Personal Expense Tracker Data Analysis

Prepared by:

Geethika Meka
Data Analyst

Date:

November 1, 2023

Table of Contents

1. Introduction

1.1 Background and Context

Effective expense tracking is vital for personal financial management. Developing a personal expense tracking application requires a robust dataset that mirrors real-world financial transactions, enabling the analysis of spending habits and income patterns. This report outlines the initial phase of the project, focusing on data gathering and preparation to create a realistic dataset of both income and expenses. The primary objective was to establish a foundation for meaningful data analysis and actionable financial insights.

1.2 Purpose of the Project

The primary purpose of this project is to design and develop a personal expense tracking application by:

Simulating Realistic Data: Creating a synthetic dataset that mirrors actual financial transactions, including both income and expenses, to serve as the backbone for analysis and application functionality.

- **Ensuring Data Quality:** Introducing and addressing common data quality issues, such as missing values and inconsistencies, to enhance the reliability and accuracy of subsequent analyses.
- **Demonstrating Technical Proficiency:** Showcasing skills in data generation, database management, SQL querying, and data visualization to provide a comprehensive solution for personal financial tracking.

1.3 Scope of the Project

This report focuses on the following key areas:

- **Data Collection:** Generation of a dataset of personal financial transactions using Python and relevant libraries.
- **Data Management and Cleaning:** Establishing a MySQL database to store the data, performing data cleaning using Python to address quality issues, and ensuring data integrity.
- **Data Analysis:** Utilizing SQL to extract meaningful financial insights from the dataset, and Exploratory Data Analysis for addressing business questions related to income, expenses, and financial trends.
- **Data Visualization:** Preparing the dataset for visualization through tools like Power BI, Microsoft Excel, and AWS QuickSight to create interactive dashboards and reports.

2. Data Collection

Generate a realistic and diverse dataset of personal financial transactions to emulate real-world financial activity.

Techniques and Tools:

- **Python Programming:** Leveraged Python for data generation, utilizing libraries such as:
    - pandas for data manipulation and storage.
    - numpy for numerical operations.
    - faker for generating realistic synthetic data.
    - random for introducing variability in data selection.

Process:

1. **Initialization:**
    - Initialized the Faker instance to generate realistic names and transaction details.

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from faker import Faker
import random
```

```
[ ] #initialize Fake
    fake = Faker()
```

2. **Defining Categories and Merchants:**
    - Established a list of expense and income categories (e.g., Food, Shopping, Utilities).
    - Mapped each category to relevant merchants or income sources.
3. **Setting Record Counts:**
    - Simulated 100 income transactions and 200 expense transactions to reflect typical personal financial activity distribution over a span of 1 year.

```
# Number of records to generate
num_income_records = 100
num_expense_records = 200
total_records = num_income_records + num_expense_records

# Define possible categories and payment methods
expense_categories = ['Food', 'Shopping', 'Travel', 'Transportation', 'Groceries', 'Bills & Utilities',
                      'Entertainment', 'Health', 'Education', 'Miscellaneous']
income_categories = ['Salary', 'Freelance', 'Investment Returns', 'Gifts', 'Tax Refund', 'Cash Back Rewards', 'From Parents']
payment_methods = ['Cash', 'Credit Card', 'Debit Card', 'Mobile Payment', 'Bank Transfer']


# Define merchants/sources for each category
merchant_dict = {
    # Expense categories
    'Food': ['McDonald\'s', 'KFC', 'Subway', 'Starbucks', 'Pizza Hut', 'IHOP', 'Happy Lemon', 'Panda Express', 'Local restaurant'],
    'Shopping': ['Outlet mall', 'Target', 'Amazon', 'Best Buy', 'Adidas', 'Nike'],
    'Travel': ['American Airlines', 'United Airlines', 'Delta Airlines', 'Southwest Airlines', 'Continental Hotel', 'Marriott Hotel', 'Hyatt', 'Holiday Inn
    'Transportation': ['Uber', 'Lyft', 'Taxi Service', 'Public Transit'],
    'Groceries': ['Walmart', 'Costco', 'Kroger', 'Whole Foods', 'Safeway', 'HEB', 'Indian store', 'Trader Joe\'s', 'Sam\'s Club'],
    'Bills & Utilities': ['AT&T', 'Verizon', 'Comcast', 'Electric Company', 'Water Company'],
    'Entertainment': ['Netflix', 'Spotify', 'AMC Theatres', 'Hulu', 'Disney+', 'Cinemark'],
    'Health': ['CVS Pharmacy', 'Walgreens', 'Rite Aid', 'Doctor\'s Office', 'Dental Clinic'],
    'Education': ['Udemy', 'Coursera', 'University Tuition Fees', 'Bookstore', 'Online Course', 'LinkedIn'],
    'Miscellaneous': ['Gift Shop', 'Charity Donation', 'Miscellaneous Purchase', 'Orange Theory Fitness', 'Spa', 'Salon'],

    # Income categories
    'Salary': ['Employer Inc.', 'Company LLC', 'Business Corp.'],
    'Freelance': ['Client A', 'Client B', 'Client C'],
    'Investment Returns': ['Investment Bank', 'Stock Brokerage'],
    'Gifts': ['Family Member', 'Friend', 'Relative'],
    'Tax Refund': ['IRS', 'State Tax Agency'],
    'Cash Back Rewards': ['Credit Card Rewards', 'Bank Rewards Program'],
    'From Parents': ['Mom', 'Dad', 'Parents']
}
```

4. **Data Generation:**
    - **Income Transactions:**
        - Randomly selected income categories and associated merchants.
        - Generated amounts between $100 and $3,000.
        - Created descriptive notes tailored to income types.

```python
# List to hold generated data
data = []

# Function to introduce missing values
def introduce_missing_value(value, missing_probability):
    return None if random.random() < missing_probability else value

# Set the probability of missing values (5%)
missing_prob = 0.05

# Generate income data
for _ in range(num_income_records):
    category = random.choice(income_categories)
    merchant = random.choice(merchant_dict.get(category, ['Unknown Source']))
    amount = round(random.uniform(100.0, 3000.0), 2)
    payment_method = random.choice(payment_methods)
    date = fake.date_between(start_date='-1y', end_date='today')
    transaction_id = fake.uuid4()
    user_id = fake.uuid4()

    # Generate a meaningful note
    notes_template = random.choice(notes_dict.get(category, ['Income from {}']))
    note = notes_template.format(merchant)

    # Introduce missing values
    merchant = introduce_missing_value(merchant, missing_prob)
    payment_method = introduce_missing_value(payment_method, missing_prob)
    note = introduce_missing_value(note, missing_prob)
    category = introduce_missing_value(category, missing_prob)

    transaction = {
        'Transaction_ID': transaction_id,
        'Date': date,
        'Amount': amount,
        'Category': category,
        'Payment_Method': payment_method,
```

    - **Expense Transactions:**
        - Randomly selected expense categories and associated merchants.
        - Generated amounts between $1 and $800.
        - Created descriptive notes tailored to expense types.
5. **Data Compilation:**
    - Combined all transactions into a single DataFrame.
    - Saved the dataset as expenses_data.csv.

3. Data Cleaning and Transformation

- **Identify Data Quality Issues**: Detect missing values, duplicates, incorrect data types, and inconsistencies.
- **Clean the Dataset**: Apply appropriate methods to handle identified issues.

Dataset description:

- **Transaction_ID**: Unique identifier for each transaction.
- **Date**: Date of the transaction within the past year.
- **Amount**: Monetary value of the transaction
- **Category**: Classification of the transaction (e.g., Food, Salary).
- **Payment_Method**: Method used for the transaction (e.g., Credit Card, Cash).
- **Merchant**: Name of the merchant or source associated with the transaction.
- **Notes**: Additional details about the transaction.
- **Type**: Indicates whether the transaction is an 'Income' or an 'Expense'.

