(df[['Region']])
df4["Category"]=oe.fit_transform(df[["Category"]])
df4["Sub-Category"]=le.fit_transform(df[["Sub-Category"]])

scaling:
from sklearn.preprocessing import RobustScaler
sc=RobustScaler()
df5=pd.DataFrame(sc.fit_transform(df4),columns=['Ship Mode', 'Segment', 'City', 'State','Region',
                                   'Category','Sub-Category','Sales','Quantity','Discount','Profit'])
Heatmap:
plt.subplots(figsize=(12,7))
sns.heatmap(df5.corr(),cmap="PuBu",annot=True)
plt.show()
```

## OUTPUT

**Initial Dataset:**

**Cleaned Dataset:**

Updated dataset

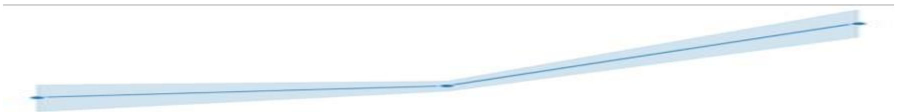
| | Ship Mode | Customer ID | Segment | City | State | Region | Category | Sub-Category | Sales | Quantity | Discount | Profit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Second Class | CG-12520 | Consumer | Henderson | Kentucky | South | Furniture | Bookcases | 261.9600 | 2 | 0.00 | 41.9136 |
