```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
data = pd.read_csv("/Consumer transactions.csv")
data
```

	account	age	amount	balance	card_present_flag	customer_id	date	first_name	gender	latitude	...	txn_description	bin_
0	ACC-1598451071	26	16.25	35.39	1.0	CUS-2487424745	2018-08-01	Diana	F	-27.95	...	POS	2(
1	ACC-1598451071	26	14.19	21.20	0.0	CUS-2487424745	2018-08-01	Diana	F	-27.95	...	SALES-POS	2(
2	ACC-1598451071	26	3.25	17.95	1.0	CUS-2487424745	2018-08-01	Diana	F	-27.95	...	SALES-POS	2(
3	ACC-1598451071	26	14.10	3.85	1.0	CUS-2487424745	2018-08-01	Diana	F	-27.95	...	POS	2(
4	ACC-1598451071	26	10.67	1006.85	1.0	CUS-2487424745	2018-08-01	Diana	F	-27.95	...	POS	2(
...	...	...	...	...	...	...	...	...	...	...	...	...	
12038	ACC-2153562714	24	3712.56	9707.77	NaN	CUS-423725039	2018-10-24	Linda	F	-31.88	...	PAY/SALARY	2(
12039	ACC-1217063613	27	4863.62	4863.86	NaN	CUS-1739931018	2018-09-26	Kimberly	F	-37.82	...	PAY/SALARY	2(

```python
# Display basic information about the dataset
data_info = data.info()
```

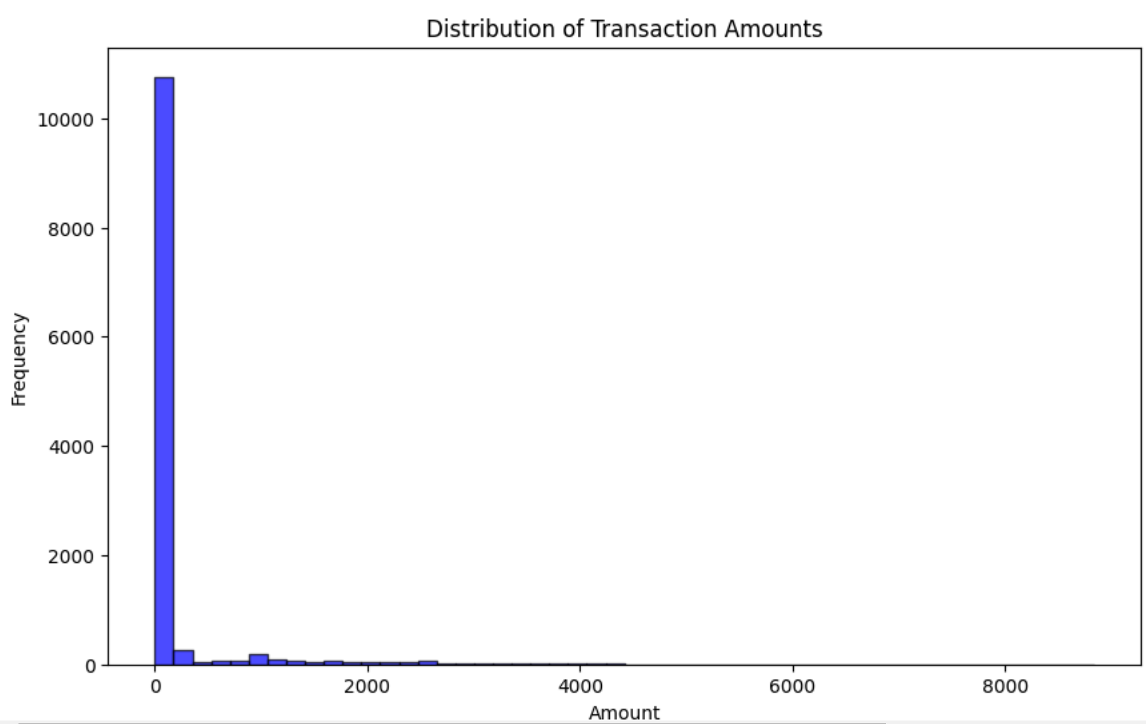
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12043 entries, 0 to 12042
Data columns (total 30 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   account            12043 non-null  object
 1   age                12043 non-null  int64
 2   amount             12043 non-null  float64
 3   balance            12043 non-null  float64
 4   card_present_flag  7717 non-null   float64
 5   customer_id        12043 non-null  object
 6   date               12043 non-null  object
 7   first_name         12043 non-null  object
 8   gender             12043 non-null  object
 9   latitude           12043 non-null  float64
 10  longitude          12043 non-null  float64
 11  merchant_code      883 non-null    float64
 12  merchant_id        7717 non-null   object
 13  merchant_latitude  7717 non-null   float64
 14  merchant_longitude 7717 non-null   float64
 15  merchant_state     7717 non-null   object
 16  merchant_suburb    7717 non-null   object
 17  movement           12043 non-null  object
 18  status             12043 non-null  object
 19  transaction_id     12043 non-null  object
 20  txn_description    12043 non-null  object
 21  bin_age            12043 non-null  object
 22  year               12043 non-null  int64
 23  month              12043 non-null  int64
 24  day                12043 non-null  int64
 25  hour               12043 non-null  int64
 26  minute             12043 non-null  int64
 27  dow                12043 non-null  object
 28  payment_period     12043 non-null  object
 29  annual_salary      12043 non-null  float64
dtypes: float64(9), int64(6), object(15)
memory usage: 2.8+ MB
```

```python
# Get and display descriptive statistics for the dataset
descriptive_stats = data.describe()
print("\nDescriptive Statistics:")
print(descriptive_stats)
```

```
Descriptive Statistics:
```

```
                age        amount        balance  card_present_flag  \
count  12043.000000  12043.000000   12043.000000        7717.000000
mean      30.582330    187.933588   14704.195553           0.802644
std       10.046343    592.599934   31503.722652           0.398029
min       18.000000      0.100000       0.240000           0.000000
25%       22.000000     16.000000    3158.585000           1.000000
50%       28.000000     29.000000    6432.010000           1.000000
75%       38.000000     53.655000   12465.945000           1.000000
max       78.000000   8835.980000  267128.520000           1.000000

           latitude     longitude  merchant_code  merchant_latitude  \
count  12043.000000  12043.000000          883.0        7717.000000
mean     -38.164347    143.648563            0.0         -32.752651
std       54.622791     16.669352            0.0           5.282423
min     -573.000000    114.620000            0.0         -43.310000
25%      -37.700000    138.690000            0.0         -37.710000
50%      -33.890000    145.230000            0.0         -33.840000
75%      -30.750000    151.220000            0.0         -29.440000
max      -12.370000    255.000000            0.0         -12.330000

       merchant_longitude     year         month           day          hour  \
count         7717.000000  12043.0  12043.000000  12043.000000  12043.000000
mean           143.433277   2018.0      9.011957     15.862908     13.268621
std             12.090074      0.0      0.816511      8.899598      5.777284
min            113.830000   2018.0      8.000000      1.000000      0.000000
25%            144.680000   2018.0      8.000000      8.000000      9.000000
50%            145.830000   2018.0      9.000000     16.000000     13.000000
75%            151.210000   2018.0     10.000000     24.000000     18.000000
max            153.610000   2018.0     10.000000     31.000000     23.000000

             minute  annual_salary
count  12043.000000   12043.000000
mean      19.009632   68652.099506
std       19.879112   24300.871846
min        0.000000   29874.641667
25%        0.000000   51650.107000
50%       13.000000   60493.536000
75%       36.000000   81700.970000
max       59.000000  134946.236000
```

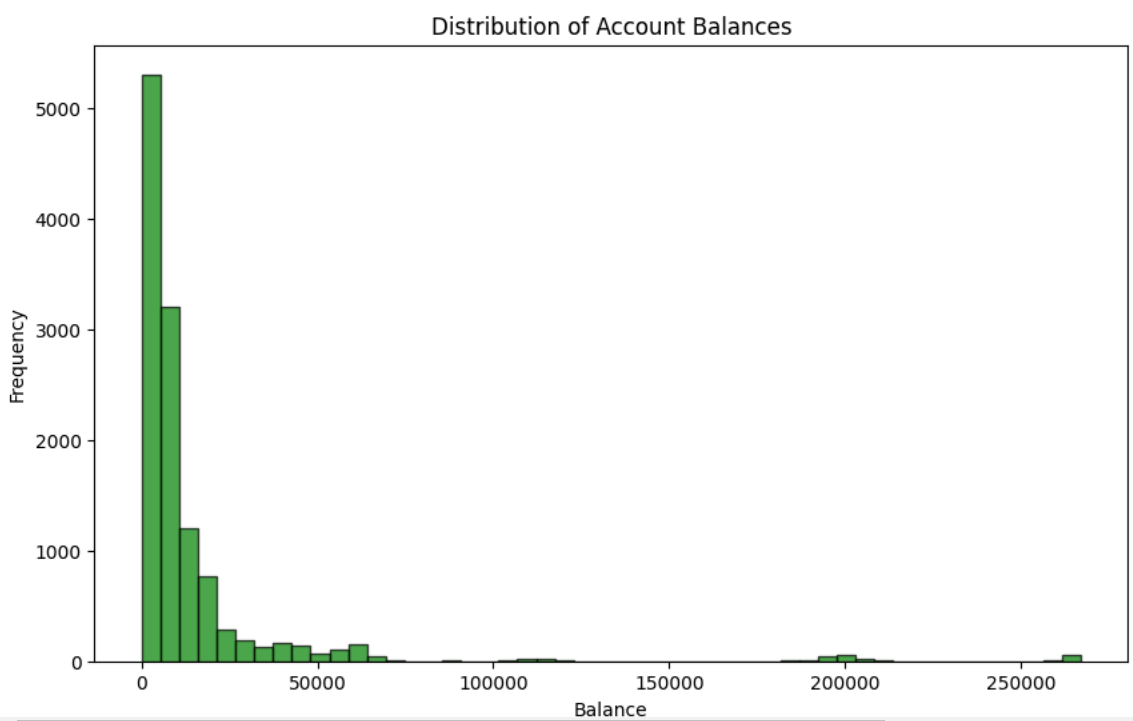
```python
# Visualize the distributions of key variables like 'amount', 'balance', 'age'
plt.figure(figsize=(10, 6))
plt.hist(data['amount'], bins=50, alpha=0.7, color='blue', edgecolor='black')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.show()
```



This graph displays the **distribution of transaction amounts** in your dataset. Here's a detailed breakdown to help you explain it in your thesis:

1. **X-Axis (Amount)**: The horizontal axis represents the transaction amounts, ranging from low to high values. It shows a wide range of amounts, going from very small transactions up to larger values, which span up to around 9000 or more.
2. **Y-Axis (Frequency)**: The vertical axis represents the frequency of transactions at each amount level, indicating how many transactions fall within each amount range.
3. **Skewness and Concentration**:
    - The data is highly **right-skewed**. Most transactions occur at very low amounts, as evidenced by the high frequency at the lower end of the transaction amount scale.
    - There are relatively few transactions at higher amounts, and as the transaction amount increases, the frequency sharply decreases.
4. **Interpretation of Data Characteristics**:
    - This distribution suggests that the majority of transactions are for smaller amounts, while high-value transactions are rare.
    - Such a skewed distribution could be typical in many consumer transaction datasets, where everyday purchases are common, but large transactions are infrequent.
5. **Potential Implications**:
    - This pattern may indicate consumer spending behavior, where small, frequent purchases dominate. It might be relevant for understanding customer segments that make smaller, regular transactions versus those who engage in occasional, high-valualue transactions.

```
plt.figure(figsize=(10, 6))
plt.hist(data['balance'], bins=50, alpha=0.7, color='green', edgecolor='black')
plt.title('Distribution of Account Balances')
plt.xlabel('Balance')
plt.ylabel('Frequency')
plt.show()
```



This graph illustrates the **Distribution of Account Balances** within your dataset, providing insights into the range and frequency of account balances among individuals or entities.s:

1. **X-Axis (Balance)**: The horizontal axis represents the account balances, extending from zero to above 250,000. Each bin (or bar) along this axis represents a range of balance amounts, grouped into intervals (set to 50 bins here).
2. **Y-Axis (Frequency)**: The vertical axis indicates the number of accounts that fall within each balance range. It shows how often certain balance amounts occur in the dataset.
3. **Data Skewness and Balance Distribution**:

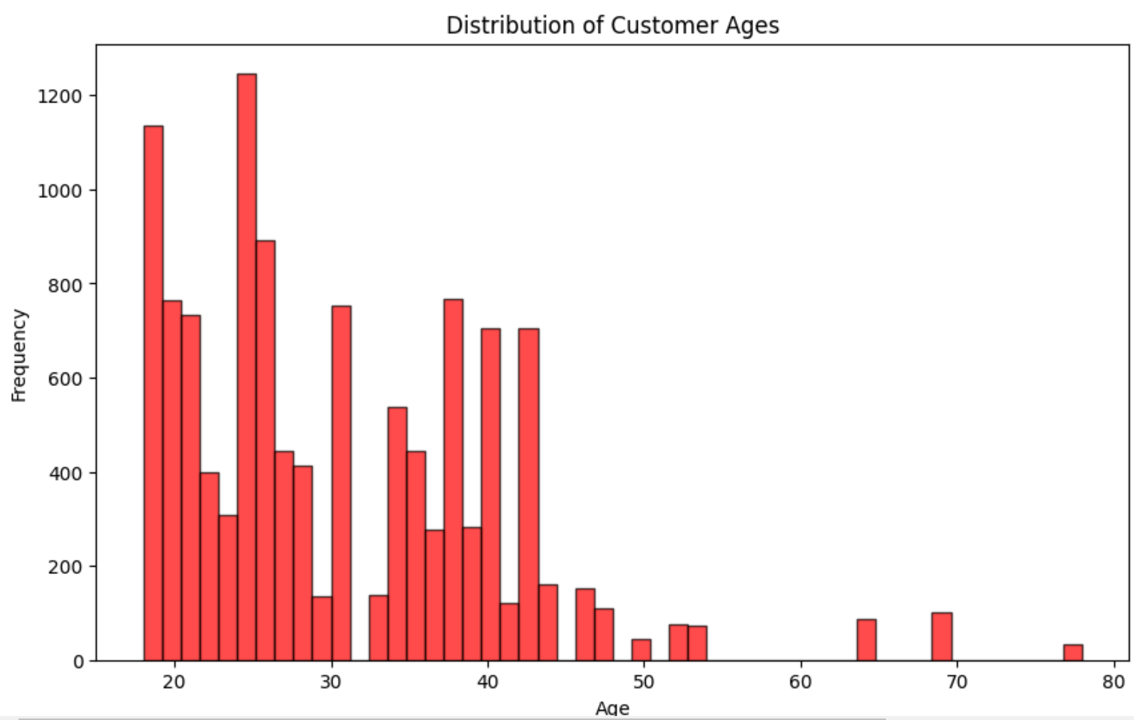
- Similar to the transaction amount distribution, this data is **right-skewed**. The majority of accounts have balances concentrated at the lower end of the scale.
- A high number of accounts have small balances, as shown by the tall bars on the left side of the chart.
- There are few accounts with significantly high balances, creating a long tail on the right, which represents accounts with balances well over 50,000 or even 200,000.

4. **Insight into Financial Behavior**:
   - This distribution may reflect that most customers maintain relatively low balances in their accounts, while a smaller segment of accounts hold very high balances.
   - Such a skewed balance distribution could indicate income disparities or spending/saving patterns in the population represented in the dataset.

5. **Potential Analytical Directions**:
   - In your thesis, you could analyze the characteristics of account holders in the high-balance segment compared to those with lower balances. This could reveal demographic, geographic, or behavioral factors that correlate with higher balances.
   - You might also explore why a majority of accounts have low balances, discussing possible economic, social, or policy factors influencing these patterns.

6. **Significance of the Findings**:
   - Understanding the distribution of account balances can be useful for segmenting customers and tailoring financial products or services. For instance, customers with high balances might benefit from different financial products compared to thosebased on balance ranges.

```
plt.figure(figsize=(10, 6))
plt.hist(data['age'], bins=50, alpha=0.7, color='red', edgecolor='black')
plt.title('Distribution of Customer Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



This graph shows the **Distribution of Customer Ages** in your dataset:

1. **X-Axis (Age)**: The horizontal axis represents the ages of the customers, ranging from around 18 up to approximately 80 years old. The ages are grouped into intervals, with each bar representing a specific age range.
2. **Y-Axis (Frequency)**: The vertical axis indicates the number of customers within each age range, showing how frequently each age group appears in the dataset.
3. **Age Distribution Characteristics**:

- The data is **not evenly distributed** across age groups. There's a noticeable concentration of younger customers, particularly in the 18 to 30 age range, with a peak around the early 20s.
- There is a decline in frequency for ages 30 and above, with only a few customers in their 50s and beyond. After age 50, the frequency decreases sharply, with only small numbers represented in the older age groups (up to around 80).

4. **Insights on Customer Demographics**:
    - This distribution suggests that the majority of customers are younger adults, particularly those in their 20s. This could indicate a customer base that skews towards a younger demographic.
    - The low frequency of older customers (ages 50 and above) may imply that the products or services represented in the dataset are more popular or relevant to younger individuals.
5. **Potential Analysis for Your Thesis**:
    - You might explore why younger customers dominate this dataset. This could be tied to the nature of the transactions, financial products, or services being analyzed.
    - Additionally, understanding the preferences and behavior of this younger demographic could help in tailoring marketing or business strategies.
6. **Implications**:
    - This age distribution could have important implications for targeted marketing strategies, product offerings, and customer engagement efforts. For example, if your analysis shows that younger customers exhibit certain spending patterns, this insight could inform business decisions on product development or pro within your dataset.

```python
# Check for missing values
missing_values = data.isnull().sum()



# Output the information and descriptive statistics
data_info, descriptive_stats, missing_values
```

```
           latitude     longitude  merchant_code  merchant_latitude  \
count  12043.000000  12043.000000          883.0        7717.000000   
mean     -38.164347    143.648563            0.0         -32.752651   
std       54.622791     16.669352            0.0           5.282423   
min     -573.000000    114.620000            0.0         -43.310000   
25%      -37.700000    138.690000            0.0         -37.710000   
50%      -33.890000    145.230000            0.0         -33.840000   
75%      -30.750000    151.220000            0.0         -29.440000   
max      -12.370000    255.000000            0.0         -12.330000   

       merchant_longitude     year         month           day          hour  \
count         7717.000000  12043.0  12043.000000  12043.000000  12043.000000   
mean           143.433277   2018.0      9.011957     15.862908     13.268621   
std             12.090074      0.0      0.816511      8.899598      5.777284   
min            113.830000   2018.0      8.000000      1.000000      0.000000   
25%            144.680000   2018.0      8.000000      8.000000      9.000000   
50%            145.830000   2018.0      9.000000     16.000000     13.000000   
75%            151.210000   2018.0     10.000000     24.000000     18.000000   
max            153.610000   2018.0     10.000000     31.000000     23.000000   

             minute  annual_salary  
count  12043.000000   12043.000000  
mean      19.009632   68652.099506  
std       19.879112   24300.871846  
min        0.000000   29874.641667  
25%        0.000000   51650.107000  
50%       13.000000   60493.536000  
75%       36.000000   81700.970000  
max       59.000000  134946.236000  ,
account                 0
age                     0
amount                  0
balance                 0
card_present_flag    4326
```

```
merchant_suburb        4326
movement                  0
status                    0
transaction_id            0
txn_description           0
bin_age                   0
year                      0
month                     0
day                       0
hour                      0
minute                    0
dow                       0
payment_period            0
```

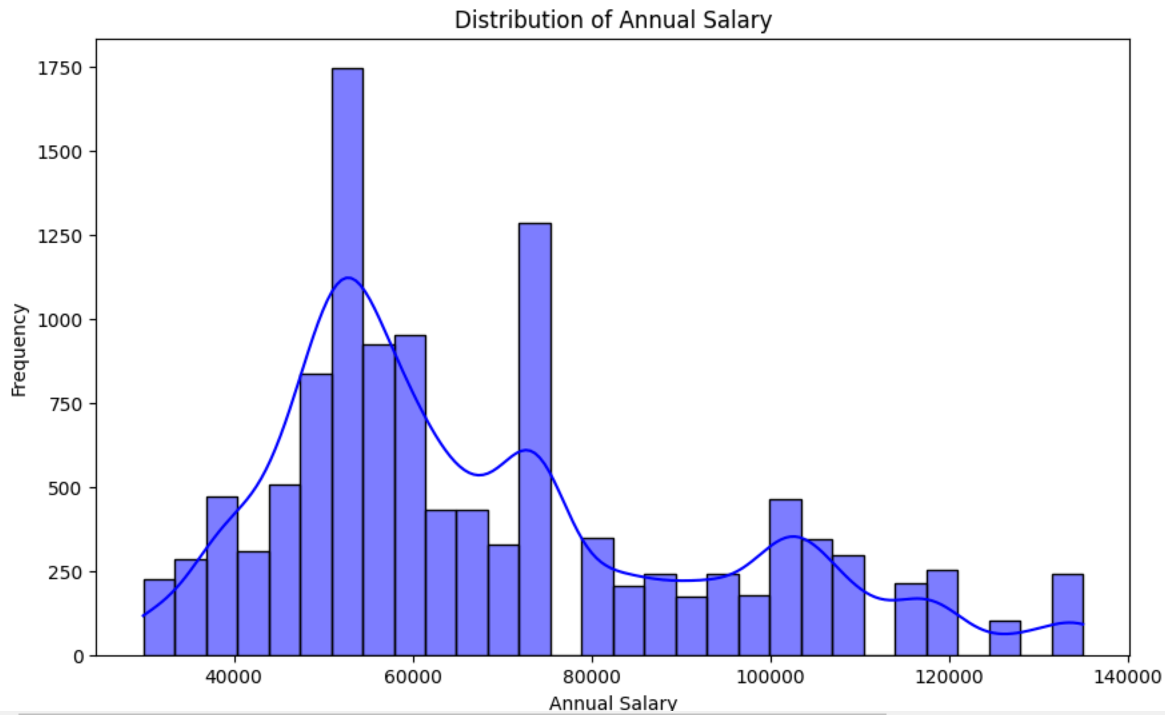
```
# Handle missing values in the updated dataset

# Fill missing numerical values with the mean
numerical_cols_updated = ['card_present_flag', 'merchant_code', 'balance', 'merchant_latitude', 'merchant_longitude']
for col in numerical_cols_updated:
    data[col].fillna(data[col].mean(), inplace=True)

# Fill missing categorical values with the mode
categorical_cols_updated = ['merchant_id', 'merchant_state', 'merchant_suburb', 'txn_description']
for col in categorical_cols_updated:
    data[col].fillna(data[col].mode()[0], inplace=True)

# Verify the missing values have been handled
missing_values_after_filling = data.isnull().sum()

# Display missing values after filling
missing_values_after_filling