```
df.head(5)
```

	Transaction_ID	Date	Amount	Category	Payment_Method	Merchant	User_ID	Notes	Type
0	b66c14cf-69c6-4f52-a540-ea81d3a1a1c5	2024-03-26	1430.20	Investment Returns	Credit Card	Investment Bank	d59b1b46-a964-4859-8443-d069cc71bd53	Dividend from Investment Bank	Income
1	0ef9593f-b7c5-4349-b3d8-625b4f3d6167	2024-10-18	2206.00	Tax Refund	Credit Card	State Tax Agency	eb566992-bcaf-406c-b1f3-df72bb01a225	Tax refund from State Tax Agency	Income
2	23104313-4867-4902-b284-c817308de987	2024-09-25	1204.93	Cash Back Rewards	Mobile Payment	Credit Card Rewards	05ec22b3-493e-4d6b-ac0a-e561058d5dbe	Cash back from Credit Card Rewards	Income
3	31b61466-4e46-43b5-a4fb-324222b3ca4f	2024-03-19	464.27	Food	Cash	Happy Lemon	8d84b9bc-e240-4545-9759-836a9392bae4	Coffee from Happy Lemon	Expense
4	ae85751c-1481-4ffd-8023-22fa70ab0ca7	2024-07-26	160.94	Bills & Utilities	Mobile Payment	AT&T	e0b3d19d-5fe1-4fae-b866-66147f26ca9d	Paid bill to AT&T	Expense

Data Validation:

1. Data Type Verification

- Converted the 'Date' column to datetime format.
- Ensured 'Amount' is of float type.

```
#Check data types
df.dtypes
```

	0
Transaction_ID	object
Date	datetime64[ns]
Amount	float64
Category	object
Payment_Method	object
Merchant	object
User_ID	object
Notes	object
Type	object

dtype: object

2. Missing Values Identification

- Filled missing 'Category' values with mode.
- Filled missing 'Payment_Method' values with mode.
- Filled missing 'Merchant' values with 'Unknown Merchant'.
- Filled missing 'Notes' values with an ‘no notes available’.

```
#check missing values
missing_values = df.isnull().sum()
missing_values
```

	0
Transaction_ID	0
Date	0
Amount	0
Category	20
Payment_Method	12
Merchant	12
User_ID	0
Notes	12
Type	0

dtype: int64

```
#replace missing values in category
mode_category = df['Category'].mode()[0]
df['Category'].fillna(mode_category, inplace=True)
```

```
[16] #replace payment method
mode_payment = df['Payment_Method'].mode()[0]
df['Payment_Method'].fillna(mode_payment, inplace=True)
```

```
[17] #replace merchant
df['Merchant'].fillna('Unknown Merchant', inplace=True)
```

```
[18] #replace notes
df['Notes'].fillna('No notes available', inplace=True)
```

3. Duplicate Records Identification

- Duplicates were identified based on the 'Transaction_ID' field and removed.

```
#check duplicates
df.duplicated().sum()
```

```
9
```

+ Code + Text

Add code cell ⌘/Ctrl+M B

```
[11] #drop duplicates
df.drop_duplicates(subset='Transaction_ID', keep='first', inplace=True)
df.duplicated().sum()
```

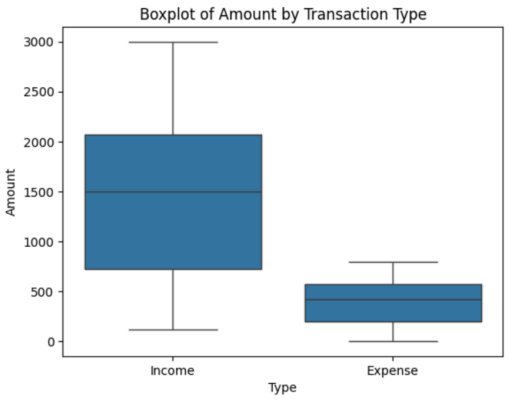
```
0
```

4. Outlier Detection

- Analyzed the 'Amount' field to identify potential outliers.
- Amounts were within the expected ranges, removal of outliers was not necessary.

```
#Outlier Detection and Handling
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x='Type', y='Amount', data=df)
plt.title('Boxplot of Amount by Transaction Type')
plt.show()
```



5. Data Transformation

- Applied ordinal encoding to month names to handle categorical variables.
- Extracted year, month, date from date field.

```
[36] #Extract 'Year' and 'Month' from 'Date'

df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Month_Name'] = df['Date'].dt.month_name()
```

```
#data transformation
#ordical encoding

# Define the chronological order of months
months_order = ['January', 'February', 'March', 'April', 'May', 'June',
                'July', 'August', 'September', 'October', 'November', 'December']

# Create a mapping dictionary for ordinal encoding
month_map = {month: index for index, month in enumerate(months_order, start=1)}

# Perform ordinal encoding on the 'Month' column
df['Month_Num'] = df['Month'].map(month_map)

# Verify the encoding
print(f"After Ordinal Encoding:\n{df[['Month']].head()}\n")
```

```
After Ordinal Encoding:
            Month
Date
2024-03-26      3
2024-10-18     10
2024-09-25      9
2024-03-19      3
2024-07-26      7
```

4. Exploratory Data Analysis

4.1. Descriptive Statistics

Summary of Transactions

- **Total Number of Transactions:** Identified 300 records in 1 year

```
[20] #total number of transactions
     total_transactions = len(df)
     total_transactions
```

```
300
```

```
[21] #date range of transactions
     start_date = df['Date'].min()
     end_date = df['Date'].max()
     print(f"Date Range: {start_date} to {end_date}")
```

```
Date Range: 2023-11-08 00:00:00 to 2024-11-06 00:00:00
```

- **Income Transactions:**
  - Represented approximately 33.3% of all transactions.
  - Average income transaction amount was notably higher than expenses.

```
[22] #total amount of transactions

     total_income = df[df['Type'] == 'Income']['Amount'].sum()
     total_expenses = df[df['Type'] == 'Expense']['Amount'].sum()
     net_income = total_income + total_expenses  # Expenses are negative
     print(f"Total Income: ${total_income:.2f}")
     print(f"Total Expenses: ${total_expenses:.2f}")
     print(f"Net Income: ${net_income:.2f}")
```

```
Total Income: $143389.53
Total Expenses: $79531.32
Net Income: $222920.85
```

- **Expense Transactions:**
  - Represented approximately 67% of all transactions, average expense transaction amount was lower but occurred more frequently.

```
[24] avg_income = df[df['Type'] == 'Income']['Amount'].mean()
     avg_expenses = df[df['Type'] == 'Expense']['Amount'].mean()
     print(f"avg_income: ${avg_income:.2f}")
     print(f"avg_expenses: ${avg_expenses:.2f}")
```

```
avg_income: $1433.90
avg_expenses: $397.66
```

Transaction Amount

- **Overall Statistics:**
  - **Mean Transaction Amount:** Average income is $1540 and average expense is $788.
  - **Median Transaction Amount:** Median income is $1559 and median expense is $407.
  - **Minimum and Maximum Amount:**
  - Income: Minimum $100, Maximum $3,000
  - Expenses: Minimum $1, Maximum $800

```
[27] #Highest income and expense
     highest_income = df[df['Type'] == 'Income']['Amount'].max()
     highest_expense = df[df['Type'] == 'Expense']['Amount'].max()
     print(f"Highest Income: ${highest_income:.2f}")
     print(f"Highest Expense: ${highest_expense:.2f}")
```

```
Highest Income: $2997.12
Highest Expense: $794.38
```

```
[28] #Lowest income and expense
     lowest_income = df[df['Type'] == 'Income']['Amount'].min()
     lowest_expense = df[df['Type'] == 'Expense']['Amount'].min()
     print(f"lowest_income: ${lowest_income:.2f}")
     print(f"lowest_expense: ${lowest_expense:.2f}")
```

```
lowest_income: $120.27
lowest_expense: $2.25
```

Category Analysis

- **Top Expense Categories by Total Amount:**
  - **Highest Amount:** ‘Bills & Utilities’ for the largest portion of expenses.
  - **Lowest Amount:** ‘Entertainment’’ for the smallest portion of expenses.
- **Top Income Categories:**
  - **Highest Amount:** ‘Investment Returns’ for the largest portion of income.
  - **Lowest Amount:** ‘Tax Refund’ and ‘From Parents’ for the smallest portion of income.