| 1 | Second Class | CG-12520 | Consumer | Henderson | Kentucky | South | Furniture | Chairs | 731.9400 | 3 | 0.00 | 219.5820 |
| 2 | Second Class | DV-13045 | Corporate | Los Angeles | California | West | Office Supplies | Labels | 14.6200 | 2 | 0.00 | 6.8714 |
| 3 | Standard Class | SO-20335 | Consumer | Fort Lauderdale | Florida | South | Furniture | Tables | 957.5775 | 5 | 0.45 | -383.0310 |
| 4 | Standard Class | SO-20335 | Consumer | Fort Lauderdale | Florida | South | Office Supplies | Storage | 22.3680 | 2 | 0.20 | 2.5164 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9989 | Second Class | TB-21400 | Consumer | Miami | Florida | South | Furniture | Furnishings | 25.2480 | 3 | 0.20 | 4.1028 |
| 9990 | Standard Class | DB-13060 | Consumer | Costa Mesa | California | West | Furniture | Furnishings | 91.9600 | 2 | 0.00 | 15.6332 |
| 9991 | Standard Class | DB-13060 | Consumer | Costa Mesa | California | West | Technology | Phones | 258.5760 | 2 | 0.20 | 19.3932 |
| 9992 | Standard Class | DB-13060 | Consumer | Costa Mesa | California | West | Office Supplies | Paper | 29.6000 | 4 | 0.00 | 13.3200 |
| 9993 | Second Class | CC-12220 | Consumer | Westminster | California | West | Office Supplies | Appliances | 243.1600 | 2 | 0.00 | 72.9480 |

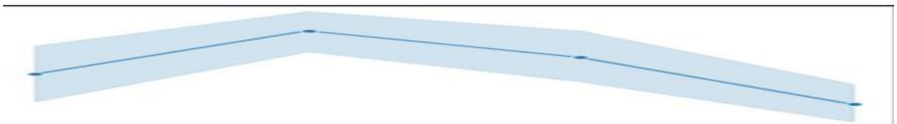
9994 rows × 12 columns

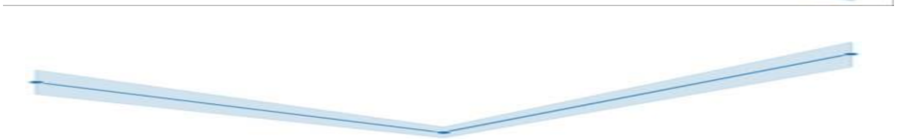
```
Ship Mode        0
Customer ID      0
Segment          0
City             0
State            0
Region           0
Category         0
Sub-Category     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64
```

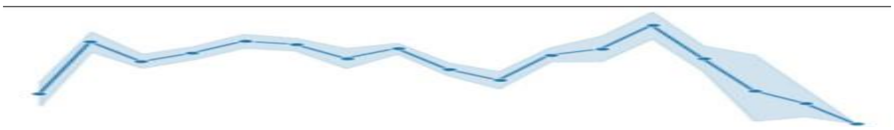
**Removing Outliers:**

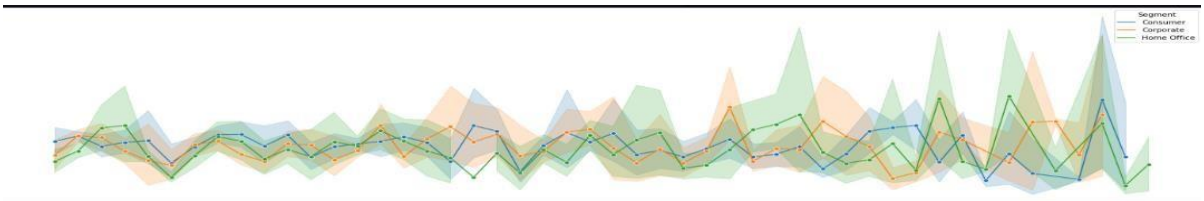
**Line PLot:**



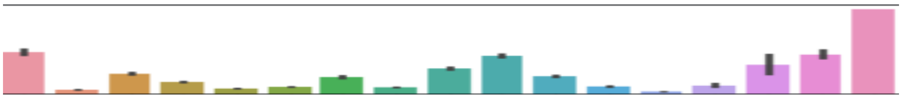