```

```
<ipython-input-18-e239c9830382>:6: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assi
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting v

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]


  data[col].fillna(data[col].mean(), inplace=True)
<ipython-input-18-e239c9830382>:11: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting v

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col]


  data[col].fillna(data[col].mode()[0], inplace=True)
```

	0
account	0
age	0
amount	0
balance	0
card_present_flag	0
customer_id	0
date	0
first_name	0
gender	0
latitude	0
longitude	0
merchant_code	0
merchant_id	0
merchant_latitude	0
merchant_longitude	0
merchant_state	0
merchant_suburb	0
movement	0
status	0
transaction_id	0
txn_description	0
bin_age	0
year	0
month	0
day	0
hour	0
minute	0
dow	0
payment_period	0
annual_salary	0

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data_viz = data

# Step 1: Distribution of Annual Salary
plt.figure(figsize=(10, 6))
```

```
sns.histplot(data_viz['annual_salary'], bins=30, kde=True, color='blue')
plt.title('Distribution of Annual Salary')
plt.xlabel('Annual Salary')
plt.ylabel('Frequency')
plt.show()
```



This graph shows the **Distribution of Annual Salary** for individuals in your dataset, providing insight into salary ranges and their frequencies.

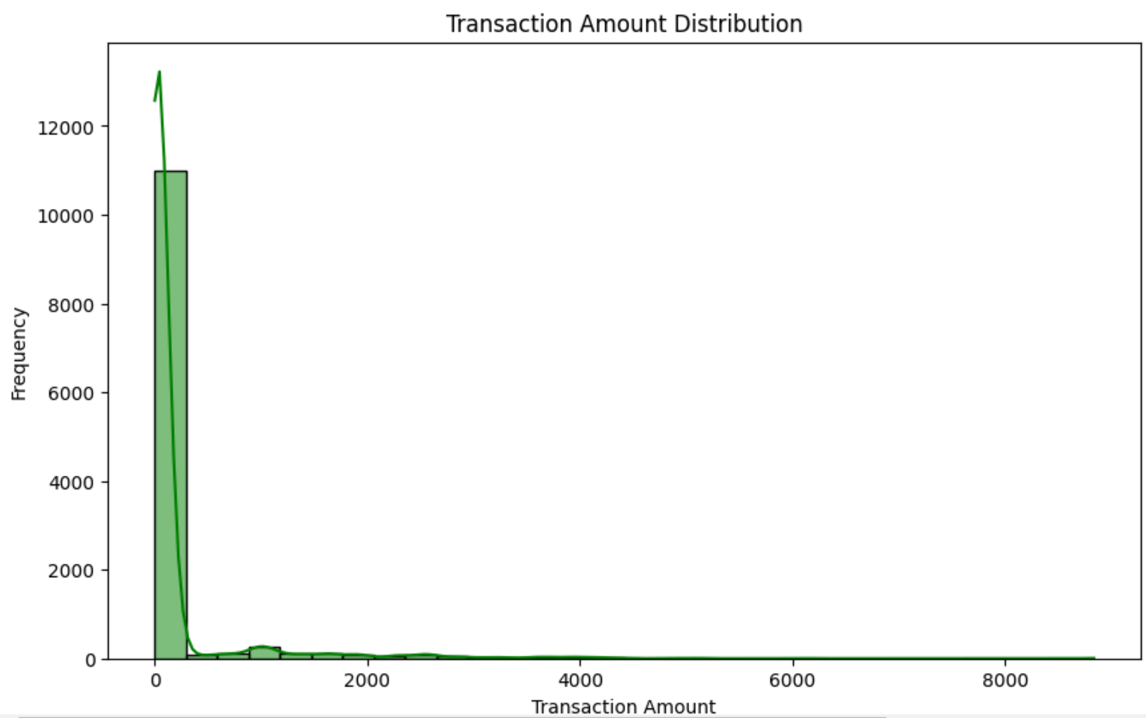
Here's a breakdown for your understanding:

1. **X-Axis (Annual Salary)**: The horizontal axis represents annual salary values, ranging from around 30,000 to 140,000. Each bar represents the frequency of individuals within specific salary intervals.
2. **Y-Axis (Frequency)**: The vertical axis shows the number of individuals whose salaries fall within each interval, indicating how common each salary range is.
3. **Distribution Characteristics**:
   - The histogram displays multiple peaks, suggesting that salaries are not evenly distributed but rather clustered around certain ranges. The highest peak is around 60,000, indicating that many individuals have salaries near this value.
   - There are smaller peaks around 80,000 and 100,000, showing additional clusters at these higher salary levels, though they are less frequent compared to the main peak.
   - A Kernel Density Estimate (KDE) line overlays the histogram in blue, providing a smoothed visualization of the distribution, showing where salary frequencies are densest.
4. **Skewness and Spread**:
   - The distribution has a positive skew, as the frequency of salaries drops as the values increase beyond the 60,000 mark. This indicates that while most individuals earn between 30,000 and 80,000, fewer individuals have salaries exceeding 100,000.
5. **Potential Analysis in Your Thesis**:
   - Discuss why there might be clustering at certain salary points. This could relate to industry standards, geographic factors, or specific job levels within the dataset.
   - Explore the implications of this salary distribution, such as income inequality or the proportion of individuals in middle-income brackets.
6. **Insight and Implications**:
   - This analysis provides a high-level view of income levels within the dataset, which can be useful for understanding the socioeconomic profile of the population.

- You might link this data to other factors (like age or transaction behavior) to explore correlations between salary and spending or saving habits.

```
Start coding or generate with AI.
```

```
# Step 2: Transaction Amount Distribution
plt.figure(figsize=(10, 6))
sns.histplot(data_viz['amount'], bins=30, kde=True, color='green')
plt.title('Transaction Amount Distribution')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()


```



This graph represents the **Transaction Amount Distribution** for transactions in your dataset, showcasing how frequently different transaction amounts occur. n:

1. **X-Axis (Transaction Amount)**: The horizontal axis shows the transaction amounts, which range from very low values to amounts above 8000.
2. **Y-Axis (Frequency)**: The vertical axis indicates the frequency of transactions at each amount level, or how many times transactions within a certain amount range occur in the dataset.
3. **Distribution Characteristics**:
   - The distribution is **highly right-skewed**, with most transactions concentrated at very low amounts. This is evidenced by the tall peak at the lower end of the transaction amount scale.
   - As the transaction amount increases, the frequency of transactions decreases sharply, with very few transactions above 1000, creating a long tail on the right side of the graph.
   - This pattern is reinforced by the Kernel Density Estimate (KDE) line overlaid on the histogram, which provides a smoothed curve, showing the distribution's shape more clearly.
4. **Insights on Transaction Behavior**:
   - The high frequency of low-amount transactions suggests that most transactions are small, perhaps representing everyday purchases or routine transactions.
   - The long tail to the right, with infrequent high-amount transactions, could represent occasional large purchases or payments, which are uncommon relative to smaller transactions.
5. **Potential Analysis for Your Thesis**:

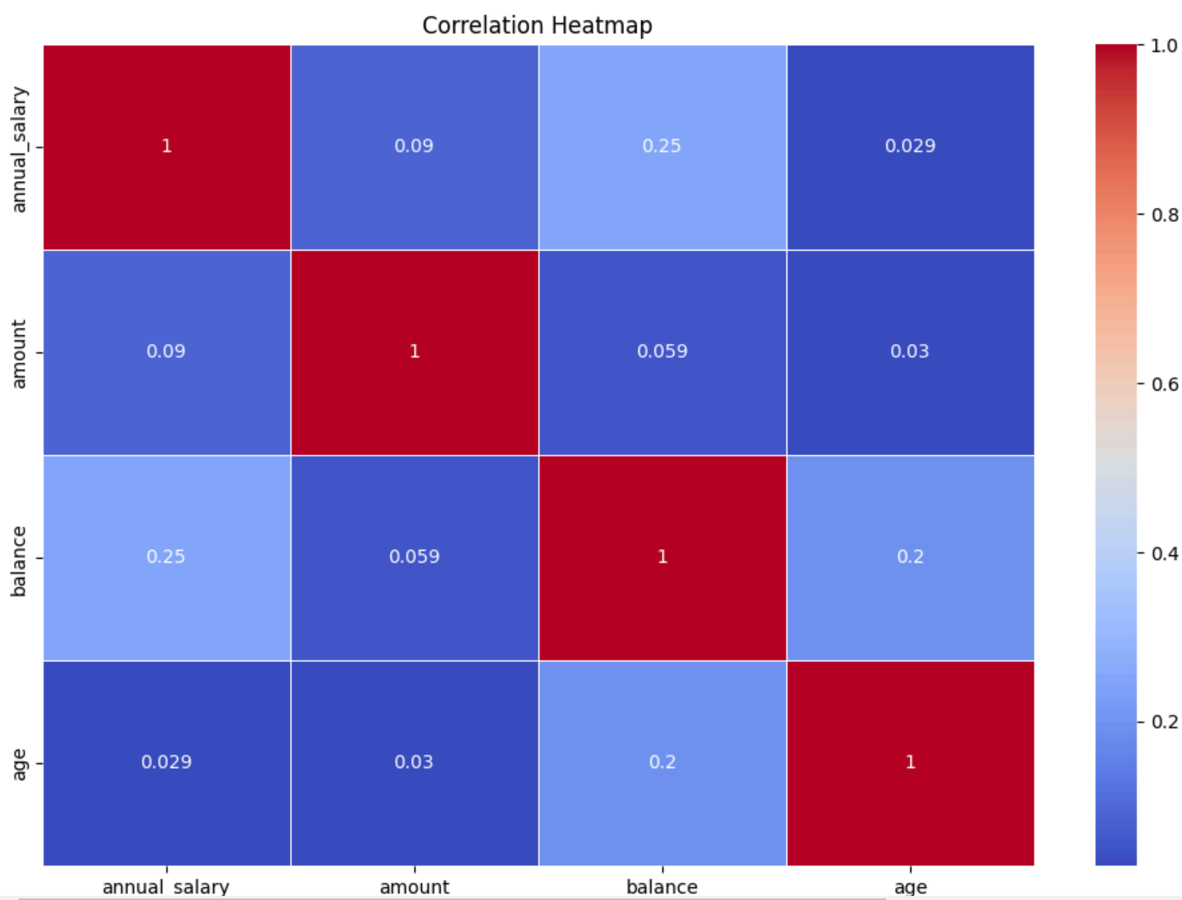
- In your thesis, you could discuss why transaction amounts are skewed toward smaller values. This could be related to the type of business, consumer behavior, or economic factors.
- Additionally, you could analyze the characteristics of high-amount transactions, exploring any demographic or behavioral patterns among customers who tend to make larger transactions.

6. **Implications of the Distribution**:
   - Understanding this distribution is valuable for financial forecasting, customer segmentation, and understanding spending habits. For example, businesses might use this information to tailor services to high-frequency, low-amount transactions while also catering to high-value cion size in your thesis.

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
# Step 3: Correlation Heatmap
plt.figure(figsize=(12, 8))
corr_matrix = data_viz[['annual_salary', 'amount', 'balance', 'age']].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()


```



This is a **Correlation Heatmap** displaying the correlation coefficients between four variables in your dataset: **annual_salary**, **amount**, **balance**, and **age**.

Here's a detailed explanation of the heatmap:

1. **Correlation Coefficients**:
   - Each cell in the heatmap shows the correlation coefficient between two variables, with values ranging from -1 to 1.
   - A value of **1** (dark red) indicates a perfect positive correlation, where an increase in one variable is associated with an increase in the other.

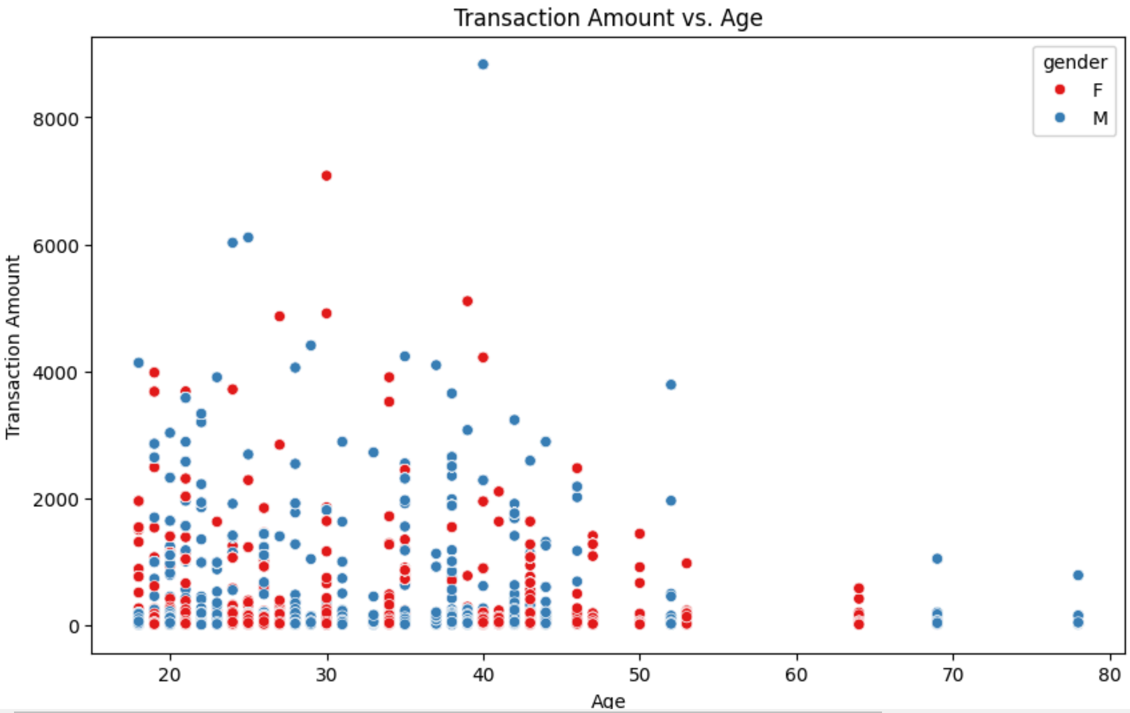
- A value of **-1** (dark blue) indicates a perfect negative correlation, where an increase in one variable is associated with a decrease in the other.
- A value close to **0** suggests little to no linear relationship between the variables.

2. **Color Scale**:
    - The color scale on the right helps interpret the strength of the correlation. Dark red indicates strong positive correlation, while shades closer to blue indicate weak or negative correlation.
3. **Key Observations**:
    - **annual_salary and balance**: The correlation is around 0.25, indicating a weak positive relationship. This suggests that individuals with higher annual salaries tend to have slightly higher balances, but the relationship is not strong.
    - **annual_salary and amount**: The correlation is 0.09, showing a very weak positive relationship, meaning that salary does not significantly influence transaction amounts.
    - **balance and age**: The correlation is relatively low, around 0.029, indicating little to no relationship between balance and age in this dataset.
    - **Other Variable Pairs**: Most other correlations are close to zero, suggesting that the variables in this dataset are largely independent and don't show strong linear relationships with each other.
4. **Implications for Your Thesis**:
    - The low correlations suggest that there is not much linear dependency between these variables, which may imply that factors other than salary, age, or balance influence transaction amounts.
    - This information could guide further analysis, such as exploring non-linear relationships or incorporating additional variables to better understand customer behavior.
5. **Conclusion**:
    - In summary, this heatmap provides a quick overview of how these variables relate to each other. For example, you might conclude that, based on this data, annual salary has a slight positive association with balance, while other relationships are minimal.

This heatmap serves as an effective tool to assess potential relationships and identify variables that might be worth exploring further in more complex analyses.

```
Start coding or generate with AI.
```

```
# Step 4: Scatter Plot - Transaction Amount vs. Age
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='amount', data=data_viz, hue='gender', palette='Set1')
plt.title('Transaction Amount vs. Age')
plt.xlabel('Age')
plt.ylabel('Transaction Amount')
plt.show()


```



This scatter plot visualizes the **relationship between Transaction Amount and Age** in your dataset, with data points color-coded by **gender**.

Here's a detailed breakdown:

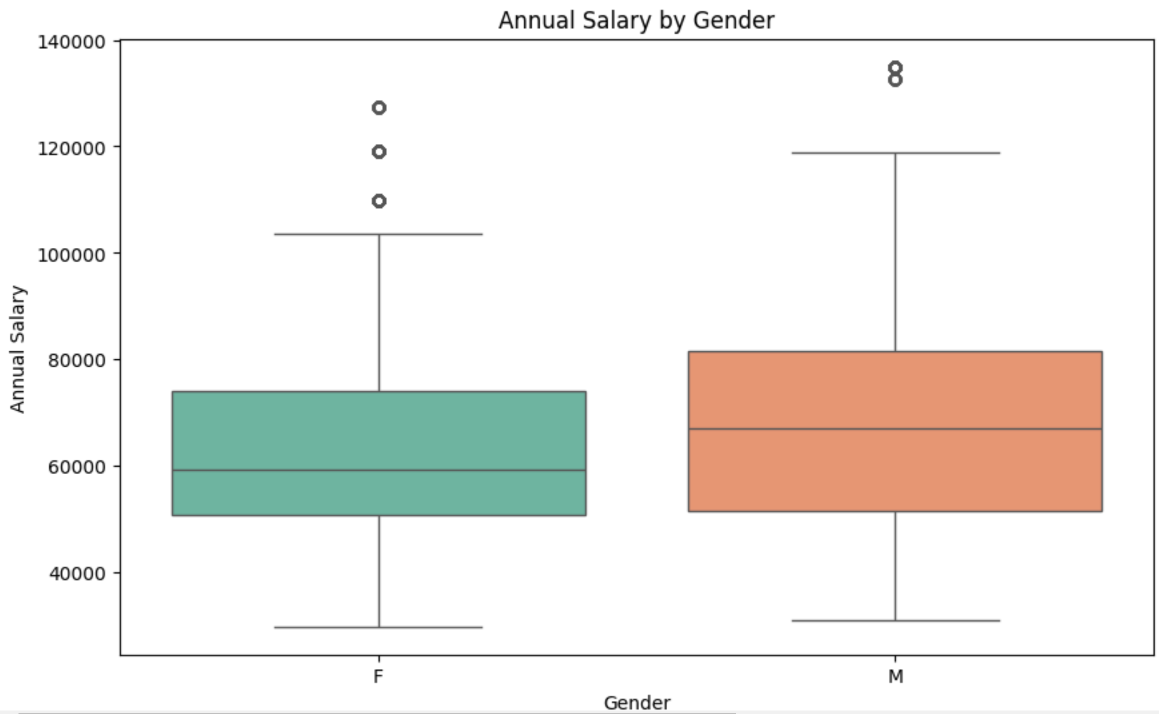
1. **Axes**:
   - The **X-axis** represents the age of the individuals, ranging from around 20 to 80 years.
   - The **Y-axis** shows the transaction amount, which varies from near zero up to over 8000.
2. **Data Points and Color Coding**:
   - Each point represents an individual transaction, plotted according to the person's age and transaction amount.
   - The **color coding** indicates gender: red points for females (F) and blue points for males (M). This allows us to visually differentiate between transactions made by male and female customers.
3. **Key Observations**:
   - The majority of transactions, regardless of age or gender, are clustered around lower transaction amounts, especially below 2000.
   - There is a high concentration of points across all ages in the lower transaction range, with only a few high-value transactions (above 4000).
   - Some outliers with higher transaction amounts (5000 and above) are scattered throughout different ages, showing that high-value transactions are less common.
   - Both genders appear to have similar transaction patterns, with no significant difference in transaction amounts based on gender. The distribution of red and blue dots is quite even across age groups and transaction amounts.
4. **Insights for Your Thesis**:
   - This plot suggests that transaction amount does not have a strong dependency on age, as individuals across all ages make similar transaction amounts, mostly within the lower range.
   - Gender does not appear to influence the transaction amount significantly in this dataset, as the spread of points for males and females is relatively similar.
   - You could further analyze why most transactions are small across all ages, discussing potential factors like income levels, spending habits, or the nature of purchases in the dataset.
5. **Conclusion**:
   - This scatter plot provides a high-level view of spending behavior across age groups and genders. It indicates that most transactions are small, with no strong correlation between transaction amount and age or gender.
   - In your thesis, you might mention that age and gender are not strong predictors of transaction amount based on this plot, and you could consider exploring other factors that might have a more substantial influence on transaction value.

