walmart-case-study

January 11, 2024

Walmart Business Case Study

[ ]:
```
# Walmart Business Case Study
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import calendar as cl
```

[ ]:
```
walmart_df = pd.read_csv("/content/walmart_data.txt")
walmart_df_copy = walmart_df.copy()
walmart_df
```

[ ]:
```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
0       1000001  P00069042      F   0-17          10             A
1       1000001  P00248942      F   0-17          10             A
2       1000001  P00087842      F   0-17          10             A
3       1000001  P00085442      F   0-17          10             A
4       1000002  P00285442      M    55+          16             C
…           …          …      …     …            …
550063  1006033  P00372445      M  51-55          13             B
550064  1006035  P00375436      F  26-35           1             C
550065  1006036  P00375436      F  26-35          15             B
550066  1006038  P00375436      F    55+           1             C
550067  1006039  P00371644      F  46-50           0             B

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                               2               0                 3      8370
1                               2               0                 1     15200
2                               2               0                12      1422
3                               2               0                12      1057
4                              4+               0                 8      7969
…                             …              …                 …        …
550063                          1               1                20       368
550064                          3               0                20       371
550065                         4+               1                20       137
550066                          2               0                20       365
550067                         4+               1                20       490
```

```
[550068 rows x 10 columns]
```

It has 550068 rows and 10 columns.

1. Problem statement:

Analyze Walmart's Black Friday sales data to understand customer spending behavior and improve sales strategy for increased business growth and profitability.

```
[ ]: walmart_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

```
[ ]: walmart_df.shape
```

```
[ ]: (550068, 10)
```

```
[ ]: walmart_df.isnull().count()
```

```
[ ]: User_ID                       550068
     Product_ID                    550068
     Gender                        550068
     Age                           550068
     Occupation                    550068
     City_Category                 550068
     Stay_In_Current_City_Years    550068
     Marital_Status                550068
     Product_Category              550068
     Purchase                      550068
     dtype: int64
```

```
[ ]: walmart_df.isnull().sum()
```

```
[ ]: User_ID                       0
     Product_ID                    0
     Gender                        0
     Age                           0
     Occupation                    0
     City_Category                 0
     Stay_In_Current_City_Years    0
     Marital_Status                0
     Product_Category              0
     Purchase                      0
     dtype: int64
```

There are no missing values in any column, eliminating the need for data imputation.

```
[ ]: walmart_df.describe(include="all")
```

```
[ ]:             User_ID Product_ID  Gender     Age     Occupation City_Category  \
     count  5.500680e+05     550068  550068  550068  550068.000000        550068
     unique          NaN       3631       2       7            NaN             3
     top             NaN  P00265242       M   26-35            NaN             B
     freq            NaN       1880  414259  219587            NaN        231173
     mean   1.003029e+06        NaN     NaN     NaN       8.076707           NaN
     std    1.727592e+03        NaN     NaN     NaN       6.522660           NaN
     min    1.000001e+06        NaN     NaN     NaN       0.000000           NaN
     25%    1.001516e+06        NaN     NaN     NaN       2.000000           NaN
     50%    1.003077e+06        NaN     NaN     NaN       7.000000           NaN
     75%    1.004478e+06        NaN     NaN     NaN      14.000000           NaN
     max    1.006040e+06        NaN     NaN     NaN      20.000000           NaN

           Stay_In_Current_City_Years  Marital_Status  Product_Category  \
     count                     550068   550068.000000     550068.000000
     unique                         5             NaN               NaN
     top                            1             NaN               NaN
     freq                      193821             NaN               NaN
     mean                         NaN        0.409653          5.404270
     std                          NaN        0.491770          3.936211
     min                          NaN        0.000000          1.000000
     25%                          NaN        0.000000          1.000000
     50%                          NaN        0.000000          5.000000
     75%                          NaN        1.000000          8.000000
     max                          NaN        1.000000         20.000000

                  Purchase
     count   550068.000000
     unique            NaN
     top               NaN
     freq              NaN
```

```
mean      9263.968713
std       5023.065394
min         12.000000
25%       5823.000000
50%       8047.000000
75%      12054.000000
max      23961.000000
```

2) Non-Graphical Analysis: Value counts and unique attributes

3) Visual Analysis - Univariate & Bivariate

value count and their visual analysis are done together.

[ ]:
```
walmart_df["User_ID"].nunique()
```

[ ]:
```
5891
```

[ ]:
```
walmart_df["Product_ID"].nunique()
```

[ ]:
```
3631
```

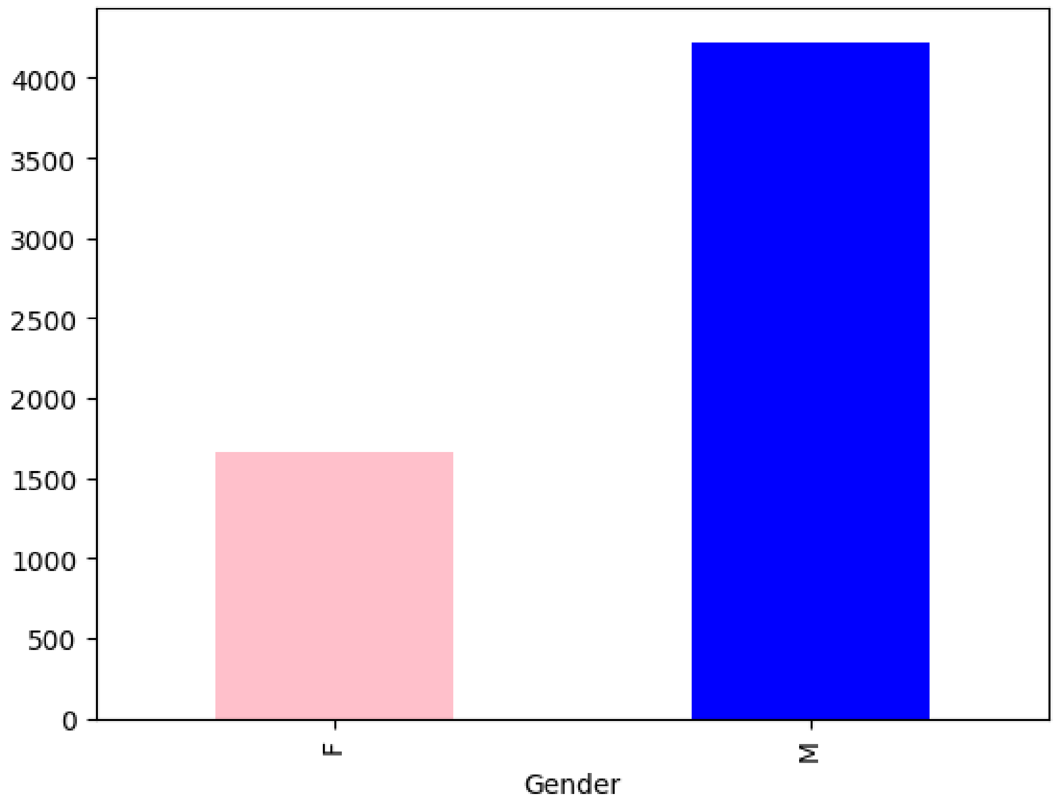
[ ]:
```
walmart_df.groupby("Gender")["User_ID"].nunique()
```

[ ]:
```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

The dataset comprises **5891** distinct customers, out of which **4225 (75.31%)** are male customers and **1666 (24.69%)** are female customers. The dataset also includes **3631** unique products.

[ ]:
```
walmart_df.groupby("Gender")["User_ID"].nunique().plot(kind="bar",
    color=["pink","blue"])
```