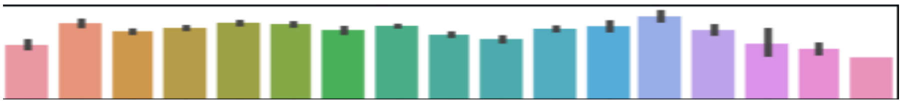


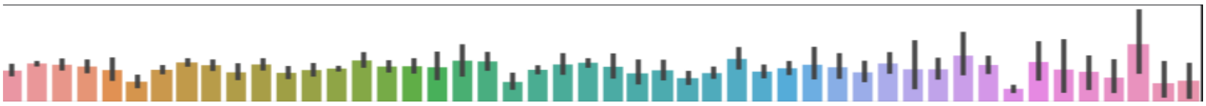


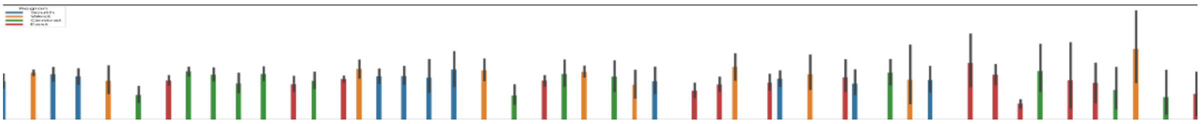



**Bar Plots:**

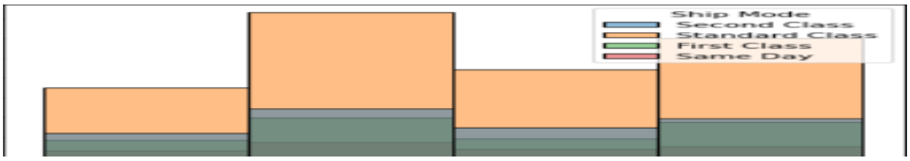


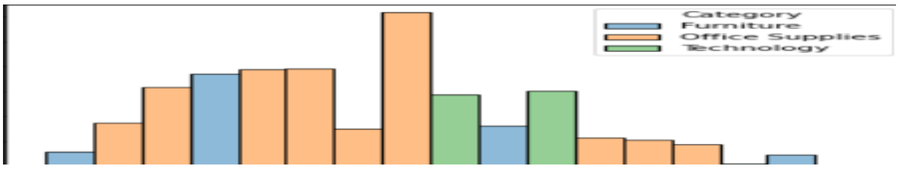


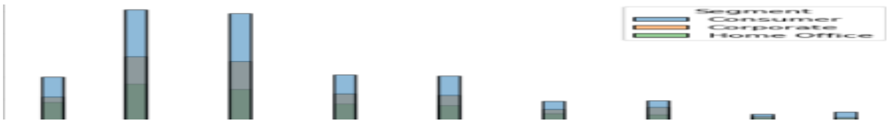




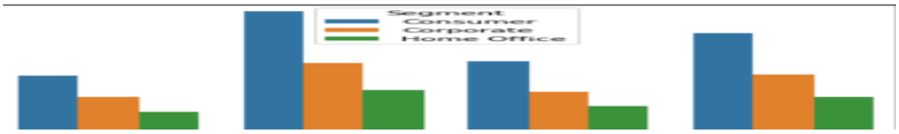
**Histograms:**

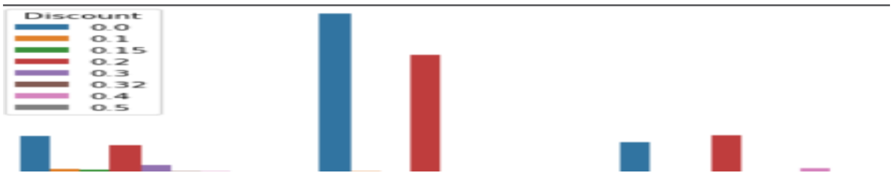



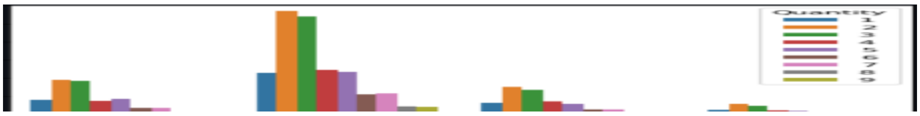



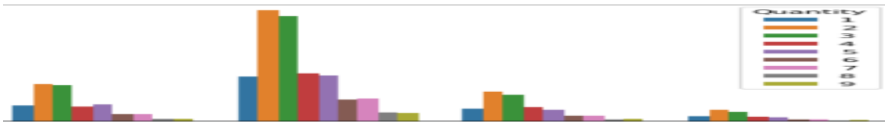




**Count plots:**

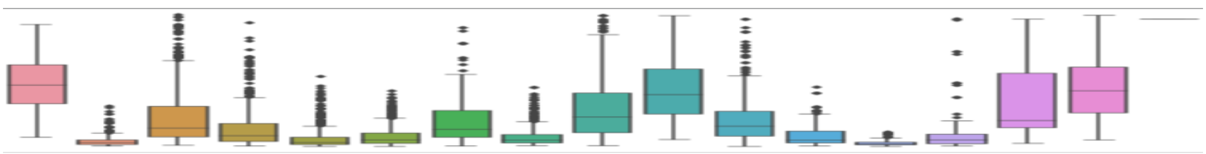


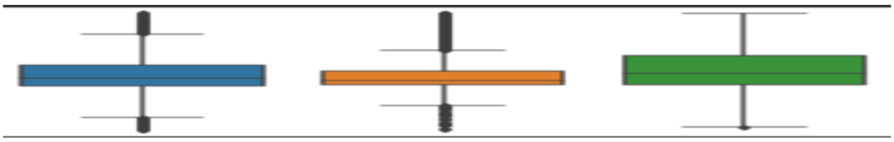


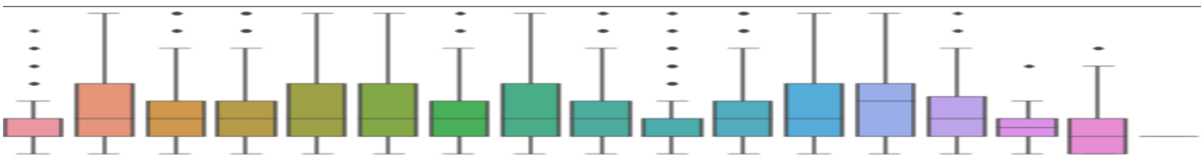



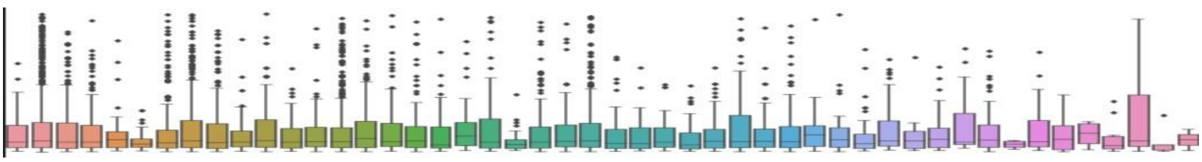



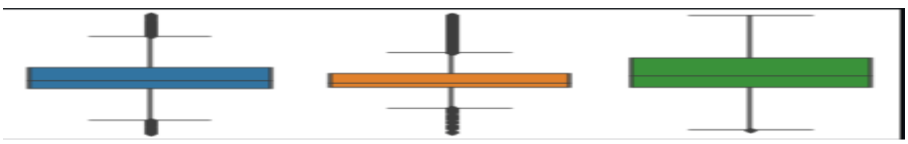
**Bar Charts:**

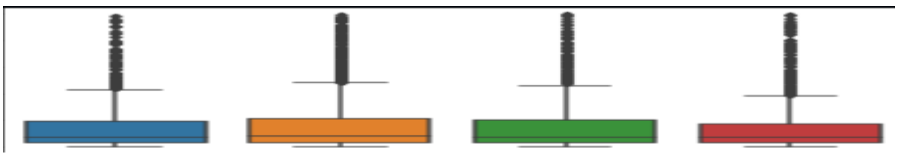


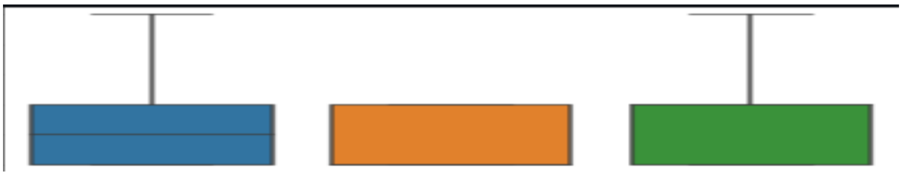




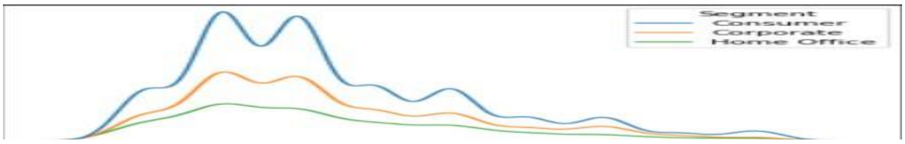