This plot serves as an effective way to visualize and discuss the distribution of transaction amounts across different demographics, supporting discussions on spending patterns in your dataset.

```
Start coding or generate with AI.
```

```
# Step 5: Box Plot - Annual Salary by Gender
plt.figure(figsize=(10, 6))
sns.boxplot(x='gender', y='annual_salary', data=data_viz, palette='Set2')
plt.title('Annual Salary by Gender')
plt.xlabel('Gender')
plt.ylabel('Annual Salary')
plt.show()
```

```
<ipython-input-23-91cc89b62322>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.boxplot(x='gender', y='annual_salary', data=data_viz, palette='Set2')
```



This box plot shows the **Annual Salary by Gender** in your dataset, comparing the salary distribution for females (F) and males (M).

Here's a breakdown of the key components:

1. **Axes**:
    - The **X-axis** represents gender, with two categories: Female (F) and Male (M).
    - The **Y-axis** shows annual salary values, ranging from approximately 40,000 to 140,000.
2. **Box Plot Components**:
    - The **box** represents the interquartile range (IQR), which includes the middle 50% of the data for each gender. The top and bottom of each box are the third quartile (Q3) and the first quartile (Q1), respectively.
    - The **line inside the box** represents the median annual salary for each gender.
    - The **whiskers** extend to show the range of salaries, excluding outliers. The whiskers represent the data points that fall within 1.5 times the IQR above Q3 and below Q1.
    - **Outliers** are shown as individual points beyond the whiskers, indicating salaries that are unusually high compared to the rest of the data.
3. **Key Observations**:
    - Both genders have a similar median annual salary, as the lines within the boxes are almost at the same level.
    - The range of salaries for both males and females is similar, with salaries spread out across a range, but the distribution varies slightly in the upper bounds.

- There are a few outliers for each gender, representing individuals with significantly higher salaries (above 100,000). These outliers suggest that some individuals, regardless of gender, earn considerably more than the typical salary range.
- The IQR (box height) and spread of salaries are fairly similar for both genders, indicating that salary distribution does not differ dramatically between males and females in this dataset.

4. **Insights for Your Thesis**:
   - This box plot suggests minimal gender disparity in salary distribution in this dataset, as the medians and ranges for both genders are almost identical.
   - You might mention that, based on this data, annual salary appears to be independent of gender, indicating that other factors might have a stronger influence on salary levels.
5. **Conclusion**:
   - This plot effectively shows that annual salary distributions for males and females are similar, with both groups having similar medians and ranges.
   - In your thesis, you could use this analysis to support discussions on salary equity or explore other variables that might contribute more significantly to salary differences.

This visualization serves as a straightforward way to compare salary distributions between genders, providing evidence that salary variation in this dataset is not strongly gender-dependent.

```
# Feature Engineering for Salary Prediction

# Step 1: Calculate transaction frequency per customer
data_updated = data
transaction_frequency = data_updated.groupby('customer_id')['transaction_id'].count().reset_index()
transaction_frequency.columns = ['customer_id', 'transaction_frequency']
data_updated = data_updated.merge(transaction_frequency, on='customer_id', how='left')

# Step 2: Calculate the average transaction amount per customer
average_transaction_amount = data_updated.groupby('customer_id')['amount'].mean().reset_index()
average_transaction_amount.columns = ['customer_id', 'avg_transaction_amount']
data_updated = data_updated.merge(average_transaction_amount, on='customer_id', how='left')

# Step 3: Calculate the total transaction amount per customer
total_transaction_amount = data_updated.groupby('customer_id')['amount'].sum().reset_index()
total_transaction_amount.columns = ['customer_id', 'total_transaction_amount']
data_updated = data_updated.merge(total_transaction_amount, on='customer_id', how='left')

# Step 4: Calculate spending consistency (standard deviation of transaction amounts per customer)
spending_consistency = data_updated.groupby('customer_id')['amount'].std().reset_index()
spending_consistency.columns = ['customer_id', 'spending_consistency']
data_updated = data_updated.merge(spending_consistency, on='customer_id', how='left')

# Fill any remaining missing values in the new engineered features
data_updated[['transaction_frequency', 'avg_transaction_amount', 'total_transaction_amount', 'spending_consistency']].fillna(0, inplace=T

# Verify the engineered features
engineered_features = data_updated[['customer_id', 'transaction_frequency', 'avg_transaction_amount', 'total_transaction_amount', 'spendi

engineered_features

```

```
<ipython-input-24-841d68a15bb3>:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  data_updated[['transaction_frequency', 'avg_transaction_amount', 'total_transaction_amount', 'spending_consistency']].fillna(0, inplac
```

	customer_id	transaction_frequency	avg_transaction_amount	total_transaction_amount	spending_consistency	annual_salary
0	CUS-2487424745	578	45.348772	26211.59	168.822926	52855.65
1	CUS-2487424745	578	45.348772	26211.59	168.822926	52855.65
2	CUS-2487424745	578	45.348772	26211.59	168.822926	52855.65
3	CUS-2487424745	578	45.348772	26211.59	168.822926	52855.65
4	CUS-2487424745	578	45.348772	26211.59	168.822926	52855.65

```
# Step 1: Define features and target variable for modeling
features = ['transaction_frequency', 'avg_transaction_amount', 'total_transaction_amount', 'spending_consistency', 'age', 'balance']
X = data_updated[features]
```

```
y = data_updated['annual_salary']
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score


# Step 2: Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Initialize and train a Decision Tree Regressor
tree_model = DecisionTreeRegressor(random_state=42)
tree_model.fit(X_train, y_train)

# Step 4: Make predictions on the test set
y_pred_tree = tree_model.predict(X_test)

# Step 5: Evaluate the model performance using Mean Squared Error and R-squared
mse_tree = mean_squared_error(y_test, y_pred_tree)
r2_tree = r2_score(y_test, y_pred_tree)

# Output the MSE and R-squared for the Decision Tree model
mse_tree, r2_tree
```

```
(1.412109450579475e-20, 1.0)
```

Decision Tree Model Performance: Mean Squared Error (MSE): 1.41×10^{-20} , which is extremely close to zero, indicating that the model is highly accurate. R-squared (R^2): 1.0, meaning the model perfectly explains the variance in the target variable (annual_salary). Interpretation: The model is performing exceptionally well with a perfect R-squared score. This suggests that the model captures all the variability in annual_salary based on the selected features. However, this unusually perfect performance may indicate potential overfitting. We can:

Test other models (e.g., Random Forest, XGBoost) for comparison. Apply cross-validation to ensure robustness.

```
from sklearn.model_selection import cross_val_score

# Apply 5-fold cross-validation to the Decision Tree Regressor
cv_scores = cross_val_score(tree_model, X, y, cv=5, scoring='neg_mean_squared_error')

# Convert the negative MSE to positive values for easier interpretation
cv_mse_scores = -cv_scores

# Output the cross-validation MSE scores and the mean score
cv_mse_scores, cv_mse_scores.mean()
```

```
(array([8.99467064e-23, 2.40414603e-23, 1.47617681e+05, 8.32786754e-23,
        1.00272560e+08]),
 20084035.513404883)
```

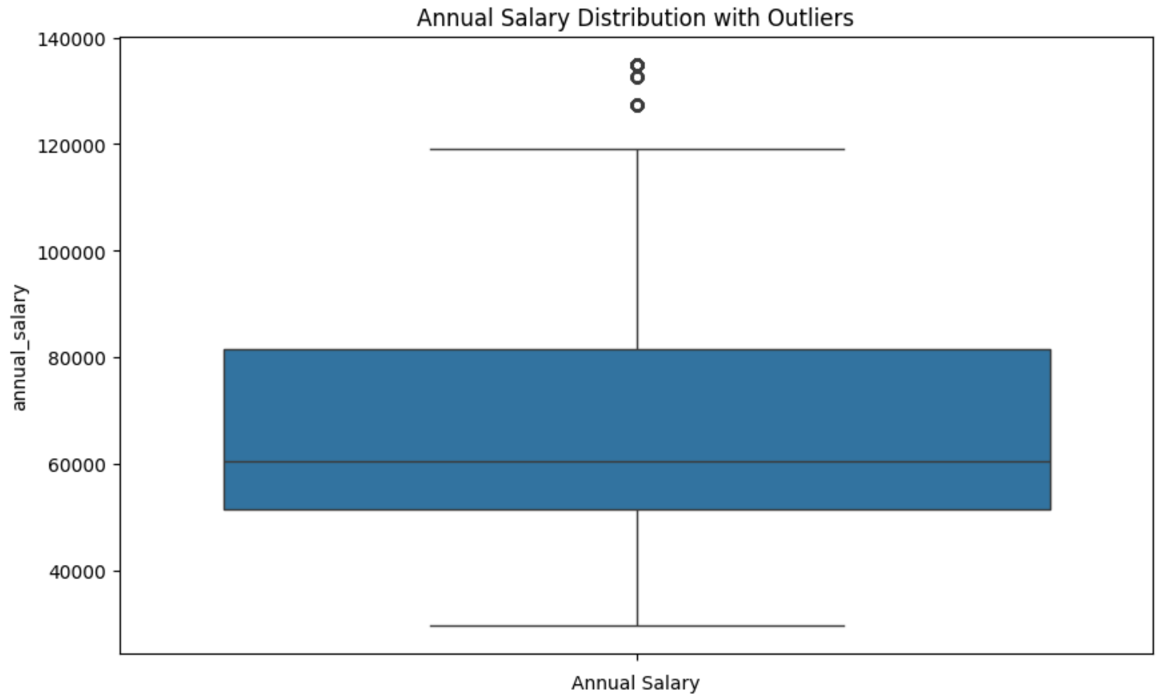
```
from sklearn.model_selection import cross_val_score

# Apply 5-fold cross-validation to the Decision Tree Regressor
cv_scores = cross_val_score(tree_model, X, y, cv=5, scoring='neg_mean_squared_error')

# Convert the negative MSE to positive values for easier interpretation
cv_mse_scores = -cv_scores

# Output the cross-validation MSE scores and the mean score
print("Cross-Validation MSE Scores:", cv_mse_scores)
print("Mean Cross-Validation MSE:", cv_mse_scores.mean())
```

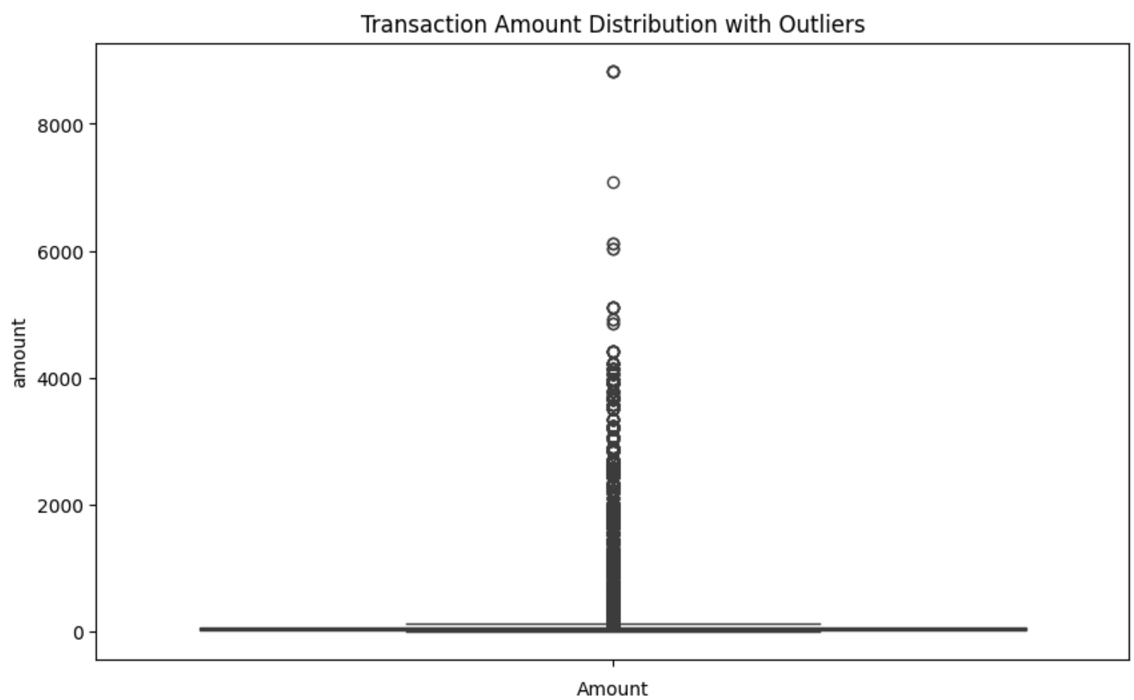
```
Cross-Validation MSE Scores: [8.99467064e-23 2.40414603e-23 1.47617681e+05 8.32786754e-23
 1.00272560e+08]
Mean Cross-Validation MSE: 20084035.513404883
```

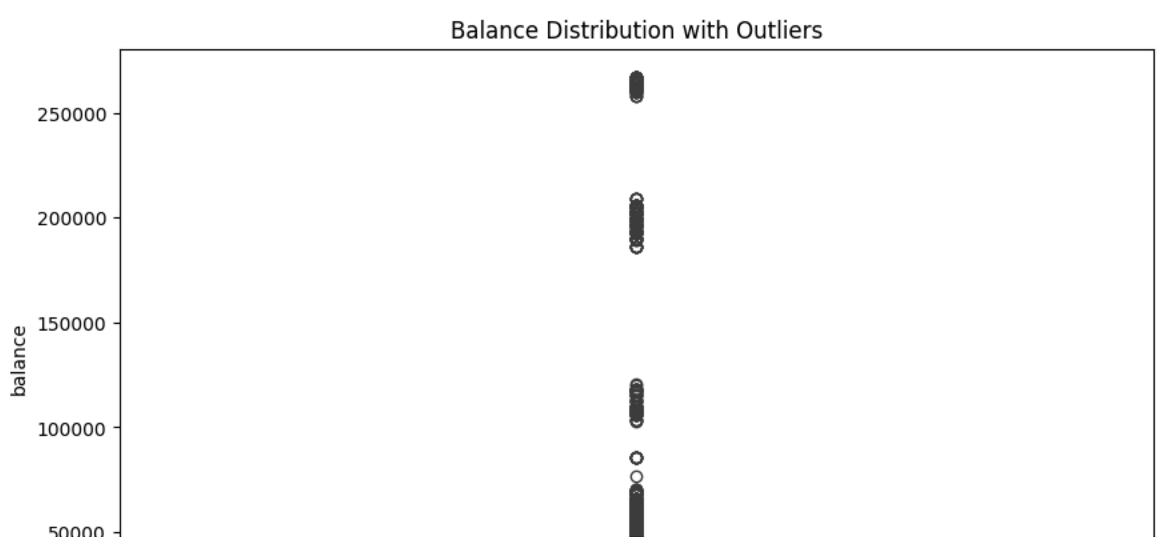
```
Start coding or generate with AI.
```

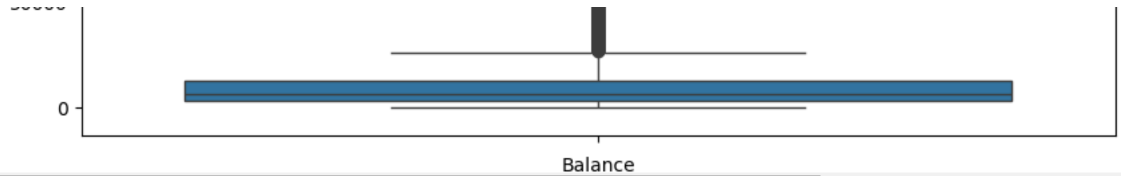
```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the distribution of 'annual_salary' to check for outliers
```

```
plt.figure(figsize=(10, 6))
sns.boxplot(data_updated['annual_salary'])
plt.title('Annual Salary Distribution with Outliers')
plt.xlabel('Annual Salary')
plt.show()

# Visualize the distribution of key features like 'amount' and 'balance'
plt.figure(figsize=(10, 6))
sns.boxplot(data_updated['amount'])
plt.title('Transaction Amount Distribution with Outliers')
plt.xlabel('Amount')
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(data_updated['balance'])
plt.title('Balance Distribution with Outliers')
plt.xlabel('Balance')
plt.show()

```


Annual Salary Distribution with Outliers
140000
120000
100000
80000
60000
40000
annual_salary
Annual Salary

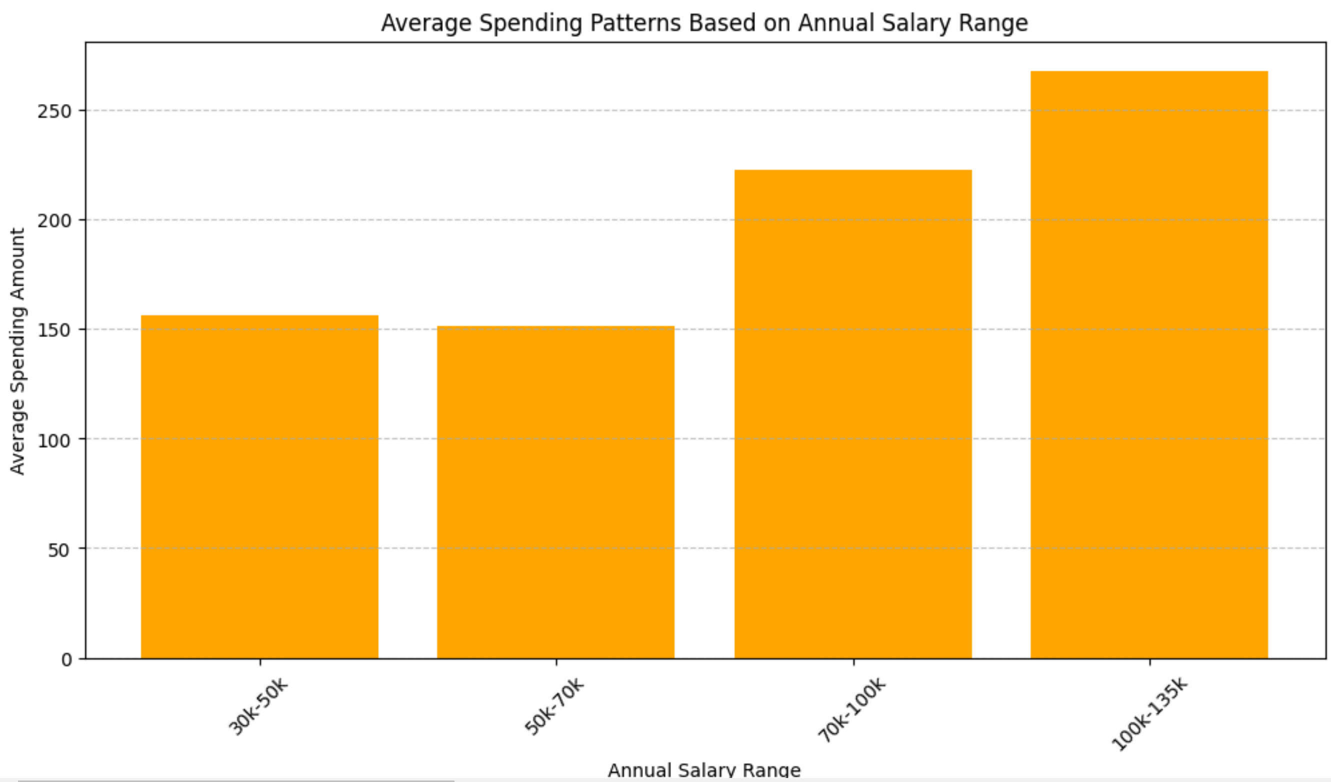

Transaction Amount Distribution with Outliers
8000
6000
4000
2000
0
amount
Amount


Balance Distribution with Outliers
250000
200000
150000
100000
50000
balance




```
# Adjusting the salary bins based on the observed minimum and maximum values
salary_bins = [29000, 50000, 70000, 100000, 135000]
salary_labels = ['30k-50k', '50k-70k', '70k-100k', '100k-135k']
data['salary_range'] = pd.cut(data['annual_salary'], bins=salary_bins, labels=salary_labels)

# Group the data by salary range and calculate the average spending
salary_spending = data.groupby('salary_range')['amount'].mean()

# Create a bar chart to visualize average spending based on annual salary ranges
plt.figure(figsize=(12, 6))
plt.bar(salary_spending.index, salary_spending.values, color='orange')
plt.xlabel('Annual Salary Range')
plt.ylabel('Average Spending Amount')
plt.title('Average Spending Patterns Based on Annual Salary Range')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```