```
category_stats = df.groupby('Category')['Amount'].agg(['sum', 'mean', 'count'])
category_stats
```

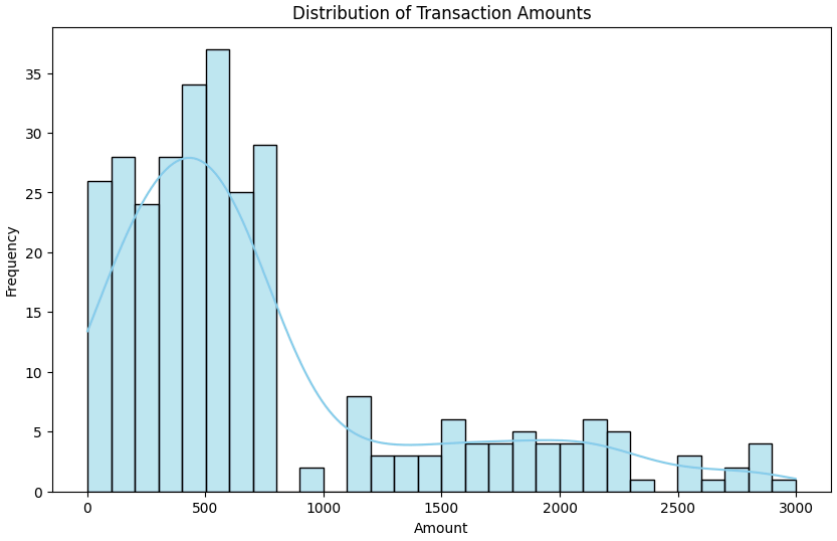
Category	sum	mean	count
Bills & Utilities	7873.69	414.404737	19
Cash Back Rewards	20929.87	1395.324667	15
Education	6792.54	485.181429	14
Entertainment	6432.59	321.629500	20
Food	20637.65	491.372619	42
Freelance	18305.37	1307.526429	14
From Parents	21085.92	1757.160000	12
Gifts	14889.24	1353.567273	11
Groceries	7419.70	370.985000	20
Health	8767.92	461.469474	19
Investment Returns	17764.00	1184.266667	15
Miscellaneous	6407.48	400.467500	16
Salary	13782.46	1531.384444	9
Shopping	6971.81	366.937368	19
Tax Refund	28418.98	1671.704706	17
Transportation	9125.71	414.805000	22
Travel	7315.92	457.245000	16

4.2. Univariate Analysis

Analyze individual variables to understand their distributions.

Transaction Amount

- **Distribution:**The distribution of transaction amount was right-skewed, with most transactions being of smaller amount.



Interpretation for skewness:

Skewness > 0: Right-skewed distribution.

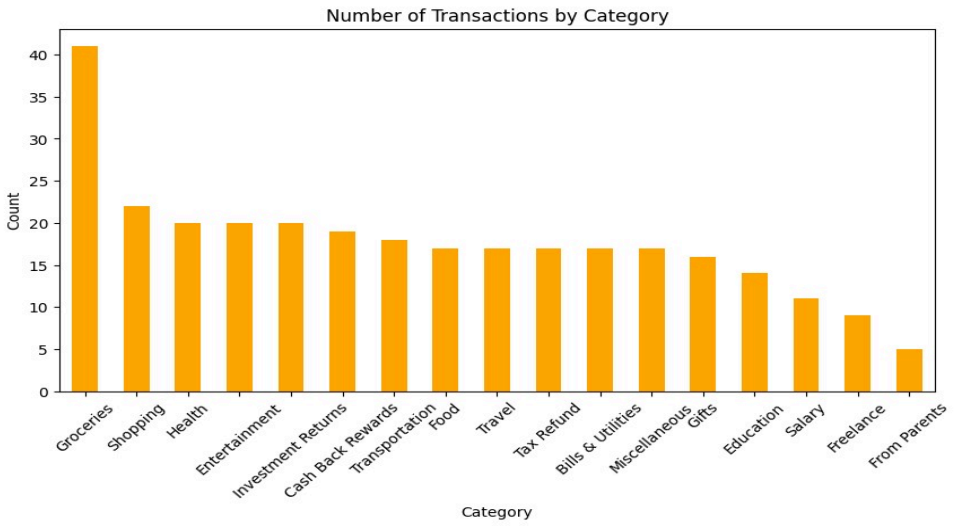
Skewness < 0: Left-skewed distribution.

Skewness ≈ 0: Symmetrical distribution.

```
skewness = df['Amount'].skew()
print(f'Skewness of transaction amounts: {skewness}')
```

