

Personal Finance & Spending Pattern Analysis

Course: Python Programming End-To-End Project **Date:** 23rd February 2026

Made By: Sanyam Chanana

Data Description

This dataset comprises a comprehensive collection of personal financial transaction records, capturing both income and expenditure activities across an extended time period. Each record is characterized by multiple attributes — including the transaction date, spending category (such as Food, Fuel, Shopping, Rent, Entertainment, and more), payment mode (UPI, Card, Cash, or Bank Transfer), transaction amount, and transaction type (Income or Expense).

The richness of the dataset — spanning both numerical variables (transaction amounts, dates) and categorical variables (categories, payment modes, transaction types) — makes it well-suited for in-depth exploratory analysis. It facilitates the examination of spending behavior, monthly financial trends, savings accumulation, and overall fiscal discipline, while also enabling robust data visualization and statistical interpretation.

Problem Statement

Personal financial management is a critical life skill, yet spending patterns often go unexamined without the aid of structured data analysis. This project aims to bridge that gap by applying data analytics techniques to real-world personal transaction data.

The core objectives of this study are to:

- 1. Analyze Monthly Income and Expenses** — Track and compare income against expenditure on a month-by-month basis to reveal periods of surplus or deficit.
- 2. Identify Major Spending Categories** — Determine which categories (e.g., Food, Rent, Shopping) consume the largest share of total expenditure, enabling targeted financial awareness.
- 3. Evaluate Savings Trends** — Assess how savings evolve over time and identify patterns that reflect sound or concerning financial behavior.
- 4. Understand Payment Behavior** — Examine the distribution of payment modes to understand preferences and their implications for financial tracking and security.
- 5. Provide Actionable Insights for Financial Management** — Translate analytical findings into practical, data-backed recommendations for improved budgeting and financial decision-making.

Through systematic Exploratory Data Analysis (EDA) and meaningful visualization, this project demonstrates how Python-driven data analytics can serve as a powerful tool for personal financial planning and long-term fiscal health.

1 Import Libraries & Load Dataset

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

# Load dataset (update filename if needed)
data = pd.read_csv("/Users/sanyamchanana/Desktop/personal-
finance.csv")

# Display dataset structure
print(data.head())
print(data.tail())
print(data.info())
print(data.describe())
```

The screenshot shows a Jupyter Notebook interface with a terminal tab active. The terminal output displays the following:

```
/usr/local/bin/python3 /Users/sanyamchanana/Desktop/Garvit.py
• (base) sanyamchanana@SANYAM-MacBook-Air ~ % /usr/local/bin/python3 /Users/sanyamchanana/Desktop/Garvit.py
      Date          Transaction Description   Category    Amount     Type
 0  2020-01-02           Score each. Food & Drink 1485.69 Expense
 1  2020-01-02           Quality throughout. Utilities 1475.58 Expense
 2  2020-01-04 Instead ahead despite measure ago. Rent 1185.08 Expense
 3  2020-01-05 Information last everything thank serve. Investment 2291.00 Income
 4  2020-01-13 Future choice whatever from. Food & Drink 1126.88 Expense
      Date          Transaction Description   Category    Amount     Type
1495 2024-12-28           Quite as when. Rent 514.09 Expense
1496 2024-12-28 Right analysis mention. Entertainment 727.25 Expense
1497 2024-12-28 No couple debate must. Investment 1425.00 Income
1498 2024-12-29 Discussion black follow. Shopping 655.78 Expense
1499 2024-12-29 Pressure activity defense detail. Other 1480.00 Income
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
 0   Date              1500 non-null   object 
 1   Transaction Description  1500 non-null  object 
 2   Category          1500 non-null   object 
 3   Amount             1500 non-null   float64
 4   Type              1500 non-null   object 
dtypes: float64(1), object(4)
memory usage: 58.7+ KB
None
      Amount
count 1500.000000
mean 1307.520913
std 982.283361
min 14.370000
25% 629.340000
50% 1156.285000
75% 1712.932500
max 4996.000000
```

2 Data Preprocessing

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

# Create Month column
data['Month'] = data['Transaction_Date'].dt.to_period('M')

# Create Income column
data['Income'] = data.apply(
```

```

lambda row: row['Amount'] if row['Transaction_Type'] == 'Income' else 0,
           axis=1
)

# Create Expense column
data['Expense'] = data.apply(
    lambda row: row['Amount'] if row['Transaction_Type'] == 'Expense' else 0,
    axis=1
)