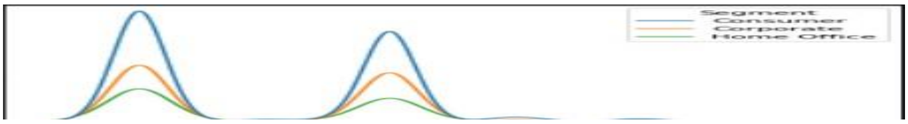




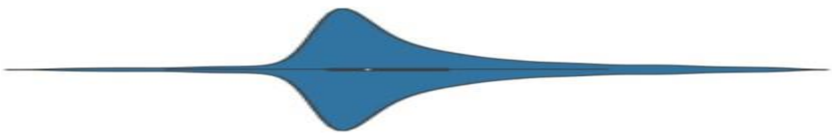


## KDE Plots:

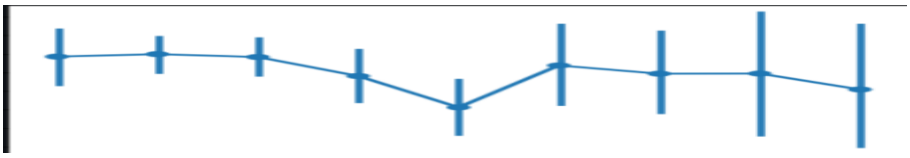


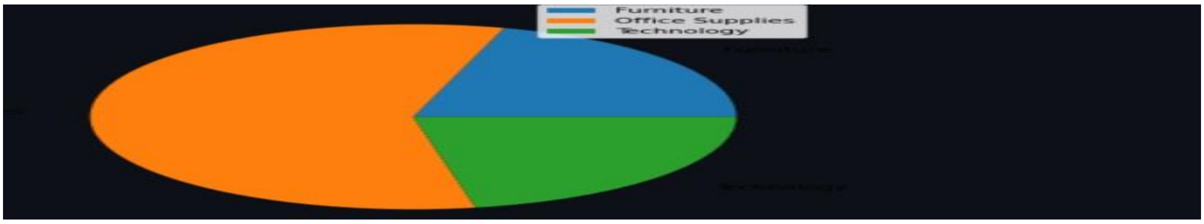

### Violin Plot:

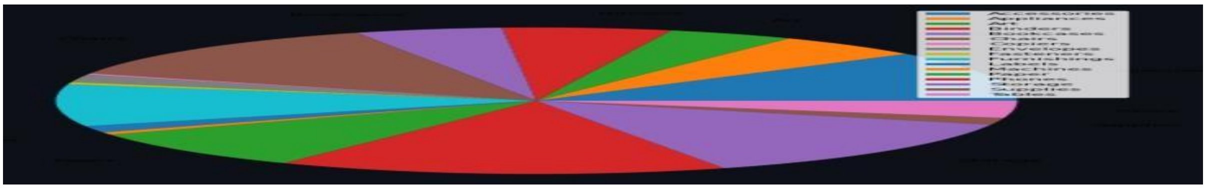


### Point Plots:

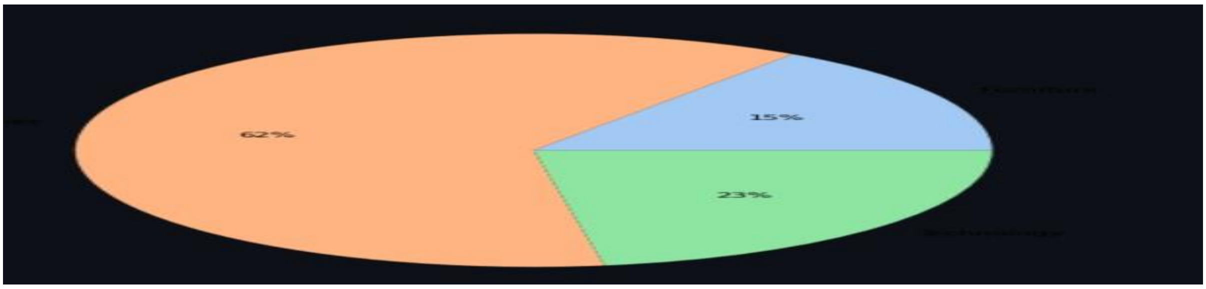


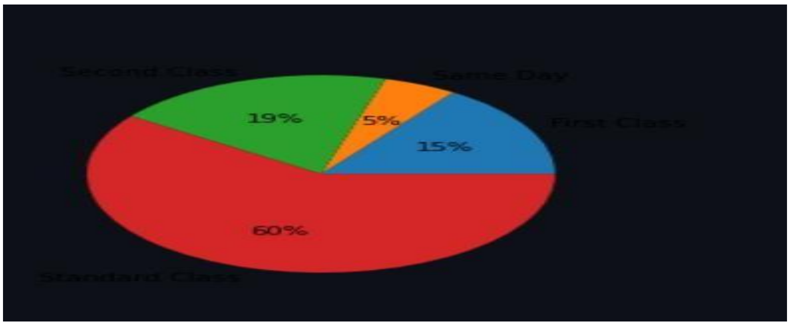
## Pie Charts:



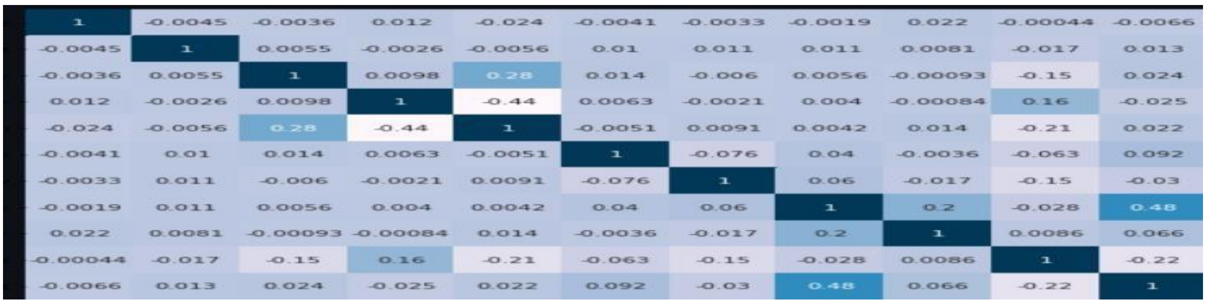








## HeatMap:




**Result:**

Hence,Data Visualization is applied on the complex dataset using libraries like Seaborn and Matplotlib successfully and the data is saved to file

# MINI PROJECT

# Mini-Project

## Aim:

›

To implement data science techniques in weight-height dataset

›

## Methodologies:

**1.Importing Libraries**

›

**2.Loading Data**

›

**3.Performing Simple EDA**

›

›

**4.Feature Engineering & Selection**

›

**5.Model Visulization**

›

## Program:

```
import numpy as np import pandas
as pdimport io
from scipy import stats

from google.colab import filesuploaded =
files.upload()

df = pd.read_csv(io.BytesIO(uploaded['weight-height.csv']))print(df)

df.head() df.isnull().sum()
df.drop("Gender",axis=1,inplace=True)
df.head() df1=df.copy()
df.boxplot() df.shape

z=np.abs(stats.zscore(df))z

df1=df1[(z<3).all(axis=1)]df1

df2=df.copy()
```

df2.head() q1=df2.quantile(0.25)
q3=df2.quantile(0.75)q1
q3
iqr=q3-q1iqr
df2_new=df2[((df2>=(q1-1.5*iqr))&(df2<=(q3+1.5*iqr))).all(axis=1)] df2_new.shape
df2_new.boxplot()
df2

# Output:

```
import numpy as np
import pandas as pd
import io
from scipy import stats
```

```
[ ] from google.colab import files
    uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving weight-height.csv to weight-height.csv
```

```
[ ] df = pd.read_csv(io.BytesIO(uploaded['weight-height.csv']))
    print(df)
```

```
      Gender     Height      Weight
0       Male  73.847017  241.893563
1       Male  68.781904  162.310473
2       Male  74.110105  212.740856
3       Male  71.730978  220.042470
4       Male  69.881796  206.349801
...      ...        ...         ...
9995  Female  66.172652  136.777454
9996  Female  67.067155  170.867906
9997  Female  63.867992  128.475319
9998  Female  69.034243  163.852461
9999  Female  61.944246  113.649103

[10000 rows x 3 columns]
```

```
[ ] df.head()
```

| | Gender | Height | Weight |
|---|---|---|---|
| 0 | Male | 73.847017 | 241.893563 |
| 1 | Male | 68.781904 | 162.310473 |
| 2 | Male | 74.110105 | 212.740856 |
| 3 | Male | 71.730978 | 220.042470 |
| 4 | Male | 69.881796 | 206.349801 |

```
[ ] df.isnull().sum()
```

```
Gender    0
Height    0
Weight    0
dtype: int64
```

```
[ ] df.drop("Gender",axis=1,inplace=True)
```

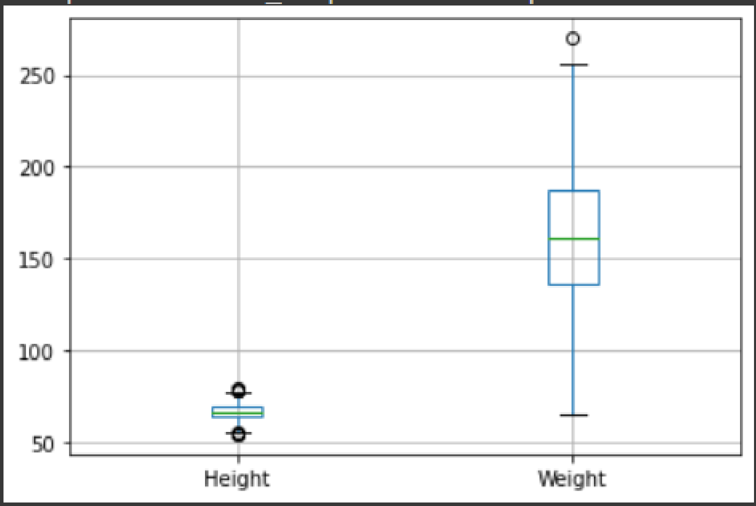
```
[ ] df.head()
```

| | Height | Weight |
|---|---|---|
| 0 | 73.847017 | 241.893563 |
| 1 | 68.781904 | 162.310473 |
| 2 | 74.110105 | 212.740856 |
| 3 | 71.730978 | 220.042470 |
| 4 | 69.881796 | 206.349801 |

```
[ ] df1=df.copy()
```

```
[ ] df.boxplot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f939c474a50>



```
[ ] df.shape
```

```
(10000, 2)
```

```
[ ] z=np.abs(stats.zscore(df))
    z
```

| | Height | Weight |
|---|---|---|
| **0** | 1.944061 | 2.505797 |
| **1** | 0.627537 | 0.027101 |
| **2** | 2.012443 | 1.597806 |
| **3** | 1.394060 | 1.825222 |
| **4** | 0.913421 | 1.398750 |
| **...** | ... | ... |
| **9995** | 0.050660 | 0.768151 |
| **9996** | 0.181839 | 0.293631 |
| **9997** | 0.649688 | 1.026730 |
| **9998** | 0.693125 | 0.075127 |
| **9999** | 1.149708 | 1.488507 |