```
Skewness of transaction amounts: 1.4770310906278759
```

- **Frequency**: Transaction frequency was high for ‘Groceries’ and ‘Shopping’ and was low for ‘Salary’.

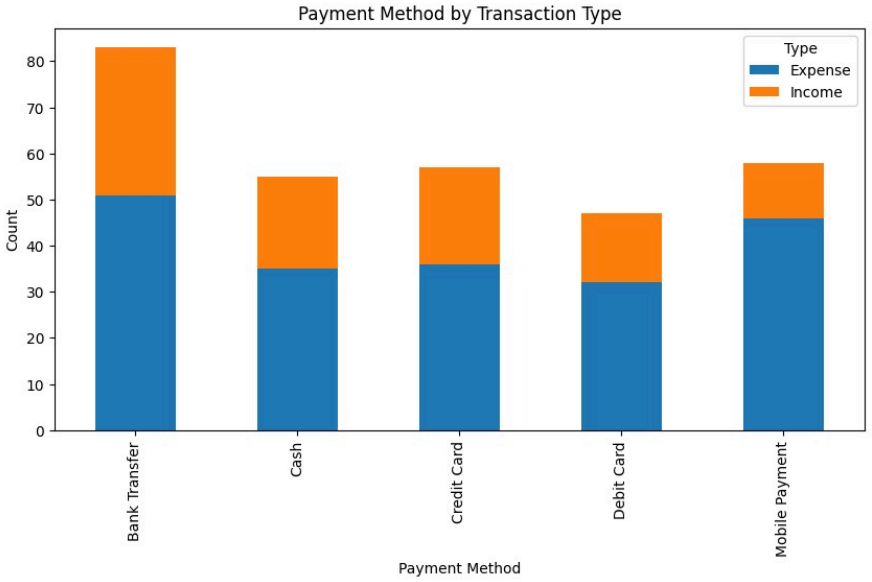


4.3. Bivariate Analysis

Explore relationships between pairs of variables.

Payment Method

- **Most frequent**: ‘Bank transfers’ are the most frequently used for expenses and income transactions.
- **Least frequent**: ‘Debit card’ transactions are the least utilized for both categories.

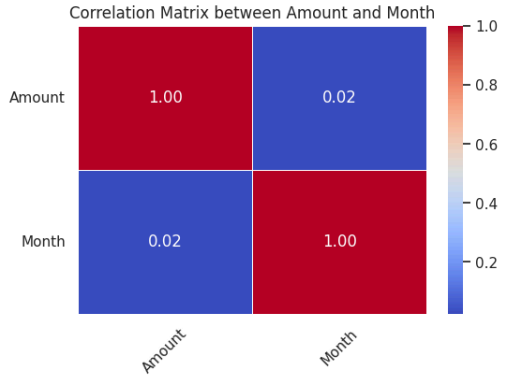


4.4. Multivariate Analysis

Examine interactions among multiple variables.

Correlation Matrix

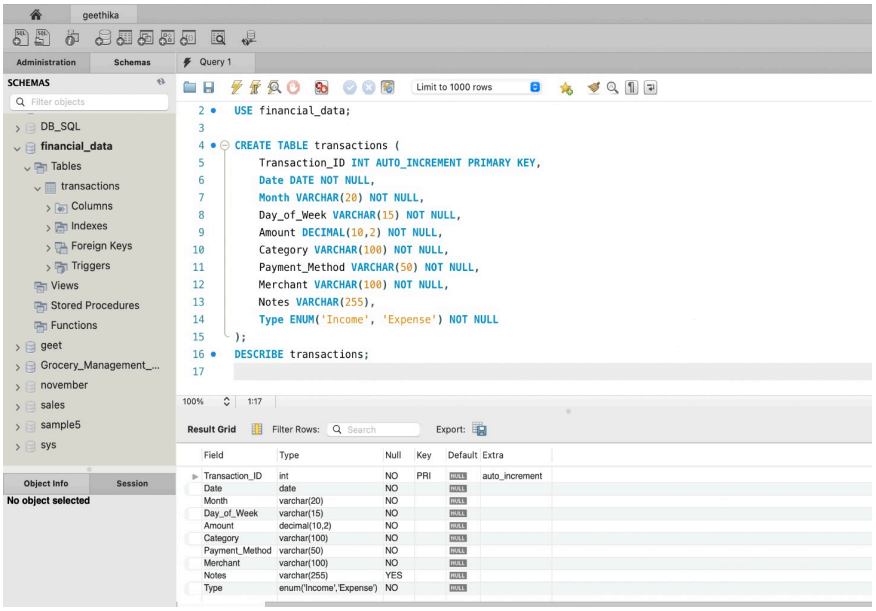
- The correlation matrix showed a strong negative correlation of **-0.57** between transaction amount and month numbers, indicating that amount decrease from January to December. This reflects a seasonal trend in spending or income over the year.



5. Data Management

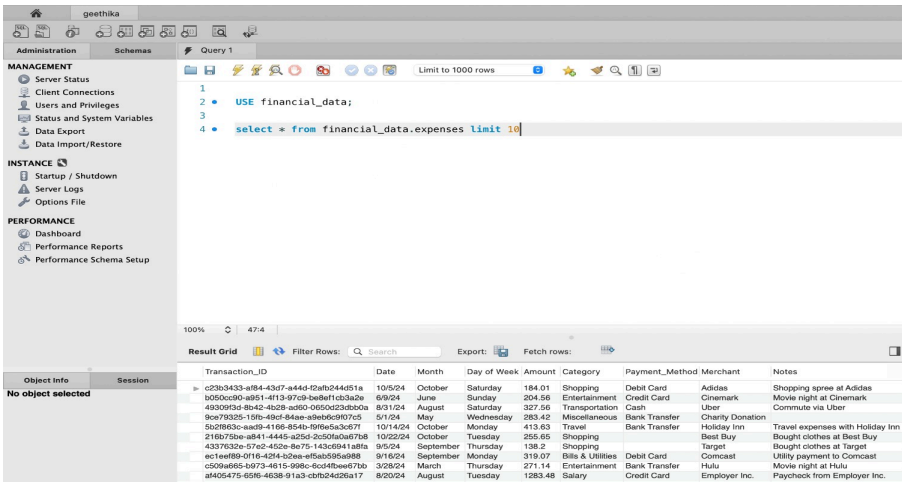