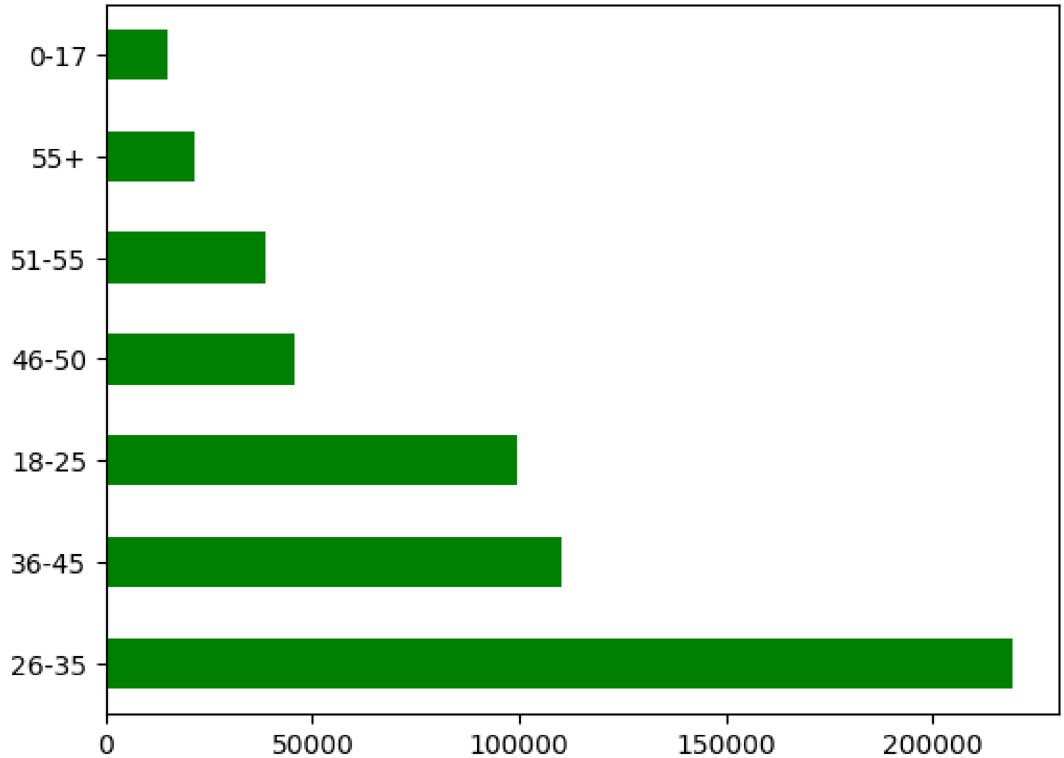
[ ]:
```
<Axes: xlabel='Gender'>
```



From the analysis, More men shop at Walmart, and they make more transactions compared to women.

[ ]:
```
walmart_df["Age"].value_counts(), walmart_df["Age"].value_counts().
 ↪plot(kind="barh", color =["green"])
```

[ ]:
```
(26-35    219587
 36-45    110013
 18-25     99660
 46-50     45701
 51-55     38501
 55+       21504
 0-17      15102
 Name: Age, dtype: int64,
 <Axes: >)
```

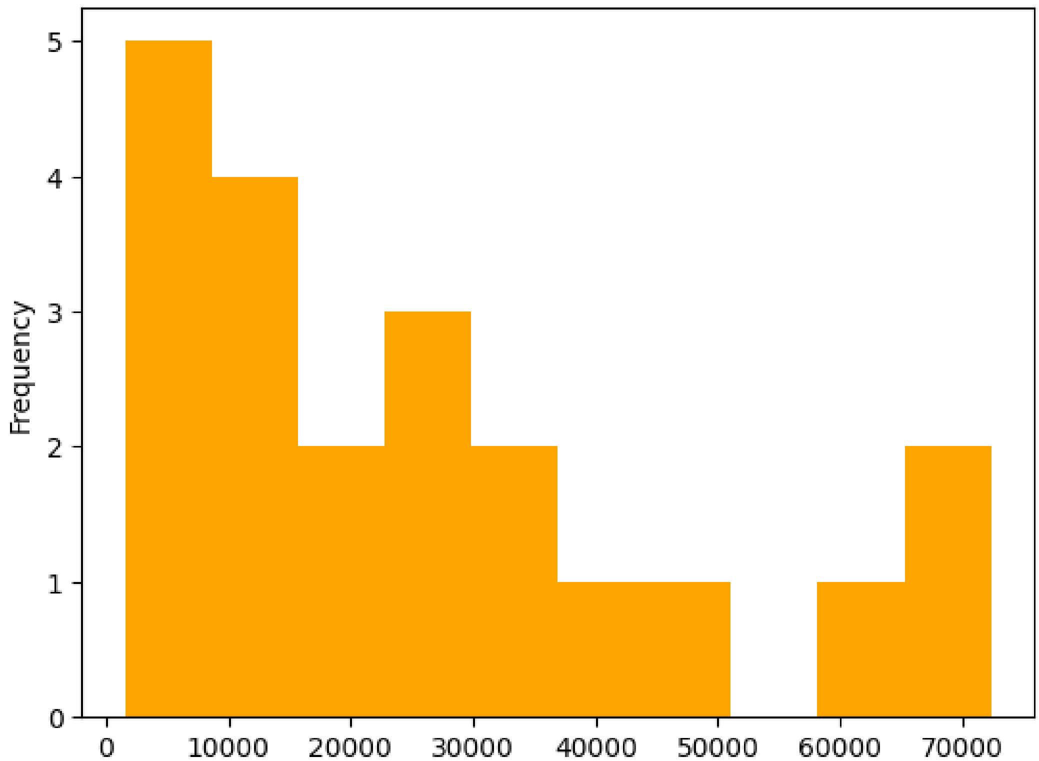


From this Analysis, Most purchases at Walmart are made by people between **18 and 45** years old, especially people aged **26 to 35**

[ ]:
```
walmart_df["Occupation"].value_counts(), walmart_df["Occupation"].
 ↪value_counts().plot(kind="hist", color = "orange")
```

[ ]:
```
(4     72308
 0     69638
 7     59133
 1     47426
 17    40043
 20    33562
 12    31179
 14    27309
 2     26588
 16    25371
 6     20355
 3     17650
 10    12930
 5     12177
 15    12165
 11    11586
```

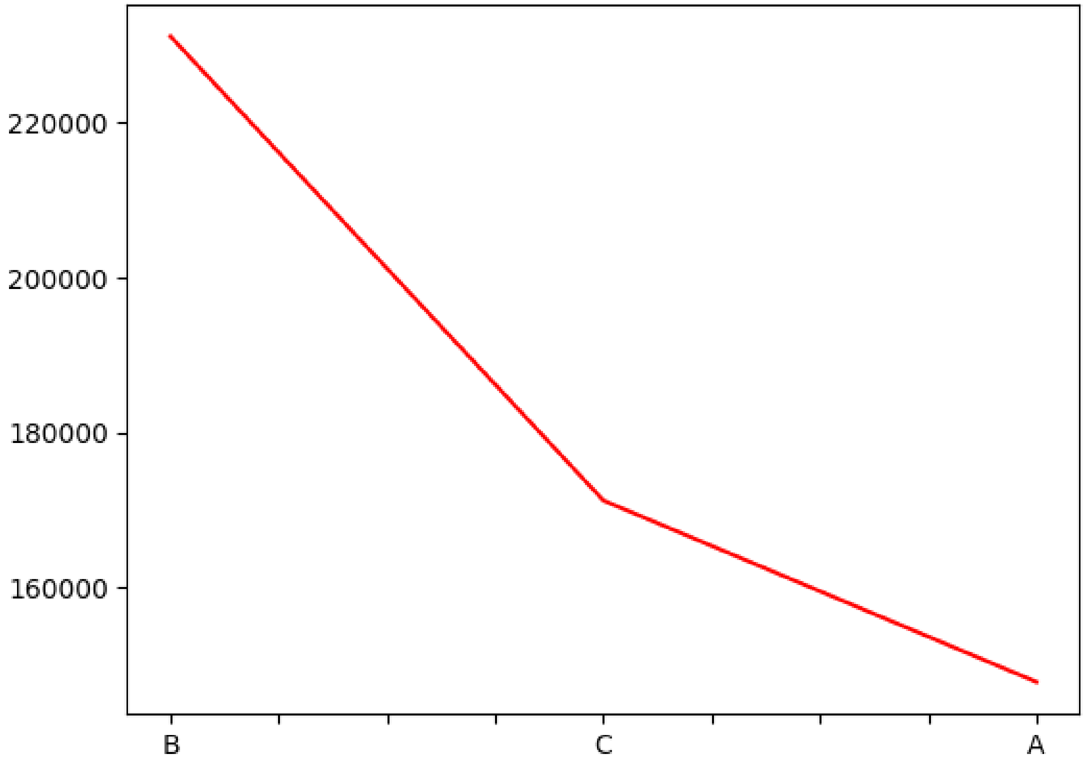
```
19      8461
13      7728
18      6622
9       6291
8       1546
Name: Occupation, dtype: int64,
<Axes: ylabel='Frequency'>)
```



Occupations categorized as 4, 0, and 7 show the highest transaction counts among customers

[ ]:
```
walmart_df["City_Category"].value_counts(), walmart_df["City_Category"].
 ↪value_counts().plot(kind="line", color="red")
```

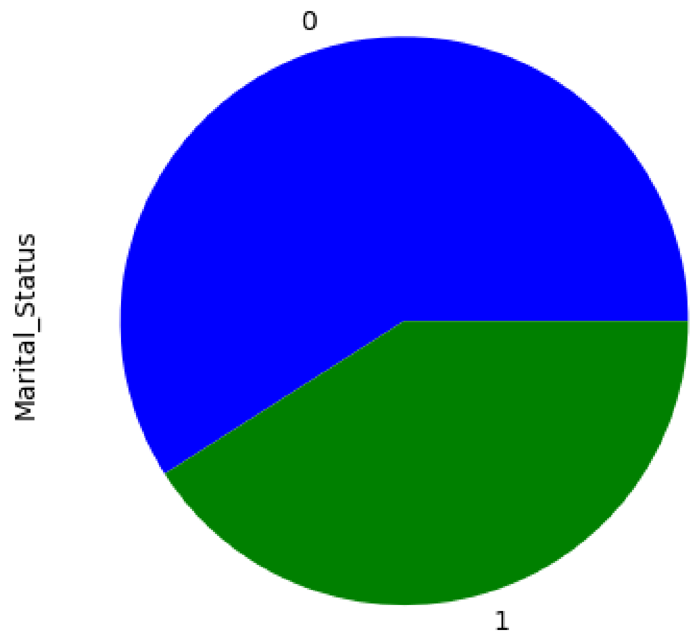
[ ]:
```
(B    231173
 C    171175
 A    147720
 Name: City_Category, dtype: int64,
 <Axes: >)
```



