Personal Transaction Analysis

HDFC Bank Account

```
# load the required Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


# Initilise the drive for dataset
from google.colab import drive

# initilise tyhe path
drive.mount('/content/drive')

# Setting Address
import os
os.chdir('/content/drive/My Drive/dummy folder')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Load the data (HDFC bank Statement)
# This data is from 1st April 2022 to 31st March 2023
# Load the data
df_HDFC = pd.read_excel("my bank statement.xlsx")
df_HDFC.head()
```

	Date	Narration	Withdrawal Amt.	Deposit Amt.	Closing Balance	Month	Year
0	2022-04-04	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	NaN	200.0	442.19	4	2022
1	2022-04-07	UPI-ANIL NARBAT BISEN-ANILBISEN229@OKSBI	NaN	50.0	492.19	4	2022
2	2022-04-07	UPI-SHRIRAM GENERAL STOR-GPAY-1119808673	50.0	NaN	442.19	4	2022
3	2022-04-12	UPI-IDRISH SHHA-BHARATPE.9051322211@FBP	30.0	NaN	412.19	4	2022
4	2022-04-13	UPI-RAMESHWAR VASANTRAO	NaN	100.0	512.19	4	2022

```
# Convert 'Date' column to datetime data type
df_HDFC['Date'] = pd.to_datetime(df_HDFC['Date'])

# Handling missing values (if any)
df_HDFC.fillna(0, inplace=True)  # Replace NaN with 0 for Withdrawal and Deposit Amounts

# Drop unnecessary columns (if needed)
df_HDFC.drop(['Month', 'Year'], axis=1, inplace=True)


df_HDFC.head()
```

	Date	Narration	Withdrawal Amt.	Deposit Amt.	Closing Balance
0	2022-04-04	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	0.0	200.0	442.19
1	2022-04-07	UPI-ANIL NARBAT BISEN-ANILBISEN229@OKSBI	0.0	50.0	492.19
2	2022-04-07	UPI-SHRIRAM GENERAL STOR-GPAY-1119808673	50.0	0.0	442.19
3	2022-04-12	UPI-IDRISH SHHA-BHARATPE.9051322211@FBP	30.0	0.0	412.19
4	2022-04-13	UPI-RAMESHWAR VASANTRAO	0.0	100.0	512.19

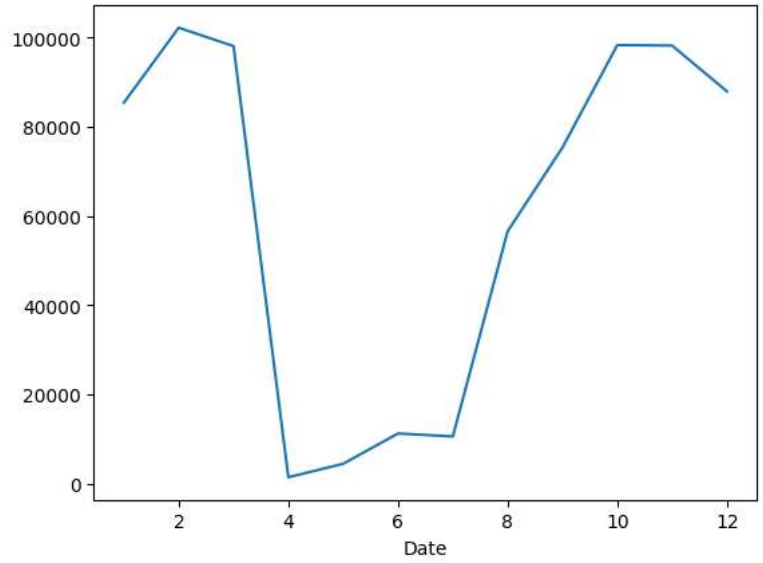
```
# Total income and total expenses
total_income = df_HDFC['Deposit Amt.'].sum()
total_expenses = df_HDFC['Withdrawal Amt.'].sum()

# Print those
print("total_income: ",total_income)
```

```
print("total_expenses: ",total_expenses )
```

```
total_income:  729578.0
total_expenses:  729503.33
```

```
# Monthly spending patterns
monthly_expenses = df_HDFC.groupby(df_HDFC['Date'].dt.month)['Withdrawal Amt.'].sum()
monthly_expenses.plot()
```

```
<Axes: xlabel='Date'>
```

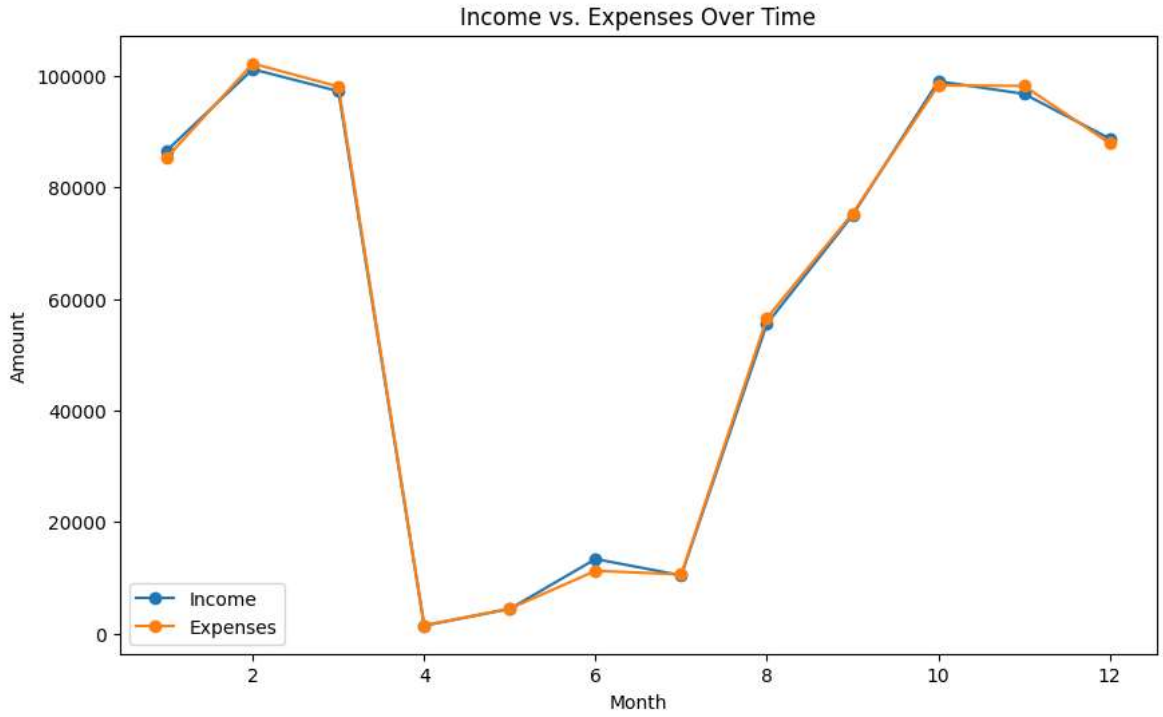



```
# Major expenses
top_expenses = df_HDFC.nlargest(5, 'Withdrawal Amt.')
top_expenses
```