```
<ipython-input-29-8e582b1065d0>:7: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future ve
  salary_spending = data.groupby('salary_range')['amount'].mean()
```



This bar chart displays **Average Spending Patterns Based on Annual Salary Range**, providing insights into how average spending varies across different income levels.

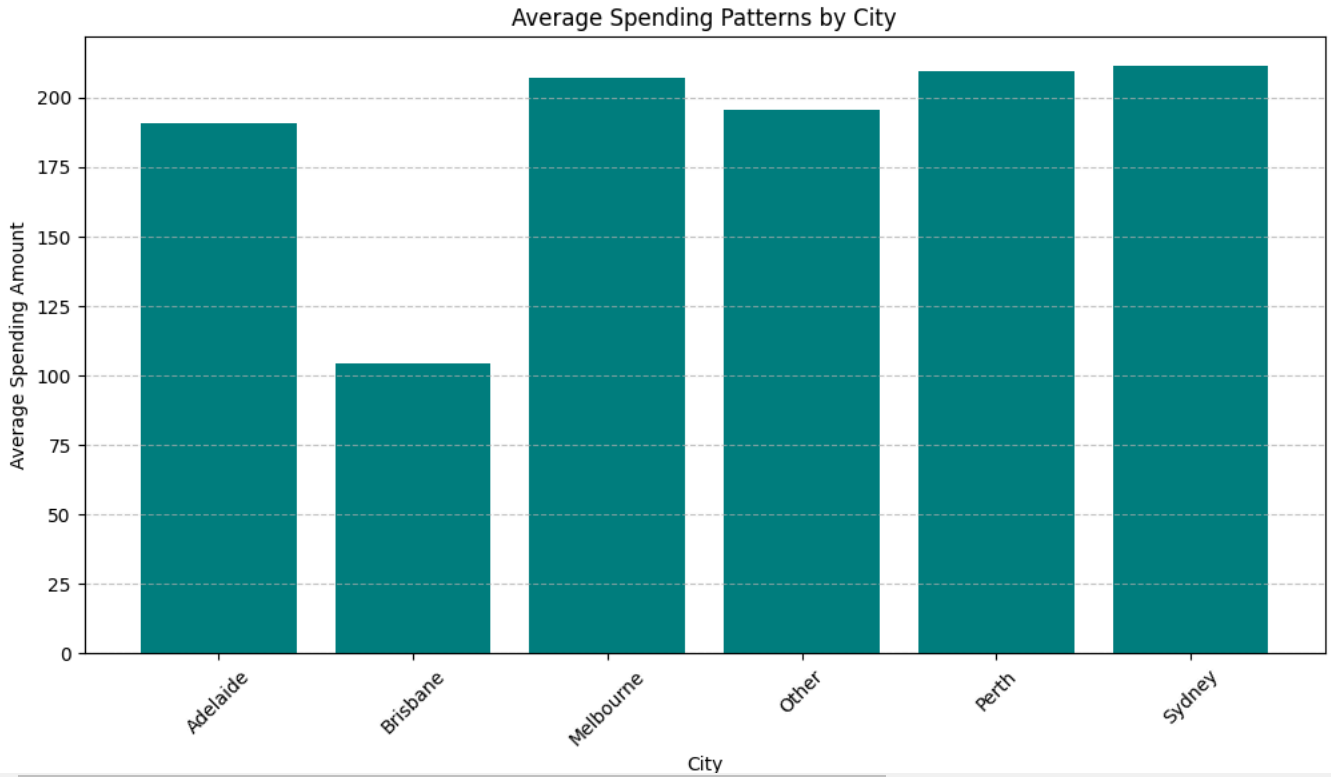
Here's a breakdown of the chart:

1. **X-Axis (Annual Salary Range)**:
    - The horizontal axis categorizes individuals into four annual salary ranges: 30K–50K, 50K–70K, 70K–100K, and 100K–135K.
    - These ranges allow for comparison of average spending across different income levels.
2. **Y-Axis (Average Spending Amount)**:
    - The vertical axis represents the average spending amount, with values ranging from 0 to 250.
    - Each bar's height indicates the average spending amount for each salary range.

3. **Key Observations**:
    - There is a clear upward trend in average spending as annual salary increases.
    - Individuals with a salary range of 30K–50K and 50K–70K have similar average spending amounts, around 150.
    - For individuals in the 70K–100K salary range, average spending increases significantly to above 200.
    - The highest spending occurs among individuals in the 100K–135K salary range, with an average spending amount close to 250.
4. **Insights for Your Thesis**:
    - This chart suggests a positive correlation between income and average spending, with higher earners generally spending more on average.
    - You could discuss this finding in terms of consumer behavior, noting that individuals with higher salaries may have more disposable income, which allows for increased spending.
5. **Implications**:
    - Businesses could use this information to segment customers by income and tailor marketing efforts or products to meet the spending patterns of different income groups.
    - This data might also imply that as individuals' income grows, they become more valuable customers for certain types of products or services.
6. **Conclusion**:
    - This bar chart provides a straightforward visualization of the relationship between income and spending, showing that higher income groups have higher average spending. This insight could be valuable in developing targeted strategies based on income levels.

```
Start coding or generate with AI.
```

```
# Grouping the dataset by city and calculating the average spending amount
# The dataset doesn't have a direct 'city' column, so using latitude and longitude to approximate cities if available

# Checking unique values in latitude and longitude to determine if cities can be differentiated
location_summary = data[['latitude', 'longitude']].drop_duplicates()
location_summary.head()
# Mapping latitude and longitude to city names based on known coordinates
def map_city(latitude, longitude):
    if -28 <= latitude <= -27 and 153 <= longitude <= 154:
        return 'Brisbane'
    elif -34 <= latitude <= -33 and 151 <= longitude <= 152:
        return 'Sydney'
    elif -38 <= latitude <= -37 and 144 <= longitude <= 145:
        return 'Melbourne'
    elif -35 <= latitude <= -34 and 138 <= longitude <= 139:
        return 'Adelaide'
    elif -32 <= latitude <= -31 and 115 <= longitude <= 116:
        return 'Perth'
    else:
        return 'Other'

# Apply the city mapping function to create a new column 'city'
data['city'] = data.apply(lambda row: map_city(row['latitude'], row['longitude']), axis=1)

# Group the data by city and calculate the average spending
city_spending = data.groupby('city')['amount'].mean()

# Create a bar chart to visualize average spending based on city
plt.figure(figsize=(12, 6))
plt.bar(city_spending.index, city_spending.values, color='teal')
plt.xlabel('City')
plt.ylabel('Average Spending Amount')
plt.title('Average Spending Patterns by City')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



This bar chart illustrates **Average Spending Patterns by City**, showing the average spending amount for customers residing in different cities.

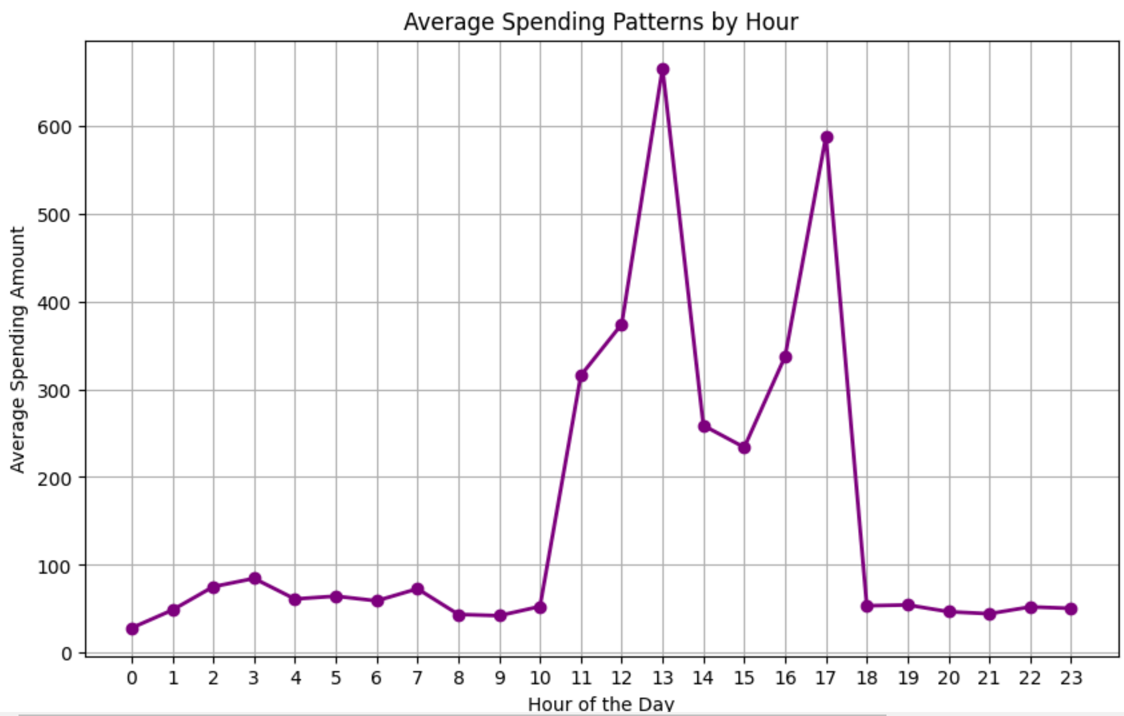
Here's a breakdown:

1. **X-Axis (City)**:
   - The horizontal axis lists the cities where customers are located: Adelaide, Brisbane, Melbourne, Other (representing smaller cities or unspecified locations), Perth, and Sydney.
   - This categorization allows for a comparison of average spending across different geographical locations.
2. **Y-Axis (Average Spending Amount)**:
   - The vertical axis represents the average spending amount, with values ranging from 0 to just above 200.
   - Each bar's height indicates the average spending for customers in each city.
3. **Key Observations**:
   - **Melbourne, Perth, and Sydney** have the highest average spending amounts, all around or slightly above 200.
   - **Adelaide** has a slightly lower average spending amount, though still relatively close to the top cities.
   - **Brisbane** stands out with the lowest average spending amount, close to 100, which is significantly lower than the other cities.
   - The **Other** category has an average spending amount close to the main cities but slightly lower than Melbourne, Perth, and Sydney.
4. **Insights for Your Thesis**:
   - This chart suggests that spending behavior varies by city, with customers in Melbourne, Perth, and Sydney spending more on average than those in Brisbane.
   - The high spending in cities like Melbourne, Perth, and Sydney could be influenced by factors like cost of living, income levels, or available goods and services.
   - In contrast, the lower spending in Brisbane could imply different consumer behavior patterns or economic conditions.
5. **Implications**:
   - Businesses might use this data for targeted marketing efforts, prioritizing high-spending cities for premium products or services.
   - This analysis could also inform location-based pricing or promotional strategies to cater to the different spending capacities of residents in each city.
6. **Conclusion**:
   - In your thesis, you can use this chart to discuss the impact of geographic location on consumer spending. It provides evidence that spending behavior can differ significantly by city, which could be linked to regional economic and demographic factors.

This bar chart offers a clear visualization of how average spending differs across cities, highlighting geographic patterns in consumer behavior that could be valuable for market segmentation and strategy development.

```
Start coding or generate with AI.
```

```
# Group the data by hour and calculate the average spending amount
hourly_spending = data.groupby('hour')['amount'].mean()

# Create a line chart to visualize the average spending amount for each hour of the day
plt.figure(figsize=(10, 6))
plt.plot(hourly_spending.index, hourly_spending.values, marker='o', linestyle='-', linewidth=2, color='purple')
plt.xlabel('Hour of the Day')
plt.ylabel('Average Spending Amount')
plt.title('Average Spending Patterns by Hour')
plt.grid(True)
plt.xticks(range(0, 24))
plt.show()

```



This line chart displays **Average Spending Patterns by Hour of the Day**, showing how average spending varies at different hours.

Here's a breakdown:

1. **X-Axis (Hour of the Day)**:
   - The horizontal axis represents each hour of the day, from 0 (midnight) to 23 (11 PM).
   - This allows us to see how spending fluctuates across the 24-hour period.
2. **Y-Axis (Average Spending Amount)**:
   - The vertical axis represents the average spending amount, ranging from 0 to above 600.
   - Each point on the line represents the average spending amount for a specific hour.
3. **Key Observations**:
   - **Peak Spending Hours**: There are two notable peaks in average spending. The first peak occurs around 13:00 (1 PM), and the second peak occurs around 17:00 (5 PM). These times show the highest average spending amounts, reaching above 600 and close to 500, respectively.
   - **Low Spending Hours**: Spending is lowest during the early morning and late-night hours (0:00–9:00 and 18:00–23:00), where the average spending amount stays below 100.
   - There is a steady increase in spending leading up to the peak around midday, followed by a dip after 1 PM, and then another increase leading up to the 5 PM peak.
4. **Insights for Your Thesis**:

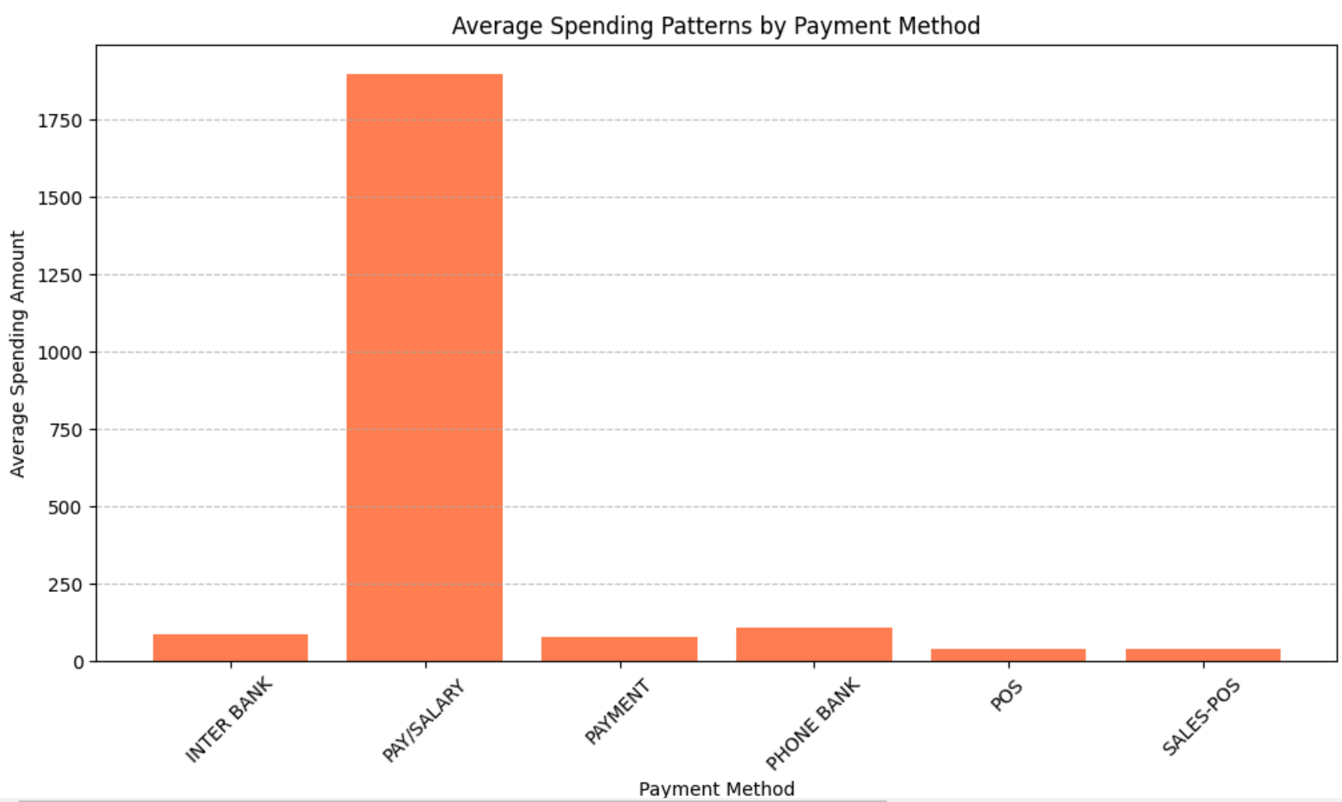
- The peaks around 1 PM and 5 PM suggest that spending activity is highest during these hours, possibly due to lunchtime and end-of-workday spending patterns. This could indicate that people tend to make purchases or transactions during lunch breaks and after finishing work.
- The low spending during early morning and late evening hours may indicate that fewer transactions occur at these times, which could be typical for non-24-hour businesses.

5. **Implications**:
   - Businesses can use this information to optimize operations and staffing around peak spending hours to better meet customer demand.
   - Marketing efforts could also be targeted during peak hours to capitalize on times when customers are more likely to make purchases.
6. **Conclusion**:
   - In your thesis, you could use this analysis to discuss consumer behavior patterns related to time of day, providing insights into when spending is most and least active.

This line chart effectively shows the times of day when spending is highest, offering valuable insights into daily spending patterns that could inform operational or marketing strategies.

```
Start coding or generate with AI.
```

```
# Checking the unique values in the 'txn_description' column to understand the different payment methods
payment_methods = data['txn_description'].unique()
payment_methods
# Group the data by payment method and calculate the average spending
payment_spending = data.groupby('txn_description')['amount'].mean()

# Create a bar chart to visualize average spending based on payment method
plt.figure(figsize=(12, 6))
plt.bar(payment_spending.index, payment_spending.values, color='coral')
plt.xlabel('Payment Method')
plt.ylabel('Average Spending Amount')
plt.title('Average Spending Patterns by Payment Method')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



This bar chart shows **Average Spending Patterns by Payment Method**, illustrating how the average spending amount varies depending on the type of payment method used.

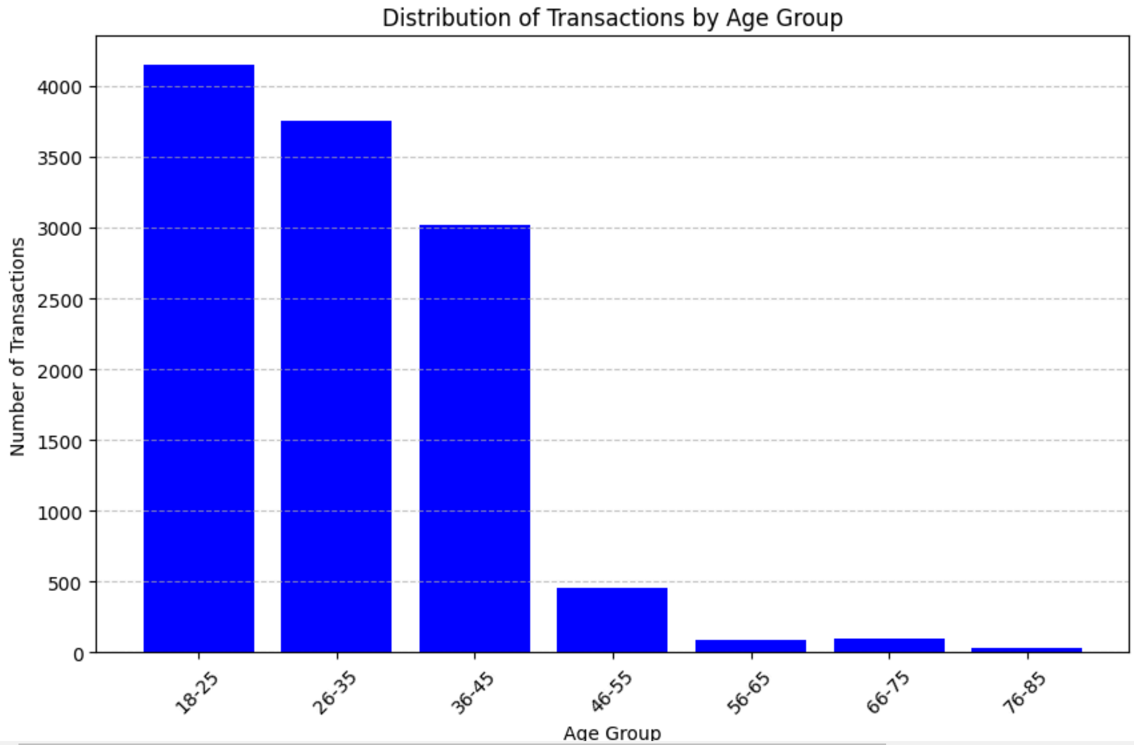
Here's a breakdown of the chart:

1. **X-Axis (Payment Method)**:
   - The horizontal axis lists different payment methods: INTER BANK, PAY/SALARY, PAYMENT, PHONE BANK, POS (Point of Sale), and SALES-POS.
   - This categorization allows for comparison of average spending across these payment types.
2. **Y-Axis (Average Spending Amount)**:
   - The vertical axis represents the average spending amount, with values ranging from 0 to above 1750.
   - Each bar's height indicates the average spending associated with each payment method.
3. **Key Observations**:
   - **PAY/SALARY** shows a significantly higher average spending amount than all other methods, with an average above 1750. This likely indicates that transactions labeled as PAY/SALARY involve larger sums, potentially due to salary deposits or high-value payments.
   - Other payment methods, such as INTER BANK, PAYMENT, PHONE BANK, POS, and SALES-POS, have much lower average spending amounts, all around or below 250.
   - The stark difference between PAY/SALARY and the other payment methods suggests that PAY/SALARY transactions are typically much larger than routine payments made through other channels.
4. **Insights for Your Thesis**:
   - This chart suggests that the nature of the PAY/SALARY payment type is distinct, likely encompassing higher-value transactions, possibly salary deposits or other large transfers.
   - The other methods—INTER BANK, PAYMENT, PHONE BANK, POS, and SALES-POS—are likely used for regular spending or smaller transactions, as indicated by their lower average amounts.
5. **Implications**:
   - Businesses could use this insight to classify transaction types based on payment methods, helping to identify high-value transactions versus routine expenses.
   - Financial institutions may analyze spending patterns by payment type to design services or set transaction limits based on typical usage.
6. **Conclusion**:
   - In your thesis, you could mention that different payment methods are associated with varying spending patterns, with PAY/SALARY transactions standing out due to their higher average values. This could help in understanding transaction behavior and categorizing transactions based on value.

This bar chart effectively highlights the disparity in spending patterns by payment method, revealing that PAY/SALARY transactions are likely associated with larger financial movements than other methods. This insight can be valuable for financial analysis and transaction categorization.

```
Start coding or generate with AI.
```

```
# Define age bins to categorize the age groups
age_bins = [18, 25, 35, 45, 55, 65, 75, 85]
age_labels = ['18-25', '26-35', '36-45', '46-55', '56-65', '66-75', '76-85']
data['age_group'] = pd.cut(data['age'], bins=age_bins, labels=age_labels)

# Group the data by age group and count the number of transactions
age_group_distribution = data['age_group'].value_counts().sort_index()

# Create a bar chart to visualize the distribution of transactions by age group
plt.figure(figsize=(10, 6))
plt.bar(age_group_distribution.index, age_group_distribution.values, color='blue')
plt.xlabel('Age Group')
plt.ylabel('Number of Transactions')
plt.title('Distribution of Transactions by Age Group')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



This bar chart displays the **Distribution of Transactions by Age Group**, illustrating the number of transactions made by customers within different age ranges.

