```
#import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#upload files from device
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving clean_exp - exp.csv to clean_exp - exp.csv

```
#read the data
df = pd.read_csv("clean_exp - exp.csv")
```

```
#first 5 entries
print(df.head())
```

```
       Date         Expense Category   Type   Amount  Closing Balance
0       NaN             NaN      NaN    NaN      NaN              NaN
1  04/05/25     Jio Prepaid     Bill  Debit    349.0         27383.38
2  04/05/25     Sent to Dad   Family  Debit    500.0         26883.38
3  05/05/25         PG Rent  Housing  Debit  10000.0         16883.38
4  07/05/25  Sarees for Mom   Family  Debit   2300.0         14583.38
```

```
# df.info() gives a concise summary of the dataframe
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 218 entries, 0 to 217
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             217 non-null    object
 1   Expense          217 non-null    object
 2   Category         217 non-null    object
 3   Type             217 non-null    object
 4   Amount           217 non-null    float64
 5   Closing Balance  217 non-null    float64
dtypes: float64(2), object(4)
memory usage: 10.3+ KB
None
```

```
# df.describe() gives summary statistics for numerical columns
print(df.describe())
```

```
             Amount  Closing Balance
count    217.000000       217.000000
mean    1532.251014     23603.938756
std     5128.279436     11192.048282
min       10.000000      5715.590000
25%       30.000000     14932.380000
50%      121.000000     23314.080000
75%      349.000000     26100.750000
max    43510.000000     50984.960000
```

```
# Convert Amounts into signed values:
# - Keep as positive if transaction Type is 'Credit'
# - Make negative if transaction Type is 'Debit'
df['Amount'] = df.apply(lambda x: x['Amount'] if x['Type'] == 'Credit' else -x['Amount'], axis=1)
```

```
# Calculate total income: sum of all Credit transaction amounts
total_income = df[df['Type'] == 'Credit']['Amount'].sum()

# Calculate total expenses: sum of all Debit transaction amounts (made positive with - sign)
total_expenses = -df[df['Type'] == 'Debit']['Amount'].sum()

# Net balance: total of all signed Amounts (income - expenses)
net_balance = df['Amount'].sum()
```

```
print("Total Income:", total_income)
print("Total Expenses:", total_expenses)
print("Net Balance:", net_balance)
```

```
Total Income: 162597.38
Total Expenses: 169901.09
Net Balance: -7303.709999999992
```

```
# Expenses by category
expense_by_cat = df[df['Type'] == 'Debit'].groupby('Category')['Amount'].sum().abs().sort_values(ascending=False)
```

```
# Income by category
income_by_cat = df[df['Type'] == 'Credit'].groupby('Category')['Amount'].sum().sort_values(ascending=False)
```

```
print("Expenses by Category:\n", expense_by_cat)
print("Income by Category:\n", income_by_cat)

#top categories
top_exp = expense_by_cat.head(3).sum() / total_expenses * 100
print(f"\n Top 3 categories account for {top_exp:.2f}% of total expenses")
```

```
Expenses by Category:
 Category
Bill                  55948.00
Housing               30000.00
Personal Transfer     21795.00
Family                16600.00
Food & Eating out     13698.80
Misc                  10297.10
Shopping               6572.00
Travel & Transport     6040.85
Education              2394.00
Groceries              2339.61
Spiritual              2251.00
Utilities              1809.73
Medical                 155.00
Name: Amount, dtype: float64
Income by Category:
 Category
Income               100512.00
Personal Transfer     58774.20
Rewards                3311.18
Name: Amount, dtype: float64

 Top 3 categories account for 63.42% of total expenses
```

```
# Convert 'Date' column from string/object to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Sort transactions by Date to ensure chronological order
df = df.sort_values('Date')
```

```
/tmp/ipython-input-2493865281.py:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back
  df['Date'] = pd.to_datetime(df['Date'])
```

```
# Daily cashflow
daily = df.groupby('Date')['Amount'].sum()

# Weekly cashflow
weekly = df.resample('W-Mon', on='Date')['Amount'].sum()

# Running balance
df['Running_Balance'] = df['Amount'].cumsum()
```

```
print("\n Highest Spending Days:\n", df[df['Type']=="Debit"].groupby('Date')['Amount'].sum().abs().nlargest(5))
print("\n Weekly Net Cashflow:\n", weekly)
```