5.1 Data Importation:

- Created a MySQL database named ‘financial_data’.
- Designed the expenses table with appropriate data types and constraints.



5.2 Data Quality Assessment:

- Verified data types (e.g., Date as DATE, Amount as DECIMAL).
- Removed duplicate records based on Transaction_ID.

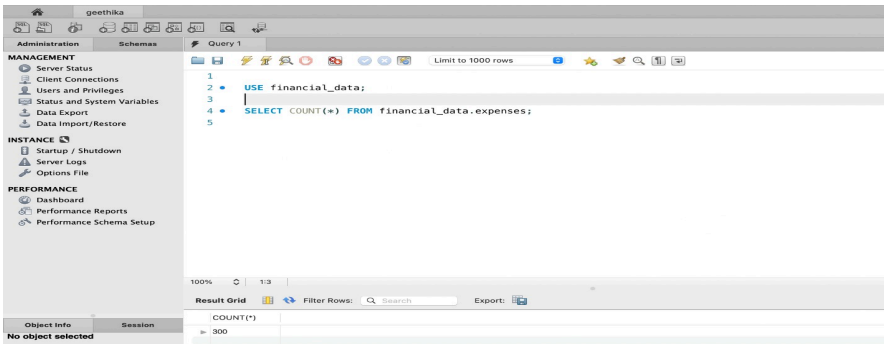


5.3 SQL Queries:

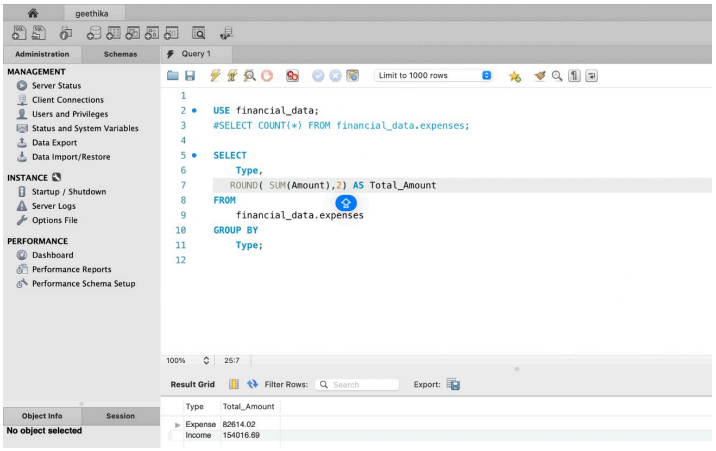
- Utilized MySQL to perform data aggregation, grouping, and filtering to address key business questions.

Key Analyses Conducted:

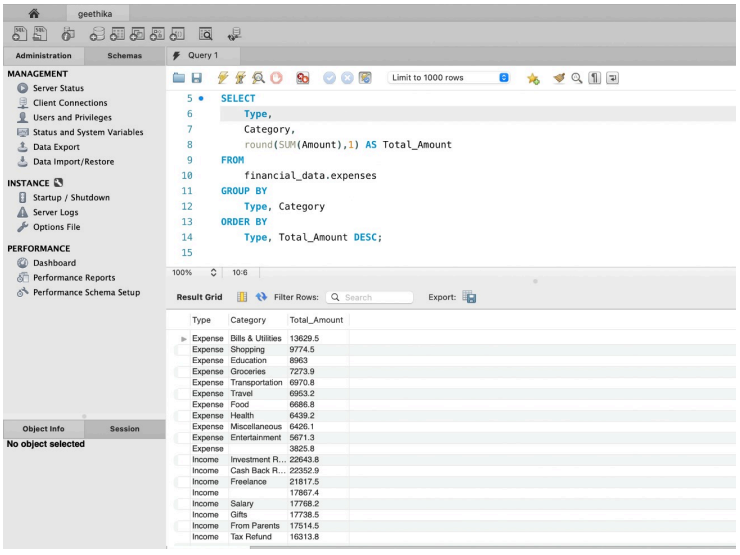
1. Count of records:



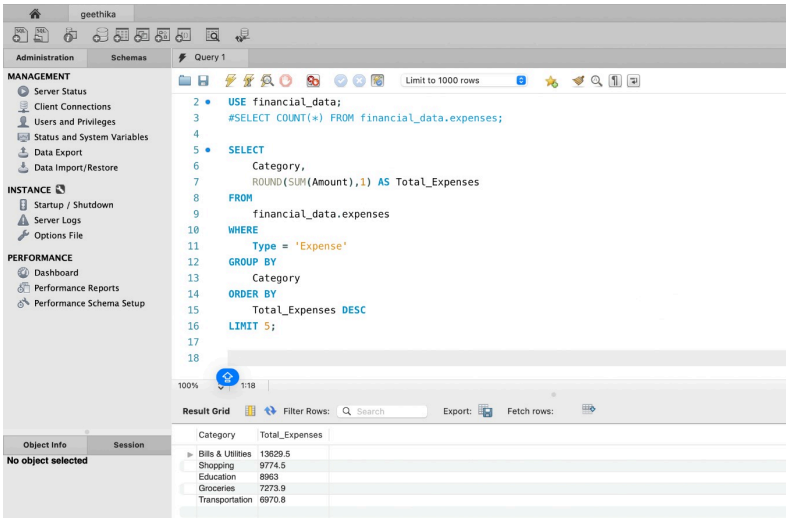
2. Total Expenses and Income



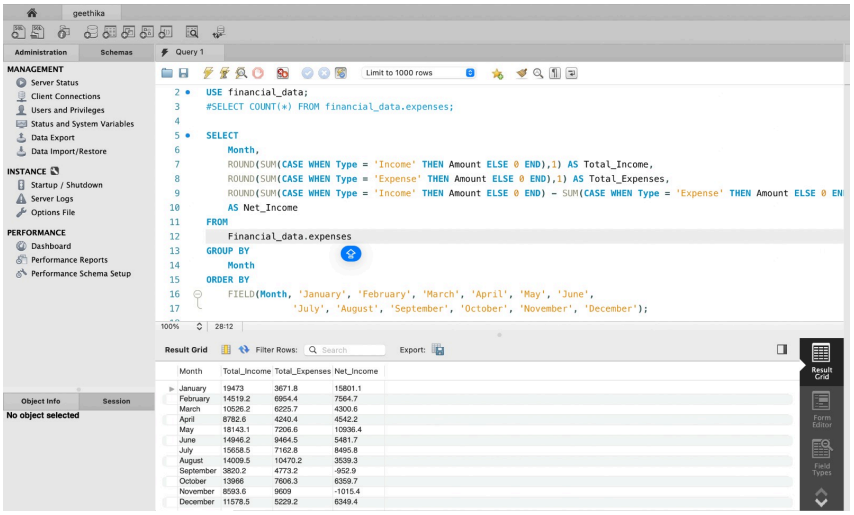
3. Expenses by Category



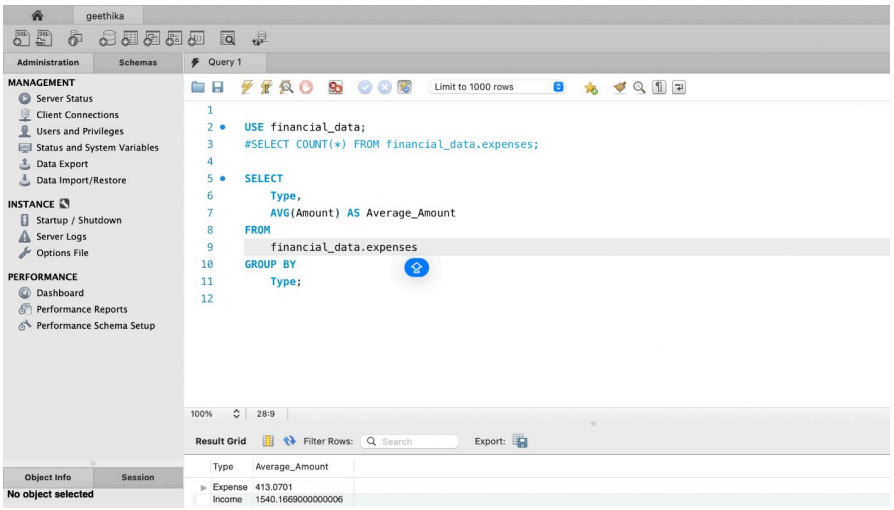
4. Top 5 Spending categories



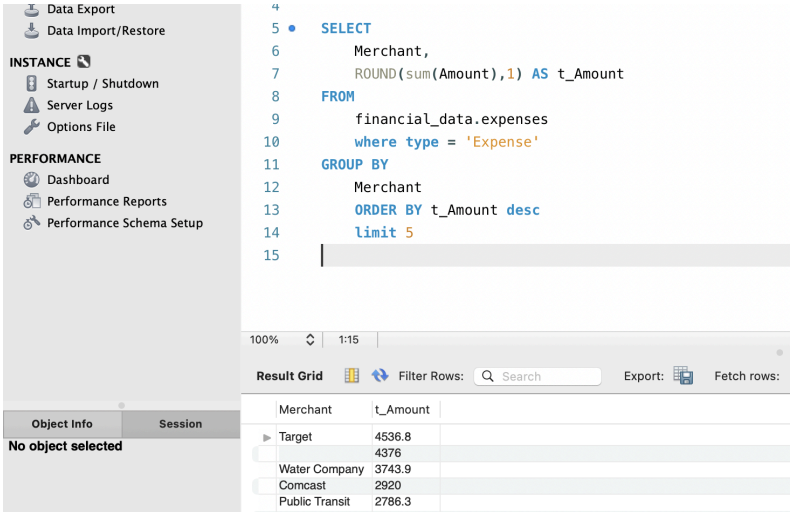
5. Monthly Expense, Income Trends



6. Average Expenses and Income



7. Top 5 Spending Merchants



8. Average Expense per Category