Here's a detailed breakdown:

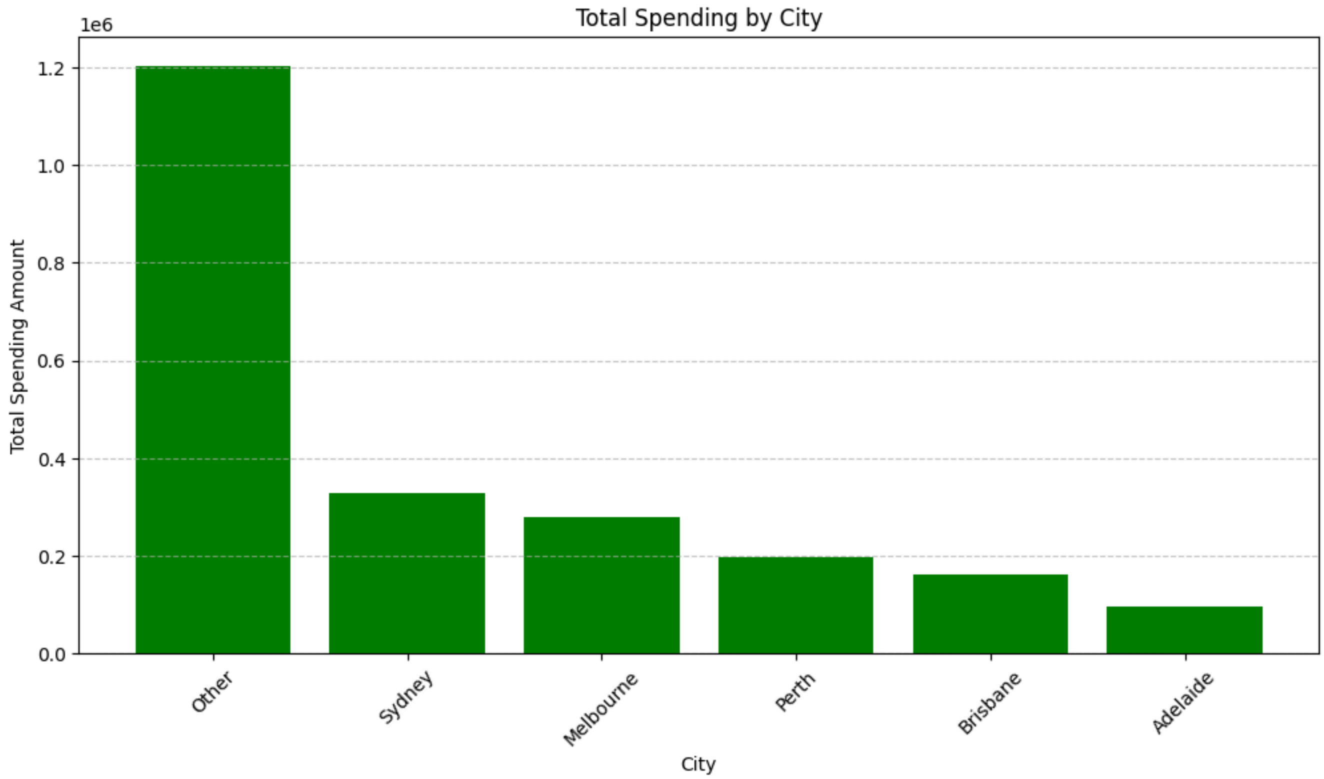
1. **X-Axis (Age Group)**:
   - The horizontal axis categorizes customers into different age groups: 18-25, 26-35, 36-45, 46-55, 56-65, 66-75, and 76-85.
   - This categorization allows for a comparison of transaction frequency across age groups.
2. **Y-Axis (Number of Transactions)**:
   - The vertical axis represents the number of transactions, ranging from 0 to above 4000.
   - Each bar’s height indicates the total number of transactions made by customers in each age group.
3. **Key Observations**:
   - **18-25 Age Group**: This age group has the highest number of transactions, with more than 4000 transactions.
   - **26-35 Age Group**: The second-highest number of transactions, slightly lower than the 18-25 group, with close to 3800 transactions.
   - **36-45 Age Group**: This age group also has a substantial number of transactions, around 3000, though lower than the two younger groups.
   - **Older Age Groups**: For age groups 46-55 and above, the number of transactions drops sharply, with each group having fewer than 1000 transactions. The transaction count continues to decrease as age increases, with the lowest transaction counts in the 66-75 and 76-85 age groups.
4. **Insights for Your Thesis**:
   - This chart suggests that younger age groups (particularly those between 18 and 35) are more active in terms of transaction frequency. This could be due to a combination of factors, such as lifestyle, spending habits, or disposable income levels.
   - The decrease in transactions as age increases may indicate that older individuals are less active in this dataset, which could reflect differences in financial behavior, technology adoption, or other demographic factors.
5. **Implications**:
   - Businesses could use this information for targeted marketing efforts, focusing more on younger age groups who engage more frequently in transactions.
   - Financial institutions might consider designing products or services that appeal to younger customers based on their high transaction activity.
6. **Conclusion**:

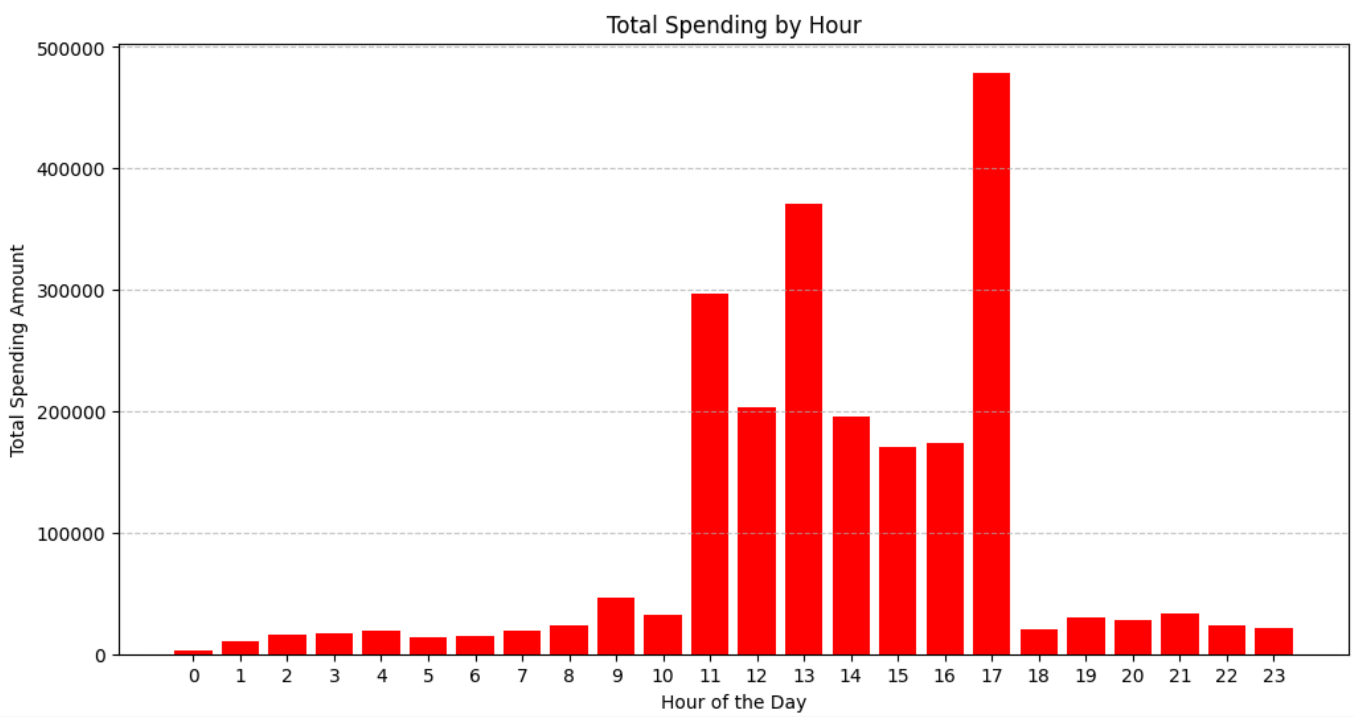
- In your thesis, you could use this chart to support discussions on age-based transaction behavior, showing that younger individuals have higher transaction activity. This information could be useful for customer segmentation and understanding the financial habits of different age groups.

This chart provides a clear visualization of transaction frequency by age, indicating that younger customers are more transactionally active, which can inform marketing, product design, and customer engagement strategies.

```
Start coding or generate with AI.
```

```
# Grouping the data by city and calculating the total spending to find the highest spending cities
city_total_spending = data.groupby('city')['amount'].sum().sort_values(ascending=False)

# Grouping the data by hour and calculating the total spending to find the highest spending hours
hour_total_spending = data.groupby('hour')['amount'].sum().sort_values(ascending=False)

# Visualizing the highest spending cities
plt.figure(figsize=(12, 6))
plt.bar(city_total_spending.index, city_total_spending.values, color='green')
plt.xlabel('City')
plt.ylabel('Total Spending Amount')
plt.title('Total Spending by City')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

# Visualizing the highest spending hours
plt.figure(figsize=(12, 6))
plt.bar(hour_total_spending.index, hour_total_spending.values, color='red')
plt.xlabel('Hour of the Day')
plt.ylabel('Total Spending Amount')
plt.title('Total Spending by Hour')
plt.xticks(range(0, 24))
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```





This bar chart shows the **Total Spending by City**, illustrating the aggregate spending amounts for customers across different cities.

Here's a breakdown of the chart:

1. **X-Axis (City)**:
   - The horizontal axis represents various cities: Other, Sydney, Melbourne, Perth, Brisbane, and Adelaide.
   - "Other" likely represents smaller cities or unspecified locations not included in the primary city categories.
2. **Y-Axis (Total Spending Amount)**:
   - The vertical axis represents the total spending amount, with the scale ranging up to 1.2 million.
   - Each bar's height indicates the cumulative spending amount for each city.
3. **Key Observations**:

- **"Other"** category has by far the highest total spending, close to 1.2 million. This indicates that a significant portion of total spending comes from locations outside the primary cities listed.
- **Sydney** and **Melbourne** follow with similar total spending amounts, each contributing a substantial amount but far less than the "Other" category.
- **Perth** shows moderate total spending, lower than Sydney and Melbourne.
- **Brisbane** and **Adelaide** have the lowest total spending amounts among the cities listed.

4. **Insights for Your Thesis**:
   - This chart suggests that a considerable portion of total spending is concentrated in locations outside the major cities (as indicated by the "Other" category). This could mean that there is significant spending activity in smaller towns or regional areas.
   - Major cities like Sydney and Melbourne also contribute substantial spending, likely due to their larger populations and economic activity.
   - The lower spending in cities like Brisbane and Adelaide might reflect smaller customer bases or different spending habits in those areas.
5. **Implications**:
   - Businesses could use this information to understand where spending is concentrated and to allocate resources accordingly. For example, marketing or expansion efforts might focus more on regions outside major cities or target high-spending cities like Sydney and Melbourne.
   - This data could also be useful for regional analysis, helping to identify areas with high spending potential beyond major metropolitan centers.
6. **Conclusion**:
   - In your thesis, you could use this chart to support discussions on geographic spending patterns, emphasizing that spending is not confined to large cities but is also strong in smaller towns or less-defined areas (represented by "Other").

This bar chart shows **Total Spending by Hour of the Day**, illustrating how the total spending amount varies at different times throughout the day.

Here’s a breakdown:

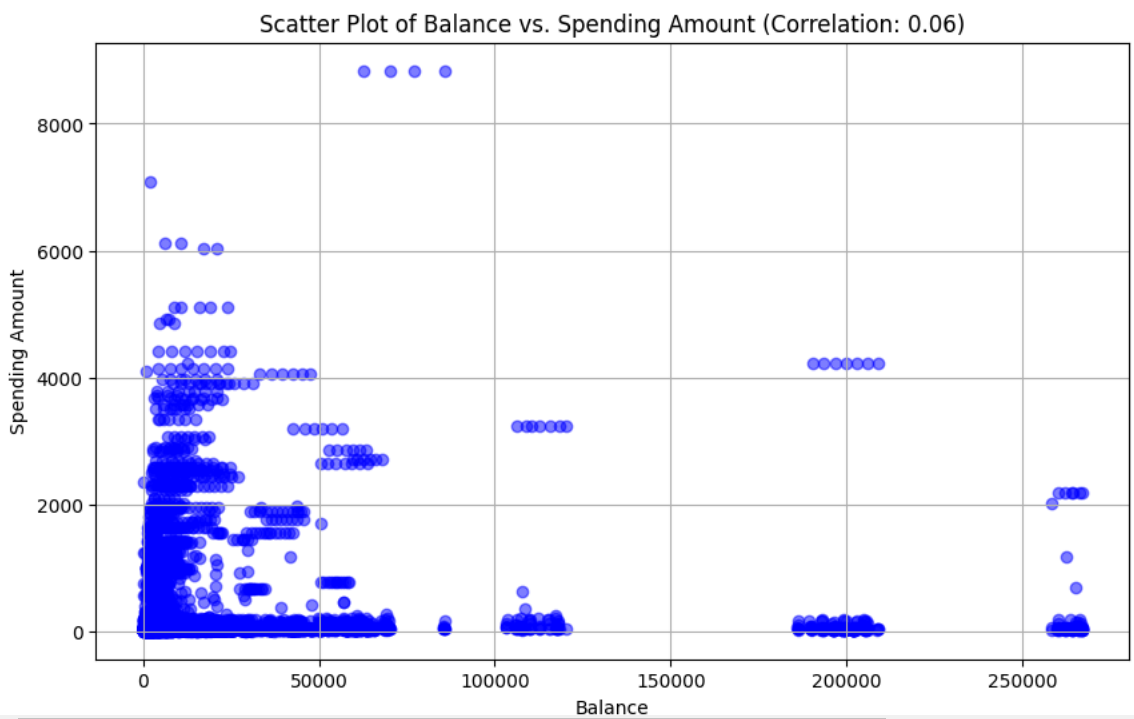
1. **X-Axis (Hour of the Day)**:
   - The horizontal axis represents each hour of the day, from 0 (midnight) to 23 (11 PM).
   - This allows us to see when the total spending amount is highest and lowest across the 24-hour period.
2. **Y-Axis (Total Spending Amount)**:
   - The vertical axis represents the total spending amount, with values up to 500,000.
   - Each bar’s height indicates the cumulative spending amount for each hour of the day.
3. **Key Observations**:
   - **Peak Spending Hours**: The highest total spending occurs during two peak times:
     - Around 13:00 (1 PM), with spending above 400,000.
     - Around 17:00 (5 PM), which shows the highest total spending, reaching close to 500,000.
   - **Other High-Spending Periods**: There is also significant spending from around 11:00 to 16:00, though these amounts are lower than the two main peaks.
   - **Low Spending Hours**: Early morning and late-night hours (0:00–9:00 and 18:00–23:00) show minimal spending, with very low total amounts compared to peak hours.
4. **Insights for Your Thesis**:
   - The peaks around midday and late afternoon suggest that spending activity is highest during these times. This could correspond to lunchtime and the end-of-workday periods, times when people are more likely to make purchases or conduct transactions.
   - The low spending during the early morning and late evening hours indicates less commercial activity during these periods.
5. **Implications**:
   - Businesses could use this data to optimize their operations, staffing, and marketing efforts around peak spending hours.
   - Financial institutions or payment processing services might consider preparing for high transaction volumes during peak hours to ensure efficient processing.
6. **Conclusion**:

- In your thesis, you could use this chart to support discussions on time-based spending behavior, showing that spending is concentrated around certain hours, which likely corresponds to typical consumer routines.

This chart provides a clear view of spending patterns by hour, indicating that consumer spending is heavily concentrated during specific times of the day. This information could be valuable for operational planning, marketing, and understanding consumer behavior.

```
Start coding or generate with AI.
```

```
import numpy as np

# Calculating the correlation between balance and amount
correlation = data['balance'].corr(data['amount'])

# Creating a scatter plot to visualize the relationship between balance and spending amount
plt.figure(figsize=(10, 6))
plt.scatter(data['balance'], data['amount'], alpha=0.5, color='blue')
plt.xlabel('Balance')
plt.ylabel('Spending Amount')
plt.title(f'Scatter Plot of Balance vs. Spending Amount (Correlation: {correlation:.2f})')
plt.grid(True)
plt.show()

```



This scatter plot shows the **relationship between Balance and Spending Amount**, with a calculated correlation of 0.06.

Here's a detailed breakdown:

1. **Axes**:
    - The **X-axis** represents the balance amount, which ranges from 0 to over 250,000.
    - The **Y-axis** represents the spending amount, which ranges from 0 to over 8,000.
2. **Correlation**:
    - The correlation value is **0.06**, which is very close to zero. This suggests that there is almost no linear relationship between balance and spending amount in this dataset.
    - A near-zero correlation implies that higher or lower balances do not significantly affect the spending amount.
3. **Distribution of Points**:
    - Most data points are clustered in the lower range of both balance and spending, particularly with balances below 50,000 and spending amounts below 2,000.
    - There are a few isolated groups of points with higher balances (around 100,000, 150,000, and 200,000), but these do not correspond to a notable increase in spending.

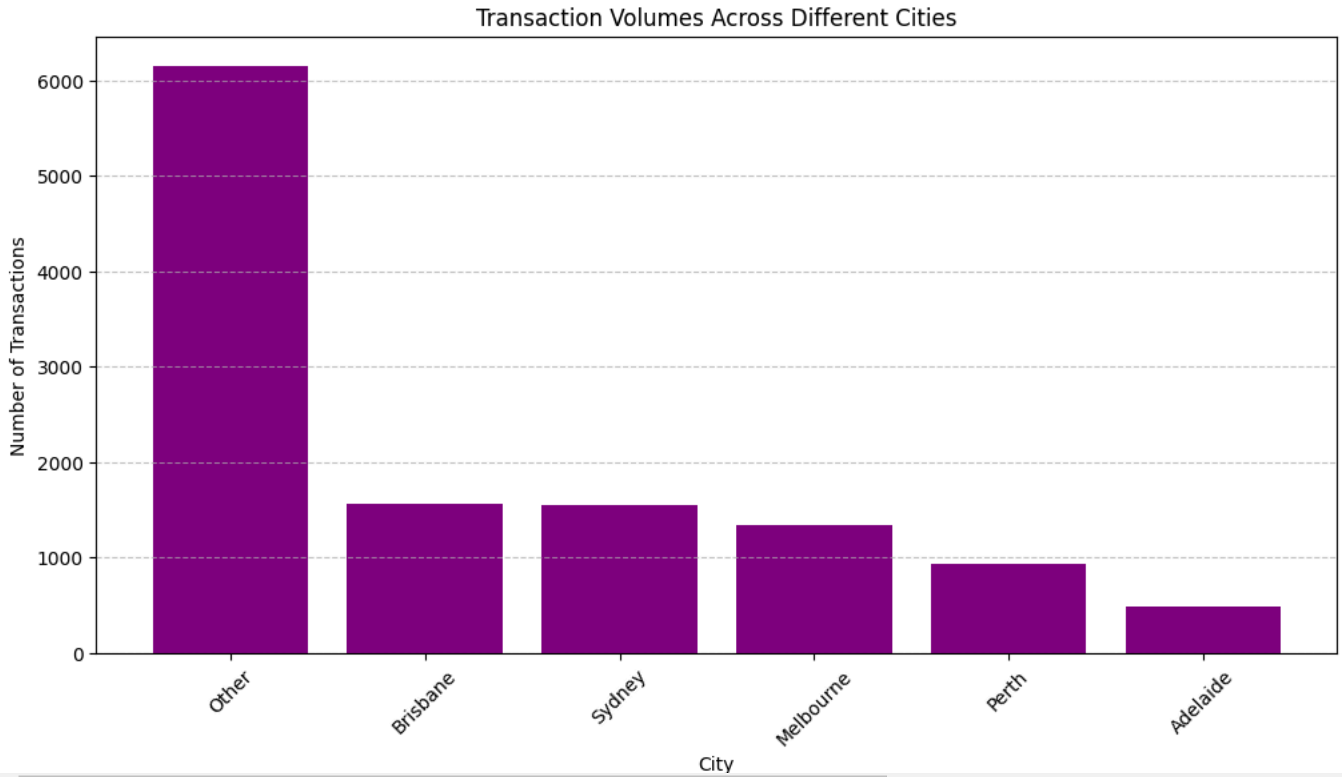
- Some outliers appear at higher spending amounts (above 4,000), but these are spread across different balance levels, indicating no clear pattern or trend.

4. **Insights for Your Thesis**:
   - The lack of a strong relationship between balance and spending amount suggests that factors other than account balance may influence spending behavior in this dataset.
   - You could explore other variables that might be better predictors of spending or discuss the possible reasons why balance does not directly correlate with spending.
5. **Implications**:
   - This analysis implies that individuals with higher balances are not necessarily spending more, which could be important for financial institutions when segmenting customers or offering products.
   - Understanding that balance does not predict spending might prompt further investigation into what drives spending behavior, such as income, age, or transaction patterns.
6. **Conclusion**:
   - In your thesis, you could discuss the significance of this weak correlation and suggest other potential avenues of analysis. This scatter plot provides evidence that balance alone is not a determining factor for spending, highlighting the complexity of consumer financial behavior.

This scatter plot effectively illustrates the lack of a relationship between balance and spending amount, which could support discussions on the varied factors influencing consumer spending beyond just account balance.

