walmart

October 26, 2024

[9]:
```
# Import required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
import statsmodels.api as sm

# Load dataset
df = pd.read_csv("C:/Users/shrad/Downloads/Walmart_data.csv")

# Check the structure and data types of the dataset
print(df.info())

# Display the first few rows
df.head()
```

```
<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  int64 
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(6), object(4)
memory usage: 42.0+ MB
None
```

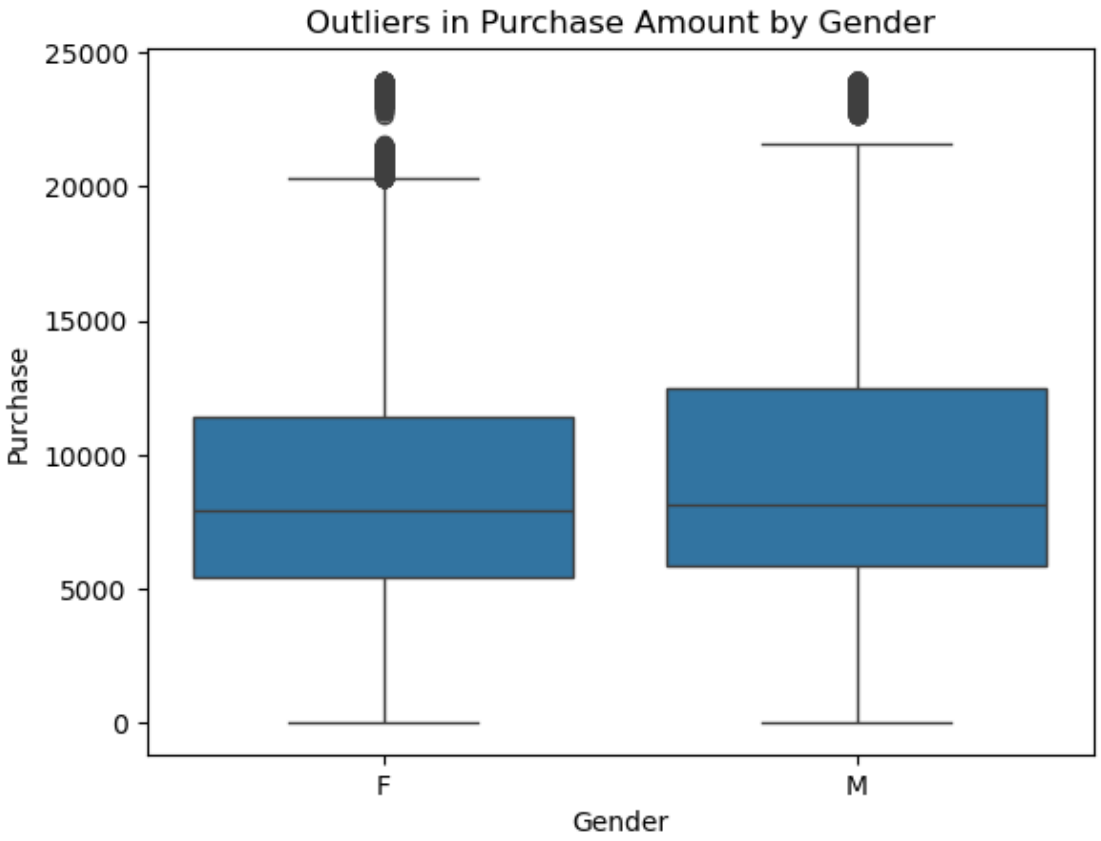
[9]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4	0	8	7969

[11]:
```
# Describe dataset for numerical summary
df.describe()

# Check for categorical columns and convert them if necessary
categorical_columns = ['Gender', 'Age', 'City_Category', 'Marital_Status']
for col in categorical_columns:
    df[col] = df[col].astype('category')
```

[13]:
```
# Check for null values
print(df.isnull().sum())

# Detect outliers in 'Purchase' column
sns.boxplot(x='Gender', y='Purchase', data=df)
plt.title("Outliers in Purchase Amount by Gender")
plt.show()

# Calculate mean and median to detect skew
print("Mean Purchase Amount:", df['Purchase'].mean())
print("Median Purchase Amount:", df['Purchase'].median())
```