```

PROBLEMS 1 OUTPUT TERMINAL PORTS ... | [] X

DEBUG CONSOLE

TERMINAL

```

Filter (e.g. text, !exclude, \escape)
● (base) sanyamchanana@SANYAMs-MacBook-Air ~ % /usr/local/bin/python3 /Users/sanyamchanana/Desktop/Garvit.py
Missing Values:
   Date          0
   Transaction Description  0
   Category      0
   Amount         0
   Type          0
   Month         0
dtype: int64
   Date          Transaction Description  Category  ...  Month  Income  Expense
0 2020-01-02          Score each.  Food & Drink  ...  2020-01  1485.69      0
1 2020-01-02          Quality throughout.  Utilities  ...  2020-01  1475.58      0
2 2020-01-04          Instead ahead despite measure ago.  Rent  ...  2020-01  1185.08      0
3 2020-01-05          Information last everything thank serve.  Investment  ...  2020-01  2291.00      0
4 2020-01-13          Future choice whatever from.  Food & Drink  ...  2020-01  1126.88      0
[5 rows x 8 columns]
○ (base) sanyamchanana@SANYAMs-MacBook-Air ~ %

```

3 Monthly Expense Trend

```

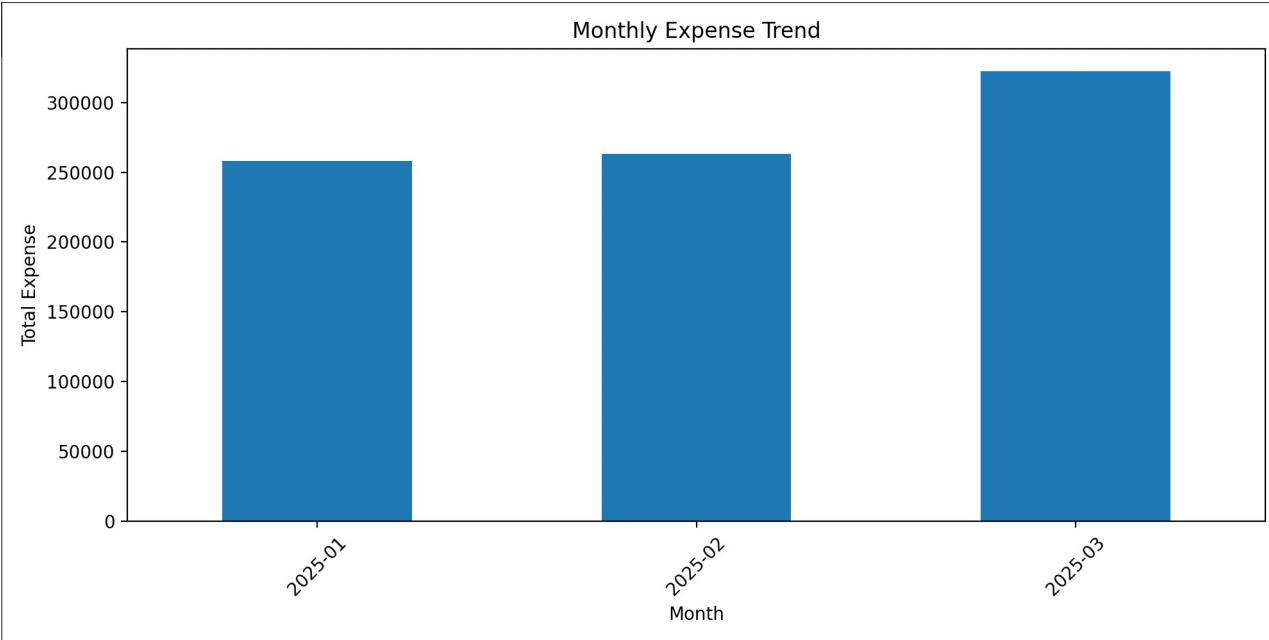
monthly_expense = data.groupby('Month')['Expense'].sum()

plt.figure(figsize=(10,5))
monthly_expense.plot(kind='bar')

plt.title("Monthly Expense Trend")
plt.xlabel("Month")
plt.ylabel("Total Expense")
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()