```
SELECT
    category,
    ROUND(AVG(Amount),1) AS Average_Amount
FROM
    financial_data.expenses
GROUP BY
    category
    ORDER BY Average_Amount ASC;
```

100% 20:2

Result Grid Filter Rows: Search Export:

category	Average_Amount
Health	357.7
Miscellaneous	378
Travel	386.3
Education	389.7
Groceries	404.1
Entertainment	405.1
Shopping	407.3
Transportation	435.7
Bills & Utilities	454.3
Food	477.6
	1141.7
Tax Refund	1254.9
Cash Back R...	1314.9
From Parents	1459.5
Gifts	1612.6
Salary	1615.3
Freelance	1678.3
Investment R...	1887

6. Data Visualization

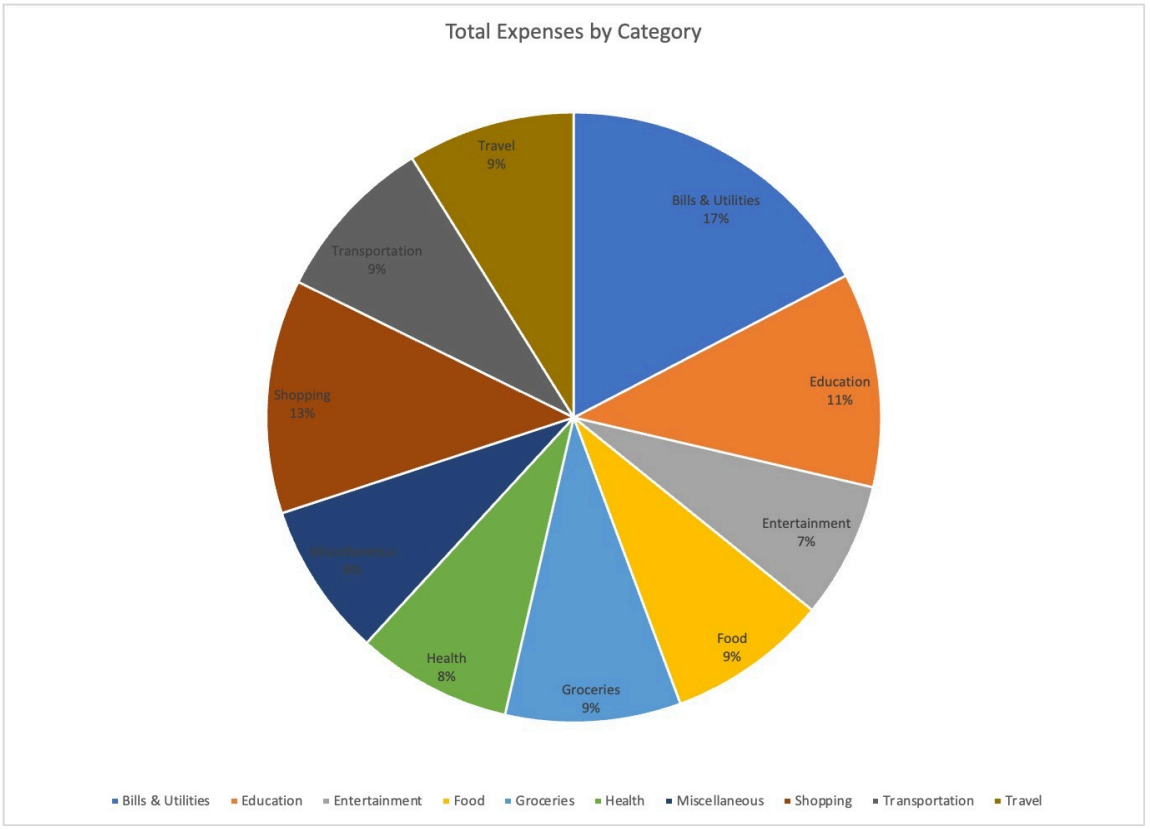
6.1 Spending Patterns

1. **Time Series Analysis:**
   - Daily and Monthly expenses were plotted to identify patterns.
   - Noted higher spending on weekends and specific dates.

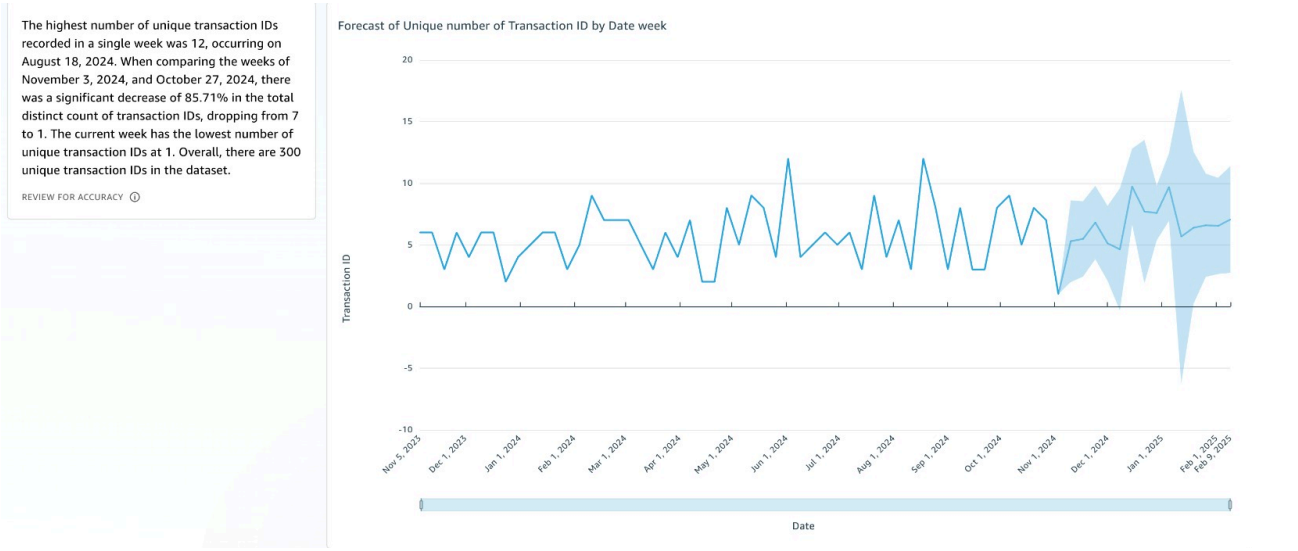
- **Seasonality:** Certain periods showed increased spending, possibly due to holidays or personal events.

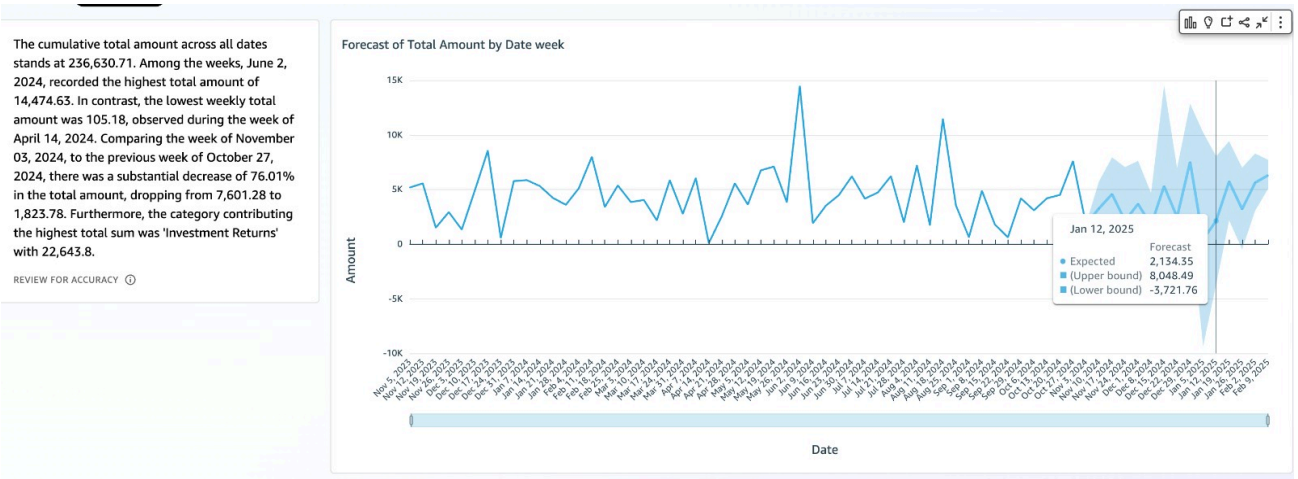
2. Expenses by Category

- Used Excel to identify 'Bills & Utilities' contribute to the highest (17%)
- 'Entertainment' is the least (7%)



3. **Predictive Models**
   - Used AWS Quick Sight to forecast the total amount by week for the next 3 months.



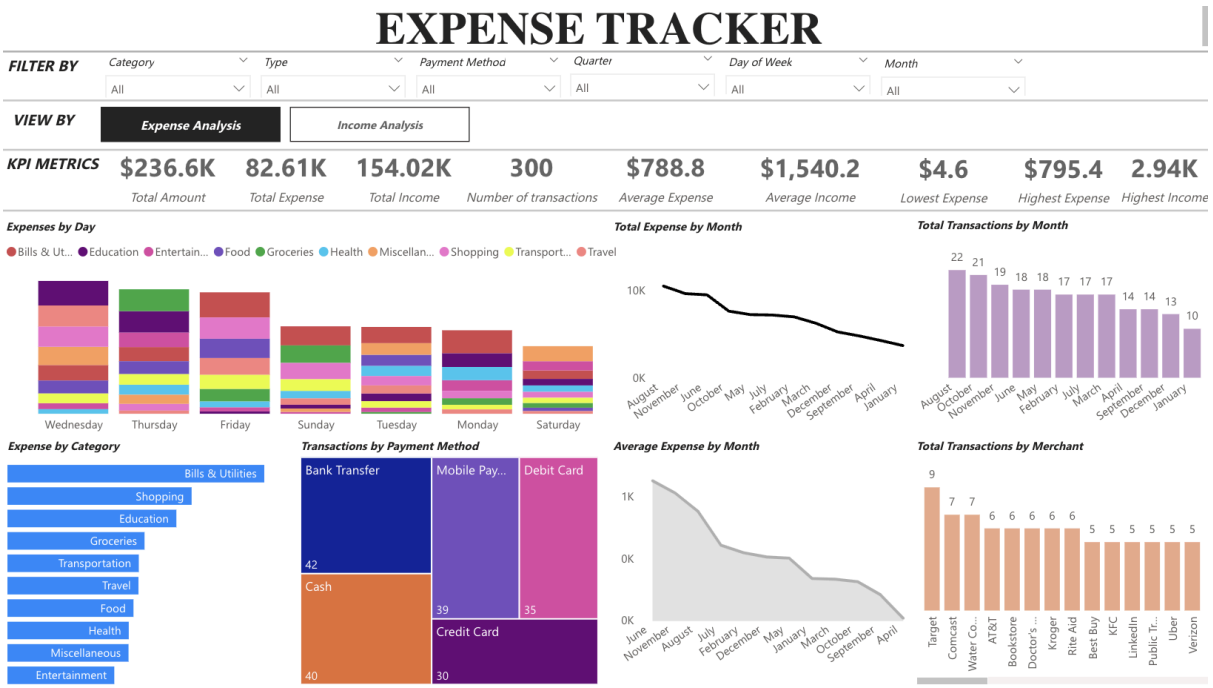


6.2 Power BI Visualizations

The visualizations are categorized into **Expenses Analysis**, **Income Analysis**, and a **Summary Dashboard** that integrates both.