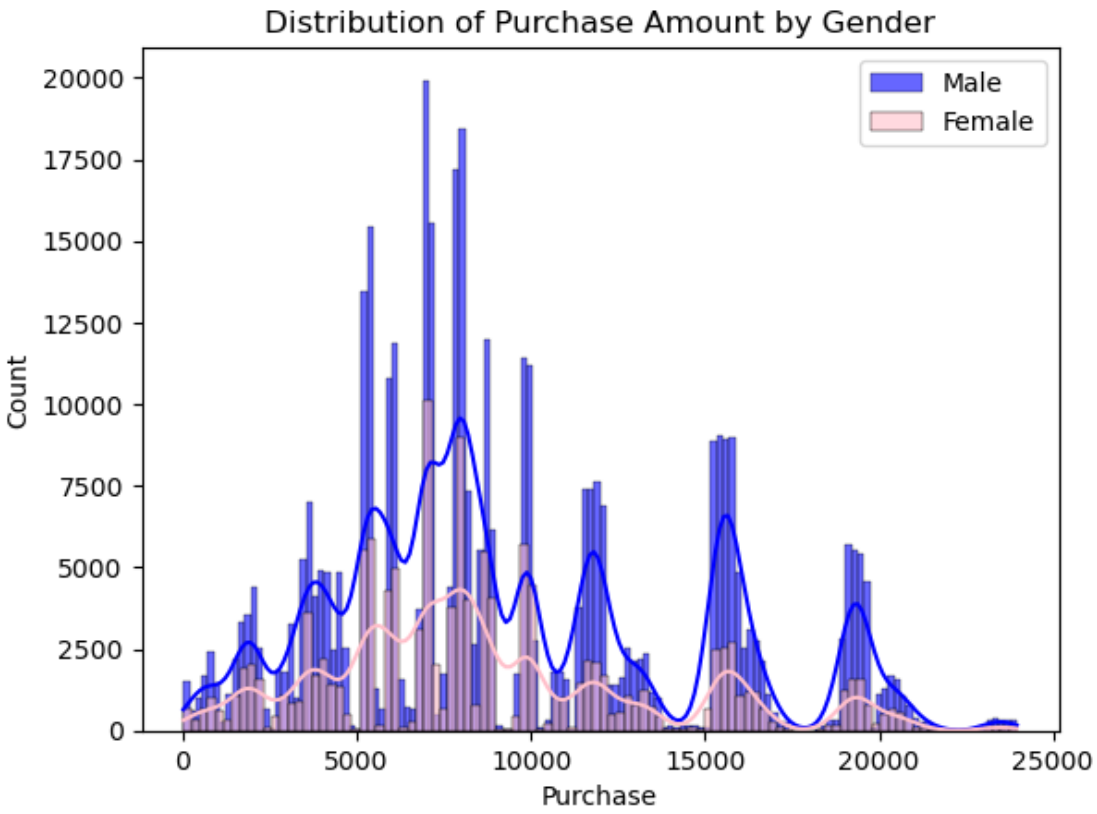
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase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