```

 Highest Spending Days:
 Date
2025-07-30    27539.14
2025-05-27    26280.09
2025-04-06    20758.10
2025-05-19    15310.55
2025-03-07    10413.85
Name: Amount, dtype: float64
```

```
 Weekly Net Cashflow:
 Date
2025-01-06     -946.39
2025-01-13      -65.86
2025-01-20        0.00
2025-01-27        0.00
2025-02-03        0.00
2025-02-10      -69.72
2025-02-17        0.00
2025-02-24        0.00
2025-03-03        0.00
2025-03-10   -10642.87
2025-03-17        0.00
2025-03-24        0.00
2025-03-31        0.00
2025-04-07   -21812.10
2025-04-14        0.00
2025-04-21        0.00
2025-04-28        0.00
2025-05-05   -10000.00
2025-05-12    -1153.53
2025-05-19     -696.35
2025-05-26    -2521.41
2025-06-02    41253.29
2025-06-09    -1351.65
2025-06-16    -1439.78
2025-06-23    -4333.74
2025-06-30    16439.80
2025-07-07    -4360.70
2025-07-14    -4600.00
2025-07-21    -3711.30
2025-07-28     8270.89
2025-08-04    -2095.14
2025-08-11     -283.00
2025-08-18        0.00
2025-08-25        0.00
2025-09-01        0.00
2025-09-08     -204.00
2025-09-15        0.00
2025-09-22        0.00
2025-09-29        0.00
2025-10-06     -299.20
2025-10-13        0.00
2025-10-20        0.00
2025-10-27        0.00
2025-11-03        0.00
2025-11-10    -2680.95
Name: Amount, dtype: float64
```

```
# Essentials vs Discretionary mapping
essentials = ["Rent", "Groceries", "Utilities", "Transport", "Bills"]
df['Spending_Type'] = df['Category'].apply(lambda x: "Essential" if x in essentials else "Discretionary")

print("\n Essentials vs Discretionary:\n", df.groupby('Spending_Type')['Amount'].sum())
```

```
 Essentials vs Discretionary:
 Spending_Type
Discretionary   -3154.37
Essential       -4149.34
Name: Amount, dtype: float64
```

```
# Average & median expense
avg_expense = df[df['Type']=="Debit"]['Amount'].abs().mean()
median_expense = df[df['Type']=="Debit"]['Amount'].abs().median()
print(f"\n Average expense per transaction: {avg_expense:.2f}")
print(f" Median expense per transaction: {median_expense:.2f}")
```

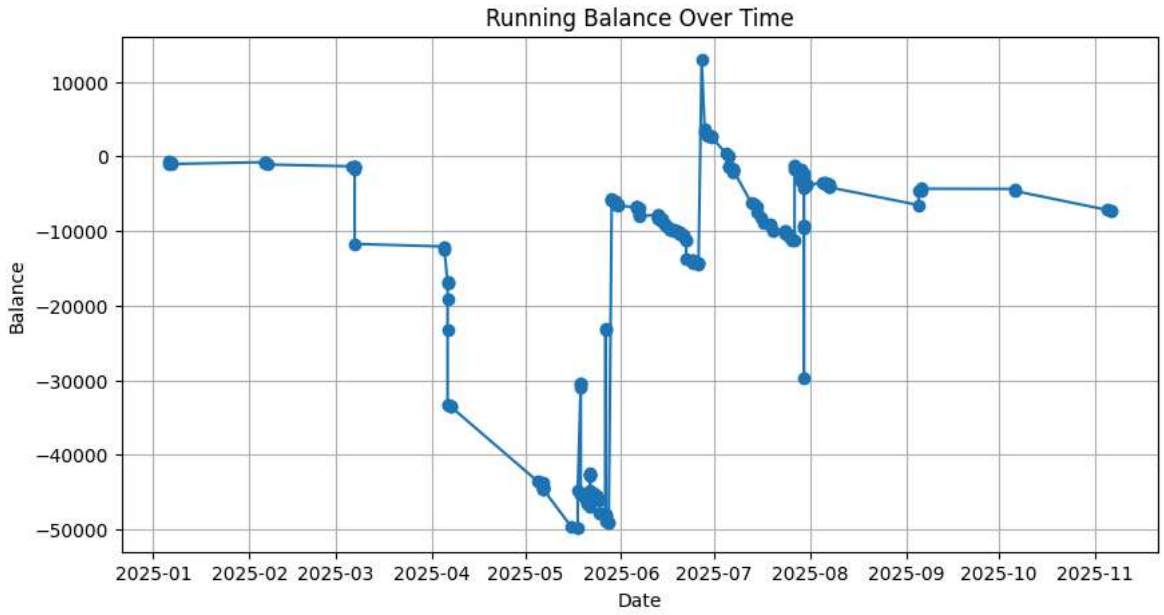
```
 Average expense per transaction: 858.09
 Median expense per transaction: 100.00
```