City **B** records the highest number of transactions among all cities

```
[ ]: walmart_df["Marital_Status"].value_counts(), walmart_df["Marital_Status"].
      ↪value_counts().plot(kind="pie")
```

```
[ ]: (0    324731
      1    225337
      Name: Marital_Status, dtype: int64,
      <Axes: ylabel='Marital_Status'>)
```

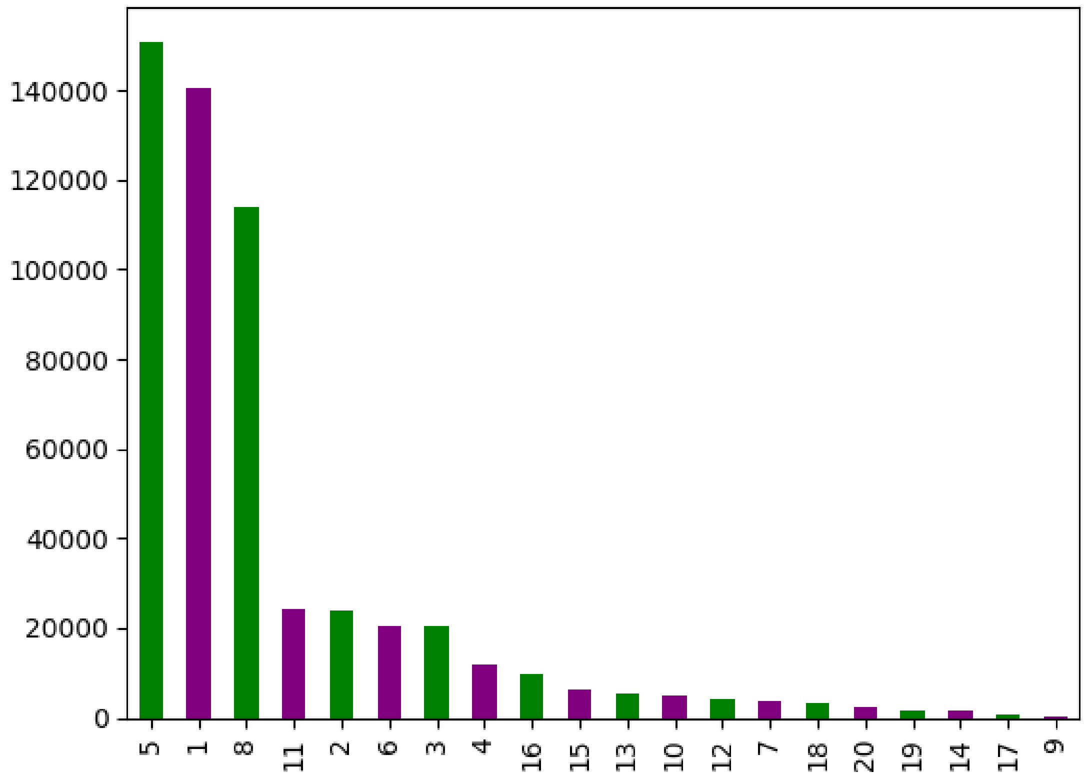


There are more transactions associated with the marital status **'0'** compared to '1'

[ ]:
```
walmart_df["Product_Category"].value_counts(), walmart_df["Product_Category"].
↪value_counts().plot(kind="bar", color =["green","purple"])
```

[ ]:
```
(5      150933
 1      140378
 8      113925
 11      24287
 2       23864
 6       20466
 3       20213
 4       11753
 16       9828
 15       6290
 13       5549
 10       5125
 12       3947
 7        3721
 18       3125
 20       2550
 19       1603
 14       1523
 17        578
```

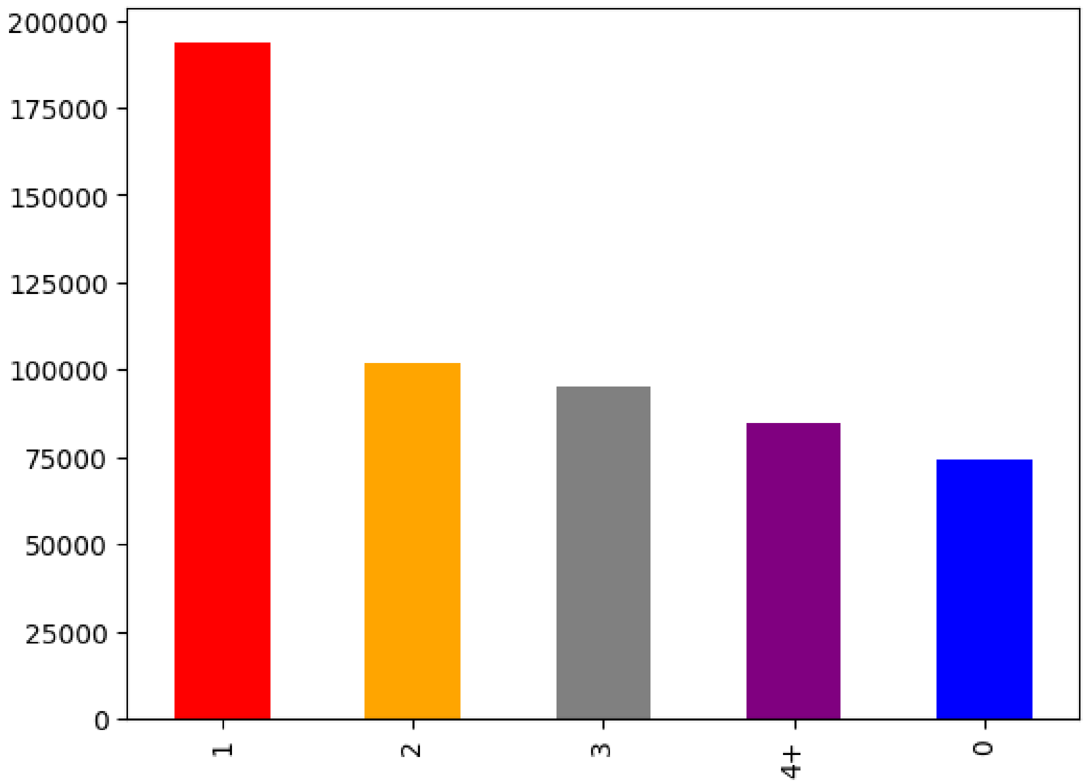
```
9         410
Name: Product_Category, dtype: int64,
<Axes: >)
```



The product categories **5, 1, and 8** exhibit significantly higher sales compared to the remaining categories.

[ ]:
```
walmart_df["Stay_In_Current_City_Years"].value_counts().plot(kind="bar", color␣
↪= (["red","orange","grey","purple","blue"]))
```

[ ]:
```
<Axes: >
```



[ ]:
```
walmart_df["Product_ID"].nunique()
```

[ ]: 3631

[ ]:
```
walmart_df.groupby("Product_ID")["Product_ID"].count().
    ↪sort_values(ascending=False)
```

[ ]:
```
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             …
P00068742       1
P00012342       1
P00162742       1
P00091742       1
P00231642       1
Name: Product_ID, Length: 3631, dtype: int64
```

[ ]:
```
walmart_df.groupby("Product_Category")["Product_Category"].count().
    ↪sort_values(ascending= False)
```

[ ]:
```
Product_Category
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
```

Product category **5** has exhibited strong performance, boasting a remarkable sales count of **150,933**.