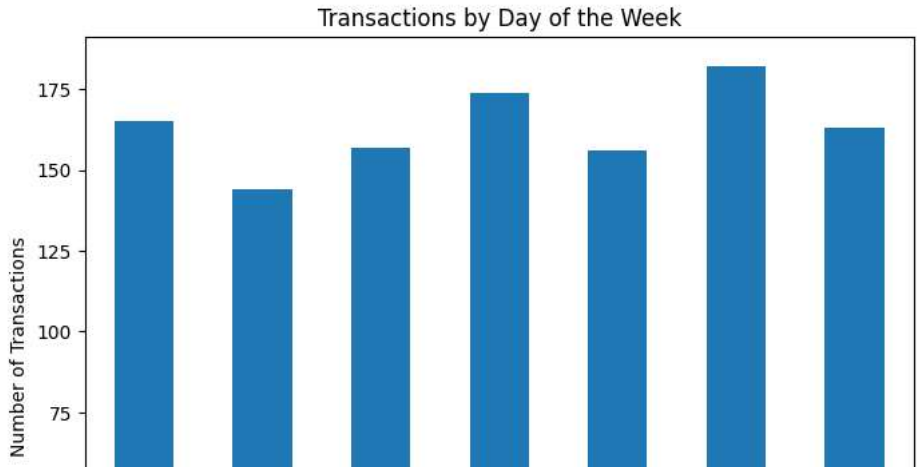
	Date	Narration	Withdrawal Amt.	Deposit Amt.	Closing Balance
563	2022-11-11	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	50000.0	0.0	18543.57
656	2022-12-02	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	50000.0	0.0	2765.81
790	2023-01-04	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	50000.0	0.0	4087.80
945	2023-02-13	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	50000.0	0.0	12579.29
1028	2023-03-02	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	50000.0	0.0	3888.27

```
# Average Monthly Balance:
# Calculate the average monthly balance based on the closing balance for each month.
df_HDFC['Month'] = df_HDFC['Date'].dt.month
average_monthly_balance = df_HDFC.groupby('Month')['Closing Balance'].mean()
average_monthly_balance.plot(kind = "bar")
plt.show()
```


```
# Income vs. Expenses Over Time:
# Visualize the trend of your income and expenses over the months.
monthly_income = df_HDFC.groupby('Month')['Deposit Amt.'].sum()
monthly_expenses = df_HDFC.groupby('Month')['Withdrawal Amt.'].sum()

plt.figure(figsize=(10, 6))
plt.plot(monthly_income.index, monthly_income, label='Income', marker='o')
plt.plot(monthly_expenses.index, monthly_expenses, label='Expenses', marker='o')
plt.xlabel('Month')
plt.ylabel('Amount')
plt.title('Income vs. Expenses Over Time')
plt.legend()
plt.show()
```



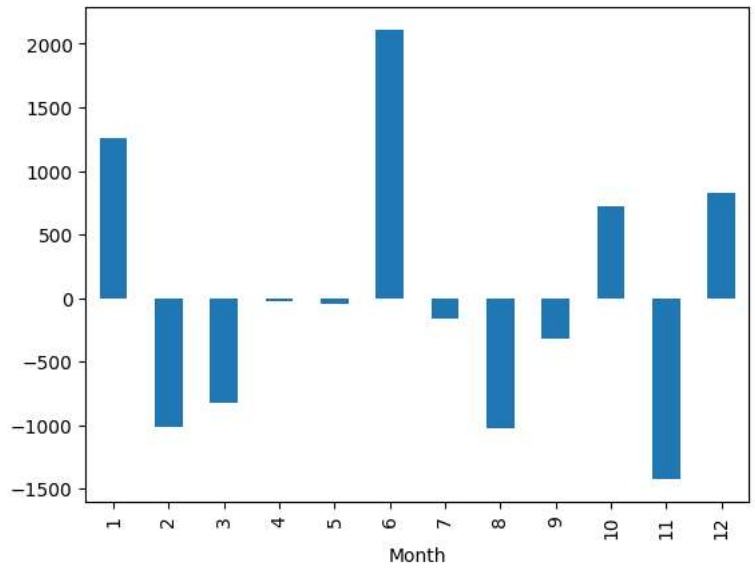

```
# Day of the Week Analysis:
# Check if there's any pattern in your transactions based on the day of the week.
df_HDFC['Day_of_Week'] = df_HDFC['Date'].dt.day_name()
transactions_by_day = df_HDFC.groupby('Day_of_Week').size()

plt.figure(figsize=(8, 6))
transactions_by_day.plot(kind='bar')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Transactions')
plt.title('Transactions by Day of the Week')
plt.show()
```




```
# Top Income Sources:
# If the 'Narration' column contains details about income sources, you can identify the top sources.
top_income_sources = df_HDFC.nlargest(5, 'Deposit Amt.')
top_income_sources
```

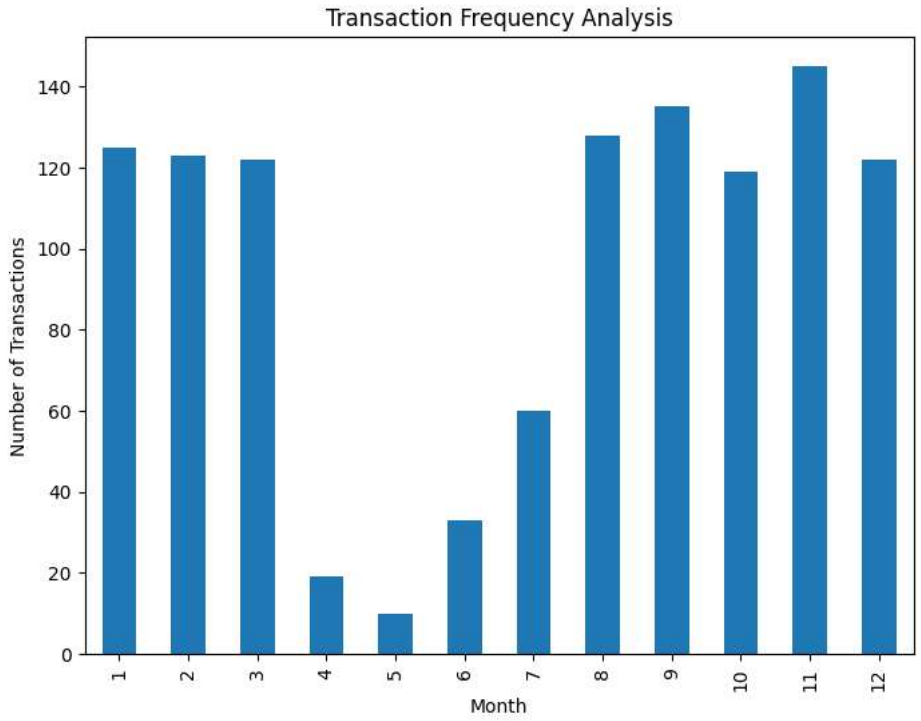
	Date	Narration	Withdrawal Amt.	Deposit Amt.	Closing Balance	Month	Day_of_Week
1027	2023-03-02	UPI-AMAN JANGID-PAYTQR6377020338@PAYTM-B	0.0	53527.0	53888.27	3	Thursday
789	2023-01-04	UPI-AMAN JANGID-PAYTQR6377020338@PAYTM-B	0.0	53500.0	54087.80	1	Wednesday
898	2023-02-02	UPI-AMAN JANGID-PAYTQR6377020338@PAYTM-B	0.0	52858.0	54798.80	2	Thursday
507	2022-11-01	UPI-AMAN JANGID-PAYTQR6377020338@PAYTM-B	0.0	52650.0	54195.57	11	Tuesday
653	2022-12-02	IMPS-233611598080-AMAN JANGID-PYTM-XXXXX	0.0	52195.0	52815.81	12	Friday

```
# Net Cash Flow:
# Calculate the net cash flow for each month (income minus expenses).
df_HDFC['Net_Cash_Flow'] = df_HDFC['Deposit Amt.'] - df_HDFC['Withdrawal Amt.']
monthly_cash_flow = df_HDFC.groupby('Month')['Net_Cash_Flow'].sum()
monthly_cash_flow.plot(kind = "bar")
plt.show()
```