```



4 Category-wise Expense Analysis

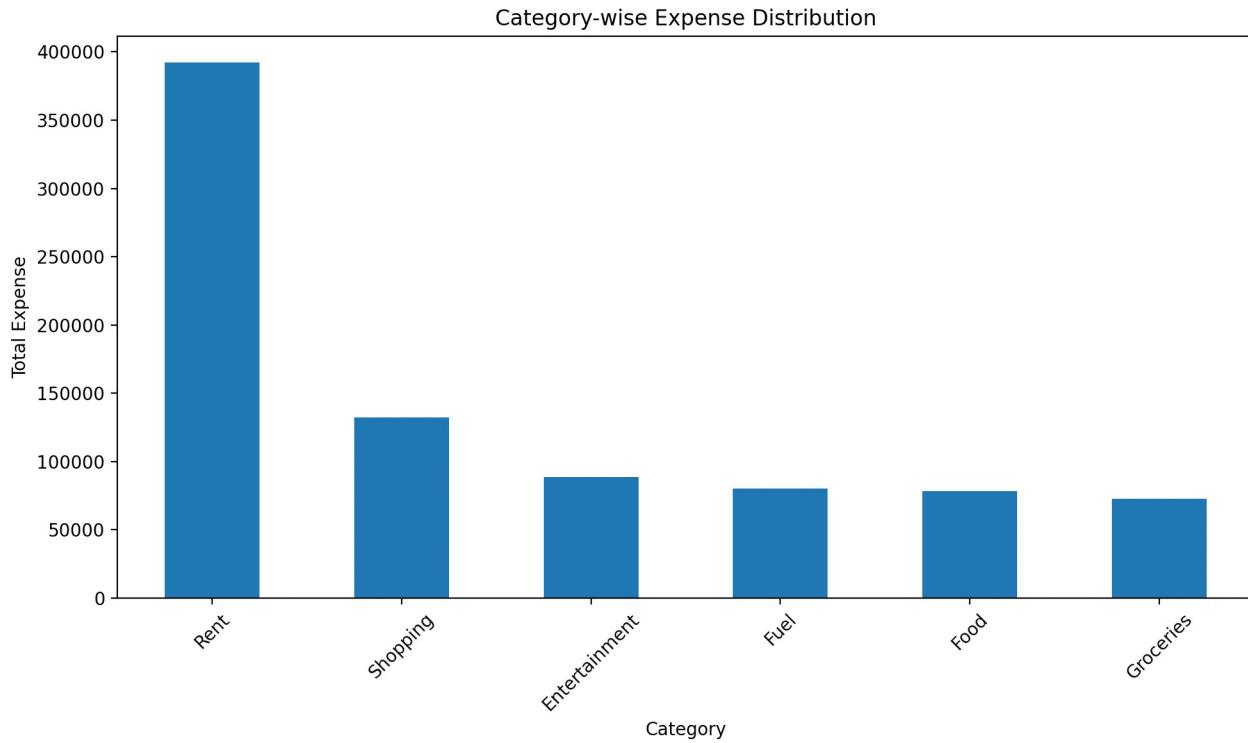
```
# Filter only Expense transactions
expense_data = data[data['Transaction_Type'] == 'Expense']

# Group by Category
category_expense = expense_data.groupby('Category')
['Amount'].sum().sort_values(ascending=False)

plt.figure(figsize=(10, 6))
category_expense.plot(kind='bar')

plt.title("Category-wise Expense Distribution")