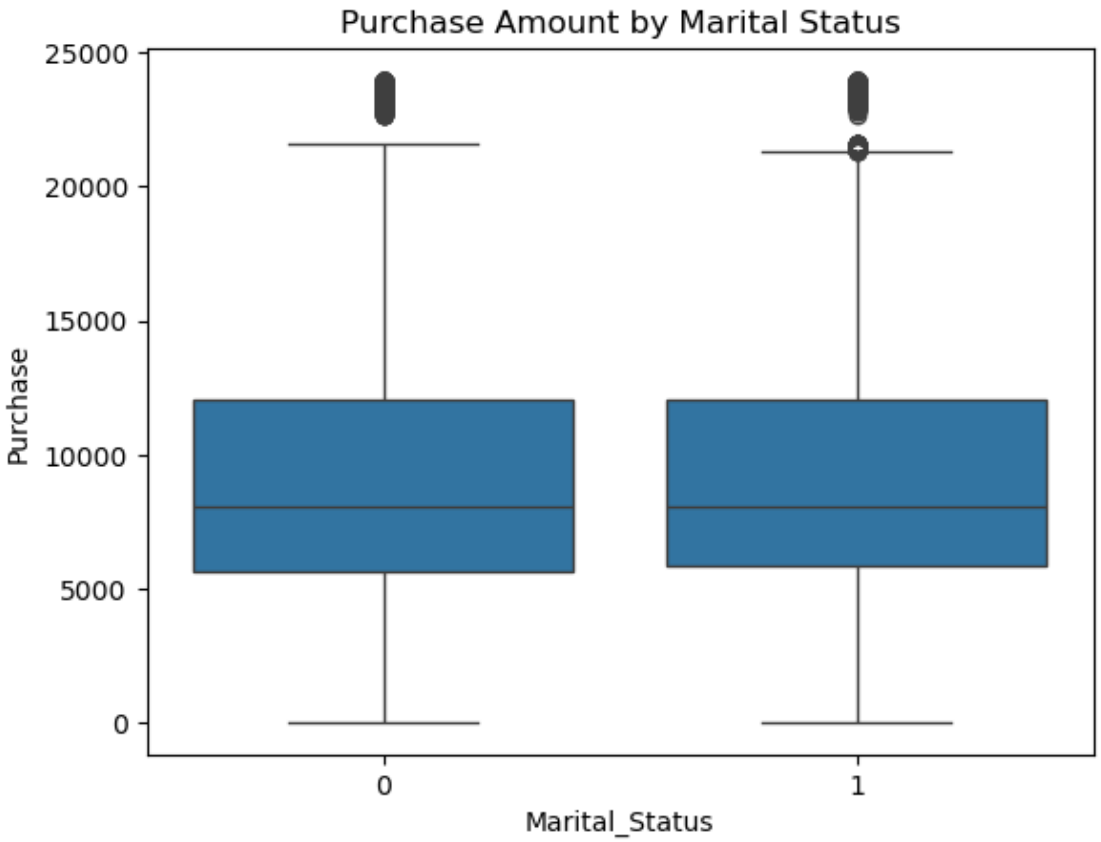

```
Mean Purchase Amount: 9263.968712959126
Median Purchase Amount: 8047.0
```

[15]:
```python
# Separate data by gender and calculate mean purchase
male_purchase = df[df['Gender'] == 'M']['Purchase']
female_purchase = df[df['Gender'] == 'F']['Purchase']

print("Average Purchase Amount (Male):", male_purchase.mean())
print("Average Purchase Amount (Female):", female_purchase.mean())

# Visualize distribution of purchase amount by gender
sns.histplot(male_purchase, kde=True, color='blue', label='Male', alpha=0.6)
sns.histplot(female_purchase, kde=True, color='pink', label='Female', alpha=0.6)
plt.legend()
plt.title("Distribution of Purchase Amount by Gender")
plt.show()
```

```
Average Purchase Amount (Male): 9437.526040472265
Average Purchase Amount (Female): 8734.565765155476
```



