Exploratory Data Analysis (EDA) on Retail Sales Data

Level 1 , Task 1

1. Data Loading and Cleaning

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.cluster import KMeans
```

```
df = pd.read_csv('retail_sales_dataset.csv')
print(df.head())
```

```
   Transaction ID        Date Customer ID  Gender  Age Product Category  \
0               1  2023-11-24     CUST001    Male   34           Beauty
1               2  2023-02-27     CUST002  Female   26         Clothing
2               3  2023-01-13     CUST003    Male   50      Electronics
3               4  2023-05-21     CUST004    Male   37         Clothing
4               5  2023-05-06     CUST005    Male   30           Beauty

   Quantity  Price per Unit  Total Amount
0         3              50           150
1         2             500          1000
2         1              30            30
3         1             500           500
4         2              50           100
```

```
df = pd.read_csv('retail_sales_dataset.csv')
print(df.head(20))
```

```
    Transaction ID        Date Customer ID  Gender  Age Product Category  \
0                1  2023-11-24     CUST001    Male   34           Beauty
1                2  2023-02-27     CUST002  Female   26         Clothing
2                3  2023-01-13     CUST003    Male   50      Electronics
3                4  2023-05-21     CUST004    Male   37         Clothing
4                5  2023-05-06     CUST005    Male   30           Beauty
5                6  2023-04-25     CUST006  Female   45           Beauty
6                7  2023-03-13     CUST007    Male   46         Clothing
7                8  2023-02-22     CUST008    Male   30      Electronics
8                9  2023-12-13     CUST009    Male   63      Electronics
9               10  2023-10-07     CUST010  Female   52         Clothing
10              11  2023-02-14     CUST011    Male   23         Clothing
11              12  2023-10-30     CUST012    Male   35           Beauty
12              13  2023-08-05     CUST013    Male   22      Electronics
13              14  2023-01-17     CUST014    Male   64         Clothing
14              15  2023-01-16     CUST015  Female   42      Electronics
15              16  2023-02-17     CUST016    Male   19         Clothing
16              17  2023-04-22     CUST017  Female   27         Clothing
17              18  2023-04-30     CUST018  Female   47      Electronics
18              19  2023-09-16     CUST019  Female   62         Clothing
19              20  2023-11-05     CUST020    Male   22         Clothing
```

```
     Quantity  Price per Unit  Total Amount
0           3              50           150
1           2             500          1000
2           1              30            30
3           1             500           500
4           2              50           100
5           1              30            30
6           2              25            50
7           4              25           100
8           2             300           600
9           4              50           200
10          2              50           100
11          3              25            75
12          3             500          1500
13          4              30           120
14          4             500          2000
15          3             500          1500
16          4              25           100
17          2              25            50
18          2              25            50
19          3             300           900
```

```
df = pd.read_csv('retail_sales_dataset.csv')
print(df.tail(10))
```

```
     Transaction ID        Date Customer ID  Gender  Age Product Category  \
990             991  2023-12-26     CUST991  Female   34         Clothing
991             992  2023-08-21     CUST992  Female   57      Electronics
992             993  2023-02-06     CUST993  Female   48      Electronics
993             994  2023-12-18     CUST994  Female   51           Beauty
994             995  2023-04-30     CUST995  Female   41         Clothing
995             996  2023-05-16     CUST996    Male   62         Clothing
996             997  2023-11-17     CUST997    Male   52           Beauty
997             998  2023-10-29     CUST998  Female   23           Beauty
998             999  2023-12-05     CUST999  Female   36      Electronics
999            1000  2023-04-12    CUST1000    Male   47      Electronics

     Quantity  Price per Unit  Total Amount
990         2              50           100
991         2              30            60
992         3              50           150
993         2             500          1000
994         1              30            30
995         1              50            50
996         3              30            90
997         4              25           100
998         3              50           150
999         4              30           120
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Transaction ID    1000 non-null   int64
 1   Date              1000 non-null   object
 2   Customer ID       1000 non-null   object
```