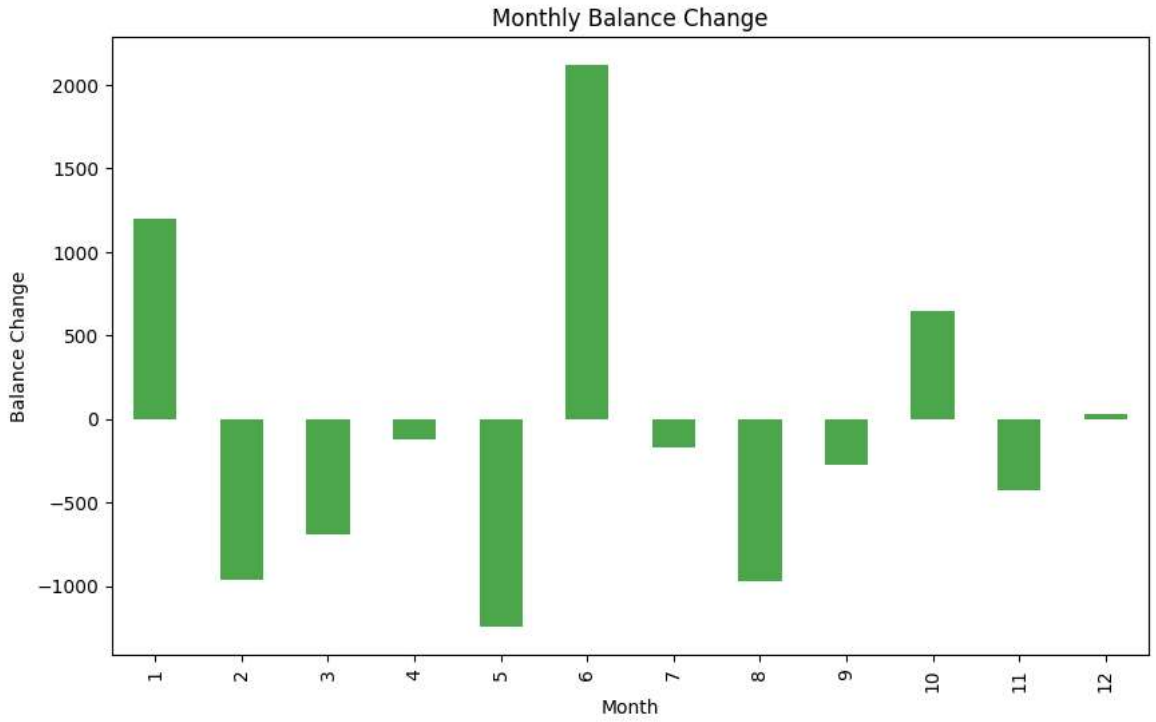

```
# Transaction Frequency Analysis:
# Analyze the frequency of transactions during the year.
transaction_frequency = df_HDFC.groupby('Month').size()

plt.figure(figsize=(8, 6))
transaction_frequency.plot(kind='bar')
plt.xlabel('Month')
plt.ylabel('Number of Transactions')
plt.title('Transaction Frequency Analysis')
```

```
plt.show()
```

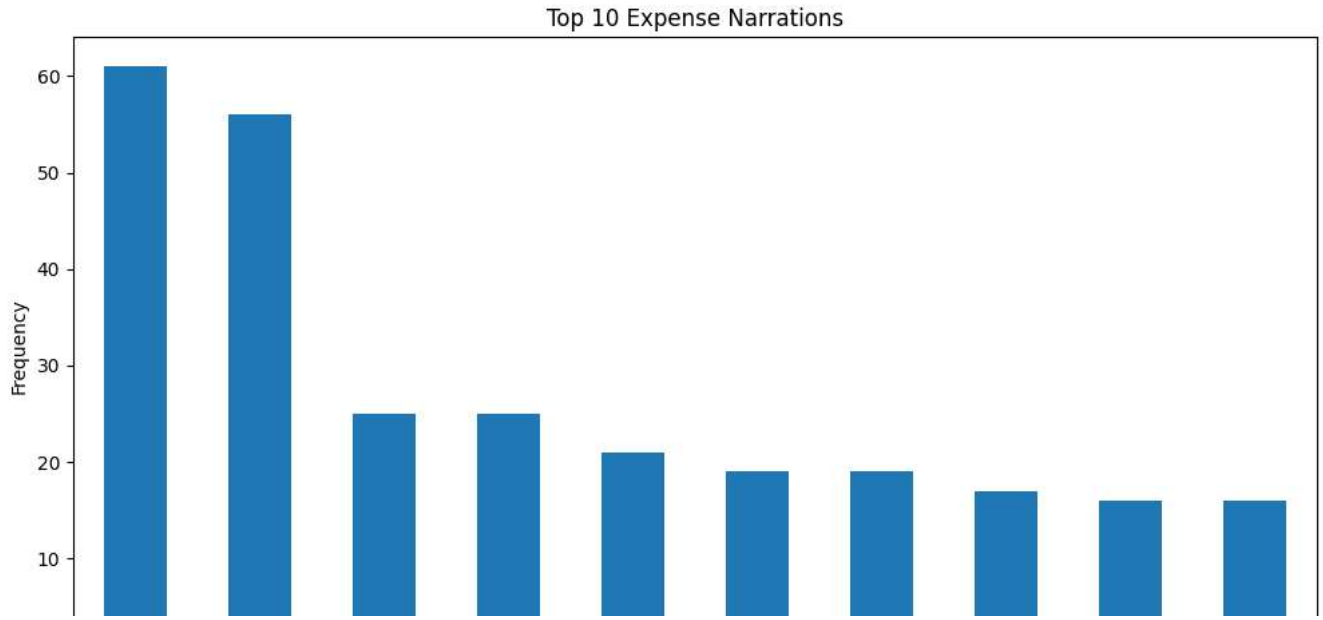



```
# Expenses by Day of the Week:
# Analyze your expenses on each day of the week to identify spending patterns.
expenses_by_day_of_week = df_HDFC[df_HDFC['Withdrawal Amt.'] > 0].groupby('Day_of_Week')['Withdrawal Amt.'].sum()

plt.figure(figsize=(8, 6))
expenses_by_day_of_week.plot(kind='bar')
plt.xlabel('Day of the Week')
plt.ylabel('Total Expenses')
plt.title('Expenses by Day of the Week')
plt.show()
```