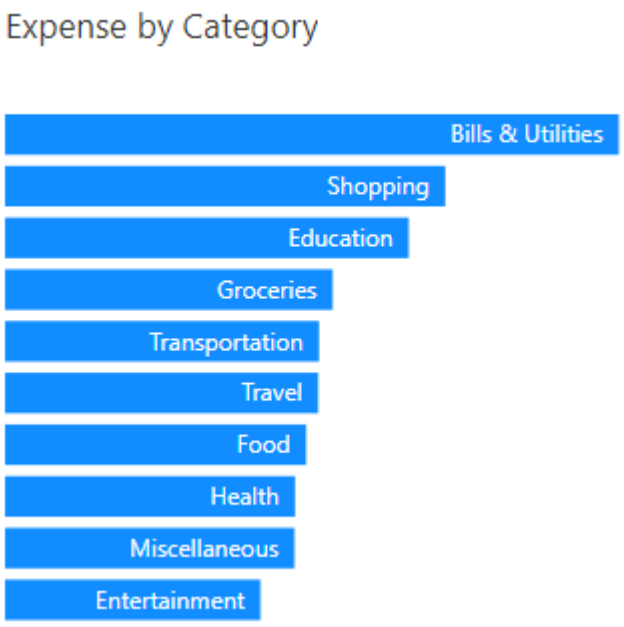
1. Expenses Dashboard

The **Expenses Dashboard** focuses on understanding spending patterns, identifying major expense categories, and tracking expenses over time.



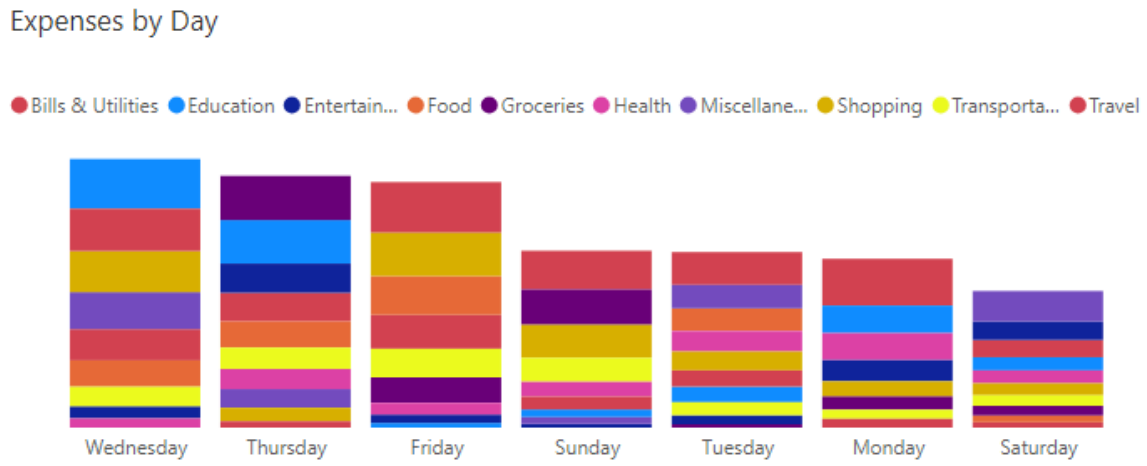
1.1 Expenses by Category (Clustered Bar Chart)

To visualize the proportion of total expenses attributed to each category, enabling quick identification of major spending areas.



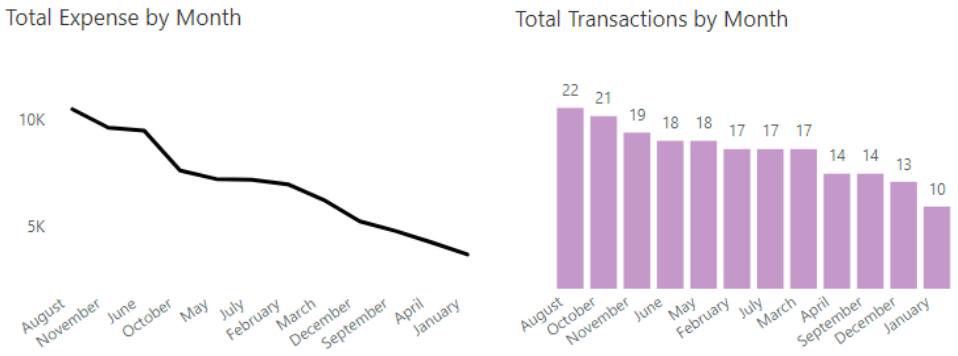
1.2 Expenses by Week Day (Stacked Column Chart)

To provide a deeper breakdown of expenses per day of the week, facilitating detailed analysis of spending.



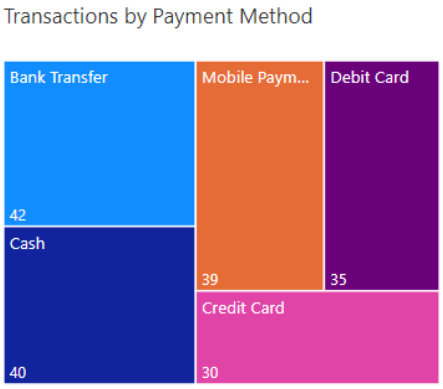
1.3 Monthly Expense Trend (Line Chart)

To track the trend of expenses over time, number of transactions, identifying patterns, peaks, and troughs in spending.



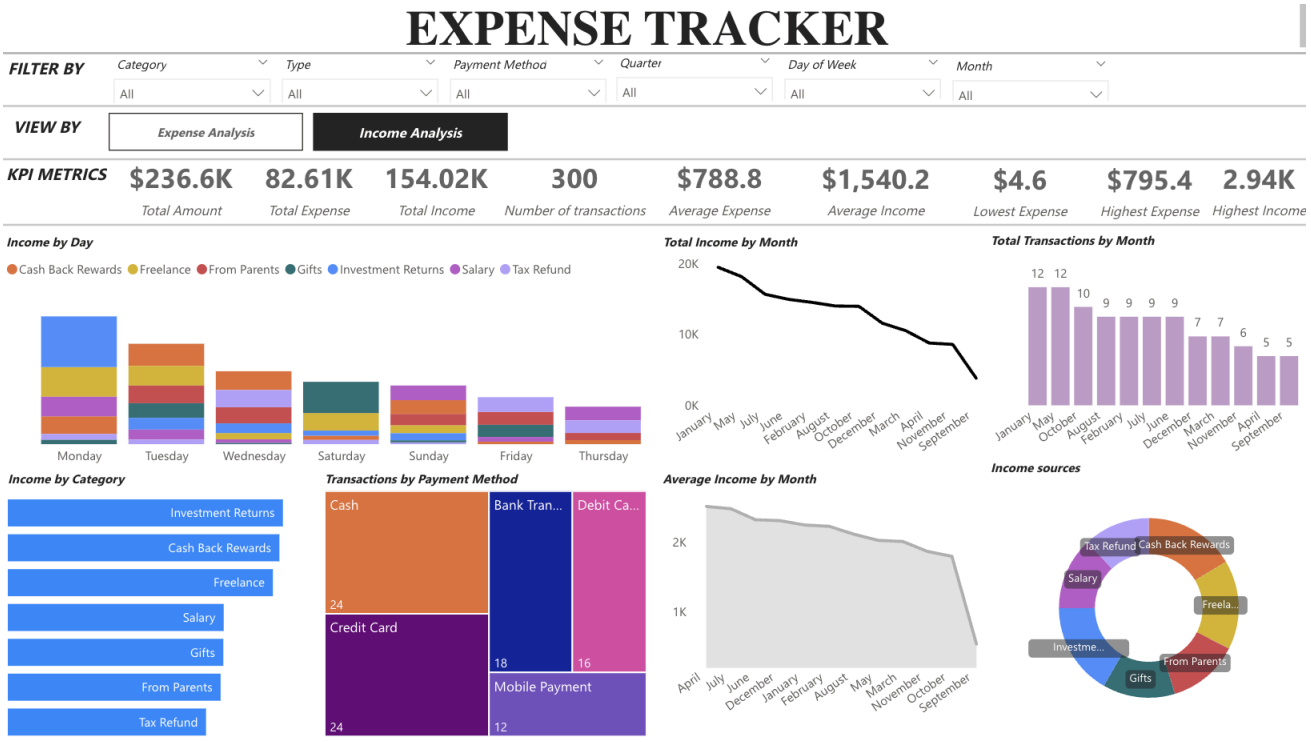
1.4 Transactions by Payment Method (Tree Map)

To identify which payment method was used for spending and ‘bank transfer’ was the most frequently used payment method.



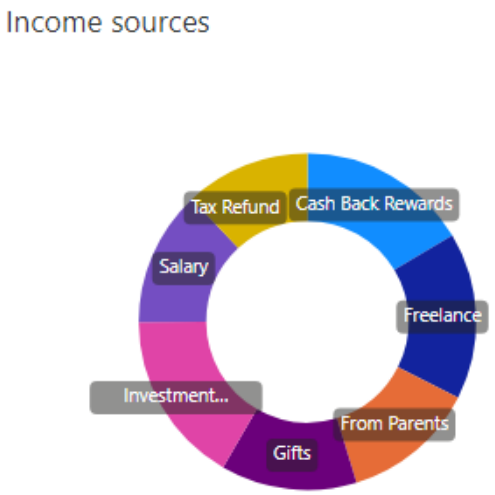
2. Income Dashboard

The **Income Dashboard** focuses on analyzing income sources, tracking income over time, and understanding income distribution.



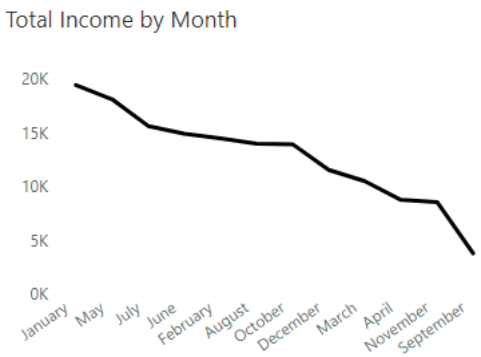
2.1 Income by Source (Pie Chart)