[ ]:
```
walmart_df.groupby("User_ID")["Purchase"].sum().sort_values(ascending=False)
```

[ ]:
```
User_ID
1004277    10536909
1001680     8699596
1002909     7577756
1001941     6817493
1000424     6573609
              …
1004991       52371
1005117       49668
1003883       49349
1000094       49288
1004464       46681
Name: Purchase, Length: 5891, dtype: int64
```

[ ]:
```
walmart_df.groupby("User_ID")["User_ID"].count().sort_values(ascending=False)
```

```
[ ]: User_ID
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           …
1002111       7
1005391       7
1002690       7
1005608       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

```python
[ ]: walmart_df.groupby("User_ID")["Purchase"].sum().sort_values(ascending=False)
```

```
[ ]: User_ID
1004277    10536909
1001680     8699596
1002909     7577756
1001941     6817493
1000424     6573609
             …
1004991       52371
1005117       49668
1003883       49349
1000094       49288
1004464       46681
Name: Purchase, Length: 5891, dtype: int64
```

The user **with ID 1001680** holds the record for the highest number of purchases, with a total of **1026 orders**.

On the other hand, user **ID 1004277** has achieved the distinction of having the highest total bill value, which amounts to an impressive **$10,536,909.**

Categorical Columns

```python
[ ]: categorical_columns = ["Gender","Age", "Occupation", "City_Category",
     ↪"Marital_Status", "Product_Category", "Stay_In_Current_City_Years"]
     categorical_columns
```

```
[ ]: ['Gender',
      'Age',
      'Occupation',
      'City_Category',
      'Marital_Status',
      'Product_Category',
```

```
'Stay_In_Current_City_Years']
```

[ ]:
```
walmart_df[categorical_columns].melt().groupby(["variable","value"])[["value"]].
↪value_counts()
```

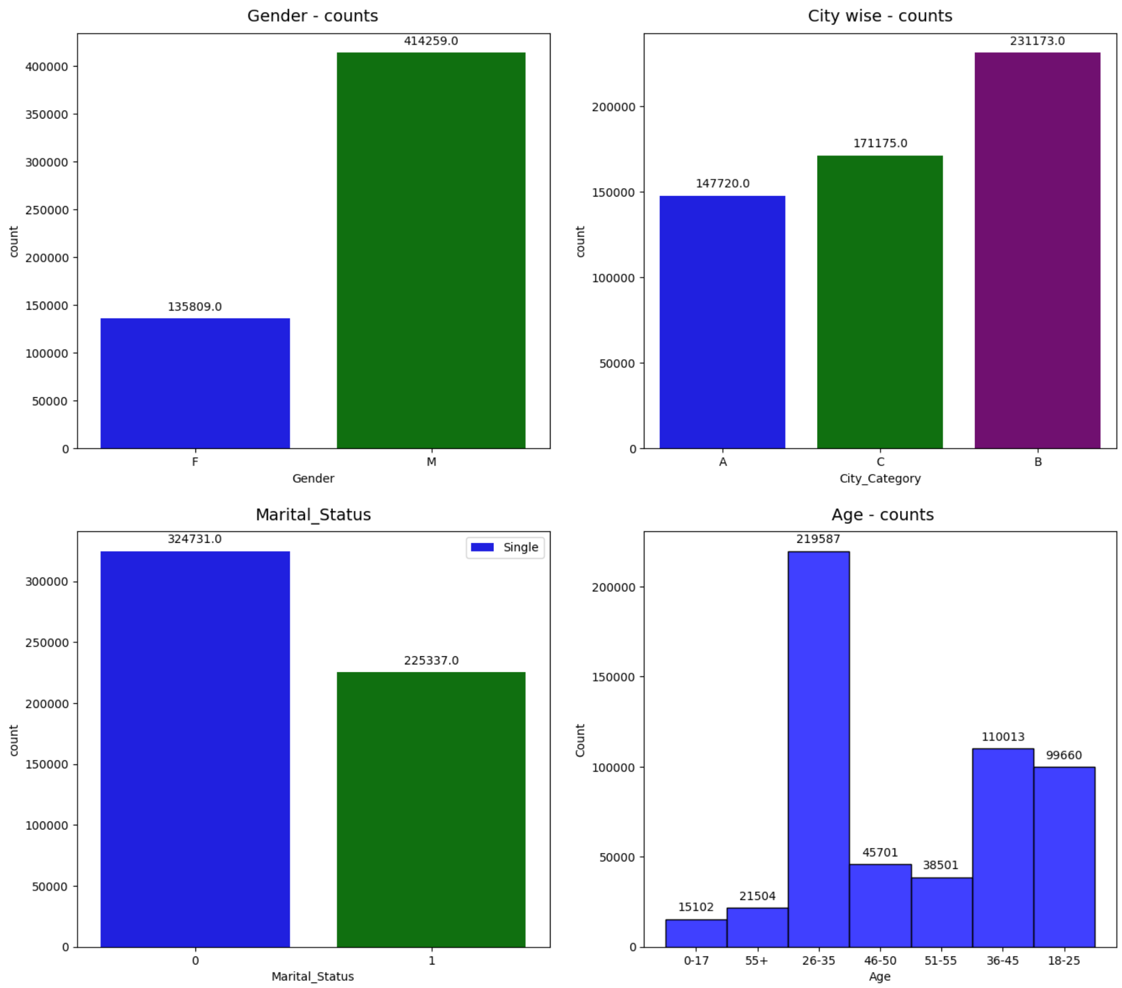
[ ]:
```
variable                    value
Age                         0-17       15102
                            18-25      99660
                            26-35     219587
                            36-45     110013
                            46-50      45701
                            51-55      38501
                            55+        21504
City_Category               A         147720
                            B         231173
                            C         171175
Gender                      F         135809
                            M         414259
Marital_Status              0         324731
                            1         225337
Occupation                  0          69638
                            1          47426
                            2          26588
                            3          17650
                            4          72308
                            5          12177
                            6          20355
                            7          59133
                            8           1546
                            9           6291
                            10         12930
                            11         11586
                            12         31179
                            13          7728
                            14         27309
                            15         12165
                            16         25371
                            17         40043
                            18          6622
                            19          8461
                            20         33562
Product_Category            1         140378
                            2          23864
                            3          20213
                            4          11753
                            5         150933
                            6          20466
```

```
                             7          3721
                             8        113925
                             9           410
                             10         5125
                             11        24287
                             12         3947
                             13         5549
                             14         1523
                             15         6290
                             16         9828
                             17          578
                             18         3125
                             19         1603
                             20         2550
Stay_In_Current_City_Years   0         74398
                             1        193821
                             2        101838
                             3         95285
                             4+        84726
dtype: int64
```

Visual Analysis - Univariate & Bivariate

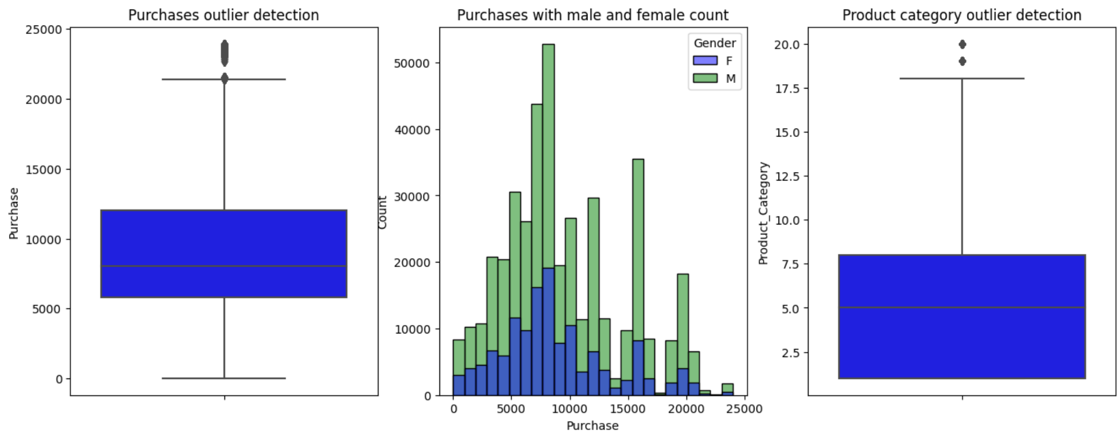
[ ]:
```
custom_palette = sns.color_palette(['blue', 'green', 'purple'])
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 10))
fig.subplots_adjust(top=1.2)
sns.set_palette(custom_palette)

# Gender-counts
ax = sns.countplot(data=walmart_df, x="Gender", ax=axis[0,0])
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='bottom', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                textcoords='offset points')
axis[0,0].set_title("Gender - counts", pad=10, fontsize=14)

# City-category
ax = sns.countplot(data=walmart_df, x="City_Category", ax=axis[0,1])
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='bottom', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                textcoords='offset points')
axis[0,1].set_title("City wise - counts", pad=10, fontsize=14)
```

```
# marital-status
ax = sns.countplot(data=walmart_df, x="Marital_Status",  ax=axis[1,0])
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='bottom', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                textcoords='offset points')
axis[1,0].set_title("Marital_Status", pad=10, fontsize=14)
axis[1,0].legend(labels=['Single','Married'],loc='upper right')

