Project: Milestone 2

Import Modules

In [1]:
```
# import pandas as pd
import matplotlib.pyplot as plt
import pandas as pd
```

Load data from .csv files

In [2]:
```
# Import data from CSV files

acData = pd.read_csv('F:/3Data/archive/Air Conditioners.csv')
appliancesData = pd.read_csv('F:/3Data/archive/All Appliances.csv')
booksData = pd.read_csv('F:/3Data/archive/All Books.csv')
electronicsData = pd.read_csv('F:/3Data/archive/All Electronics.csv')
clothingData = pd.read_csv('F:/3Data/archive/Clothing.csv')
jeansData = pd.read_csv('F:/3Data/archive/Jeans.csv')
walletsData = pd.read_csv('F:/3Data/archive/Wallets.csv')
sportsWearData = pd.read_csv('F:/3Data/archive/Sportswear.csv')
speakersData = pd.read_csv('F:/3Data/archive/Speakers.csv')
sportsShoesData = pd.read_csv('F:/3Data/archive/Sports Shoes.csv')
jewelleryData = pd.read_csv('F:/3Data/archive/Jewellery.csv')

listOfdf = [acData, appliancesData, booksData, electronicsData, clothingData, jeansDat
```

Check NULL values in all files/dataframes

Here we are checking missing values in all files/dataframes

In [3]:
```
listOfdf = [acData, appliancesData, booksData, electronicsData, clothingData, jeansDat

for i in listOfdf:
    # Check for null values in AC Data
    null_values = i.isnull().sum()
    print("Null Values in selected dataset:\n", null_values)
```

```
Null Values in selected dataset:
 name                0
main_category        0
sub_category         0
image                0
link                 0
ratings            287
no_of_ratings      287
discount_price     263
actual_price       220
dtype: int64
Null Values in selected dataset:
 name                0
main_category        0
sub_category         0
image                0
link                 0
ratings            478
no_of_ratings      478
discount_price     362
actual_price        91
dtype: int64
Null Values in selected dataset:
 name               0.0
main_category      0.0
sub_category       0.0
image              0.0
link               0.0
ratings            0.0
no_of_ratings      0.0
discount_price     0.0
actual_price       0.0
dtype: float64
Null Values in selected dataset:
 name                0
main_category        0
sub_category         0
image                0
link                 0
ratings             95
no_of_ratings       95
discount_price     484
actual_price        70
dtype: int64
Null Values in selected dataset:
 name                 0
main_category         0
sub_category          0
image                 0
link                  0
ratings            1427
no_of_ratings      1427
discount_price      849
actual_price        119
dtype: int64
Null Values in selected dataset:
 name                  0
main_category          0
sub_category           0
image                  0
```

```
link                0
ratings          13280
no_of_ratings    13280
discount_price    3543
actual_price      1418
dtype: int64
Null Values in selected dataset:
 name               0
main_category      0
sub_category       0
image              0
link               0
ratings          963
no_of_ratings    963
discount_price   317
actual_price      41
dtype: int64
Null Values in selected dataset:
 name               0
main_category      0
sub_category       0
image              0
link               0
ratings          933
no_of_ratings    933
discount_price   4587
actual_price      256
dtype: int64
Null Values in selected dataset:
 name               0
main_category      0
sub_category       0
image              0
link               0
ratings          5613
no_of_ratings    5613
discount_price    548
actual_price      302
dtype: int64
Null Values in selected dataset:
 name               0
main_category      0
sub_category       0
image              0
link               0
ratings          8191
no_of_ratings    8191
discount_price   3296
actual_price      718
dtype: int64
Null Values in selected dataset:
 name               0
main_category      0
sub_category       0
image              0
link               0
ratings          7224
no_of_ratings    7224
discount_price    644
```

```
actual_price       325
dtype: int64
```

Dataset cleaning

As this dataset has two extra columns i.e image and link we are going to remove it.

In [4]:
```
# Remove two columns from all dataframes
columns_to_remove = ['image', 'link']

acData = acData.drop(columns=columns_to_remove)
appliancesData = appliancesData.drop(columns=columns_to_remove)
booksData = booksData.drop(columns=columns_to_remove)
electronicsData = electronicsData.drop(columns=columns_to_remove)
clothingData = clothingData.drop(columns=columns_to_remove)
jeansData = jeansData.drop(columns=columns_to_remove)
walletsData = walletsData.drop(columns=columns_to_remove)
sportsWearData = sportsWearData.drop(columns=columns_to_remove)
speakersData = speakersData.drop(columns=columns_to_remove)
sportsShoesData = sportsShoesData.drop(columns=columns_to_remove)
jewelleryData = jewelleryData.drop(columns=columns_to_remove)

listOfdf = [acData, appliancesData, booksData, electronicsData, clothingData, jeansDat
```

In [5]:
```
#check dataframe
acData.describe()
```

Out[5]:

	name	main_category	sub_category	ratings	no_of_ratings	discount_price	actual_price
count	720	720	720	433	433	457	500
unique	708	1	1	31	187	295	296
top	Samsung 1.5 Ton 3 Star Wind-Free Technology In...	appliances	Air Conditioners	4	1	$39,990	$59,990
freq	3	720	720	56	45	11	15