```
Start coding or generate with AI.
```

```
# Grouping the data by city and counting the number of transactions for each city
city_transaction_volume = data['city'].value_counts()

# Creating a bar chart to visualize the transaction volumes across different cities
plt.figure(figsize=(12, 6))
plt.bar(city_transaction_volume.index, city_transaction_volume.values, color='purple')
plt.xlabel('City')
plt.ylabel('Number of Transactions')
plt.title('Transaction Volumes Across Different Cities')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



This bar chart illustrates the **Transaction Volumes Across Different Cities**, showing the number of transactions made by customers in various cities.

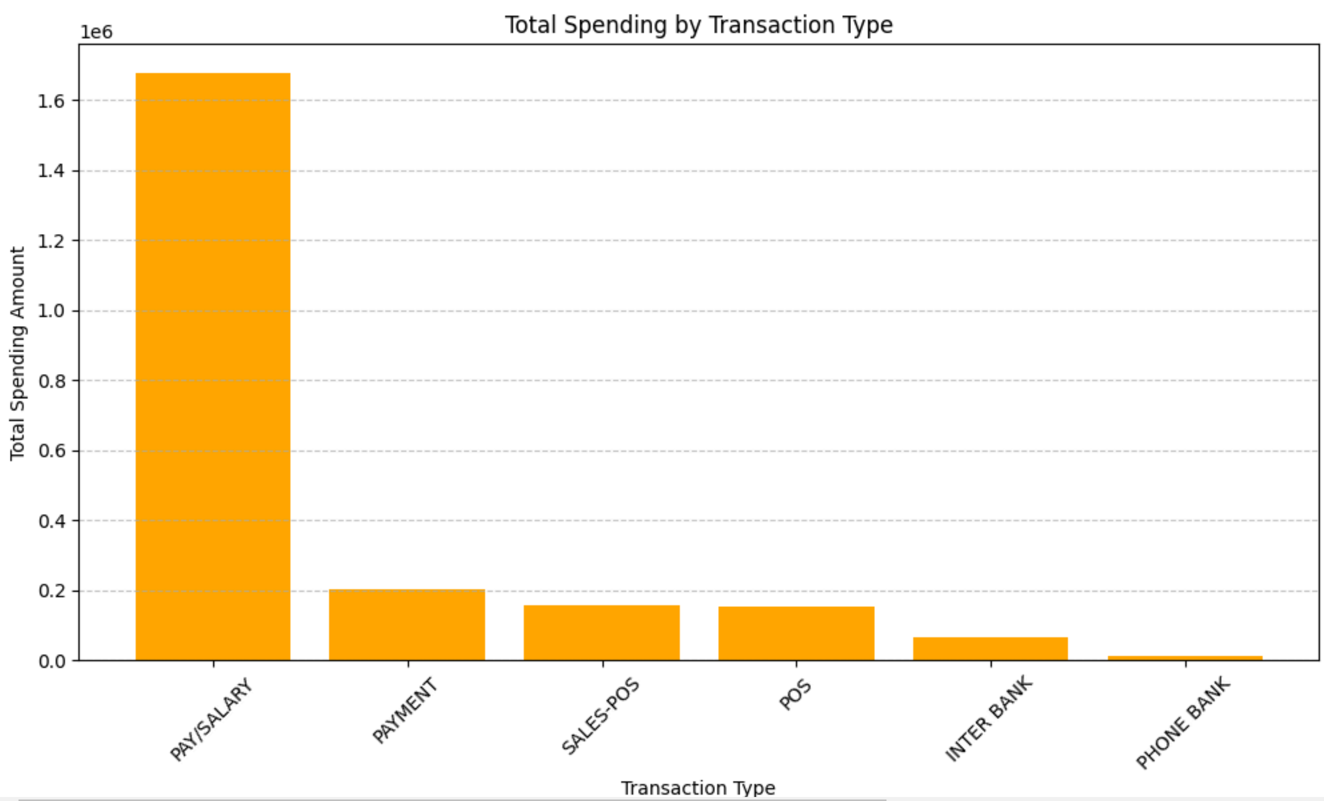
Here's a detailed explanation:

1. **X-Axis (City)**:
    - The horizontal axis lists the cities where transactions were recorded: Other, Brisbane, Sydney, Melbourne, Perth, and Adelaide.
    - "Other" likely represents transactions from smaller cities or unspecified locations not included in the primary city categories.
2. **Y-Axis (Number of Transactions)**:
    - The vertical axis shows the number of transactions, ranging up to 6,000.
    - Each bar's height represents the total transaction volume for each city.
3. **Key Observations**:
    - **"Other"** category has the highest transaction volume, with approximately 6,000 transactions. This suggests that a large portion of transactions occur outside the major cities listed.
    - **Brisbane**, **Sydney**, and **Melbourne** each have a similar number of transactions, with volumes around or slightly above 1,000.
    - **Perth** and **Adelaide** have the lowest transaction volumes among the cities, with fewer than 1,000 transactions each.
4. **Insights for Your Thesis**:
    - This chart indicates that while major cities contribute significantly to transaction volumes, a considerable number of transactions also come from smaller towns or less-defined regions, as represented by the "Other" category.
    - The similar transaction volumes in Brisbane, Sydney, and Melbourne suggest these cities have comparable levels of financial activity, whereas Perth and Adelaide have lower transaction engagement.
5. **Implications**:
    - Businesses could leverage this data to identify high-activity regions and allocate resources accordingly, focusing on both urban centers and smaller areas.
    - For companies aiming to expand, understanding transaction volumes by city could help target cities with higher or growing financial activity.
6. **Conclusion**:
    - In this thesis, we could use this chart to support discussions on geographic transaction patterns, highlighting that transaction volume is not solely concentrated in big cities, but is also substantial in smaller or unspecified regions.

This chart provides a clear view of transaction activity across various cities, suggesting that both metropolitan and smaller areas play a role in overall transaction volume. This insight can inform strategies related to regional engagement, customer targeting, and resource allocation.

```
Start coding or generate with AI.
```

```
# Grouping the data by transaction type ('txn_description') and calculating the total spending for each type
transaction_type_spending = data.groupby('txn_description')['amount'].sum().sort_values(ascending=False)

# Creating a bar chart to visualize the highest spending by transaction type
plt.figure(figsize=(12, 6))
plt.bar(transaction_type_spending.index, transaction_type_spending.values, color='orange')
plt.xlabel('Transaction Type')
plt.ylabel('Total Spending Amount')
plt.title('Total Spending by Transaction Type')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



This bar chart displays **Total Spending by Transaction Type**, showing how spending amounts vary across different types of transactions.

Here's a breakdown of the chart:

1. **X-Axis (Transaction Type)**:
   - The horizontal axis lists various transaction types: PAY/SALARY, PAYMENT, SALES-POS, POS, INTER BANK, and PHONE BANK.
   - This categorization allows for a comparison of total spending amounts associated with each transaction type.
2. **Y-Axis (Total Spending Amount)**:
   - The vertical axis represents the total spending amount, with values up to 1.6 million.
   - Each bar's height indicates the cumulative spending amount for each transaction type.
3. **Key Observations**:
   - **PAY/SALARY** has the highest total spending by far, with an amount close to 1.6 million. This indicates that transactions categorized under PAY/SALARY account for a significant portion of overall spending, likely due to salary deposits or other large-value transactions.
   - **PAYMENT**, **SALES-POS**, and **POS** show moderate total spending amounts, each below 0.2 million. These types of transactions are likely more routine, involving smaller individual amounts but potentially higher transaction frequencies.

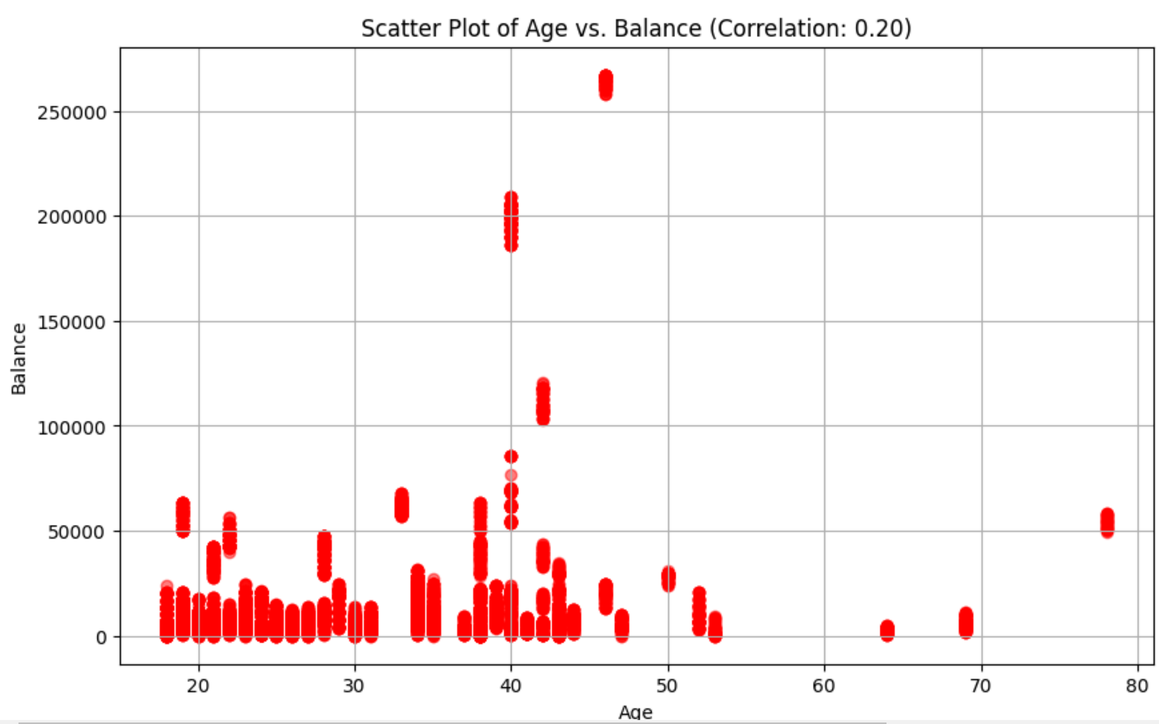
- **INTER BANK** and **PHONE BANK** have the lowest total spending amounts, with PHONE BANK being particularly low. These transaction types contribute minimally to overall spending.

4. **Insights for Your Thesis**:
    - The dominance of PAY/SALARY in total spending suggests that this transaction type involves high-value transfers, such as salary payments or significant deposits.
    - The lower spending totals for POS and SALES-POS indicate that these are likely used for day-to-day transactions involving smaller amounts.
    - The limited spending through INTER BANK and PHONE BANK could suggest that these methods are less frequently used or involve smaller transfers in this dataset.
5. **Implications**:
    - This data can help businesses or financial institutions understand which transaction types contribute most to total spending, potentially guiding decisions on where to focus transaction processing resources.
    - Marketing efforts or service enhancements might be targeted toward high-value transaction types like PAY/SALARY, whereas operational efficiencies could be explored for high-frequency but lower-value transaction types like POS.
6. **Conclusion**:
    - This chart could be used to discuss spending behavior across different transaction types, emphasizing that high-value transactions (such as PAY/SALARY) make up a large part of total spending, while routine transactions contribute smaller amounts.

This bar chart effectively highlights the disparity in spending amounts by transaction type, showing that PAY/SALARY transactions account for the majority of spending, while other transaction types represent routine, lower-value spending. This insight can inform analyses related to transaction categorization and financial behavior.

```
Start coding or generate with AI.
```

```python
# Calculating the correlation between age and balance
age_balance_correlation = data['age'].corr(data['balance'])

# Creating a scatter plot to visualize the relationship between age and balance
plt.figure(figsize=(10, 6))
plt.scatter(data['age'], data['balance'], alpha=0.5, color='red')
plt.xlabel('Age')
plt.ylabel('Balance')
plt.title(f'Scatter Plot of Age vs. Balance (Correlation: {age_balance_correlation:.2f})')
plt.grid(True)
plt.show()

```



This scatter plot shows the **relationship between Age and Balance**, with a calculated correlation of 0.20.

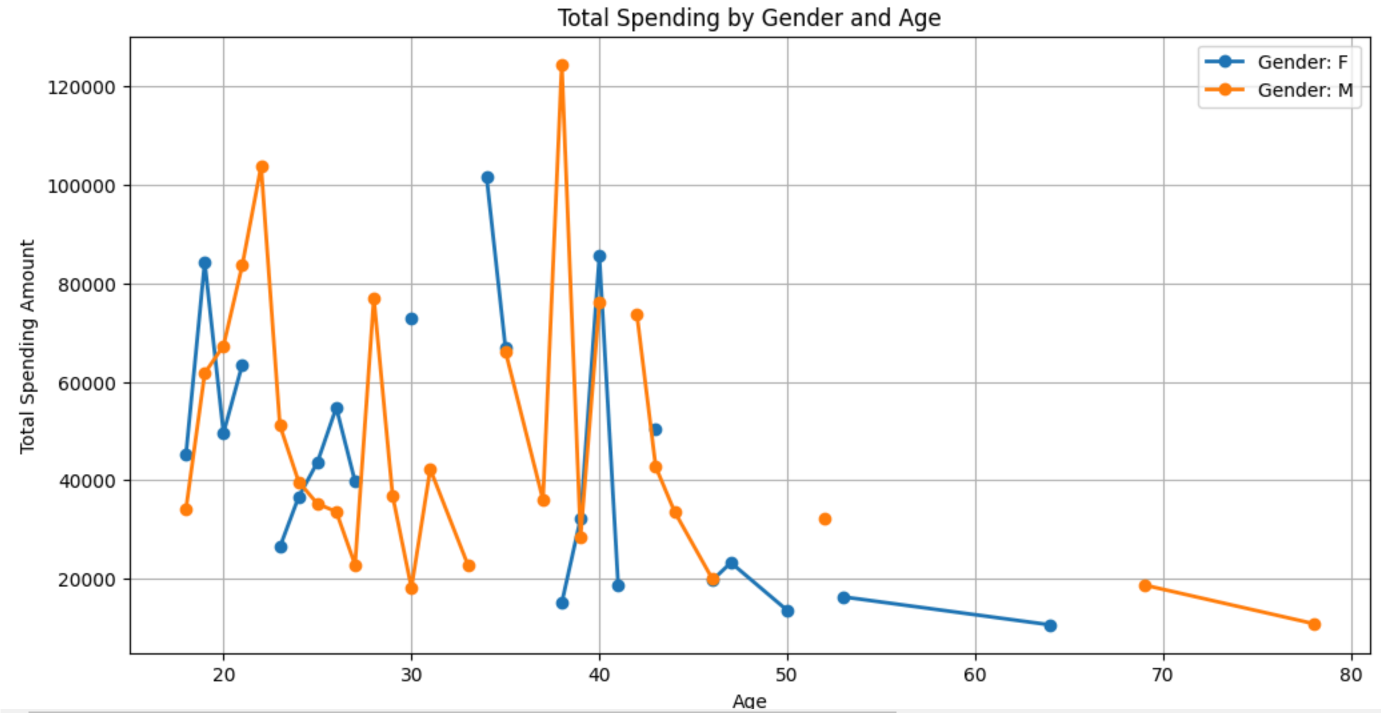
Here's a detailed explanation:

1. **Axes**:
   - The **X-axis** represents age, ranging from approximately 20 to 80.
   - The **Y-axis** represents the balance amount, ranging from 0 to over 250,000.
2. **Correlation**:
   - The correlation value is **0.20**, which is a weak positive correlation. This suggests that as age increases, there is a slight tendency for balance to increase, but the relationship is not strong.
3. **Distribution of Points**:
   - Most data points are clustered in the lower balance range (0 to 50,000) across all ages, particularly for individuals under 50.
   - A few individuals in their late 30s and early 40s have significantly higher balances, with some balances exceeding 200,000.
   - There are only a few scattered points for older individuals (above 50), and they mostly have low balances, with only a few exceptions.
4. **Insights for Your Thesis**:
   - This chart suggests that while there is a weak positive correlation between age and balance, age alone does not strongly determine the balance amount. Other factors might have a more significant influence on balance levels.
   - The presence of high-balance individuals mostly in the 30s and 40s could indicate that people tend to accumulate higher balances in mid-adulthood.
5. **Implications**:
   - Financial institutions or businesses might use this information to better understand the savings or balance trends across age groups. The weak correlation suggests age-based segmentation might not be sufficient for predicting balance levels.
   - This data could be useful for targeting financial products to individuals in mid-adulthood, as this group shows a tendency for higher balances.
6. **Conclusion**:
   - In this thesis, we use this scatter plot to discuss the limited influence of age on balance, noting that while mid-adults show a tendency for higher balances, the overall relationship between age and balance remains weak.

This scatter plot highlights that although there is a weak trend of higher balances with increasing age, the relationship is not strong, suggesting that age alone is not a strong predictor of account balance. This insight can guide discussions on financial behavior across age groups in your analysis.

```
Start coding or generate with AI.
```

```
# Grouping the data by gender and age to calculate total spending for each combination
gender_age_spending = data.groupby(['gender', 'age'])['amount'].sum().reset_index()

# Creating a pivot table for better visualization
gender_age_pivot = gender_age_spending.pivot(index='age', columns='gender', values='amount')

# Creating a line chart for each gender to visualize peak spending by age
plt.figure(figsize=(12, 6))
for gender in gender_age_pivot.columns:
    plt.plot(gender_age_pivot.index, gender_age_pivot[gender], marker='o', linestyle='-', linewidth=2, label=f'Gender: {gender}')

plt.xlabel('Age')
plt.ylabel('Total Spending Amount')
plt.title('Total Spending by Gender and Age')
plt.legend()
plt.grid(True)
plt.show()

```



This line chart shows **Total Spending by Gender and Age**, illustrating how total spending varies across different age groups for both male and female customers.

Here's a detailed explanation:

1. **Axes**:
    - The **X-axis** represents age, ranging from around 20 to 80.
    - The **Y-axis** represents the total spending amount, which goes up to approximately 120,000.
2. **Lines and Colors**:
    - There are two lines on the chart:
        - **Blue line** represents total spending for female (F) customers.
        - **Orange line** represents total spending for male (M) customers.
    - The markers along each line indicate data points for specific age groups, showing variations in total spending by gender.
3. **Key Observations**:
    - **Higher Total Spending in Younger Ages**: For both genders, the highest total spending occurs in the younger age groups, particularly between ages 20 and 40.
    - **Spending Peaks and Fluctuations**:
        - For males, there are notable peaks around ages 20, 25, 30, and 40, where total spending reaches above 100,000 in some age groups.
        - For females, spending is generally more consistent but also shows fluctuations, with peaks around ages 25 and 40.
    - **Decline in Spending with Age**: After age 40, total spending for both genders declines significantly, with few data points beyond age 50, and these points reflect relatively low spending amounts.
    - **Gender Differences**: While both genders show fluctuations in spending, males tend to have higher peaks and greater variability, especially in the younger age groups.
4. **Insights for Thesis**:
    - This chart suggests that both genders exhibit high spending activity in young adulthood (20-40 years), with a sharp decline as age increases. This pattern could indicate that younger individuals have higher financial activity or disposable income, while spending decreases as people age.
    - The peaks in male spending may suggest higher variability or spending on specific activities, while the steadier trend for females could imply more consistent spending patterns.
5. **Implications**:

- This data could be used to tailor marketing or product offerings based on age and gender, focusing on high-spending younger age groups.
- Financial institutions might find this useful for designing age and gender-specific financial products, as spending behavior shows distinct patterns between these groups.

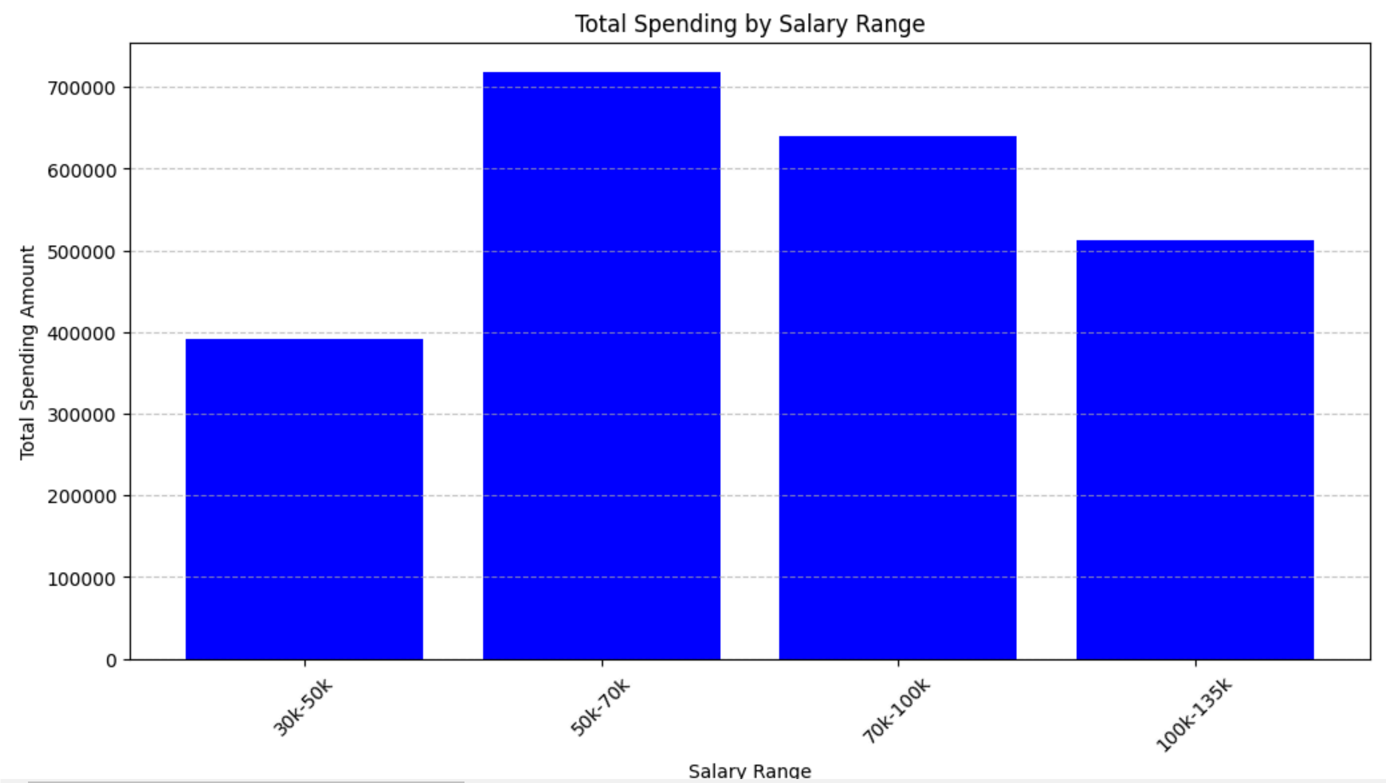
6. **Conclusion**:
   - In this thesis, you could discuss how age and gender influence spending patterns, noting the concentration of high spending in younger age groups and the more pronounced peaks in male spending. This insight could help in understanding consumer behavior and segmentation strategies.

This chart effectively illustrates how spending habits differ by age and gender, showing that total spending is highest among younger customers, with unique patterns for males and females. This insight is valuable for analyzing consumer behavior and targeting customer segments.