```
# Savings rate: what percentage of income is saved
savings_rate = (total_income - total_expenses) / total_income * 100

# Expense-to-income ratio: what percentage of income is spent
expense_income_ratio = total_expenses / total_income * 100

# Average weekly savings: mean of weekly net cashflows
weekly_savings = weekly.mean()
```

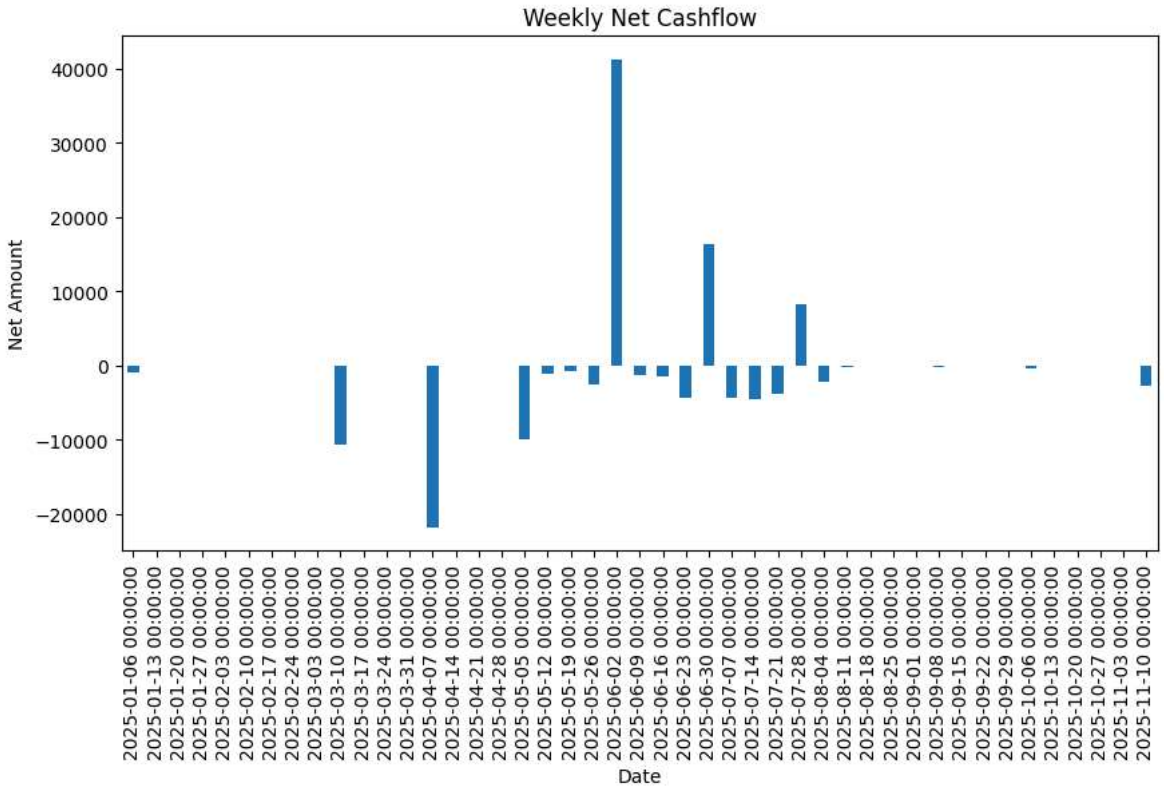
```
print(f"\n Savings Rate: {savings_rate:.2f}%")
print(f" Expense-to-Income Ratio: {expense_income_ratio:.2f}%")
print(f" Average Weekly Net Savings: {weekly_savings:.2f}")
```

```
 Savings Rate: -4.49%
 Expense-to-Income Ratio: 104.49%
 Average Weekly Net Savings: -162.30
```

```
# Running balance
plt.figure(figsize=(10,5))
plt.plot(df['Date'], df['Running_Balance'], marker='o')
plt.title("Running Balance Over Time")
plt.xlabel("Date")
plt.ylabel("Balance")
plt.grid(True)
plt.show()
```