plt.xlabel("Category")
plt.ylabel("Total Expense")
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



5 Income vs Expense Comparison

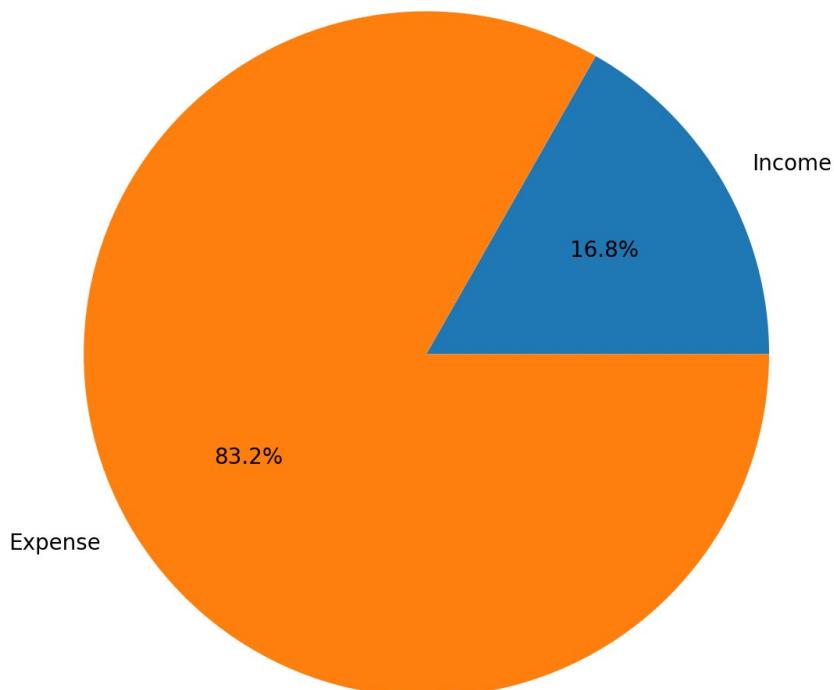
```
# Calculate totals correctly
total_income = data[data['Transaction_Type'] == 'Income']
['Amount'].sum()
total_expense = data[data['Transaction_Type'] == 'Expense']
['Amount'].sum()