Data Duplication handling

check dupliacation and remove duplication

In [6]:
```
listOfdf = [acData, appliancesData, booksData, electronicsData, clothingData, jeansDat
updateDFlist = []

for selectedDf in listOfdf:
    # Check for data duplication
    duplicated_rows = selectedDf.duplicated()
    print("Duplicated Rows:\n", duplicated_rows)
```

```
    # Remove duplicated rows
    selectedDf = selectedDf.drop_duplicates()
    selectedDf.describe()
    updateDFlist.append(selectedDf)

#test
updateDFlist[0].describe()
```

```
Duplicated Rows:
 0      False
1      False
2      False
3      False
4      False
       ...
715    False
716    False
717    False
718    False
719    False
Length: 720, dtype: bool
Duplicated Rows:
 0       False
1       False
2       False
3       False
4       False
        ...
9571    False
9572    False
9573    False
9574    False
9575    False
Length: 9576, dtype: bool
Duplicated Rows:
 Series([], dtype: bool)
Duplicated Rows:
 0       False
1       False
2       False
3       False
4       False
        ...
9595    False
9596    False
9597    False
9598    False
9599    False
Length: 9600, dtype: bool
Duplicated Rows:
 0        False
1        False
2        False
3        False
4        False
         ...
19147    False
19148    False
19149    False
19150    False
19151    False
Length: 19152, dtype: bool
Duplicated Rows:
 0        False
1        False
2        False
3        False
4        False
```

```
         ...
19195    False
19196     True
19197     True
19198    False
19199     True
Length: 19200, dtype: bool
Duplicated Rows:
 0       False
1       False
2       False
3       False
4       False
        ...
2011    False
2012    False
2013    False
2014    False
2015    False
Length: 2016, dtype: bool
Duplicated Rows:
 0       False
1       False
2       False
3       False
4       False
        ...
7366    False
7367    False
7368    False
7369    False
7370    False
Length: 7371, dtype: bool
Duplicated Rows:
 0       False
1       False
2       False
3       False
4       False
        ...
9595    False
9596    False
9597    False
9598    False
9599    False
Length: 9600, dtype: bool
Duplicated Rows:
 0        False
1        False
2        False
3        False
4        False
         ...
19195    False
19196    False
19197    False
19198    False
19199    False
Length: 19200, dtype: bool
Duplicated Rows:
```

```
 0        False
1        False
2        False
3        False
4        False
         ...
19147    False
19148    False
19149    False
19150    False
19151    False
Length: 19152, dtype: bool
```

Out[6]:

	name	main_category	sub_category	ratings	no_of_ratings	discount_price	actual_price
count	718	718	718	433	433	457	500
unique	708	1	1	31	187	295	296
top	Samsung 1.5 Ton 3 Star Wind-Free Technology In...	appliances	Air Conditioners	4	1	$39,990	$59,990
freq	3	718	718	56	45	11	15

Handle Missing and Null values

We are going to handle missing and null values by replacing mean/average value id data type in "int" or "float" and replacing mode value (most repeated) if data type is non numeric or other

for selectedDf in updateDFlist: print('test 1') for column in selectedDf.columns: print('test 2') if selectedDf[column].dtype in [float, int]: print('test 3') selectedDf[column].fillna(selectedDf[column].mean(), inplace=True) print('test 4') updateDFlist.append(selectedDf)

In [ ]:
```
# Handle null and missing values for all DataFrames

for selectedDf in updateDFlist:
    for column in selectedDf.columns:
        if selectedDf[column].dtype in [float, int]:
            selectedDf[column].fillna(selectedDf[column].mean(), inplace=True)
        else:
            selected_df[column].fillna(selected_df[column].mode()[0], inplace=True)

    updateDFlist.append(selectedDf)
```

Outliers Handling

To handle outliers in data, there are several approaches you can consider. Here are three common techniques:

Removing outliers:

You can choose to remove the outliers from your dataset. This approach is suitable when the outliers are deemed to be erroneous data points or if they significantly affect the analysis. One common method to detect outliers is using the z-score. Any data point that falls outside a certain threshold (e.g., z-score greater than 3) can be considered an outlier and removed from the dataset.

Capping outliers:

Instead of removing outliers, you can cap their values to a certain range. This approach retains the outliers in the dataset but reduces their impact on the analysis. For example, you can set a lower and upper limit and replace any outliers below the lower limit with the lower limit value and any outliers above the upper limit with the upper limit value.

Transformation:

Outliers can sometimes be addressed by applying mathematical transformations to the data. Transformations like logarithmic, square root, or Box-Cox transformations can help normalize the distribution and reduce the impact of outliers.

In [ ]:
```
# We are using capping outliers

newupdatdeDFlist = []
for data in updateDFlist:

    # Define the lower and upper limits for capping outliers
    lower_limit = 0  # Specify the lower limit
    upper_limit = 100 * np.std(data[['ratings']])  # Specify the upper limit

    # Cap outliers below the lower limit
    data[['ratings']][data[['ratings']] < lower_limit] = lower_limit

    # Cap outliers above the upper limit
    data[['ratings']][data[['ratings']] > upper_limit] = upper_limit


    newupdatdeDFlist.append(data)
```