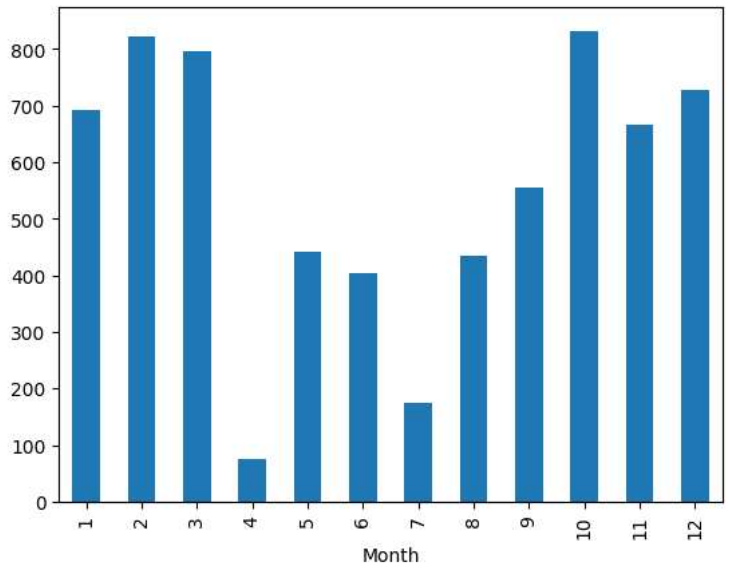
```
# Monthly Balance Change:
# Plot the change in your account balance over the months to see how it fluctuates.
monthly_balance_change = df_HDFC.groupby('Month')['Closing Balance'].last() - df_HDFC.groupby('Month')['Closing Balance'].first()

plt.figure(figsize=(10, 6))
monthly_balance_change.plot(kind='bar', color='g', alpha=0.7)
plt.xlabel('Month')
plt.ylabel('Balance Change')
plt.title('Monthly Balance Change')
plt.show()
```




```
# Expense Distribution by Narration:
# Analyze the distribution of expenses based on the descriptions in the 'Narration' column.
plt.figure(figsize=(12, 6))
top_narrations = df_HDFC['Narration'].value_counts().nlargest(10)
top_narrations.plot(kind='bar')
plt.xlabel('Narration')
plt.ylabel('Frequency')
plt.title('Top 10 Expense Narrations')
plt.show()
```

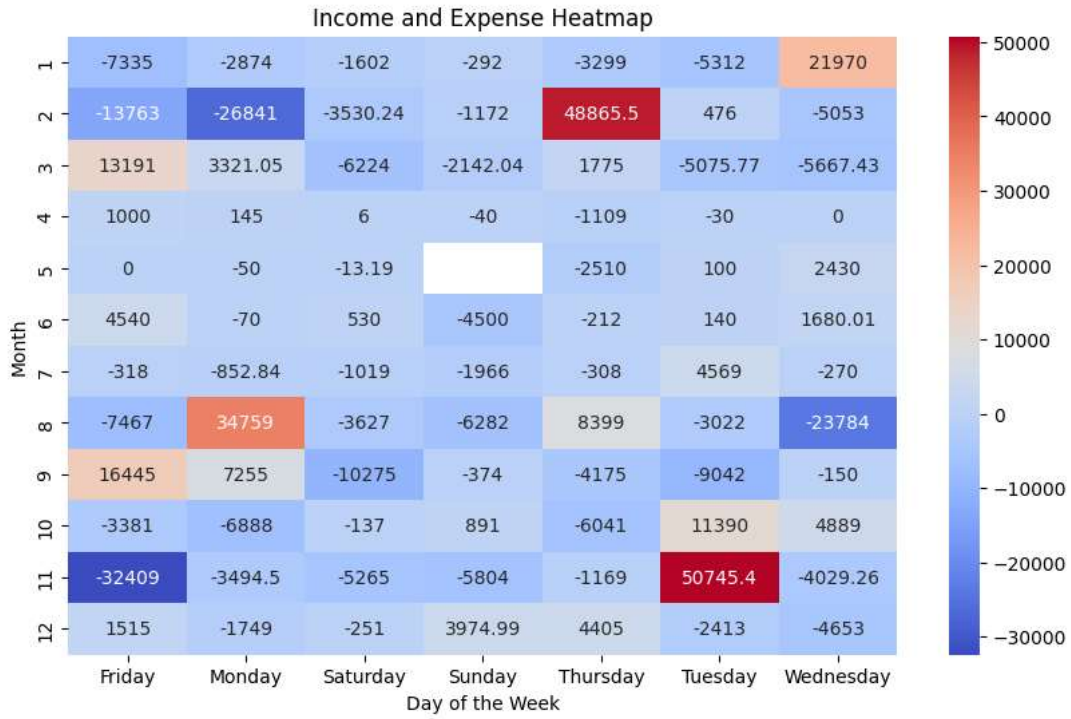



```
# Average Withdrawal and Deposit Amounts:
# Calculate the average withdrawal and deposit amounts for each month.
average_withdrawal_per_month = df_HDFC.groupby('Month')['Withdrawal Amt.'].mean()
average_deposit_per_month = df_HDFC.groupby('Month')['Deposit Amt.'].mean()
average_deposit_per_month.plot(kind = "bar")
plt.show()
```

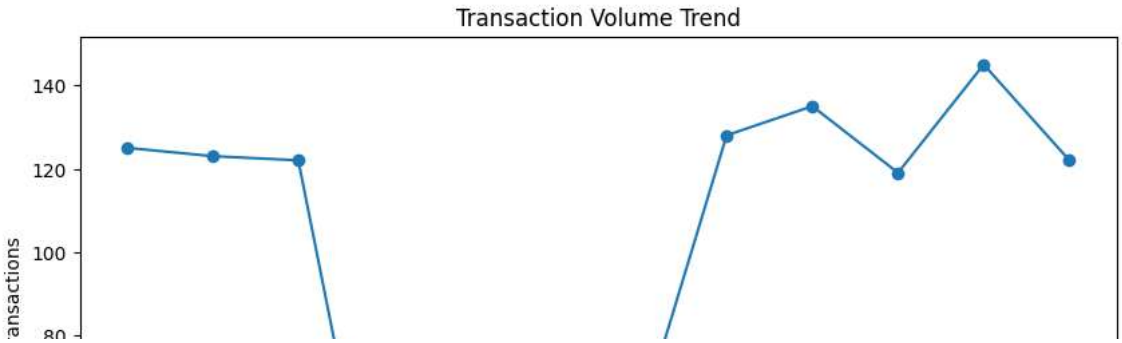



```
average_withdrawal_per_month.plot(kind = "line")
plt.show()
```


```
# Income and Expense Heatmap:
# Create a heatmap to visualize the distribution of income and expenses across different months.
income_expense_heatmap = df_HDFC.pivot_table(index='Month', columns='Day_of_Week', values='Net_Cash_Flow', aggfunc='sum')

plt.figure(figsize=(10, 6))
sns.heatmap(income_expense_heatmap, cmap='coolwarm', annot=True, fmt='g')
plt.xlabel('Day of the Week')
plt.ylabel('Month')
plt.title('Income and Expense Heatmap')
plt.show()
```