```
 3   Gender            1000 non-null   object
 4   Age               1000 non-null   int64
 5   Product Category  1000 non-null   object
 6   Quantity          1000 non-null   int64
 7   Price per Unit    1000 non-null   int64
 8   Total Amount      1000 non-null   int64
dtypes: int64(5), object(4)
memory usage: 70.4+ KB
None
```

```
# Load the dataset
df = pd.read_csv('retail_sales_dataset.csv')

# Count rows and columns
rows, columns = df.shape

print(f"Number of rows: {rows}")
print(f"Number of columns: {columns}")
```

```
Number of rows: 1000
Number of columns: 9
```

```
# Load the dataset
df = pd.read_csv('retail_sales_dataset.csv')

# Get the column names
column_names = df.columns

# Print the column names
print(column_names)
```

```
Index(['Transaction ID', 'Date', 'Customer ID', 'Gender', 'Age',
       'Product Category', 'Quantity', 'Price per Unit', 'Total Amount'],
      dtype='object')
```

```
# null values
# Load the dataset
retail_data = pd.read_csv('retail_sales_dataset.csv')

# Check for missing values
missing_values = retail_data.isnull().sum()

print(missing_values)
```

```
Transaction ID      0
Date                0
Customer ID         0
Gender              0
Age                 0
Product Category    0
Quantity            0
Price per Unit      0
Total Amount        0
dtype: int64
```

```
# Check for duplicate rows
duplicates = retail_data.duplicated()

# Number of duplicate rows
num_duplicates = duplicates.sum()

print(f"Number of duplicate rows: {num_duplicates}")
```

```
Number of duplicate rows: 0
```

There are no missing or null values in the dataset

```
# Data Cleaning: Convert the 'Date' column to DateTime format
retail_data['Date'] = pd.to_datetime(retail_data['Date'])
retail_data.head(5)
```

| | Transaction ID | Date | Customer ID | Gender | Age | Product Category | Quantity | Price per Unit | Total Amount |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2023-11-24 | CUST001 | Male | 34 | Beauty | 3 | 50 | 150 |
| **1** | 2 | 2023-02-27 | CUST002 | Female | 26 | Clothing | 2 | 500 | 1000 |
| **2** | 3 | 2023-01-13 | CUST003 | Male | 50 | Electronics | 1 | 30 | 30 |

Next steps: Generate code with retail_data | View recommended plots | New interactive sheet

2. Descriptive Statistics

```
des_stat=retail_data.describe()
des_stat
```

| | Transaction ID | Date | Age | Quantity | Price per Unit | Total Amount |
|---|---|---|---|---|---|---|
| **count** | 1000.000000 | 1000 | 1000.00000 | 1000.000000 | 1000.000000 | 1000.00000 |
| **mean** | 500.500000 | 2023-07-03 00:25:55.200000256 | 41.39200 | 2.514000 | 179.890000 | 456.00000 |
| **min** | 1.000000 | 2023-01-01 00:00:00 | 18.00000 | 1.000000 | 25.000000 | 25.00000 |
| **25%** | 250.750000 | 2023-04-08 00:00:00 | 29.00000 | 1.000000 | 30.000000 | 60.00000 |
| **50%** | 500.500000 | 2023-06-29 12:00:00 | 42.00000 | 3.000000 | 50.000000 | 135.00000 |

Next steps: Generate code with des_stat | View recommended plots | New interactive sheet

```
# AGE
AGE_stat = retail_data['Age'].describe()
AGE_stat
```

| | Age |
|---|---|
| **count** | 1000.00000 |
| **mean** | 41.39200 |
| **std** | 13.68143 |
| **min** | 18.00000 |
| **25%** | 29.00000 |
| **50%** | 42.00000 |
| **75%** | 53.00000 |
| **max** | 64.00000 |

**dtype:** float64