10000 rows × 2 columns

```
[ ] df1=df1[(z<3).all(axis=1)]
    df1
```

| | Height | Weight |
|---|---|---|
| **0** | 73.847017 | 241.893563 |
| **1** | 68.781904 | 162.310473 |
| **2** | 74.110105 | 212.740856 |
| **3** | 71.730978 | 220.042470 |
| **4** | 69.881796 | 206.349801 |
| **...** | ... | ... |
| **9995** | 66.172652 | 136.777454 |
| **9996** | 67.067155 | 170.867906 |
| **9997** | 63.867992 | 128.475319 |
| **9998** | 69.034243 | 163.852461 |
| **9999** | 61.944246 | 113.649103 |

9993 rows × 2 columns

```
[ ] df2=df.copy()
```

```
[ ] df2.head()
```

| | Height | Weight |
|---|---|---|
| 0 | 73.847017 | 241.893563 |
| 1 | 68.781904 | 162.310473 |
| 2 | 74.110105 | 212.740856 |
| 3 | 71.730978 | 220.042470 |
| 4 | 69.881796 | 206.349801 |

```
[ ] q1=df2.quantile(0.25)
```

```
[ ] q3=df2.quantile(0.75)
```

```
[ ] q1
```

```
Height     63.505620
Weight    135.818051
Name: 0.25, dtype: float64
```

```
[ ] q3
```

```
Height     69.174262
Weight    187.169525
Name: 0.75, dtype: float64
```

```
[ ] iqr=q3-q1
```

```
iqr
```

```
Height     5.668641
Weight    51.351474
dtype: float64
```

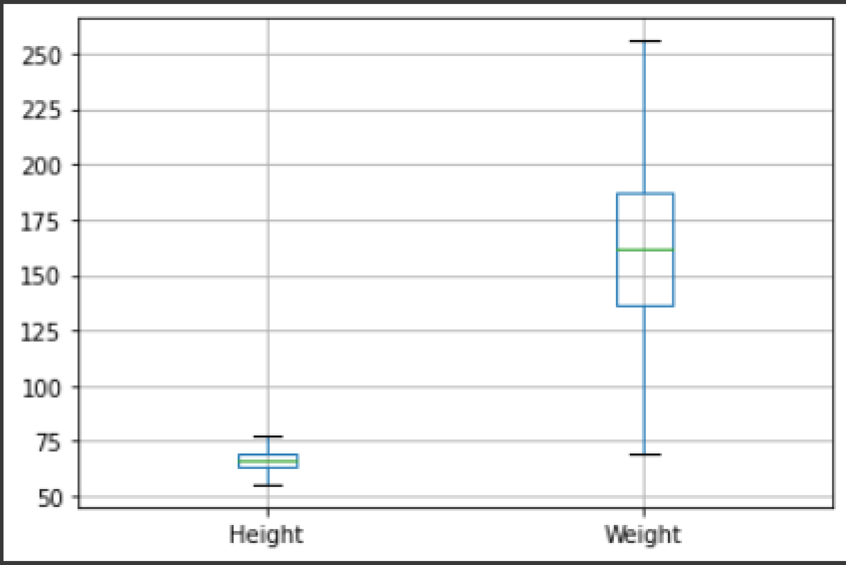
```
[ ] df2_new=df2[((df2>=(q1-1.5*iqr))&(df2<=(q3+1.5*iqr))).all(axis=1)]
```

```
[ ] df2_new.shape
```

```
(9992, 2)
```

```
[ ] df2_new.boxplot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f939c3be750>
```



```
[ ] df2
```

| | Height | Weight |
|---|---|---|
| 0 | 73.847017 | 241.893563 |
| 1 | 68.781904 | 162.310473 |
| 2 | 74.110105 | 212.740856 |
| 3 | 71.730978 | 220.042470 |
| 4 | 69.881796 | 206.349801 |
| ... | ... | ... |
| 9995 | 66.172652 | 136.777454 |
| 9996 | 67.067155 | 170.867906 |
| 9997 | 63.867992 | 128.475319 |
| 9998 | 69.034243 | 163.852461 |
| 9999 | 61.944246 | 113.649103 |

10000 rows × 2 columns

## Result:

Thus above program is required for weight-height was analyzed successfully