```
# Weekly bar chart
weekly.plot(kind='bar', figsize=(10,5))
plt.title("Weekly Net Cashflow")
plt.ylabel("Net Amount")
plt.show()
```




```
# Remove the first row which contains NaN values
df = df.iloc[1:].copy()

# Convert 'Type' column to string type
df['Type'] = df['Type'].astype(str)

# Apply the lambda function to create 'Amount_signed'
df["Amount_signed"] = df.apply(lambda x: x["Amount"] if x["Type"].lower() == "credit" else -x["Amount"], axis=1)
df.head()
```

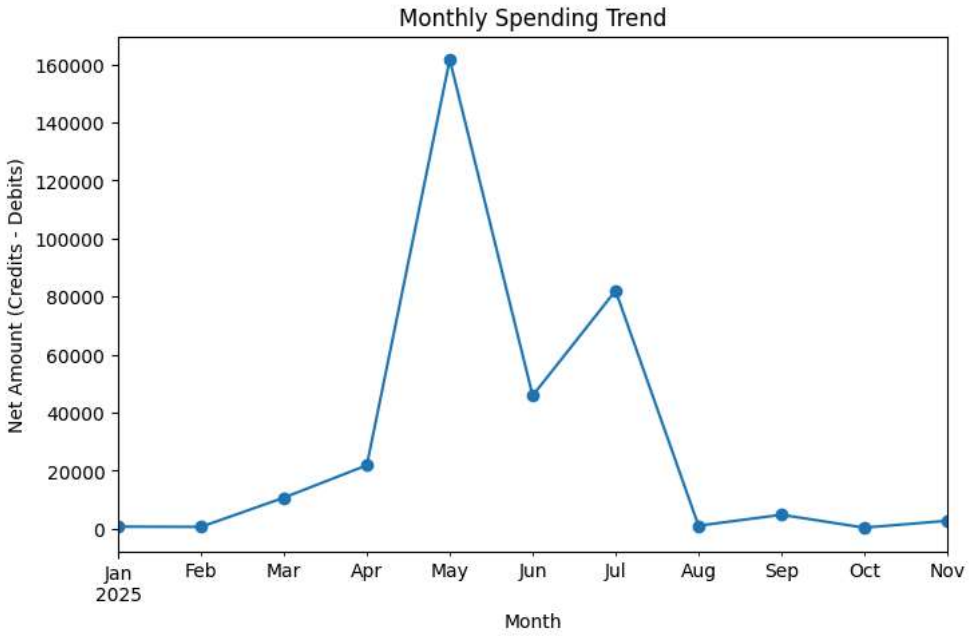
	Date	Expense	Category	Type	Amount	Closing Balance	Running_Balance	Spending_Type	Amount_signed
61	2025-01-06	Magic Pin	Food & Eating out	Debit	-347.39	49606.57	-946.39	Discretionary	347.39
141	2025-01-07	from bank	Rewards	Credit	121.00	36448.83	-825.39	Discretionary	121.00
142	2025-01-07	magic pin	Food & Eating out	Debit	-186.86	36261.97	-1012.25	Discretionary	186.86
66	2025-02-06	Zepto refund	Rewards	Credit	253.00	49110.57	-759.25	Discretionary	253.00
65	2025-02-06	BMTC Bus	Travel & Transport	Debit	-20.00	48857.57	-779.25	Discretionary	20.00