[17]:
```
# Marital Status
sns.boxplot(x='Marital_Status', y='Purchase', data=df)
plt.title("Purchase Amount by Marital Status")
plt.show()

# Age
age_bins = [0, 17, 25, 35, 50, 100]
age_labels = ['0-17', '18-25', '26-35', '36-50', '51+']
df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

sns.boxplot(x='AgeGroup', y='Purchase', data=df)
plt.title("Purchase Amount by Age Group")
plt.show()
```




```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 9
      7 age_bins = [0, 17, 25, 35, 50, 100]
      8 age_labels = ['0-17', '18-25', '26-35', '36-50', '51+']
----> 9 df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)
     11 sns.boxplot(x='AgeGroup', y='Purchase', data=df)
     12 plt.title("Purchase Amount by Age Group")

File F:\anaconda\Lib\site-packages\pandas\core\reshape\tile.py:257, in cut(x,␣
 ↪bins, right, labels, retbins, precision, include_lowest, duplicates, ordered)
    254     if not bins.is_monotonic_increasing:
    255         raise ValueError("bins must increase monotonically.")
--> 257 fac, bins = _bins_to_cuts(
    258     x_idx,
    259     bins,
    260     right=right,
    261     labels=labels,
    262     precision=precision,
    263     include_lowest=include_lowest,
```

```
    264     duplicates=duplicates,
    265     ordered=ordered,
    266 )
    268 return _postprocess_for_cut(fac, bins, retbins, original)

File F:\anaconda\Lib\site-packages\pandas\core\reshape\tile.py:452, in␣
 ↪_bins_to_cuts(x_idx, bins, right, labels, precision, include_lowest,␣
 ↪duplicates, ordered)
    449 side: Literal["left", "right"] = "left" if right else "right"
    451 try:
--> 452     ids = bins.searchsorted(x_idx, side=side)
    453 except TypeError as err:
    454     # e.g. test_datetime_nan_error if bins are DatetimeArray and x_idx
    455     #  is integers
    456     if x_idx.dtype.kind == "m":

File F:\anaconda\Lib\site-packages\pandas\core\base.py:1352, in IndexOpsMixin.
 ↪searchsorted(self, value, side, sorter)
   1348 if not isinstance(values, np.ndarray):
   1349     # Going through EA.searchsorted directly improves performance␣
 ↪GH#38083
   1350     return values.searchsorted(value, side=side, sorter=sorter)
-> 1352 return algorithms.searchsorted(
   1353     values,
   1354     value,
   1355     side=side,
   1356     sorter=sorter,
   1357 )

File F:\anaconda\Lib\site-packages\pandas\core\algorithms.py:1329, in␣
 ↪searchsorted(arr, value, side, sorter)
   1325     arr = ensure_wrapped_if_datetimelike(arr)
   1327 # Argument 1 to "searchsorted" of "ndarray" has incompatible type
   1328 # "Union[NumpyValueArrayLike, ExtensionArray]"; expected␣
 ↪"NumpyValueArrayLike"
-> 1329 return arr.searchsorted(value, side=side, sorter=sorter)

TypeError: '<' not supported between instances of 'int' and 'str'
```

[19]:
```
def calculate_confidence_interval(data, confidence=0.95):
    mean = np.mean(data)
    std_error = np.std(data, ddof=1) / np.sqrt(len(data))
    margin = std_error * norm.ppf((1 + confidence) / 2)
    return mean - margin, mean + margin

# Calculate confidence intervals for male and female purchases
male_ci = calculate_confidence_interval(male_purchase)
```

```
female_ci = calculate_confidence_interval(female_purchase)

print(f"95% Confidence Interval for Male Purchase Amount: {male_ci}")
print(f"95% Confidence Interval for Female Purchase Amount: {female_ci}")
```

```
95% Confidence Interval for Male Purchase Amount: (9422.01944736257,
9453.032633581959)
95% Confidence Interval for Female Purchase Amount: (8709.21154714068,
8759.919983170272)
```

[21]:
```
# Check overlap between male and female confidence intervals
overlap = not (male_ci[1] < female_ci[0] or female_ci[1] < male_ci[0])
print("Do male and female confidence intervals overlap?", overlap)
```

```
Do male and female confidence intervals overlap? False
```

Insights Summary and Recommendations Based on these findings:

High-Spending Groups: Target higher-spending age groups and marital statuses (e.g., married couples, older customers) with promotions or loyalty programs. Gender-Specific Campaigns: If one gender spends more, Walmart could direct relevant promotions to that demographic, especially for Black Friday or holiday sales. Family Promotions: If married customers spend more, introduce family-oriented discounts or bundles to increase sales. These visualizations and insights provide a strong foundation for understanding customer purchase behavior and crafting Walmart's marketing strategy.

Gender-Targeted Marketing: If females are found to spend more, Walmart could target promotional efforts towards women during high-spending periods. Age-Specific Campaigns: If younger customers (e.g., 18-25) are spending less, Walmart could consider special discounts or loyalty programs for this demographic.