```
Start coding or generate with AI.
```

```
# Grouping the data by salary range to calculate total spending for each salary category
salary_spending_total = data.groupby('salary_range')['amount'].sum().sort_index()

# Creating a bar chart to visualize total spending based on salary ranges
plt.figure(figsize=(12, 6))
plt.bar(salary_spending_total.index, salary_spending_total.values, color='blue')
plt.xlabel('Salary Range')
plt.ylabel('Total Spending Amount')
plt.title('Total Spending by Salary Range')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```

```
<ipython-input-40-4902fee15cdc>:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future ve
  salary_spending_total = data.groupby('salary_range')['amount'].sum().sort_index()
```



This bar chart shows **Total Spending by Salary Range**, indicating how total spending amounts vary across different income levels.

Here's a detailed explanation:

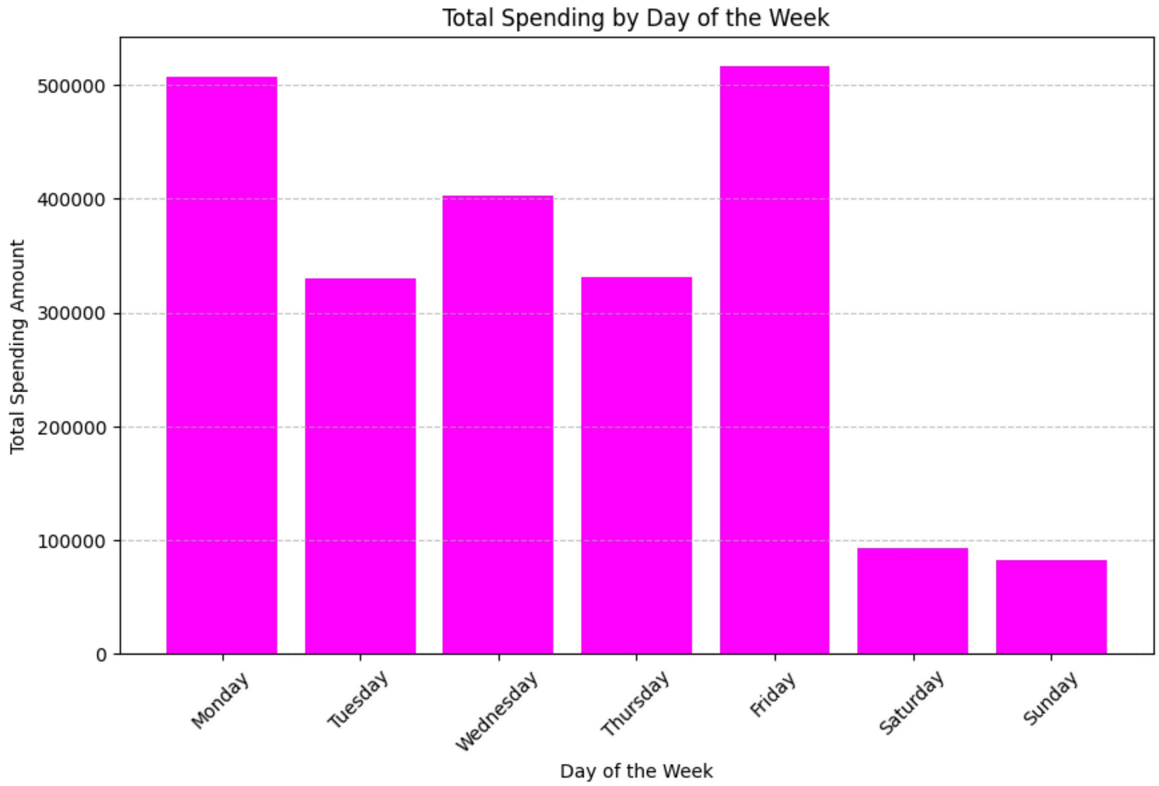
1. **X-Axis (Salary Range)**:
   - The horizontal axis categorizes individuals into four annual salary ranges: 30K–50K, 50K–70K, 70K–100K, and 100K–135K.
   - This segmentation allows for comparison of total spending across different income brackets.

2. **Y-Axis (Total Spending Amount)**:
   - The vertical axis represents the total spending amount, with values up to 700,000.
   - Each bar's height indicates the cumulative spending amount for each salary range.
3. **Key Observations**:
   - **50K–70K** has the highest total spending amount, reaching around 700,000, indicating that individuals in this income range contribute significantly to total spending.
   - **70K–100K** has the second-highest total spending, slightly lower than the 50K–70K range.
   - **100K–135K** and **30K–50K** show similar total spending amounts, both of which are lower than the middle-income ranges (50K–100K).
4. **Insights for Your Thesis**:
   - This chart suggests that total spending does not increase uniformly with higher income. Instead, there appears to be a peak in total spending for individuals in the 50K–70K income range.
   - The similar spending levels for 30K–50K and 100K–135K may indicate that individuals in these ranges have different spending behaviors or priorities compared to those in the middle-income ranges.
5. **Implications**:
   - Businesses could use this data to target marketing or financial products more effectively, focusing on the 50K–70K and 70K–100K income brackets where total spending is higher.
   - This insight could also be useful for understanding how disposable income influences spending, especially if other factors (such as debt or savings rates) affect spending behavior in the highest and lowest income groups.
6. **Conclusion**:
   - In your thesis, you could discuss the spending patterns across income levels, noting that the highest spending is not concentrated in the highest income bracket. This could support discussions on consumer behavior and spending tendencies across different salary ranges.

This bar chart effectively illustrates how spending varies by income, with the highest spending seen in middle-income brackets rather than at the extremes. This insight can be valuable for understanding consumer behavior and targeting customer segments.

```
Start coding or generate with AI.
```

```
# Grouping the data by day of the week and calculating the total spending for each day
day_spending_total = data.groupby('dow')['amount'].sum().reindex(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Su

# Creating a bar chart to visualize total spending by day of the week
plt.figure(figsize=(10, 6))
plt.bar(day_spending_total.index, day_spending_total.values, color='magenta')
plt.xlabel('Day of the Week')
plt.ylabel('Total Spending Amount')
plt.title('Total Spending by Day of the Week')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



This bar chart displays **Total Spending by Day of the Week**, showing how total spending amounts vary across the days of the week.

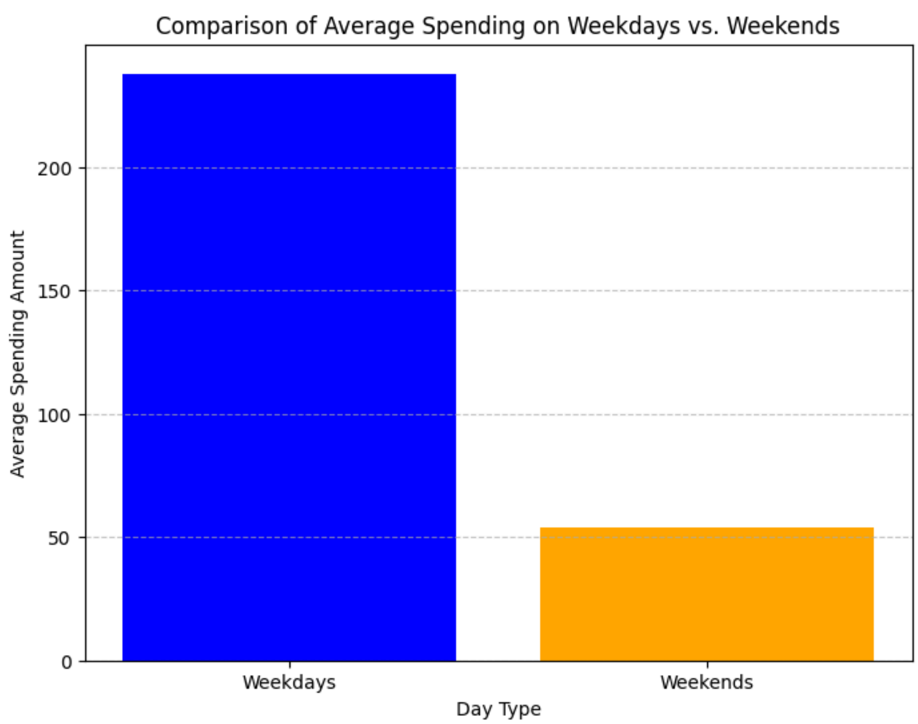
Here's a detailed explanation:

1. **X-Axis (Day of the Week)**:
    - The horizontal axis represents each day of the week, from Monday to Sunday.
    - This allows for a comparison of spending patterns on each day.
2. **Y-Axis (Total Spending Amount)**:
    - The vertical axis represents the total spending amount, with values up to 500,000.
    - Each bar's height indicates the cumulative spending amount for each day of the week.
3. **Key Observations**:
    - **Highest Spending on Monday and Friday**: The highest total spending amounts occur on Monday and Friday, both reaching around 500,000. This suggests that spending peaks at the beginning and end of the work week.
    - **Moderate Spending Midweek**: Wednesday and Thursday show moderate total spending amounts, lower than Monday and Friday but still significant.
    - **Lowest Spending on the Weekend**: Saturday and Sunday have the lowest spending amounts, with totals much lower than weekdays. This indicates reduced spending activity on weekends.
4. **Insights for Your Thesis**:
    - This chart suggests that spending activity is concentrated on weekdays, particularly on Monday and Friday. This may be due to routines such as starting the week with necessary purchases and preparing for the weekend on Fridays.
    - The lower spending on weekends could reflect changes in consumer behavior, possibly due to fewer work-related expenses or other routines that decrease spending.
5. **Implications**:
    - Businesses could use this data to optimize operations or marketing efforts, focusing on peak spending days like Monday and Friday.
    - Financial institutions might consider adjusting staffing or resources for transaction processing based on the lower spending volumes expected over the weekend.
6. **Conclusion**:
    - In your thesis, you could discuss how spending behavior varies by day, noting the concentration of spending at the start and end of the work week. This could support insights into consumer habits and spending patterns.

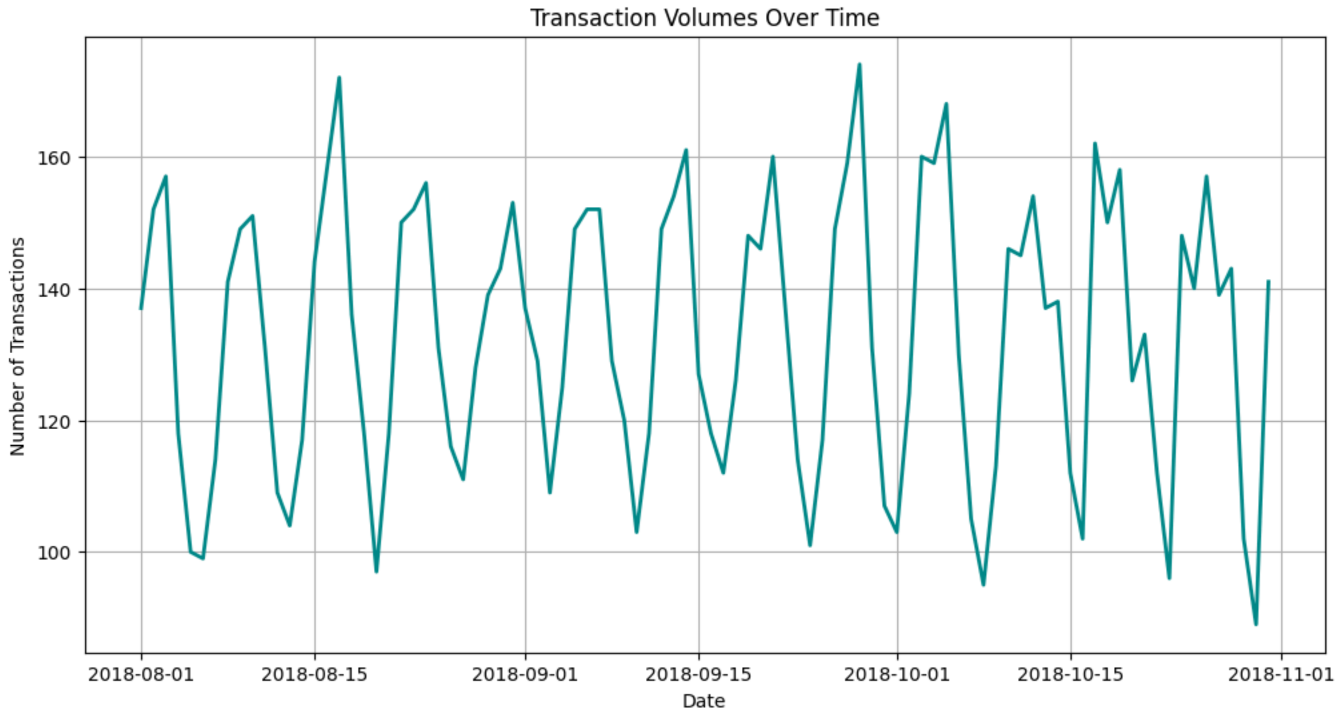
This bar chart highlights that spending is higher on weekdays, particularly at the start and end of the work week, with significantly lower activity on weekends. This insight can inform business strategies around timing and consumer engagement.

```
Start coding or generate with AI.
```

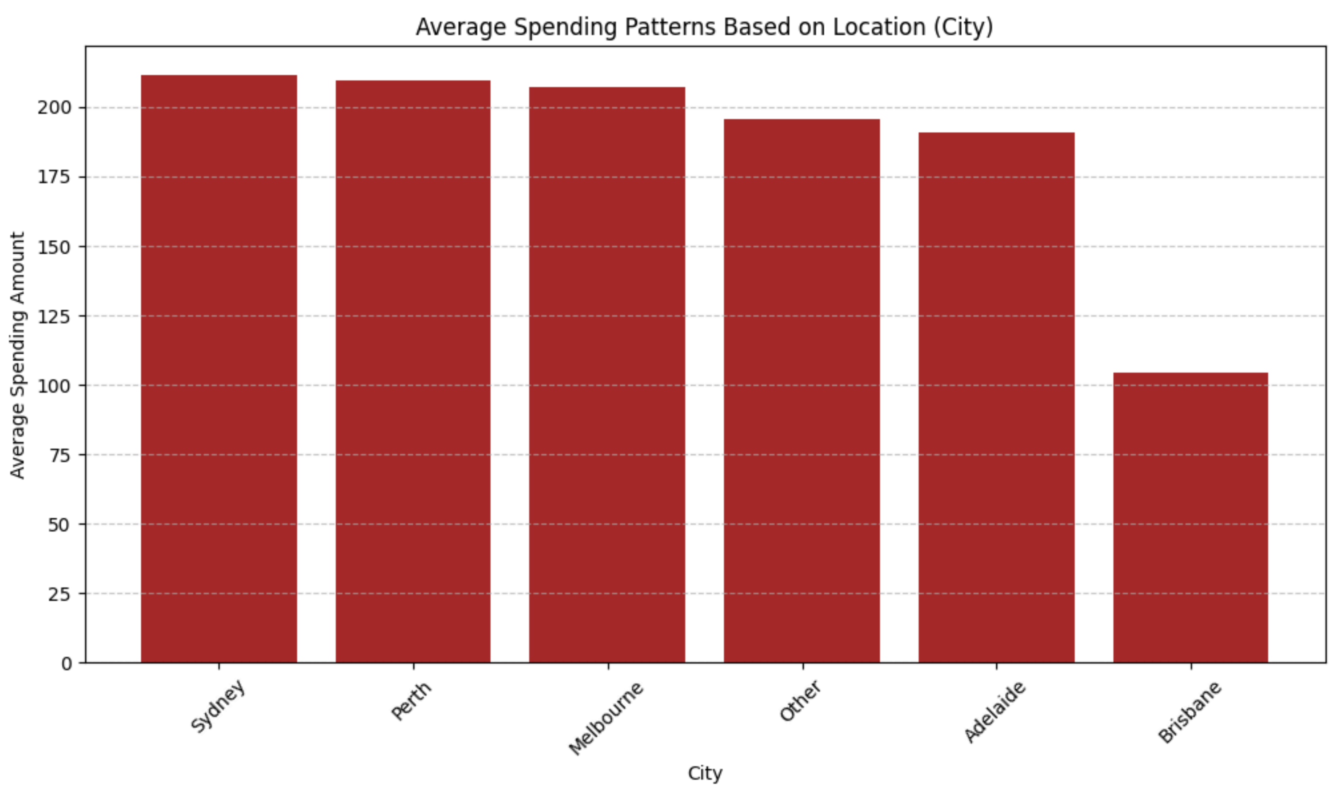
```
# Filtering the data for weekends (Saturday and Sunday)
weekend_data = data[data['dow'].isin(['Saturday', 'Sunday'])]

# Calculating the average spending on weekends
average_weekend_spending = weekend_data['amount'].mean()

# Filtering the data for weekdays (Monday to Friday)
weekday_data = data[~data['dow'].isin(['Saturday', 'Sunday'])]

# Calculating the average spending on weekdays
average_weekday_spending = weekday_data['amount'].mean()

# Creating a bar chart to compare average spending on weekends and weekdays
plt.figure(figsize=(8, 6))
plt.bar(['Weekdays', 'Weekends'], [average_weekday_spending, average_weekend_spending], color=['blue', 'orange'])
plt.xlabel('Day Type')
plt.ylabel('Average Spending Amount')
plt.title('Comparison of Average Spending on Weekdays vs. Weekends')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```



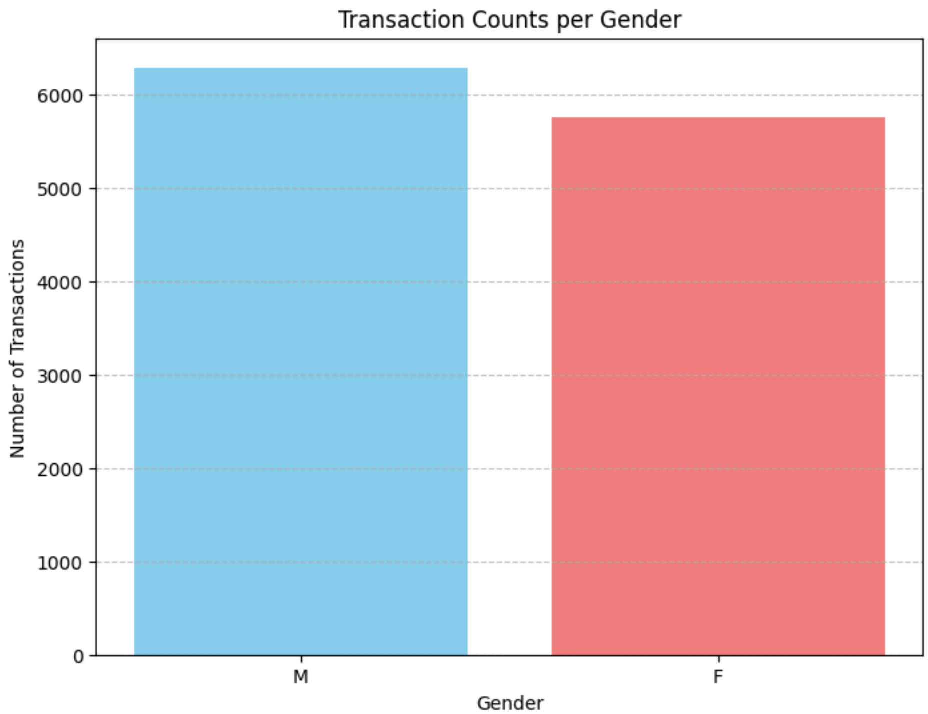

```
# Converting the 'date' column to datetime format for better time-based analysis
data['date'] = pd.to_datetime(data['date'], errors='coerce')

# Grouping the data by date and counting the number of transactions per day
daily_transaction_volume = data.groupby('date').size()

# Creating a line chart to visualize transaction volumes over time
plt.figure(figsize=(12, 6))
plt.plot(daily_transaction_volume.index, daily_transaction_volume.values, color='darkcyan', linewidth=2)
plt.xlabel('Date')
plt.ylabel('Number of Transactions')
plt.title('Transaction Volumes Over Time')
plt.grid(True)
plt.show()