```
Quality_stat=retail_data['Quantity'].describe()
Quality_stat
```

| | Quantity |
|---|---|
| **count** | 1000.000000 |
| **mean** | 2.514000 |
| **std** | 1.132734 |
| **min** | 1.000000 |
| **25%** | 1.000000 |
| **50%** | 3.000000 |
| **75%** | 4.000000 |
| **max** | 4.000000 |

**dtype:** float64

```
Total_price_stat=retail_data['Total Amount'].describe()
Total_price_stat
```

| | Total Amount |
|---|---|
| **count** | 1000.000000 |
| **mean** | 456.000000 |
| **std** | 559.997632 |
| **min** | 25.000000 |
| **25%** | 60.000000 |
| **50%** | 135.000000 |
| **75%** | 900.000000 |
| **max** | 2000.000000 |

**dtype:** float64

3. Time Series Analysis

```
# Add a 'Month' column for monthly sales analysis
retail_data['Month'] = retail_data['Date'].dt.to_period('M')
retail_data.head()
```

| | Transaction ID | Date | Customer ID | Gender | Age | Product Category | Quantity | Price per Unit | Total Amount | Month |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2023-11-24 | CUST001 | Male | 34 | Beauty | 3 | 50 | 150 | 2023-11 |
| **1** | 2 | 2023-02-27 | CUST002 | Female | 26 | Clothing | 2 | 500 | 1000 | 2023-02 |
| | | 2023- | | | | | | | | 2023- |

Next steps: Generate code with `retail_data` View recommended plots New interactive sheet

```
#### analyze sales trends over time. Group by month and calculate total sales
monthly_sales = retail_data.groupby('Month')['Total Amount'].sum()
monthly_sales
```

| Month | Total Amount |
|---|---|
| **2023-01** | 35450 |
| **2023-02** | 44060 |
| **2023-03** | 28990 |
| **2023-04** | 33870 |
| **2023-05** | 53150 |
| **2023-06** | 36715 |
| **2023-07** | 35465 |
| **2023-08** | 36960 |
| **2023-09** | 23620 |
| **2023-10** | 46580 |
| **2023-11** | 34920 |
| **2023-12** | 44690 |
| **2024-01** | 1530 |

**dtype:** int64

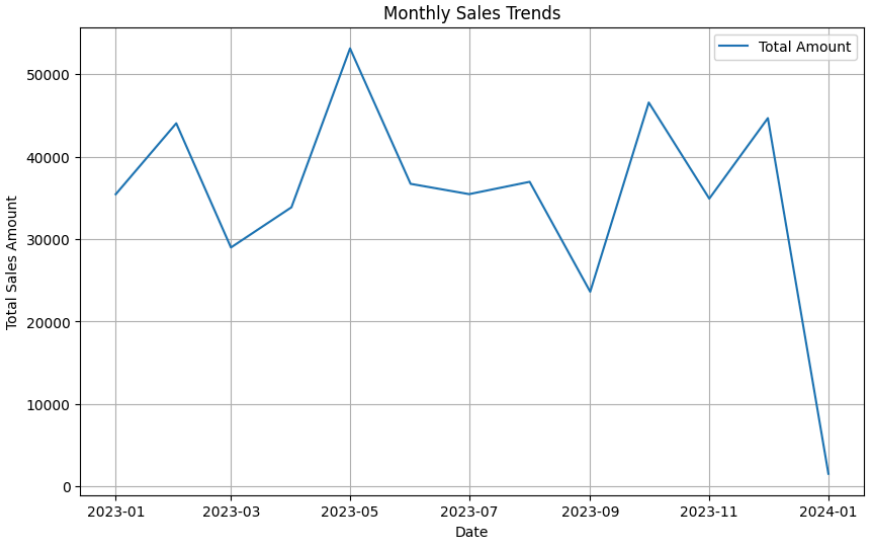
```
# Convert the Series to a DataFrame
monthly_sales_df = monthly_sales.reset_index()
monthly_sales_df.columns = ['Date', 'Total Amount']  # Rename the columns
print(monthly_sales_df)
```