# Age-counts
ax = sns.histplot(data=walmart_df, x="Age", ax=axis[1,1])
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='bottom', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                textcoords='offset points')
axis[1,1].set_title(" Age - counts", pad=10, fontsize=14)
plt.show()
```

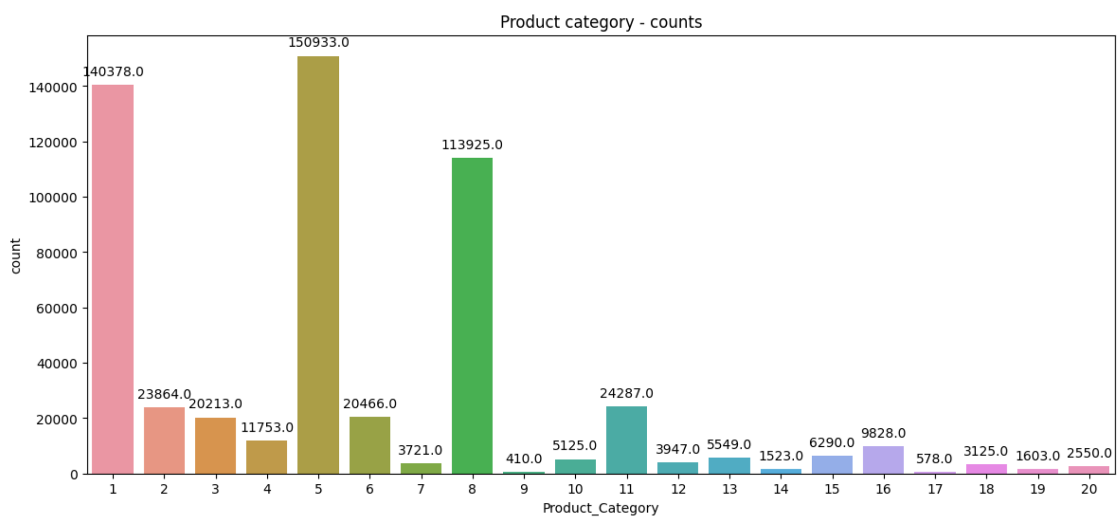


Outlier Detection

[ ]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(16, 4))
fig.subplots_adjust(top=1.2)

sns.boxplot(data=walmart_df, y="Purchase", ax=axis[0])
sns.histplot( x='Purchase', data=walmart_df, bins=25, hue='Gender',ax=axis[1])
sns.boxplot(data=walmart_df, y="Product_Category", ax=axis[2])

axis[0].set_title('Purchases outlier detection')
axis[1].set_title('Purchases with male and female count')
axis[2].set_title('Product category outlier detection')
plt.show()
```

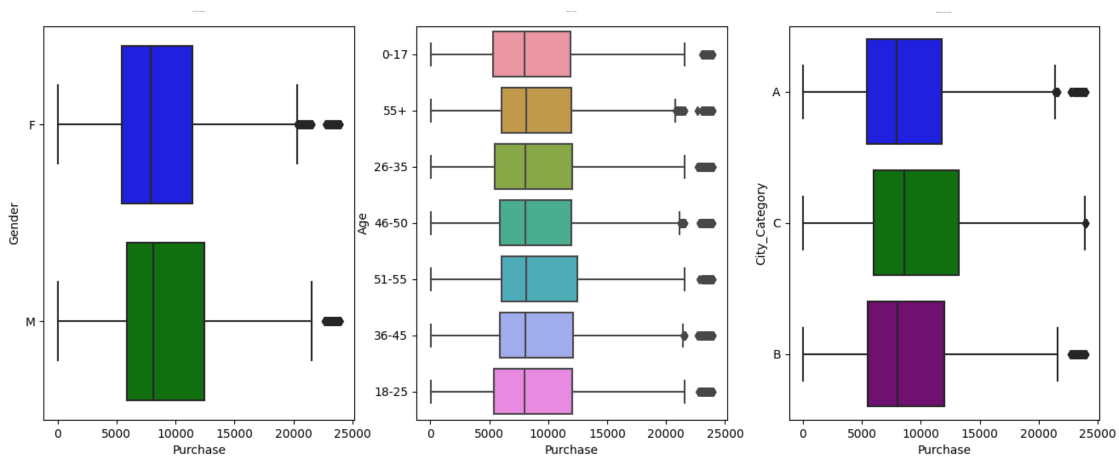



```
[ ]: plt.figure(figsize=(14, 6))
     ax = sns.countplot(data=walmart_df, x="Product_Category")
     # Add count labels on top of the bars
     for p in ax.patches:
      height = p.get_height()
      if(height > 0):
         ax.annotate(f'{height}', (p.get_x() + p.get_width() / 2., height),␣
      ↪ha='center', va='bottom', fontsize=10, color='black', xytext=(0, 5),
                     textcoords='offset points')
     plt.title("Product category - counts")
     plt.show()
```



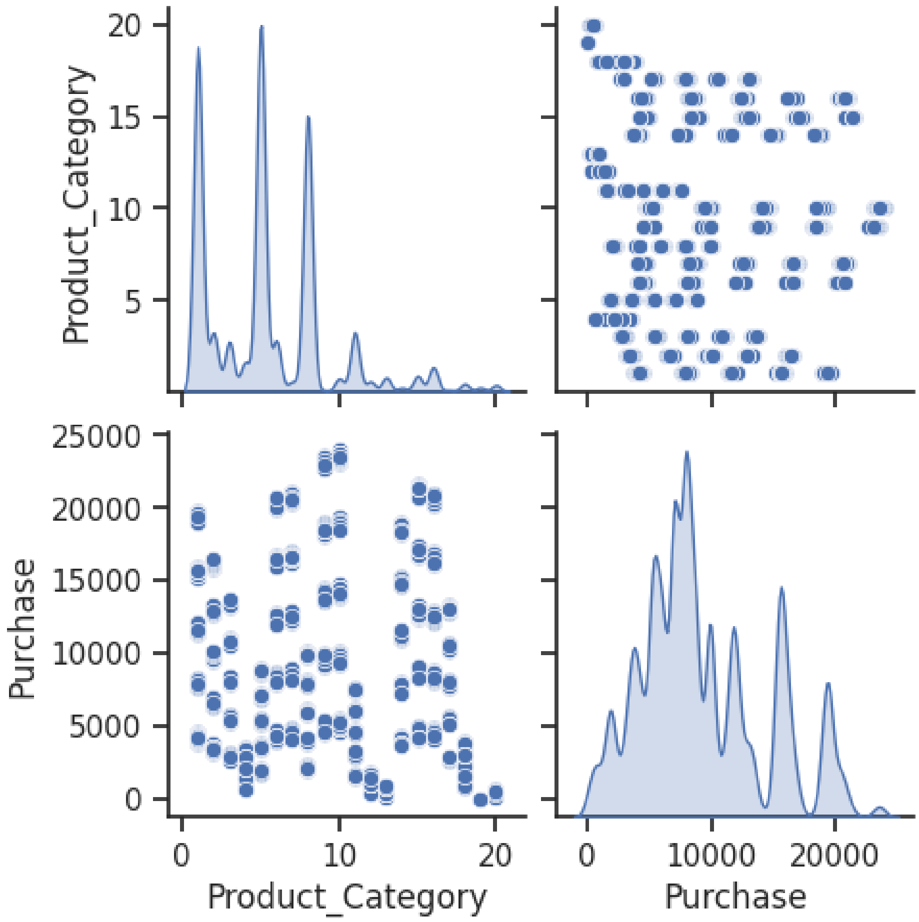

```
[ ]: y_attr = ['Gender','Age','City_Category','Product_Category']
     fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(16, 6))
```

```
count = 0
paletteCount = 1;
for i in range(3):
    paletteSet = 'Set'+str(paletteCount)
    sns.boxplot(data=walmart_df,x='Purchase',y=y_attr[count],ax=axis[i])
    axis[i].set_title(f"Purchase vs {y_attr[count]}", pad=12, fontsize=1)
    count += 1
    paletteCount +=1
    if count > 2:
        paletteCount=1
```



[ ]:
```
sns.set(style="ticks")
sns.pairplot(walmart_df[["Product_Category","Purchase"]], diag_kind= "kde")
```

[ ]: <seaborn.axisgrid.PairGrid at 0x7c99d3da1c00>



Missing Value & Outlier Detection:

There is **no missing values** in the given dataset.

From the above visuals we can see that **there is outliers** in **purchase and product category**.

[ ]:
```
def findOutliers(columnName):

# Calculate Q1 and Q3
    q1 = walmart_df[columnName].quantile(0.25)
    q3 = walmart_df[columnName].quantile(0.75)
    iqr = q3-q1

    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = walmart_df[(walmart_df[columnName] < lower_bound) |␣
 ↪(walmart_df[columnName] > upper_bound)]

    print(f"Outliers of unique {columnName}:\n{outliers[columnName].unique()}")
```

```python
    print(f"Outliers of {columnName}:\n{outliers[columnName]}")
```

[ ]:
```python
findOutliers('Purchase')
findOutliers('Product_Category')
```

```
Outliers of unique Purchase:
[23603 23792 23233 … 23945 23680 23529]
Outliers of Purchase:
343       23603
375       23792
652       23233
736       23595
1041      23341
          …
544488    23753
544704    23724
544743    23529
545663    23663
545787    23496
Name: Purchase, Length: 2677, dtype: int64
Outliers of unique Product_Category:
[20 19]
Outliers of Product_Category:
545915    20
545916    20
545917    20
545918    20
545919    20
          ..
550063    20
550064    20
550065    20
550066    20
550067    20
Name: Product_Category, Length: 4153, dtype: int64
```

Observation:

1. The data reveals a distinct trend, with **males leading** in terms of purchase frequency at **414,259** instances, while **females** have engaged in purchases **135,809** times.
2. **City B** secures the top position in purchase frequency, with a significant count of **231,173** transactions.
3. **Unmarried** individuals exhibit a **higher tendency** for purchasing in comparison to those who are married.
4. The age bracket of **26-35** demonstrates a **greater inclination** towards making purchases, closely followed by the 36-45 age group.