plt.figure(figsize=(6, 6))
plt.pie([
    total_income, total_expense],
    labels=['Income', 'Expense'],
    autopct='%.1f%%'
)

plt.title("Income vs Expense Share")

plt.tight_layout()
plt.show()
```

Income vs Expense Share



6 Savings Calculation

```
# Calculate totals properly
total_income = data.loc[data['Transaction_Type'] == 'Income',
'Amount'].sum()
total_expense = data.loc[data['Transaction_Type'] == 'Expense',
'Amount'].sum()

# Calculate savings
savings = total_income - total_expense

# Print formatted results
print(f"Total Income: ₹{total_income:.2f}")
print(f"Total Expense: ₹{total_expense:.2f}")
print(f"Total Savings: ₹{savings:.2f}")

# Calculate savings rate safely
if total_income > 0:
    savings_rate = (savings / total_income) * 100
    print(f"Savings Rate: {savings_rate:.2f}%")
else:
    print("Savings Rate: Cannot calculate (Income is 0)")
```

A screenshot of a terminal window within a code editor interface. The terminal tab is selected at the top. The output shows the execution of a Python script named 'Garvit.py' which processes a CSV file. The script calculates total income, total expense, total savings, and the savings rate. The output is as follows:

```
/usr/local/bin/python3 /Users/sanyamchanana/Desktop/Garvit.py
(base) sanyamchanana@SANYAMs-MacBook-Air ~ % /usr/local/bin/python3 /Users/sanyamchanana/Desktop/Garvit.py
Index(['Transaction_Date', 'Category', 'Payment_Mode', 'Amount',
       'Transaction_Type'],
      dtype='object')
Total Income: ₹170,119.00
Total Expense: ₹843,924.00
Total Savings: ₹-673,805.00
Savings Rate: -396.08%
(base) sanyamchanana@SANYAMs-MacBook-Air ~ %
```

7 Monthly Savings Trend

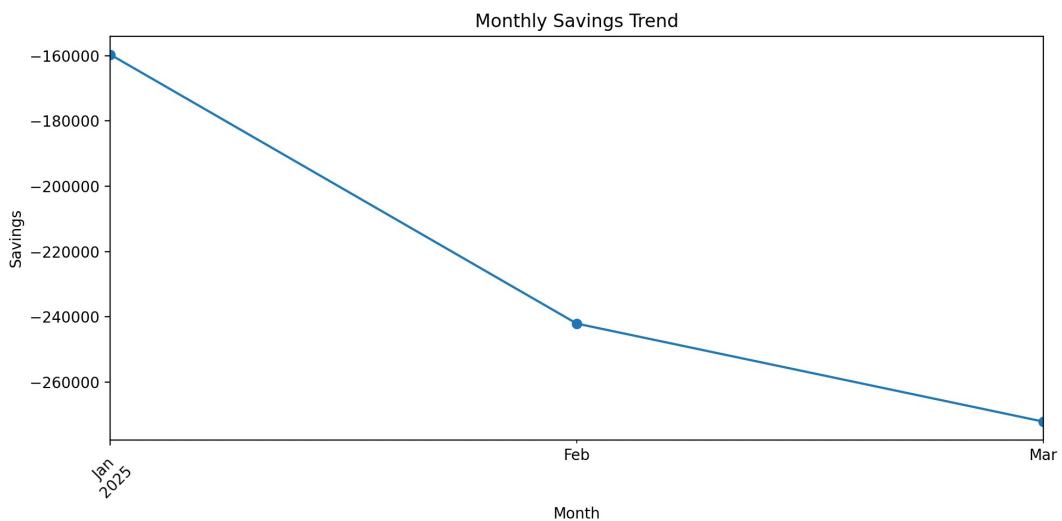
```
# Create monthly summary table
monthly_summary = (
    data.groupby(['Month', 'Transaction_Type'])['Amount']
    .sum()
    .unstack(fill_value=0)
)

# Create Savings column
monthly_summary['Savings'] = monthly_summary['Income'] -
monthly_summary['Expense']

# Plot Savings Trend
plt.figure(figsize=(10,5))
monthly_summary['Savings'].plot(marker='o')

plt.title("Monthly Savings Trend")
plt.xlabel("Month")
plt.ylabel("Savings")
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



8 Top 5 Highest Expenses

```
# Filter only expense transactions
expense_data = data[data['Transaction_Type'] == 'Expense']

# Sort by Amount (highest first)
top_expenses = expense_data.sort_values(by='Amount',
                                         ascending=False).head(5)

print("Top 5 Highest Expenses:")
print(top_expenses[['Transaction Date', 'Category', 'Amount']])
```

```
Total Savings: < 675,000.00
Savings Rate: -396.08%
Top 5 Highest Expenses:
   Transaction_Date Category  Amount
175      2025-03-19    Rent  14939
 83       2025-02-05    Rent  14435
186      2025-03-24    Rent  14228
153      2025-03-09    Rent  14161
143      2025-03-06    Rent  13571
○ (base) sanyamchanana@SANYAMs-MacBook-Air ~ %
```