```
        Date  Total Amount
0   2023-01         35450
1   2023-02         44060
2   2023-03         28990
3   2023-04         33870
4   2023-05         53150
5   2023-06         36715
6   2023-07         35465
7   2023-08         36960
8   2023-09         23620
9   2023-10         46580
10  2023-11         34920
11  2023-12         44690
12  2024-01          1530
```

```
# Convert the 'Month' column to datetime format
monthly_sales_df['Date'] = monthly_sales_df['Date'].dt.to_timestamp()

# Plot time series of sales
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales_df['Date'], monthly_sales_df['Total Amount'], label='Total Amount')
plt.title('Monthly Sales Trends')
plt.xlabel('Date')
plt.xlabel('Total Sales Amount')
```

```
plt.legend()
plt.grid(True)
plt.show()
```




4. Customer and Product Analysis

5. Visulaizations

```
# Gender distribution
gender_counts = retail_data['Gender'].value_counts()
gender_sales = retail_data.groupby('Gender')['Total Amount'].sum()
print(gender_sales)
print(gender_counts)
```

```
Gender
Female    232840
Male      223160
Name: Total Amount, dtype: int64
Gender
Female    510
Male      490
Name: count, dtype: int64
```

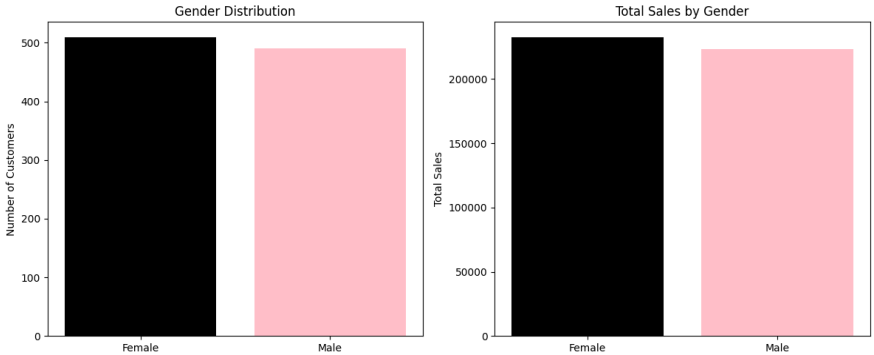
```
# Plot gender distribution
```

```
fig, ax1 = plt.subplots(1, 2, figsize=(12, 5))

ax1[0].bar(gender_counts.index, gender_counts, color=['black', 'pink'])
ax1[0].set_title('Gender Distribution')
ax1[0].set_ylabel('Number of Customers')

ax1[1].bar(gender_sales.index, gender_sales, color=['black', 'pink'])
ax1[1].set_title('Total Sales by Gender')
ax1[1].set_ylabel('Total Sales')

plt.tight_layout()
plt.show()
```




```
## Age distribution and spending by age group
age_distribution = retail_data['Age'].value_counts().sort_index()
age_sales = retail_data.groupby('Age')['Total Amount'].sum()
print(age_distribution)
print(age_sales)
```

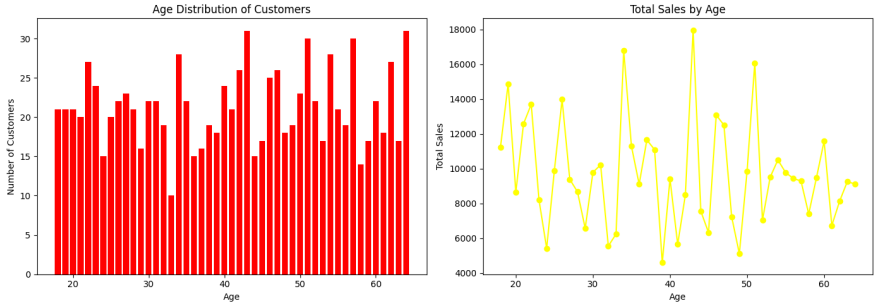
```
19    14870
20     8645
21    12585
22    13700
23     8220
24     5415
25     9900
26    13980
27     9385
28     8670
29     6570
30     9790
31    10220
32     5550
33     6240
34    16785
35    11290
36     9105
37    11650
38    11100
39     4595
40     9415
41     5650
42     8500
43    17970
44     7560
45     6325
46    13090
47    12505
48     7240
49     5110
50     9845
51    16065
52     7040
53     9510
54    10505
55     9780
56     9440
57     9290
58     7395
59     9470
60    11590
61     6730
62     8120
63     9250
64     9125
Name: Total Amount, dtype: int64
```