5. A substantial portion of purchases falls within the range of 7000$ to 8000$.
6. **Category 5** stands out by contributing the highest number of sales in comparison to the other categories.

Answering questions:

1.Are women spending more money per transaction than men? Why or Why not?

[ ]:
```
gender_user_data = walmart_df.groupby(['Gender'])['Purchase'].mean()
gender_user_data
```

[ ]:
```
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

It's evident that **men tend to spend more** on average per transaction.

Therefore, the response to the earlier question is negative

2. Confidence intervals and distribution of the mean of the expenses by female and male customers.

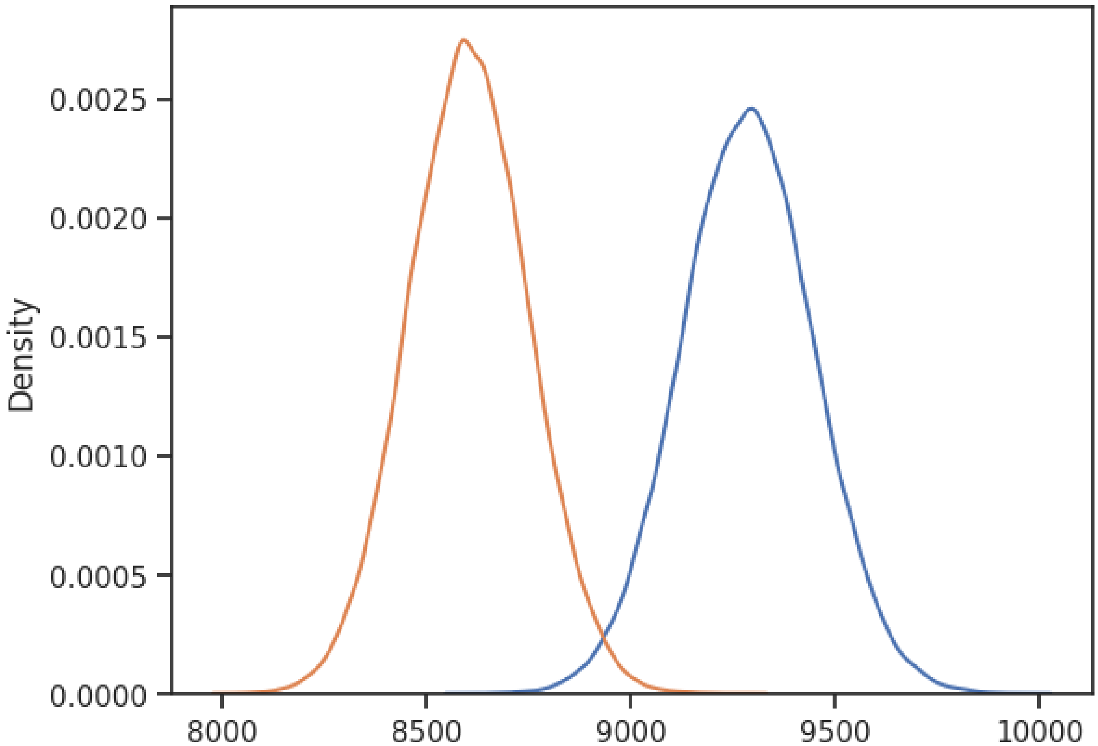
[ ]:
```
def mean_cal(DF, sample_size):
    samp = DF.sample(sample_size)
    n = samp.size
    n_trials = 100000
    bs_samples = np.random.choice(samp, (n_trials,n), replace="True")
    sample_means = bs_samples.mean(axis=1)
    return sample_means
def confidence_interval(means, conf):
    if conf == 99:
        left = np.percentile(means,0.5)
        right = np.percentile(means,99.5)
    elif conf == 95:
        left = np.percentile(means, 2.5)
        right = np.percentile(means, 97.5)
    else:
        left = np.percentile(means, 5)
        right = np.percentile(means, 95)

    return round(left,2), round(right,2)
```

[ ]:
```
male_purchase = walmart_df[walmart_df["Gender"]=="M"]["Purchase"]
female_purchase = walmart_df[walmart_df["Gender"]=="F"]["Purchase"]
```

[ ]:
```
# Analysizing male/femal purcahses for 1000 samples
male_means = mean_cal(male_purchase,1000)
female_means = mean_cal(female_purchase,1000)
```

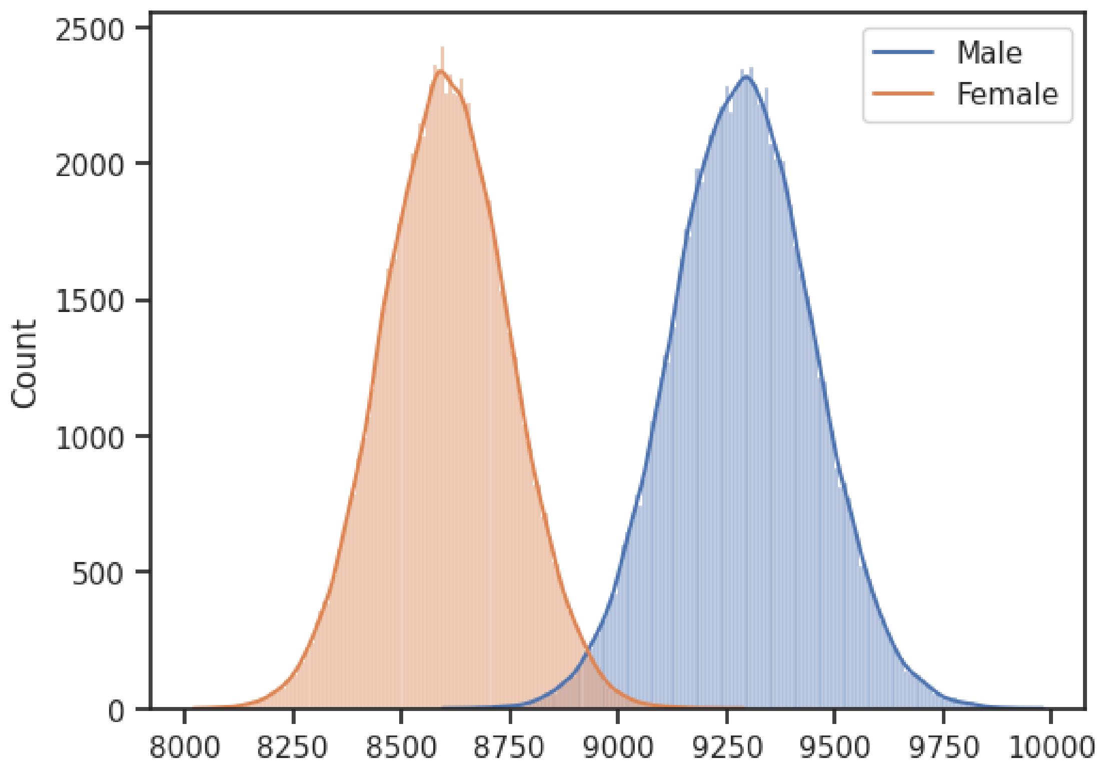
```
sns.kdeplot(data = male_means), sns.kdeplot(data=female_means)
```

```
[ ]: (<Axes: ylabel='Density'>, <Axes: ylabel='Density'>)
```




```
[ ]: sns.histplot(data = male_means,kde=True), sns.histplot(data=female_means,␣
     ↪kde=True)
     plt.legend(labels=['Male','Female'],loc='upper right')
```