```




```python
# Grouping the data by city to calculate average spending for each location
location_spending = data.groupby('city')['amount'].mean().sort_values(ascending=False)

# Creating a bar chart to visualize average spending based on location (city)
plt.figure(figsize=(12, 6))
plt.bar(location_spending.index, location_spending.values, color='brown')
plt.xlabel('City')
plt.ylabel('Average Spending Amount')
plt.title('Average Spending Patterns Based on Location (City)')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```

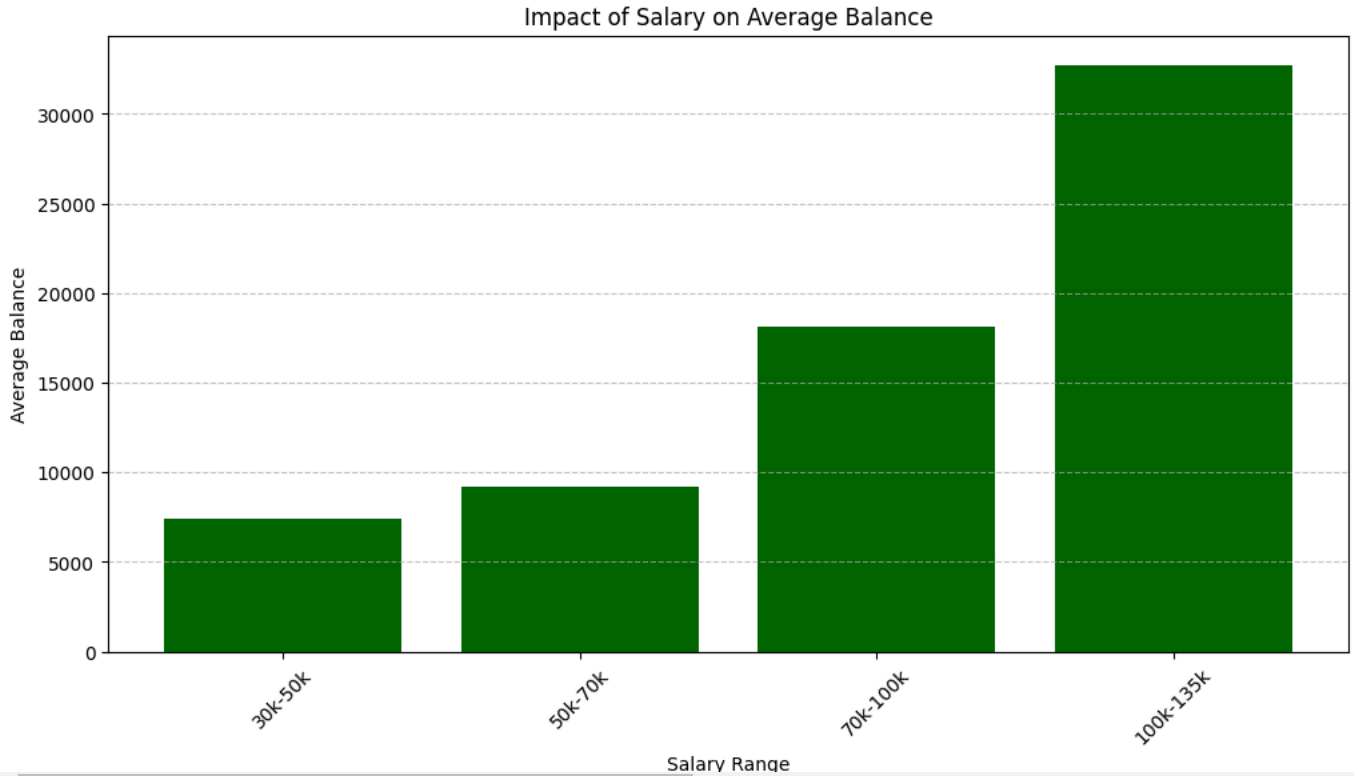


```python
# Grouping the data by gender and counting the number of transactions for each gender
transaction_counts_gender = data['gender'].value_counts()

# Creating a bar chart to visualize the transaction counts per gender
plt.figure(figsize=(8, 6))
plt.bar(transaction_counts_gender.index, transaction_counts_gender.values, color=['skyblue', 'lightcoral'])
plt.xlabel('Gender')
plt.ylabel('Number of Transactions')
plt.title('Transaction Counts per Gender')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```

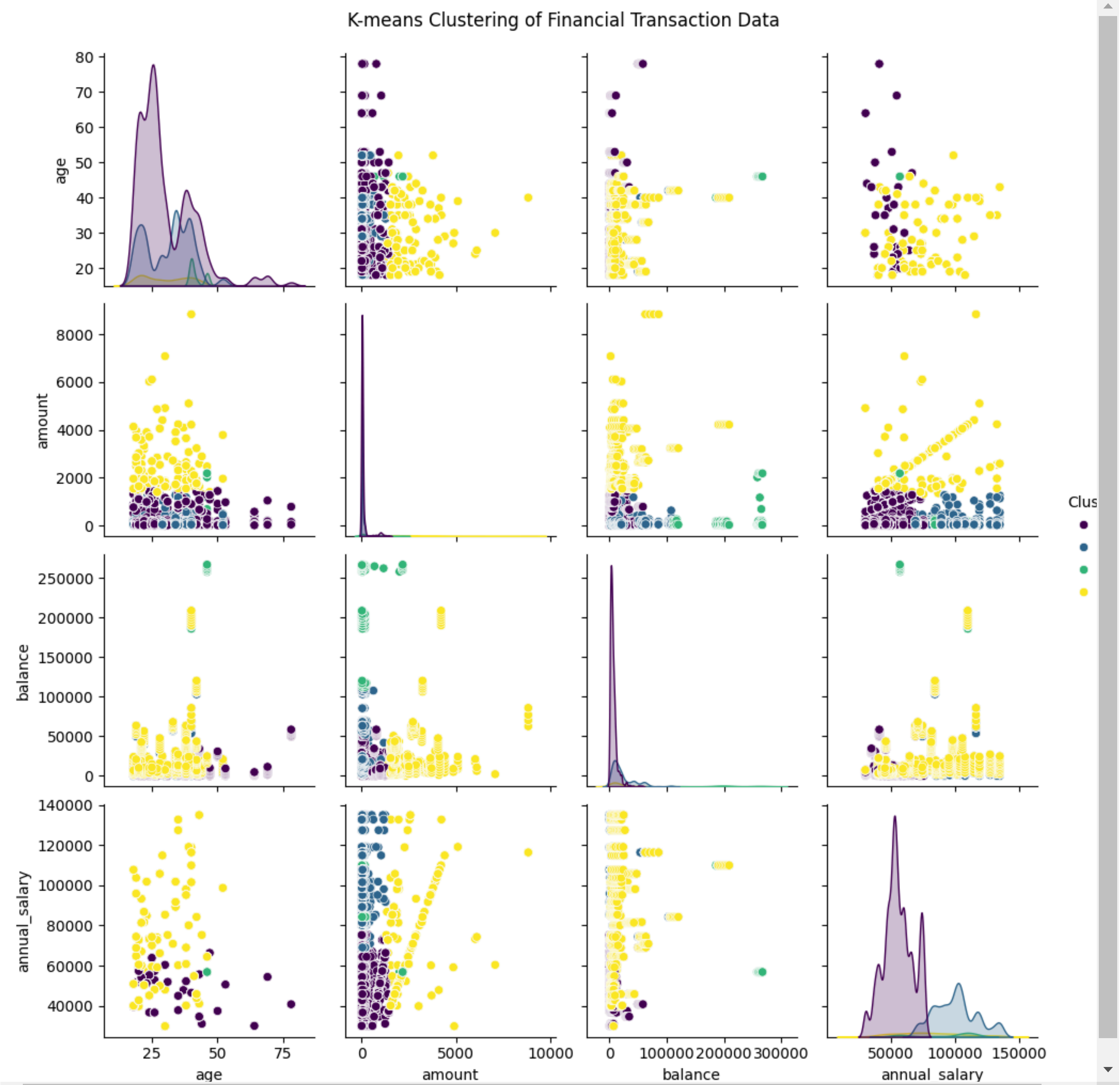



```python
# Grouping the data by salary range and calculating the average balance for each salary range
salary_balance = data.groupby('salary_range')['balance'].mean().sort_index()

# Creating a bar chart to visualize the impact of salary on balance
plt.figure(figsize=(12, 6))
plt.bar(salary_balance.index, salary_balance.values, color='darkgreen')
plt.xlabel('Salary Range')
plt.ylabel('Average Balance')
plt.title('Impact of Salary on Average Balance')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```

```
<ipython-input-46-5b111246edf9>:2: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future ve
  salary_balance = data.groupby('salary_range')['balance'].mean().sort_index()
```



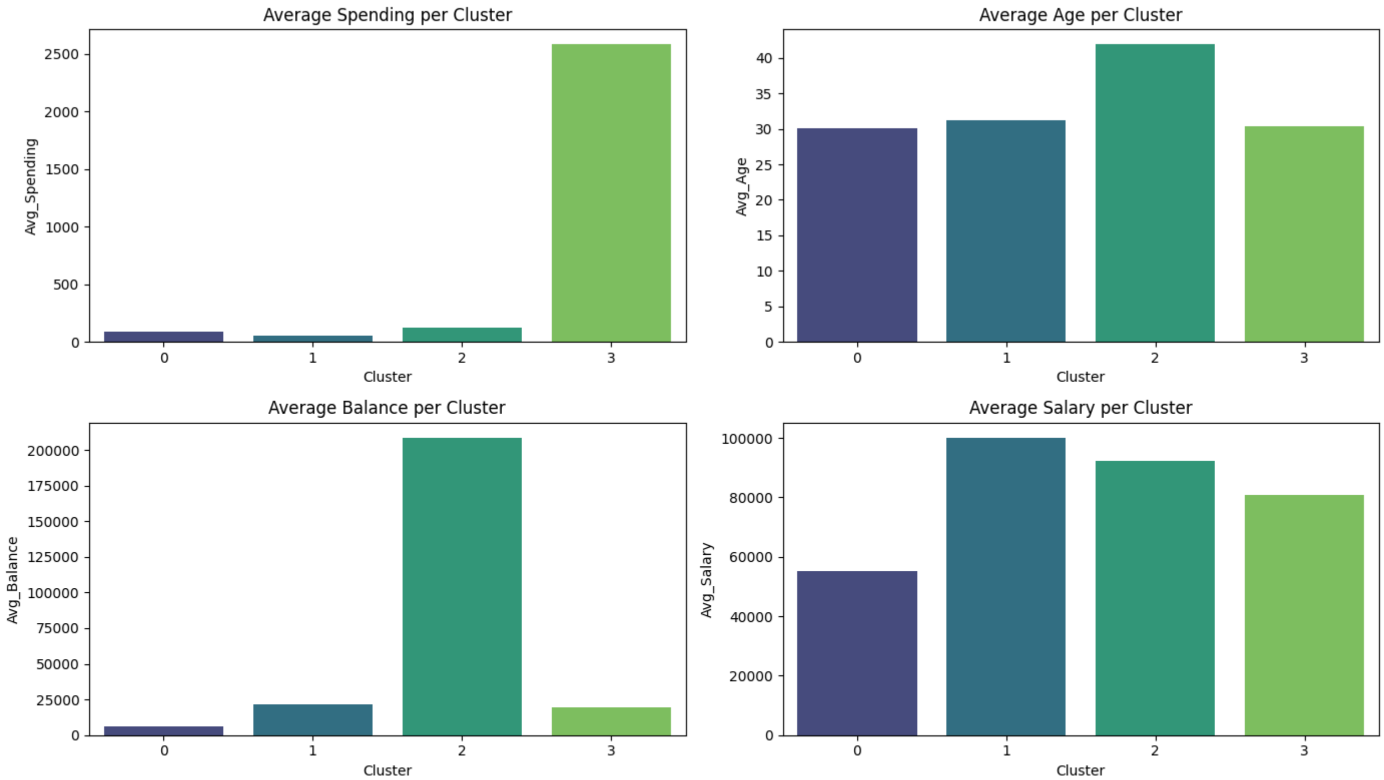

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import seaborn as sns

# Selecting relevant numerical columns for clustering
columns_for_clustering = ['age', 'amount', 'balance', 'annual_salary']

# Dropping rows with missing values in these columns for clean clustering
clustering_data = data[columns_for_clustering].dropna()

# Standardizing the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(clustering_data)

# Applying K-means clustering with an arbitrary number of clusters (e.g., 4)
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(scaled_data)

# Adding the cluster labels to the original data
clustering_data['Cluster'] = clusters

# Visualizing the clusters using a pairplot to see relationships between variables
sns.pairplot(clustering_data, hue='Cluster', diag_kind='kde', palette='viridis')
plt.suptitle('K-means Clustering of Financial Transaction Data', y=1.02)
plt.show()

```




```
# Selecting relevant numerical columns for clustering
columns_for_clustering = ['age', 'amount', 'balance', 'annual_salary']

# Dropping rows with missing values in these columns for clean clustering
clustering_data = data[columns_for_clustering].dropna()

# Standardizing the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(clustering_data)

# Applying K-means clustering with 4 clusters
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(scaled_data)

# Adding the cluster labels to the original data
clustering_data['Cluster'] = clusters
data['Cluster'] = -1  # Initialize all rows with -1 (as non-clustered)
data.loc[clustering_data.index, 'Cluster'] = clustering_data['Cluster']
```

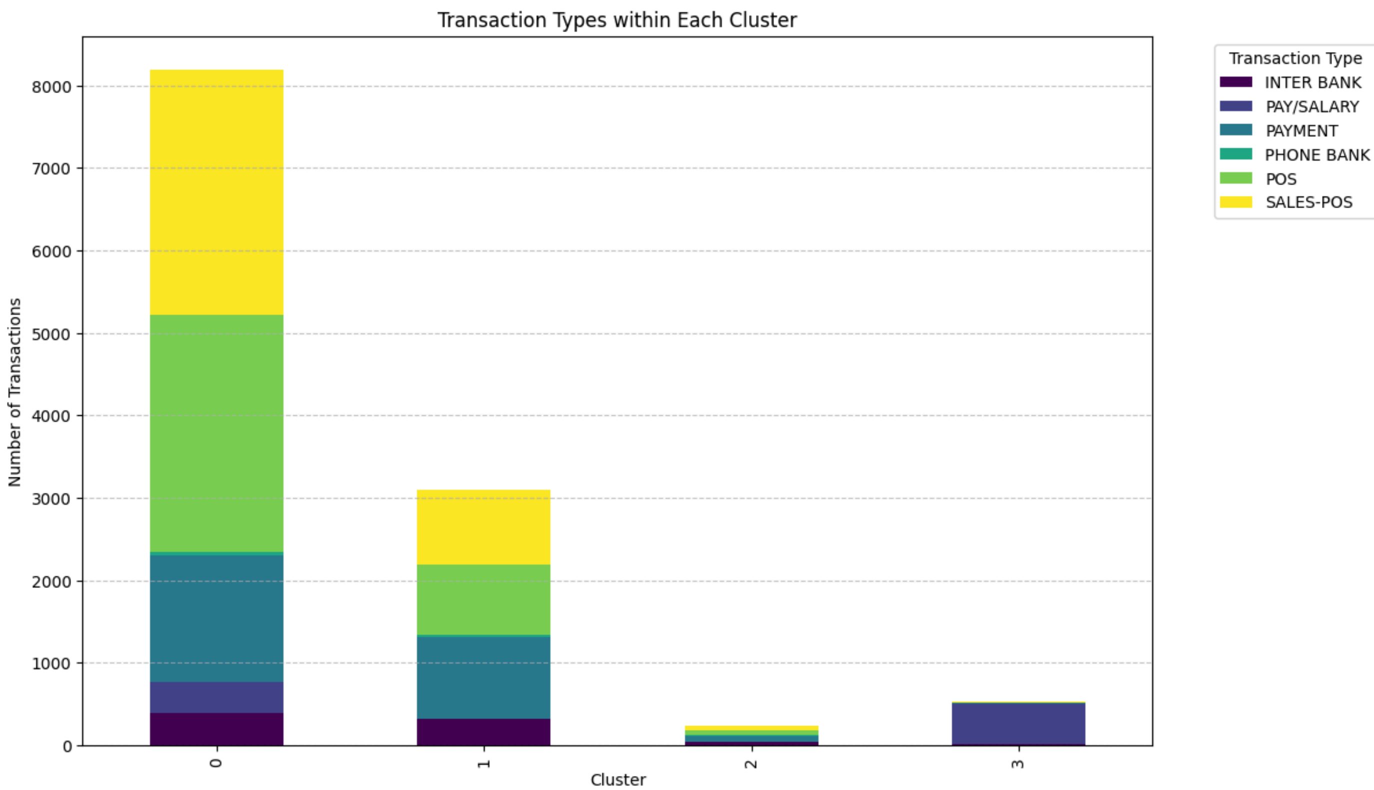
```
# Grouping data by cluster to analyze average spending behavior within each cluster
cluster_analysis = data.groupby('Cluster').agg({
    'amount': ['mean', 'median', 'std'],
    'age': ['mean', 'median'],
    'balance': ['mean', 'median'],
    'annual_salary': ['mean', 'median'],
    'customer_id': 'count'  # Count the number of transactions in each cluster
}).reset_index()

# Renaming columns for clarity
cluster_analysis.columns = ['Cluster', 'Avg_Spending', 'Median_Spending', 'Spending_StdDev',
                            'Avg_Age', 'Median_Age', 'Avg_Balance', 'Median_Balance',
                            'Avg_Salary', 'Median_Salary', 'Transaction_Count']


import matplotlib.pyplot as plt
import seaborn as sns

# Set the figure size for the plots
plt.figure(figsize=(14, 8))

# Creating subplots for each variable we want to visualize
variables = ['Avg_Spending', 'Avg_Age', 'Avg_Balance', 'Avg_Salary']
titles = ['Average Spending per Cluster', 'Average Age per Cluster',
          'Average Balance per Cluster', 'Average Salary per Cluster']

# Iterate over variables and create bar plots for each
for i, var in enumerate(variables):
    plt.subplot(2, 2, i + 1)
    sns.barplot(x='Cluster', y=var, data=cluster_analysis, palette='viridis')
    plt.title(titles[i])
    plt.xlabel('Cluster')
    plt.ylabel(var)

# Adjust the layout for clarity
plt.tight_layout()
plt.show()



```

```
<ipython-input-48-a2c1638c36e3>:49: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x='Cluster', y=var, data=cluster_analysis, palette='viridis')
<ipython-input-48-a2c1638c36e3>:49: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x='Cluster', y=var, data=cluster_analysis, palette='viridis')
<ipython-input-48-a2c1638c36e3>:49: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x='Cluster', y=var, data=cluster_analysis, palette='viridis')
<ipython-input-48-a2c1638c36e3>:49: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x='Cluster', y=var, data=cluster_analysis, palette='viridis')
```




```python

# Adding the cluster labels back to the dataset from previous clustering
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# Selecting relevant numerical columns for clustering
columns_for_clustering = ['age', 'amount', 'balance', 'annual_salary']

# Dropping rows with missing values in these columns for clean clustering
clustering_data = data[columns_for_clustering].dropna()

# Standardizing the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(clustering_data)

```

```
# Applying K-means clustering with 4 clusters
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(scaled_data)

# Adding the cluster labels to the original data
clustering_data['Cluster'] = clusters
data['Cluster'] = -1  # Initialize all rows with -1 (as non-clustered)
data.loc[clustering_data.index, 'Cluster'] = clustering_data['Cluster']

# Grouping data by cluster and transaction type to count the number of transactions in each type per cluster
transaction_type_cluster = data.groupby(['Cluster', 'txn_description']).size().unstack().fillna(0)

# Visualizing the transaction types within each cluster using a stacked bar plot
transaction_type_cluster.plot(kind='bar', stacked=True, figsize=(12, 8), colormap='viridis')
plt.xlabel('Cluster')
plt.ylabel('Number of Transactions')
plt.title('Transaction Types within Each Cluster')
plt.legend(title='Transaction Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

```




```


# Selecting relevant numerical columns for clustering
columns_for_clustering = ['age', 'amount', 'balance', 'annual_salary']

# Dropping rows with missing values in these columns for clean clustering
clustering_data = data[columns_for_clustering].dropna()

# Standardizing the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(clustering_data)

# Applying K-means clustering with 4 clusters
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(scaled_data)

# Adding the cluster labels to the original data
```

```
clustering_data['Cluster'] = clusters
data['Cluster'] = -1  # Initialize all rows with -1 (as non-clustered)
data.loc[clustering_data.index, 'Cluster'] = clustering_data['Cluster']

# Grouping data by cluster to analyze average spending behavior within each cluster
cluster_analysis = data.groupby('Cluster').agg({
    'amount': ['mean', 'median', 'std'],
    'age': ['mean', 'median'],
    'balance': ['mean', 'median'],
    'annual_salary': ['mean', 'median'],
    'customer_id': 'count'  # Count the number of transactions in each cluster
}).reset_index()

# Renaming columns for clarity
cluster_analysis.columns = ['Cluster', 'Avg_Spending', 'Median_Spending', 'Spending_StdDev'
                            'Avg_Age', 'Median_Age', 'Avg_Balance', 'Median_Balance',
                            'Avg_Salary', 'Median_Salary', 'Transaction_Count']

# Visualizing the average balance per cluster using a bar plot
plt.figure(figsize=(10, 6))
plt.bar(cluster_analysis['Cluster'], cluster_analysis['Avg_Balance'], color='teal')
plt.xlabel('Cluster')
plt.ylabel('Average Balance')
plt.title('Average Balance per Cluster')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Average Balance per Cluster