```
fig, ax2 = plt.subplots(1, 2, figsize=(14, 5))

# Plot age distribution
ax2[0].bar(age_distribution.index, age_distribution, color='red')
ax2[0].set_title('Age Distribution of Customers')
ax2[0].set_xlabel('Age')
ax2[0].set_ylabel('Number of Customers')

# Plot total sales by age
ax2[1].plot(age_sales.index, age_sales, marker='o', color='yellow')
ax2[1].set_title('Total Sales by Age')
ax2[1].set_xlabel('Age')
ax2[1].set_ylabel('Total Sales')
```

```
plt.tight_layout()
plt.show()
```




```
## 2. Product Preferences Analysis
# Total sales by product category
category_sales = retail_data.groupby('Product Category')['Total Amount'].sum()
average_sales_category = retail_data.groupby('Product Category')['Total Amount'].mean()
print(category_sales)
print(average_sales_category)
```

```
Product Category
Beauty         143515
Clothing       155580
Electronics    156905
Name: Total Amount, dtype: int64
Product Category
Beauty         467.475570
Clothing       443.247863
Electronics    458.786550
Name: Total Amount, dtype: float64
```

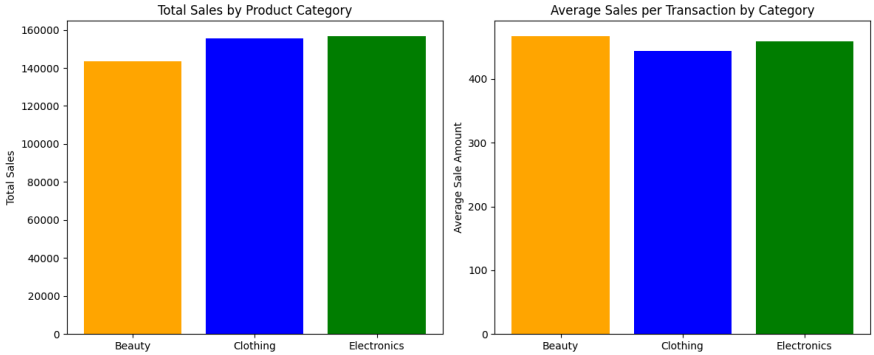
```
# Plot sales by product category
fig, ax3 = plt.subplots(1, 2, figsize=(12, 5))

# Total sales by category
ax3[0].bar(category_sales.index, category_sales, color=['orange', 'blue', 'green'])
ax3[0].set_title('Total Sales by Product Category')
ax3[0].set_ylabel('Total Sales')

# Average sales by category
ax3[1].bar(average_sales_category.index, average_sales_category, color=['orange', 'blue', '
```

```
ax3[1].set_title('Average Sales per Transaction by Category')
ax3[1].set_ylabel('Average Sale Amount')

plt.tight_layout()
plt.show()
```




```
import pandas as pd

# Load the dataset
retail_data = pd.read_csv('retail_sales_dataset.csv')

# Ensure you have the Age column in your DataFrame
if 'Age' in retail_data.columns:
    # Create age groups for segmentation
    bins = [18, 25, 35, 45, 55, 65]
    labels = ['18-25', '26-35', '36-45', '46-55', '56-65']
    retail_data['Age Group'] = pd.cut(retail_data['Age'], bins=bins, labels=labels)