```
[ ]: <matplotlib.legend.Legend at 0x7c99cb2ba0b0>
```



[ ]:
```
# at 99% confidence for male
male_left_interval_99,male_right_interval_99 =␣
  ↪confidence_interval(male_means,99)
male_left_interval_99,male_right_interval_99
```

[ ]: (9075.65, 9907.87)

[ ]:
```
# at 95 confidence for male
male_left_interval_95,male_right_interval_95 =␣
  ↪confidence_interval(male_means,95)
male_left_interval_95,male_right_interval_95
```

[ ]: (9171.63, 9806.2)

[ ]:
```
# at 90 confidence for male
male_left_interval_90,male_right_interval_90 =␣
  ↪confidence_interval(male_means,90)
male_left_interval_90,male_right_interval_90
```

[ ]: (9221.49, 9754.2)

[ ]:
```
# at 99 confidence for female
female_left_interval_99,female_right_interval_99 =␣
↪confidence_interval(female_means,99)
female_left_interval_99,female_right_interval_99
```

[ ]: (8281.96, 9044.98)

[ ]:
```
# at 95 confidence for female
female_left_interval_95,female_right_interval_95 =␣
↪confidence_interval(female_means,95)
female_left_interval_95,female_right_interval_95
```

[ ]: (8366.69, 8948.01)

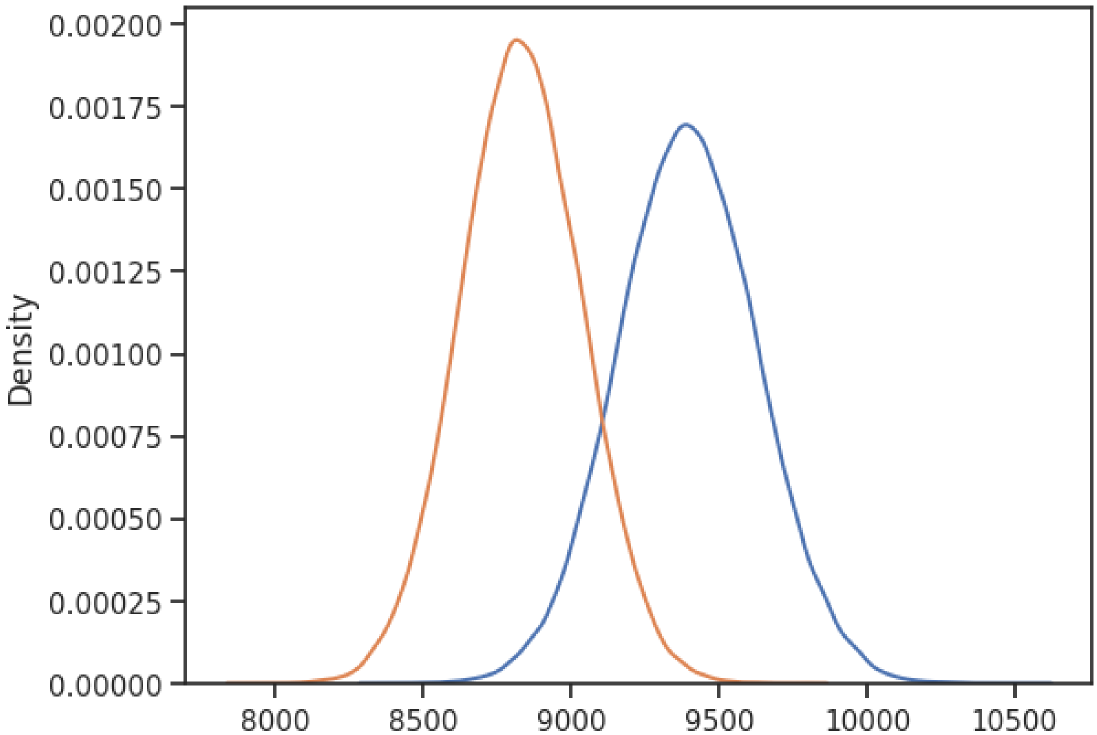
[ ]:
```
# at 90 confidence for female
female_left_interval_90,female_right_interval_90 =␣
↪confidence_interval(female_means,90)
female_left_interval_90,female_right_interval_90
```

[ ]: (8412.06, 8899.87)

Based on the findings shown above, the confidence intervals for both males and females **overlap only within the 99% interval**. This suggests that at higher spending ranges, there's a similarity in spending behavior between males and females. In essence, for larger transaction amounts, both genders exhibit comparable spending patterns.

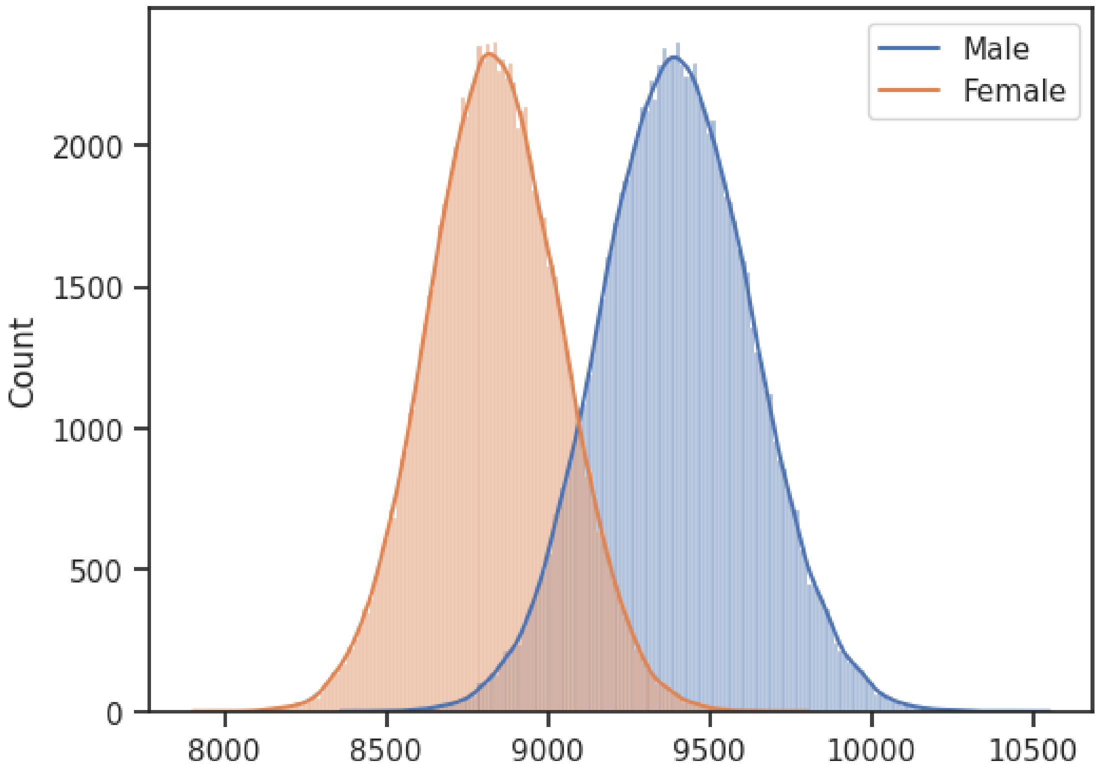
[ ]:
```
# Analysing for sample of 500
male_means_500 = mean_cal(male_purchase,500)
female_means_500 = mean_cal(female_purchase,500)
sns.kdeplot(data = male_means_500), sns.kdeplot(data=female_means_500)
```

[ ]: (<Axes: ylabel='Density'>, <Axes: ylabel='Density'>)



[ ]:
```
sns.histplot(data = male_means_500,kde=True), sns.
 ↪histplot(data=female_means_500, kde=True)
plt.legend(labels=['Male','Female'],loc='upper right')
```

[ ]: <matplotlib.legend.Legend at 0x7c99cb3e2e30>



[ ]:
```
# at 99 confidence for male
male_left_interval_99,male_right_interval_99 =␣
 ↪confidence_interval(male_means_500,99)
male_left_interval_99,male_right_interval_99
```

[ ]: (8798.72, 10002.84)

[ ]:
```
# at 95 confidence for male
male_left_interval_95,male_right_interval_95 =␣
 ↪confidence_interval(male_means_500,95)
male_left_interval_95,male_right_interval_95
```

[ ]: (8940.46, 9862.58)

[ ]:
```
# at 90 confidence for male
male_left_interval_90,male_right_interval_90 =␣
 ↪confidence_interval(male_means_500,90)
male_left_interval_90,male_right_interval_90
```

[ ]: (9014.66, 9786.51)

```
[ ]: # at 99 confidence for female
female_left_interval_99,female_right_interval_99 =␣
  ↪confidence_interval(female_means_500,99)
female_left_interval_99,female_right_interval_99
```

```
[ ]: (8315.78, 9368.19)
```

```
[ ]: # at 95 confidence for female
female_left_interval_95,female_right_interval_95 =␣
  ↪confidence_interval(female_means_500,95)
female_left_interval_95,female_right_interval_95
```