9 Transaction Count by Category

```
transaction_count = data['Category'].value_counts()

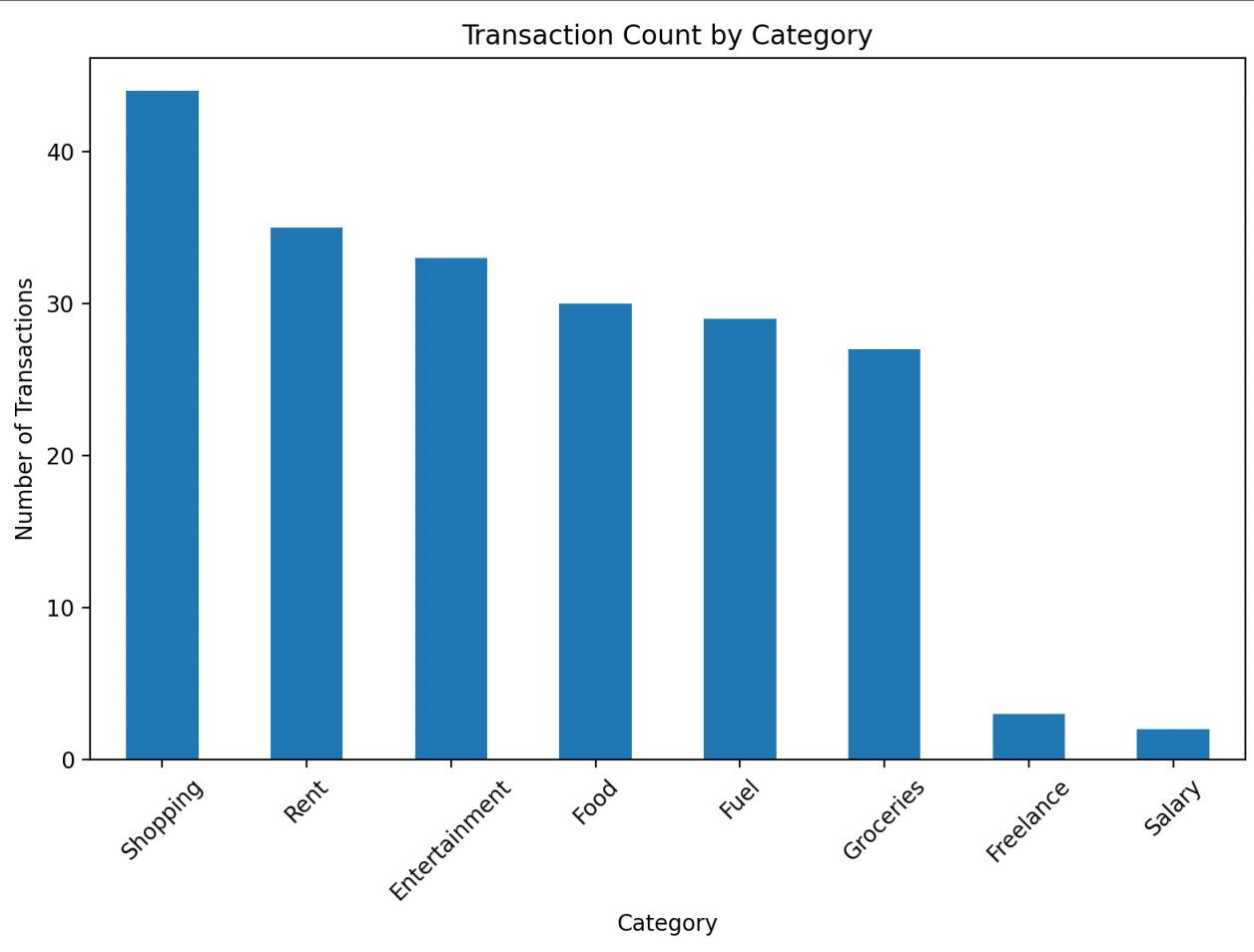
plt.figure(figsize=(8, 6))
transaction_count.plot(kind='bar')
```

```

plt.title("Transaction Count by Category")
plt.xlabel("Category")
plt.ylabel("Number of Transactions")
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()