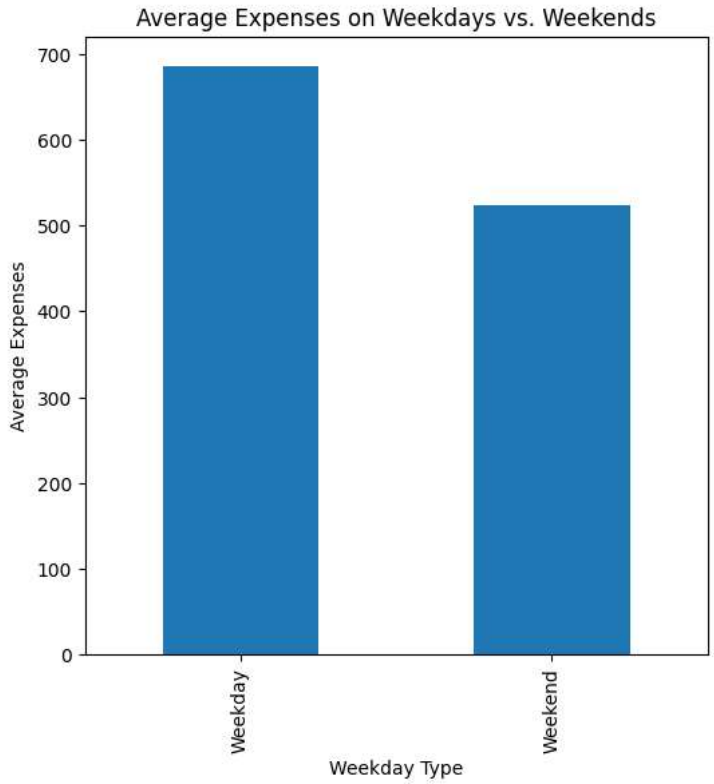

```
# Transaction Volume Trend:
# Visualize the trend of the number of transactions over the months.
monthly_transaction_count = df_HDFC.groupby('Month').size()

plt.figure(figsize=(10, 6))
monthly_transaction_count.plot(kind='line', marker='o')
plt.xlabel('Month')
plt.ylabel('Number of Transactions')
plt.title('Transaction Volume Trend')
plt.show()
```




```
# Monthly Withdrawal vs. Deposit Comparison:
# Compare the monthly total withdrawal and deposit amounts.
monthly_withdrawals = df_HDFC.groupby('Month')['Withdrawal Amt.'].sum()
monthly_deposits = df_HDFC.groupby('Month')['Deposit Amt.'].sum()

plt.figure(figsize=(10, 6))
plt.bar(monthly_withdrawals.index, monthly_withdrawals, label='Withdrawals')
plt.bar(monthly_deposits.index, monthly_deposits, label='Deposits')
plt.xlabel('Month')
plt.ylabel('Amount')
plt.title('Monthly Withdrawal vs. Deposit')
plt.legend()
plt.show()
```

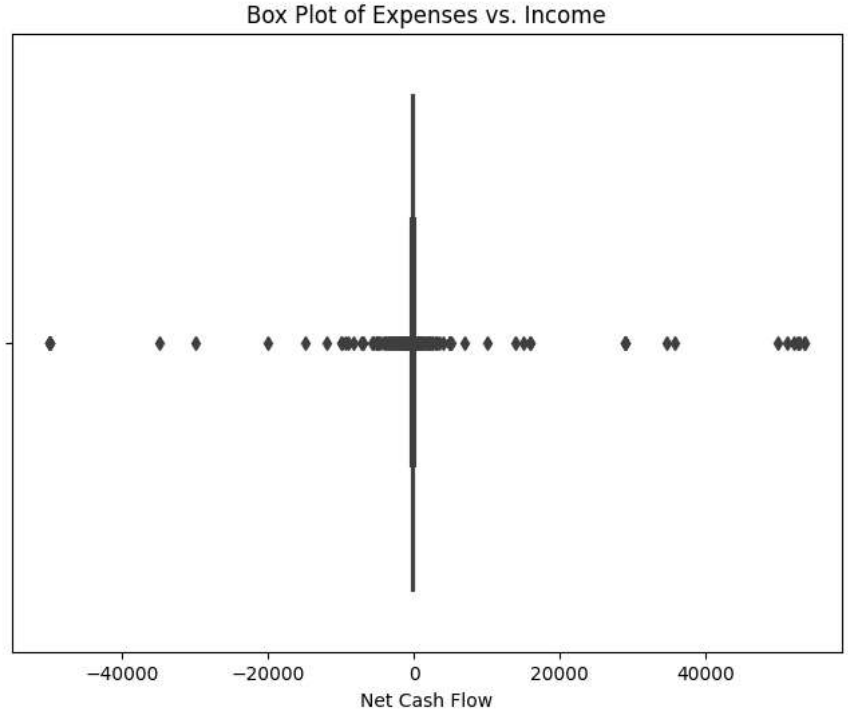

```
bank_statement_df = df_HDFC


# Average Daily Expenses:
# Calculate the average daily expenses for each month.
bank_statement_df['Day'] = bank_statement_df['Date'].dt.day
daily_expenses = bank_statement_df.groupby(['Month', 'Day'])['Withdrawal Amt.'].sum().reset_index()
average_daily_expenses = daily_expenses.groupby('Month')['Withdrawal Amt.'].mean()



# Weekday vs. Weekend Expenses:
# Compare your average expenses on weekdays and weekends.
def categorize_weekday(weekday):
    if weekday in ['Saturday', 'Sunday']:
        return 'Weekend'
    return 'Weekday'
```

```
bank_statement_df['Weekday_Type'] = bank_statement_df['Day_of_Week'].apply(categorize_weekday)
expenses_by_weekday_type = bank_statement_df.groupby('Weekday_Type')['Withdrawal Amt.'].mean()

plt.figure(figsize=(6, 6))
expenses_by_weekday_type.plot(kind='bar')
plt.xlabel('Weekday Type')
plt.ylabel('Average Expenses')
plt.title('Average Expenses on Weekdays vs. Weekends')
plt.show()
```

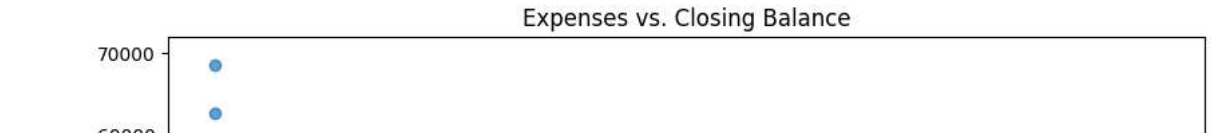



```
# Transaction Descriptions Word Cloud:
# Create a word cloud to visualize the most frequent words in transaction descriptions.
from wordcloud import WordCloud

narration_words = ' '.join(bank_statement_df['Narration'].astype(str))
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(narration_words)

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Transaction Descriptions Word Cloud')
plt.show()
```


```
# Expenses vs. Income Box Plot:
# Compare the distribution of expenses and income using a box plot.
plt.figure(figsize=(8, 6))
sns.boxplot(x='Net_Cash_Flow', data=bank_statement_df)
plt.xlabel('Net Cash Flow')
plt.title('Box Plot of Expenses vs. Income')
plt.show()
```