```
# Descriptive statistics
print("Overall Statistics for Amount_signed:")
print(df["Amount_signed"].describe())
```

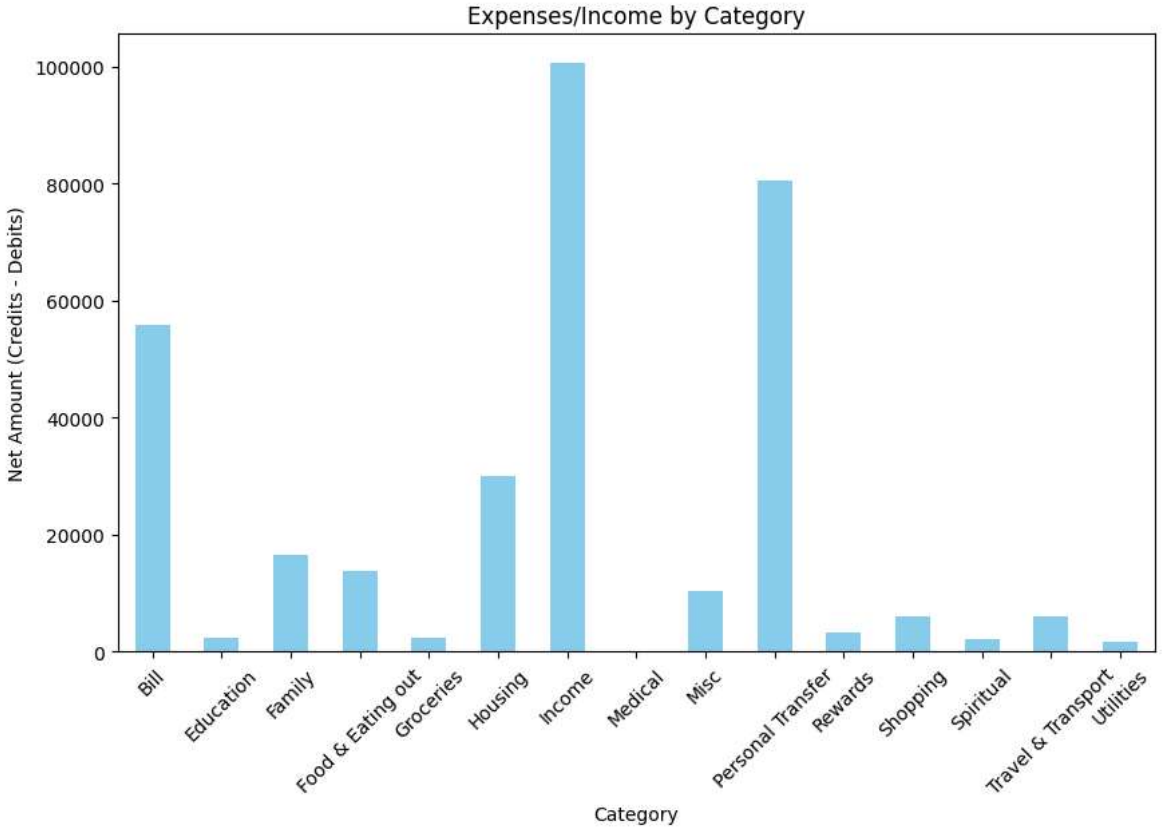
```
Overall Statistics for Amount_signed:
count      216.000000
mean      1536.571620
std       5139.795944
min         10.000000
25%         30.000000
50%        120.500000
75%        349.000000
max      43510.000000
Name: Amount_signed, dtype: float64
```

```
# Monthly trend analysis
df["Date"] = pd.to_datetime(df["Date"])
df["Month"] = df["Date"].dt.to_period("M")

monthly_trend = df.groupby("Month")["Amount_signed"].sum()
monthly_trend.plot(kind="line", marker="o", figsize=(8,5), title="Monthly Spending Trend")
```

```
plt.ylabel("Net Amount (Credits - Debits)")
plt.show()
```




```
# Category totals
category_totals = df.groupby("Category")["Amount_signed"].sum()
category_totals.plot(kind="bar", figsize=(10,6), color="skyblue")
plt.title("Expenses/Income by Category")
plt.ylabel("Net Amount (Credits - Debits)")
plt.xticks(rotation=45)
plt.show()
```




```
#Summary stats
total_income = df[df['Type'] == 'Credit']['Amount'].sum()
total_expenses = df[df['Type'] == 'Debit']['Amount'].sum()
net_balance = total_income - total_expenses
```

```
savings_rate = (net_balance / total_income) * 100 if total_income > 0 else 0
expense_income_ratio = (total_expenses / total_income) * 100 if total_income > 0 else 0

summary = pd.DataFrame({
    'Metric': ['Total Income', 'Total Expenses', 'Net Balance', 'Savings Rate (%)', 'Expense/Income Ratio (%)'],
    'Value': [total_income, total_expenses, net_balance, round(savings_rate, 2), round(expense_income_ratio, 2)]
})

print("📊 Summary:")
display(summary)
```

📊 Summary:

	Metric	Value
0	Total Income	162597.38
1	Total Expenses	169901.09
2	Net Balance	-7303.71
3	Savings Rate (%)	-4.49
4	Expense/Income Ratio (%)	104.49

```
#Category-wise expense chart
category_expenses = df[df['Type'] == 'Debit'].groupby('Category')['Amount'].sum().abs().sort_values()

plt.figure(figsize=(8,5))
category_expenses.plot(kind='barh', color='skyblue')
plt.title("Expenses by Category", fontsize=14, weight='bold')
plt.xlabel("Amount")
plt.ylabel("Category")
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```