```



10 Correlation Heatmap

```

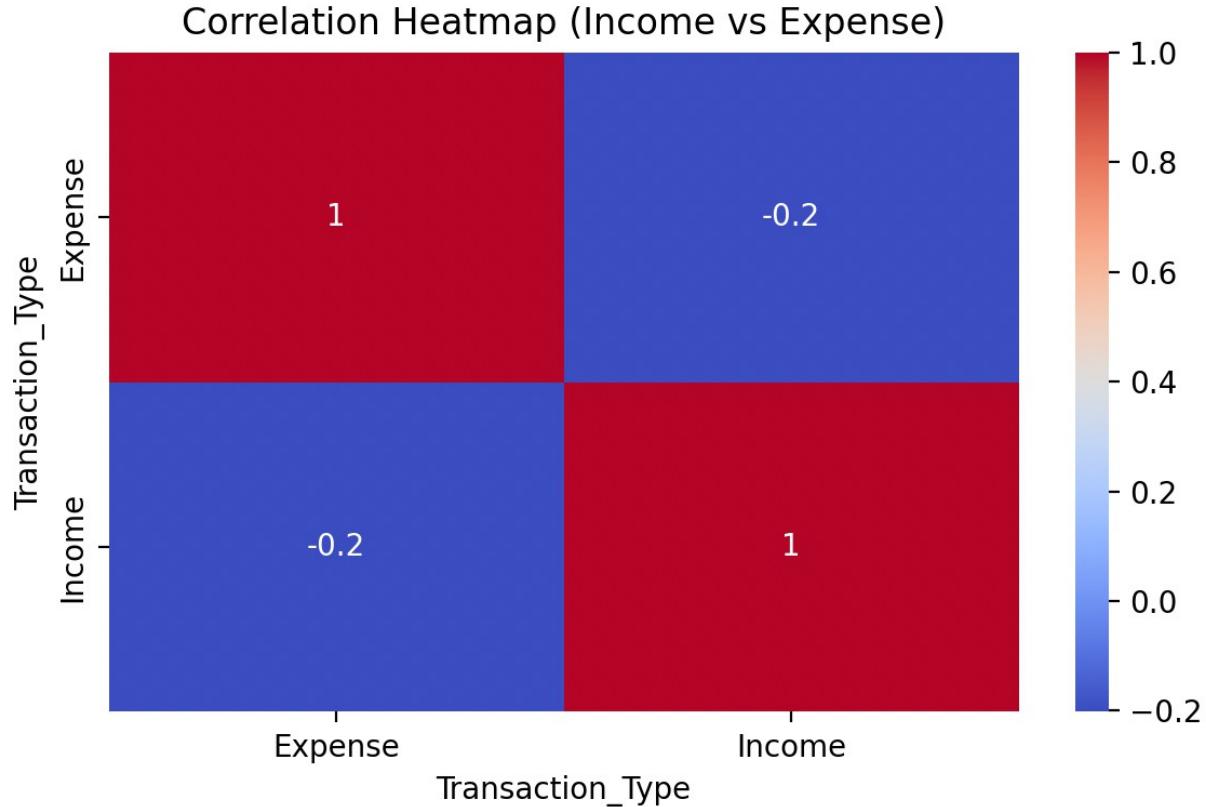
# Create monthly summary
monthly_summary = (
    data.groupby(['Month', 'Transaction_Type'])['Amount']
    .sum()
    .unstack(fill_value=0)
)

plt.figure(figsize=(6, 4))
sns.heatmap(monthly_summary.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap (Income vs Expense)")

```

```
plt.tight_layout()  
plt.show()
```



Conclusion

This project undertook a systematic analysis of personal financial transaction data with the goal of uncovering meaningful patterns in income, expenditure, and savings behavior. The findings yielded several significant insights that collectively paint a comprehensive picture of personal financial health.

Monthly expense trend analysis revealed notable fluctuations in spending behavior across different time periods, suggesting the influence of seasonal factors, irregular purchases, or varying lifestyle demands. These variations underscored the importance of continuous financial monitoring rather than relying on static, one-time assessments.

Category-wise expenditure analysis successfully pinpointed the primary drivers of total spending, highlighting which areas — such as Food, Rent, Shopping, or Fuel — command the greatest financial attention. This granular breakdown enables more targeted budgeting strategies and helps identify categories where spending optimisation is most feasible.

The comparative examination of income against expenses served as a reliable indicator of financial discipline and budget adherence. Periods of surplus and deficit were identified, offering a realistic assessment of overall fiscal management and the effectiveness of day-to-day financial decisions.

Savings and savings rate analysis further reinforced the evaluation of long-term financial stability, revealing whether income growth translated into proportionate savings accumulation or was offset by rising expenditure. This metric stands as one of the most telling indicators of sustainable financial behaviour.

In summation, this project affirms that data analytics is not merely a technical exercise — it is a practical and powerful instrument for personal financial empowerment. By transforming raw transaction records into structured, visual, and interpretable insights, this analysis demonstrates how Python-driven exploratory data analysis can meaningfully support smarter budgeting, disciplined spending, and well-informed financial decision-making. The methodologies applied here can be readily extended to larger datasets or integrated into automated personal finance dashboards for ongoing, real-time financial oversight.