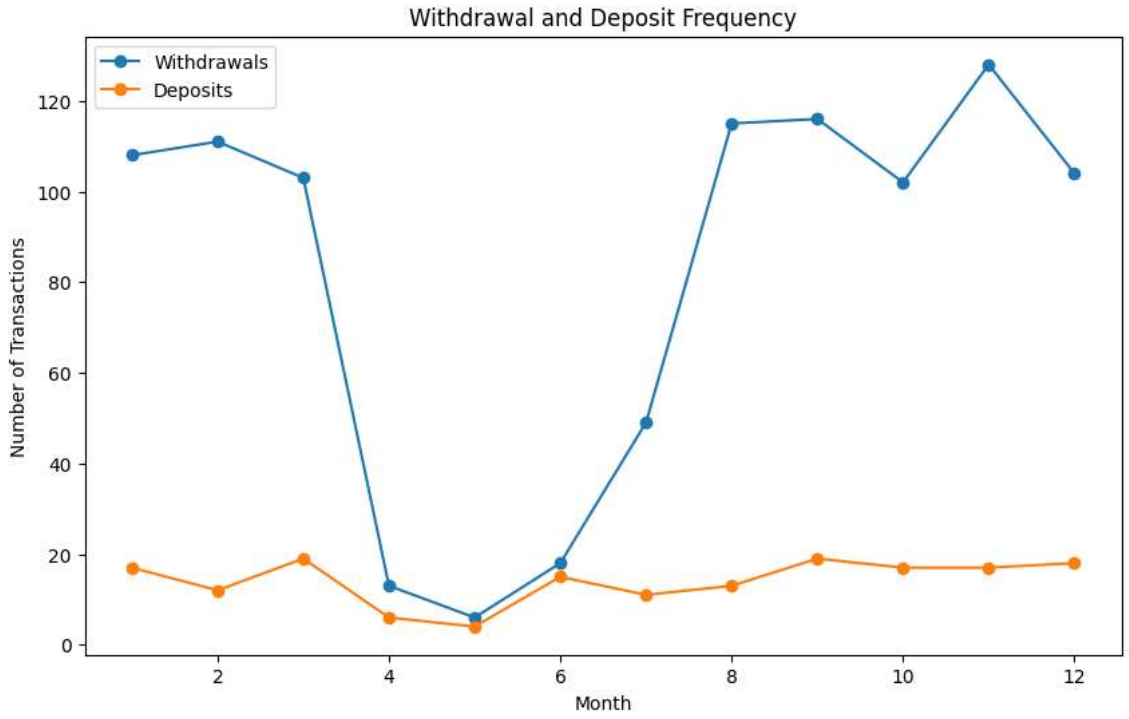

```
# Expenses vs. Closing Balance Scatter Plot:
# Visualize the relationship between expenses and the closing balance for each transaction.
plt.figure(figsize=(10, 6))
plt.scatter(bank_statement_df['Withdrawal Amt.'], bank_statement_df['Closing Balance'], alpha=0.7)
plt.xlabel('Expenses')
plt.ylabel('Closing Balance')
plt.title('Expenses vs. Closing Balance')
plt.show()
```




```
# Monthly Savings:
# Calculate your monthly savings by subtracting total expenses from total income.
monthly_savings = monthly_income - monthly_expenses
```

```
# Biggest Income and Expense Days:
# Identify the days with the highest income and expenses.
biggest_income_day = bank_statement_df.nlargest(1, 'Deposit Amt.')
biggest_expense_day = bank_statement_df.nlargest(1, 'Withdrawal Amt.')
```

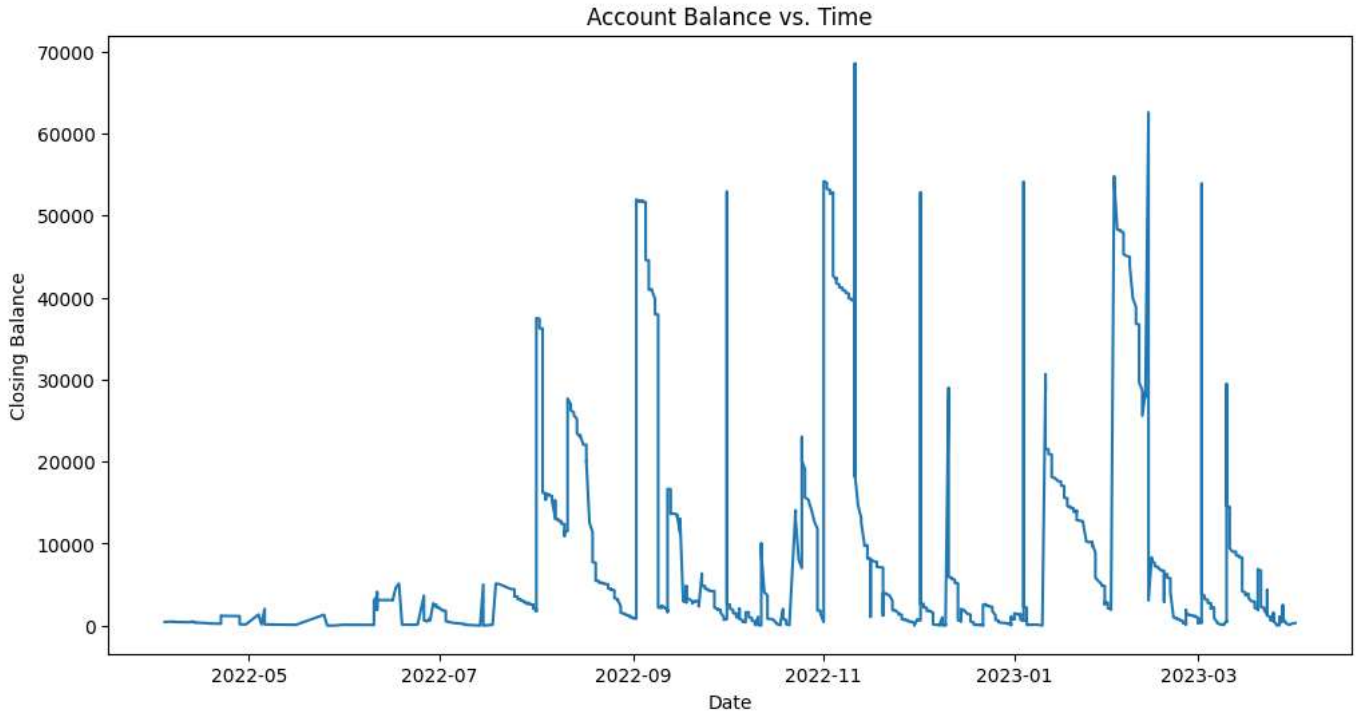
```
# Withdrawal and Deposit Frequency:
# Analyze the frequency of withdrawals and deposits over the months.
withdrawal_frequency = bank_statement_df[bank_statement_df['Withdrawal Amt.'] > 0].groupby('Month').size()
deposit_frequency = bank_statement_df[bank_statement_df['Deposit Amt.'] > 0].groupby('Month').size()

plt.figure(figsize=(10, 6))
withdrawal_frequency.plot(kind='line', marker='o', label='Withdrawals')
deposit_frequency.plot(kind='line', marker='o', label='Deposits')
plt.xlabel('Month')
plt.ylabel('Number of Transactions')
plt.title('Withdrawal and Deposit Frequency')
plt.legend()
plt.show()
```




```
# Daily Average Balance:
# Calculate the daily average balance for each month.
daily_average_balance = bank_statement_df.groupby('Month')['Closing Balance'].mean() / bank_statement_df['Day'].nunique()
```