    # Calculate total and average spending per age group
    age_group_sales = retail_data.groupby('Age Group')['Total Amount'].sum()
    age_group_avg_sales = retail_data.groupby('Age Group')['Total Amount'].mean()

    print(age_group_avg_sales)
    print("Age Group Sales\n", age_group_sales)
else:
    print("The column 'Age' does not exist in the DataFrame.")
```

```
Age Group
18-25    495.506757
26-35    480.390244
```

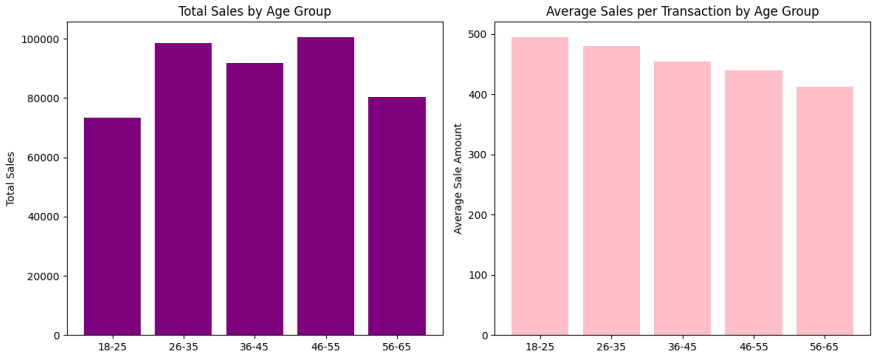
```
36-45    454.801980
46-55    439.694323
56-65    412.358974
Name: Total Amount, dtype: float64
Age Group Sales
 Age Group
18-25     73335
26-35     98480
36-45     91870
46-55    100690
56-65     80410
Name: Total Amount, dtype: int64
<ipython-input-34-2925c0bc91f7>:14: FutureWarning: The default of observed=False is de
  age_group_sales = retail_data.groupby('Age Group')['Total Amount'].sum()
<ipython-input-34-2925c0bc91f7>:15: FutureWarning: The default of observed=False is de
  age_group_avg_sales = retail_data.groupby('Age Group')['Total Amount'].mean()
```

```
# Plot spending by age group
fig, ax4 = plt.subplots(1, 2, figsize=(12, 5))

# Total sales by age group
ax4[0].bar(age_group_sales.index, age_group_sales, color='purple')
ax4[0].set_title('Total Sales by Age Group')
ax4[0].set_ylabel('Total Sales')

# Average sales by age group
ax4[1].bar(age_group_avg_sales.index, age_group_avg_sales, color='pink')
ax4[1].set_title('Average Sales per Transaction by Age Group')
ax4[1].set_ylabel('Average Sale Amount')

plt.tight_layout()
plt.show()
```




```
# 4. Heatmap for Sales by Age Group and Product Category

# Create pivot table for heatmap (sum of sales by Age Group and Product Category)
heatmap_data = retail_data.pivot_table(values='Total Amount', index='Age Group', columns='P
heatmap_data
```

```
<ipython-input-39-8e437dfc98b6>:4: FutureWarning: The default value of observed=False
  heatmap_data = retail_data.pivot_table(values='Total Amount', index='Age Group', col
```

| Product Category | Beauty | Clothing | Electronics |
|---|---|---|---|
| **Age Group** | | | |
| **18-25** | 26320 | 22425 | 24590 |
| **26-35** | 31240 | 39975 | 27265 |
| **36-45** | 28405 | 29550 | 33915 |
| **46-55** | 34720 | 30485 | 35485 |
| **56-65** | 17870 | 29060 | 33480 |

Next steps: Generate code with heatmap_data | View recommended plots | New interactive sheet

```
# Plot heatmap
plt.figure(figsize=(10,6))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt=".0f")
plt.title('Sales by Age Group and Product Category')
```

```
plt.xlabel('Product Category')
plt.ylabel('Age Group')
plt.show()
```