```
[ ]: (8315.78, 9368.19)
```

```
[ ]: # at 90 confidence for female
female_left_interval_90,female_right_interval_90 =␣
  ↪confidence_interval(female_means_500,90)
female_left_interval_90,female_right_interval_90
```

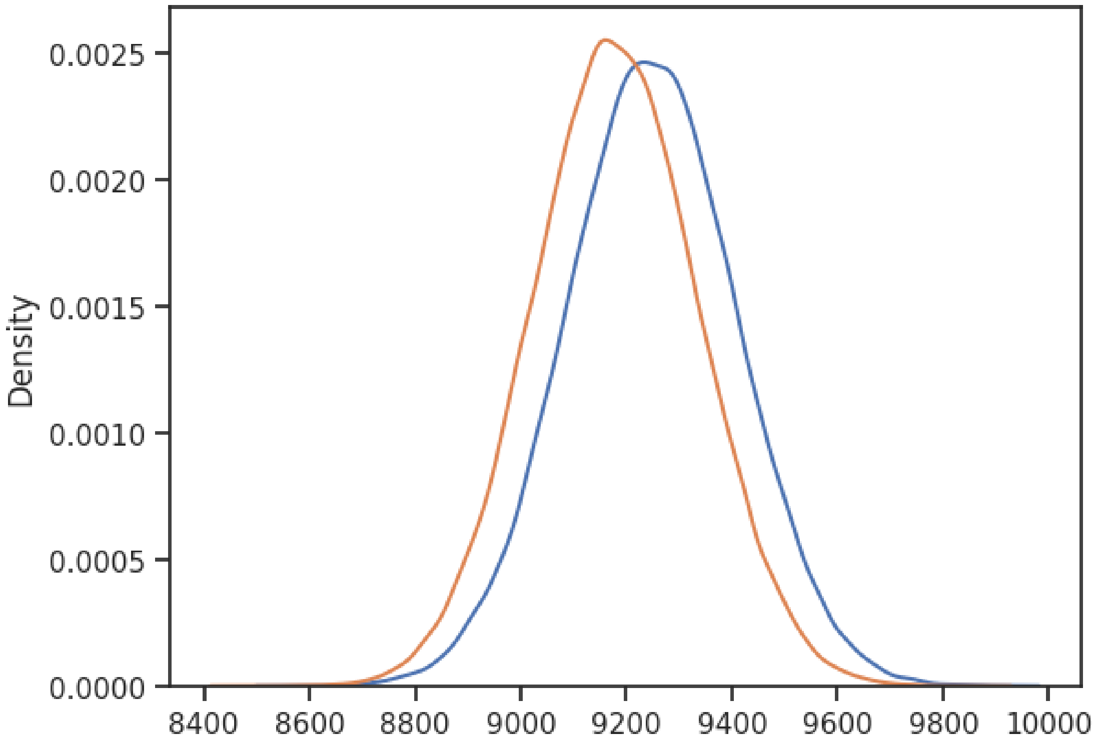
```
[ ]: (8498.34, 9173.31)
```

In the case of **500 samples**, the **overlap in the 99%** confidence intervals occurs solely for higher transaction values in both male and female purchases. This indicates that at elevated transaction amounts, the spending behavior between males and females appears to align or be comparable.

```
[ ]: # Calculating CI for Marital status
ms1_purchase = walmart_df[walmart_df["Marital_Status"]==1]["Purchase"]
ms2_purchase = walmart_df[walmart_df["Marital_Status"]==0]["Purchase"]
```

```
[ ]: # CI at 1000 samples
ms1_means = mean_cal(ms1_purchase,1000)
ms2_means = mean_cal(ms2_purchase,1000)
sns.kdeplot(data = ms1_means), sns.kdeplot(data=ms2_means)
```

```
[ ]: (<Axes: ylabel='Density'>, <Axes: ylabel='Density'>)
```



[ ]:
```
# at 99 confidence for ms1
ms1_means_left_interval_99,ms1_means_right_interval_99 =␣
↪confidence_interval(ms1_means,99)
ms1_means_left_interval_99,ms1_means_right_interval_99
```

[ ]: (8835.13, 9663.9)

[ ]:
```
# at 95 confidence for ms1
ms1_left_interval_95,ms1_right_interval_95 = confidence_interval(ms1_means,95)
ms1_left_interval_95,ms1_right_interval_95
```

[ ]: (8931.75, 9563.47)

[ ]:
```
# at 90 confidence for ms1
ms1_left_interval_90,ms1_right_interval_90 = confidence_interval(ms1_means,90)
ms1_left_interval_90,ms1_right_interval_90
```

[ ]: (8985.36, 9512.62)

[ ]:
```
# at 99 confidence for ms2
ms2_left_interval_99,ms2_right_interval_99 = confidence_interval(ms2_means,99)
ms2_left_interval_99,ms2_right_interval_99
```

[ ]: (8784.8, 9582.74)

```
[ ]: # at 95 confidence for ms2
ms2_left_interval_95,ms2_right_interval_95 = confidence_interval(ms2_means,95)
ms2_left_interval_95,ms2_right_interval_95
```

[ ]: (8875.69, 9486.12)

```
[ ]: # at 90 confidence for ms2
ms2_left_interval_90,ms2_right_interval_90 = confidence_interval(ms2_means,90)
ms2_left_interval_90,ms2_right_interval_90
```

[ ]: (8923.43, 9436.71)

Observing the confidence interval values for both **marital statuses**, it's evident that their **ranges overlap across all confidence levels**. This suggests that there isn't a substantial difference in spending behavior between the two marital statuses.

```
[ ]: # Calculating CI for age
age_26_35_purchase = walmart_df[walmart_df["Age"]=="26-35"]["Purchase"]
age_36_45_purchase = walmart_df[walmart_df["Age"]=="36-45"]["Purchase"]
age_18_25_purchase = walmart_df[walmart_df["Age"]=="18-25"]["Purchase"]
age_46_50_purchase = walmart_df[walmart_df["Age"]=="18-25"]["Purchase"]
age_51_55_purchase = walmart_df[walmart_df["Age"]=="18-25"]["Purchase"]
age_55_purchase = walmart_df[walmart_df["Age"]=="55+"]["Purchase"]
age_0_17_purchase = walmart_df[walmart_df["Age"]=="0-17"]["Purchase"]
```

```
[ ]: # analysing for 1000 samples and 99%
age_2635_means = mean_cal(age_2635_purchase,1000)
age_3645_means = mean_cal(age_3645_purchase,1000)
age_1825_means = mean_cal(age_1825_purchase,1000)
age_4650_means = mean_cal(age_4650_purchase,1000)
age_5155_means = mean_cal(age_5155_purchase,1000)
age_55_means = mean_cal(age_55_purchase,1000)
age_017_means = mean_cal(age_017_purchase,1000)
```

```
[ ]: # at 99 confidence
age_2635_left_interval_99,age_2635_right_interval_99 =
 ↪confidence_interval(age_2635_means,99)
age_3645_left_interval_99,age_3645_right_interval_99 =
 ↪confidence_interval(age_3645_means,99)
age_1825_left_interval_99,age_1825_right_interval_99 =
 ↪confidence_interval(age_1825_means,99)
age_4650_left_interval_99,age_2635_right_interval_99 =
 ↪confidence_interval(age_4650_means,99)
age_5155_left_interval_99,age_3645_right_interval_99 =
 ↪confidence_interval(age_5155_means,99)
```

```
age_55_left_interval_99,age_1825_right_interval_99 =␣
 ↪confidence_interval(age_55_means,99)
age_017_left_interval_99,age_2635_right_interval_99 =␣
 ↪confidence_interval(age_017_means,99)
```

[ ]:
```
print((age_2635_left_interval_99,␣
 ↪age_2635_right_interval_99),(age_3645_left_interval_99,␣
 ↪age_3645_right_interval_99),
(age_1825_left_interval_99,age_1825_right_interval_99),␣
 ↪(age_4650_left_interval_99,age_2635_right_interval_99),
(age_5155_left_interval_99,age_3645_right_interval_99),(age_55_left_interval_99,age_1825_right
(age_017_left_interval_99,age_2635_right_interval_99), sep="\n")
```

```
(8930.1, 9367.68)
(8662.35, 9497.5)
(8690.61, 9833.2)
(8493.51, 9367.68)
(8661.17, 9497.5)
(8998.47, 9833.2)
(8537.37, 9367.68)
```

As we can decode from above results, CI values at **99% confidence** are overlapping for all age groups. This implies there is **no significant diffence between their spending behaviour**.

RECOMMENDATIONS:

a) Given the dominance of male customers in both count and transaction numbers, Walmart could enhance its engagement with female customers **by introducing discounts or dedicating specific days, like women-only shopping days**, to attract more female shoppers.

b) City B exhibits higher purchase activity during Black Friday, indicating an opportunity for Walmart to strengthen its **marketing efforts in Cities A and C to boost purchase numbers**.

c) Customers who have resided in the city for a year or less are observed to engage in more transactions, potentially due to their initial unfamiliarity with available options. Walmart **should focus on retaining long-term customers** by understanding the competitive landscape and preferences of established residents.

d) The age group between 18 and 45 demonstrates the highest transaction volumes. Walmart could tailor its product offerings to cater more specifically to this age range while also **expanding outreach to underrepresented age groups**.

e) Notably, there's no discernible disparity between male and female spending at higher purchase amounts. **Both genders should receive equitable attention in targeted marketing strategies and discount initiatives.**

f) The analysis reveals no significant divergence in spending behavior among different marital statuses. Walmart's current approach should continue without alterations. **bold text**

g) Product categories 5, 1, and 8 emerge as the most frequently purchased. Walmart should **ensure ample stocking of items falling under these categories** to meet customer demand.