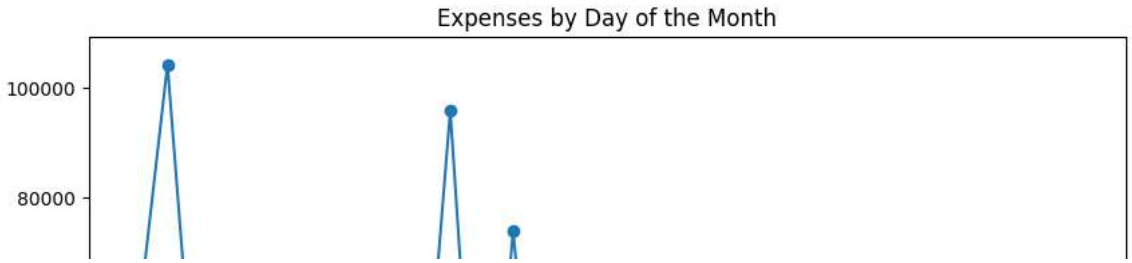
```
# Transaction Outliers:
# Identify any outliers in transactions that are significantly higher or lower than typical.
transaction_outliers = bank_statement_df[
    (bank_statement_df['Withdrawal Amt.'] > bank_statement_df['Withdrawal Amt.'].mean() + 3 * bank_statement_df['Withdrawal Amt.'].std()) |
    (bank_statement_df['Deposit Amt.'] > bank_statement_df['Deposit Amt.'].mean() + 3 * bank_statement_df['Deposit Amt.'].std())
]
```

```
# Balance vs. Time Plot:
# Visualize the change in your account balance over time.
plt.figure(figsize=(12, 6))
plt.plot(bank_statement_df['Date'], bank_statement_df['Closing Balance'])
plt.xlabel('Date')
plt.ylabel('Closing Balance')
plt.title('Account Balance vs. Time')
plt.show()
```

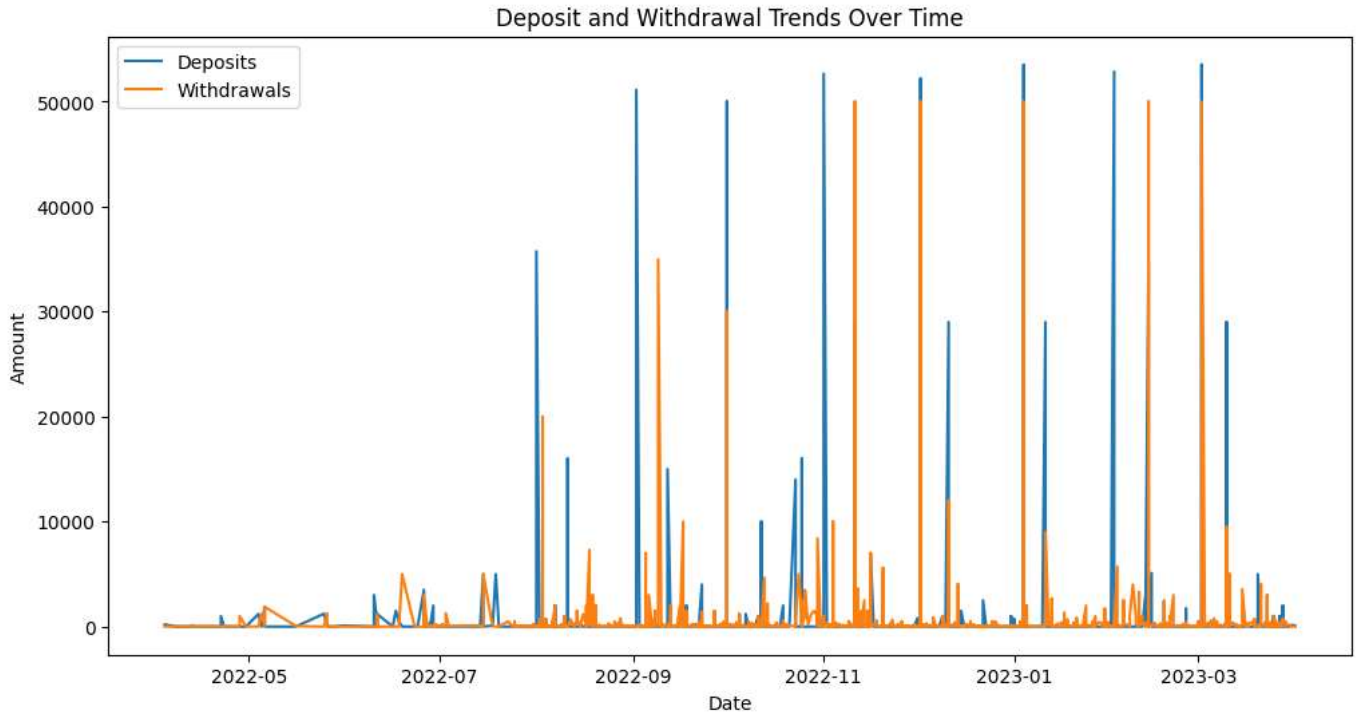



```
# Expense vs. Income Ratio:
# Calculate the ratio of total expenses to total income for each month.
expense_income_ratio = total_expenses / total_income
```

```
# Expenses by Day of the Month:
# Analyze your expenses on different days of the month.
expenses_by_day_of_month = bank_statement_df.groupby(bank_statement_df['Date'].dt.day)['Withdrawal Amt.'].sum()

plt.figure(figsize=(10, 6))
plt.plot(expenses_by_day_of_month.index, expenses_by_day_of_month, marker='o')
plt.xlabel('Day of the Month')
plt.ylabel('Expenses')
plt.title('Expenses by Day of the Month')
plt.show()
```

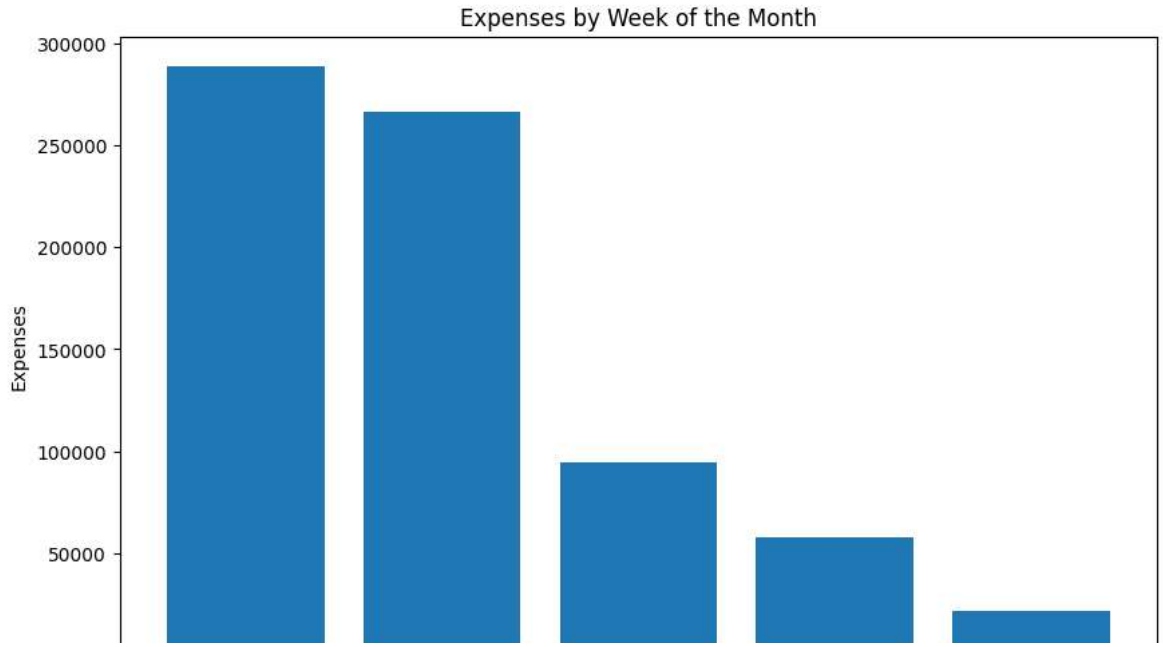



```
# Deposit and Withdrawal Trends Over Time:
# Visualize the trends of deposits and withdrawals over time.
plt.figure(figsize=(12, 6))
plt.plot(bank_statement_df['Date'], bank_statement_df['Deposit Amt.'], label='Deposits')
plt.plot(bank_statement_df['Date'], bank_statement_df['Withdrawal Amt.'], label='Withdrawals')
plt.xlabel('Date')
plt.ylabel('Amount')
plt.title('Deposit and Withdrawal Trends Over Time')
plt.legend()
plt.show()
```

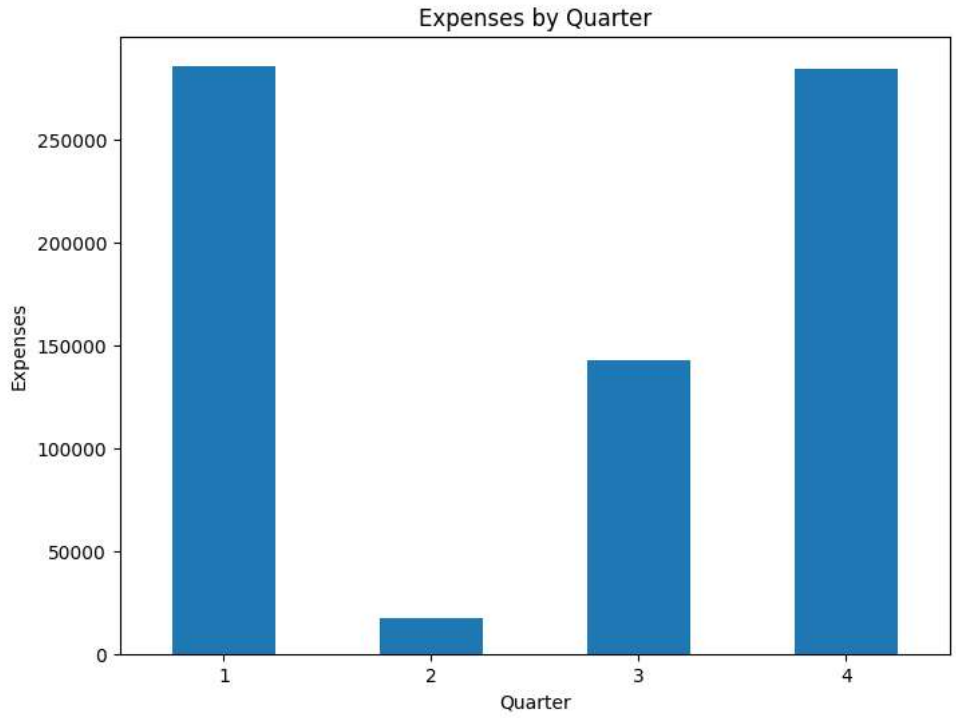



```
# Average Expense per Transaction:
# Calculate the average expense per transaction for each month.
bank_statement_df['Transaction_Amt'] = bank_statement_df['Withdrawal Amt.'] * -1
average_expense_per_transaction = bank_statement_df.groupby('Month')['Transaction_Amt'].mean()
```

```
# Expenses by Week of the Month:
# # Analyze your expenses based on the week of the month (e.g., 1st week, 2nd week, etc.).
week_of_month = bank_statement_df['Date'].apply(lambda x: (x.day - 1) // 7 + 1)
expenses_by_week_of_month = bank_statement_df.groupby(week_of_month)['Withdrawal Amt.'].sum()

plt.figure(figsize=(10, 6))
plt.bar(expenses_by_week_of_month.index, expenses_by_week_of_month)
plt.xlabel('Week of the Month')
plt.ylabel('Expenses')
plt.title('Expenses by Week of the Month')
plt.show()
```




```
# Expenses by Quarter:
# Analyze your expenses on a quarterly basis.
quarterly_expenses = bank_statement_df.groupby(bank_statement_df['Date'].dt.quarter)['Withdrawal Amt.'].sum()

plt.figure(figsize=(8, 6))
quarterly_expenses.plot(kind='bar', rot=0)
plt.xlabel('Quarter')
plt.ylabel('Expenses')
plt.title('Expenses by Quarter')
plt.show()
```




```
# Financial Ratios:
# Calculate financial ratios to assess your financial health, such as the Debt-to-Income Ratio, Savings Ratio, or Expense-to-Income Ratio.

# Debt-to-Income Ratio: Total Monthly Debt Payments / Monthly Gross Income
debt_to_income_ratio = total_expenses / monthly_income

# Savings Ratio: Monthly Savings / Monthly Income
savings_ratio = monthly_savings / monthly_income
```

```
# Expense-to-Income Ratio: Total Monthly Expenses / Monthly Gross Income
expense_to_income_ratio = total_expenses / monthly_income



from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Select relevant features for clustering (e.g., 'Withdrawal Amt.' and 'Deposit Amt.')
features_for_clustering = bank_statement_df[['Withdrawal Amt.', 'Deposit Amt.']]

# Standardize the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features_for_clustering)

# Determine the optimal number of clusters using the Elbow method or other techniques
kmeans = KMeans(n_clusters=3)  # You can choose the appropriate number of clusters
bank_statement_df['Cluster'] = kmeans.fit_predict(scaled_features)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
  warnings.warn(
```

```
bank_statement_df.head()
```

	Date	Narration	Withdrawal Amt.	Deposit Amt.	Closing Balance	Month	Day_of_Week	Net_Cash_Flow	Day	Weekday_Type	Transaction_
0	2022-04-04	UPI-VISHAL VINAYAK BHOYA-VISHALBHOYAR313	0.0	200.0	442.19	4	Monday	200.0	4	Weekday	-
1	2022-04-07	UPI-ANIL NARBAT BISEN-ANILBISEN229@OKSBI	0.0	50.0	492.19	4	Thursday	50.0	7	Weekday	-
2	2022-04-07	UPI-SHRIRAM GENERAL STOR-GPAY-1119808673	50.0	0.0	442.19	4	Thursday	-50.0	7	Weekday	-5
3	2022-04-12	UPI-IDRISH SHHA-BHARATPE.9051322211@FBP	30.0	0.0	412.19	4	Tuesday	-30.0	12	Weekday	-3
4	2022-04-13	UPI-RAMESHWAR VASANTRAO	0.0	100.0	512.19	4	Wednesday	100.0	13	Weekday	-

✓ 0s completed at 9:06 AM