To visualize the proportion of total income from different sources, aiding in identifying primary income streams.



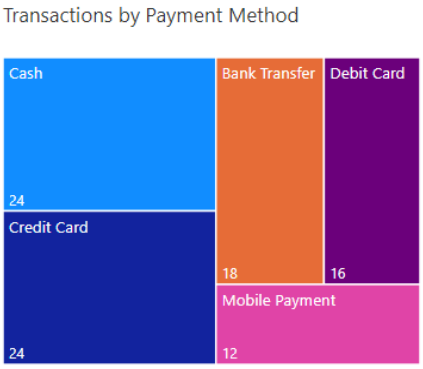
2.2 Monthly Income Trend (Line Chart)

To monitor the progression of income over time, identifying growth trends and fluctuations.



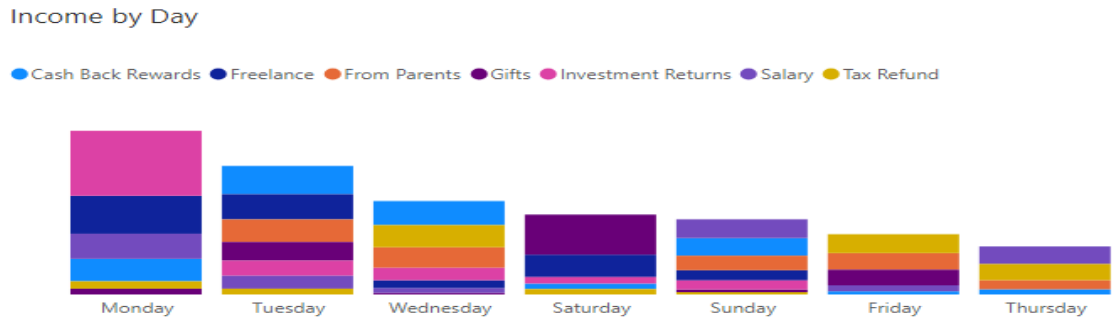
2.3 Transactions by Payment Method (Tree Map)

To identify which payment method was used for receiving income, ‘cash’ was the most frequently used payment method and ‘mobile payment’ was the least.



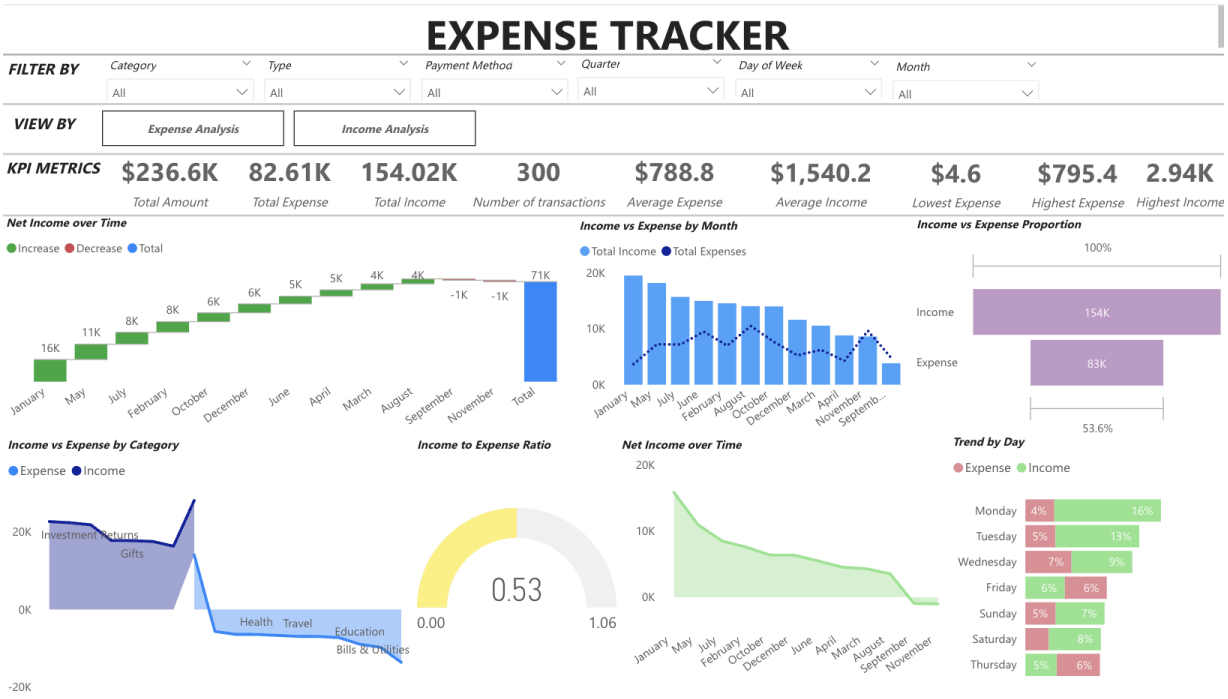
1.4 Income by Week Day (Stacked Column Chart)

To provide a deeper breakdown of income per day of the week, facilitating detailed analysis of spending.



3. Summary Dashboard

The **Summary Dashboard** integrates both **Income** and **Expenses** visualizations, providing a holistic view of financial performance and facilitating comparisons.

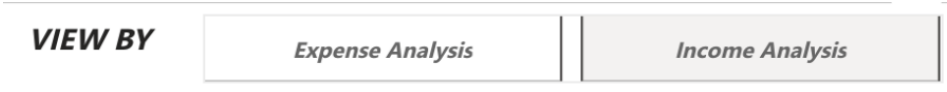


3.1 Filters

Filters enable users to refine dashboard data by selecting specific time periods, categories, or transaction types for customized insights.

3.2 Bookmark Navigator

Bookmark Navigator allows users to quickly switch between different dashboard views and predefined report states for efficient data exploration.

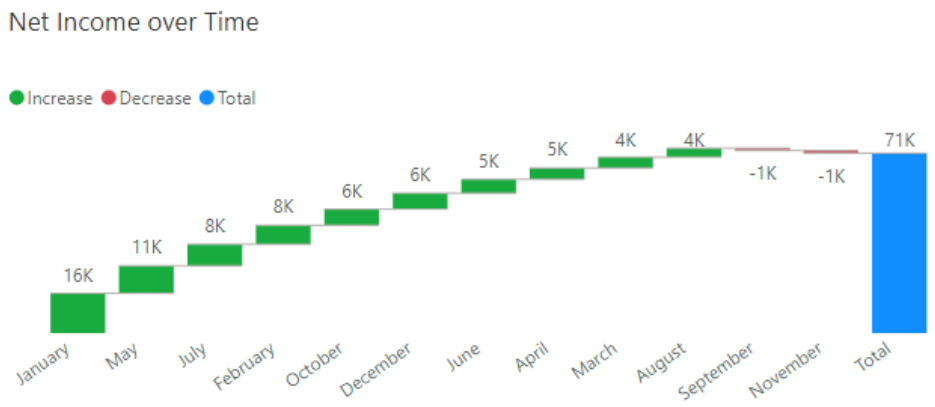


3.3 KPI (Card Visuals)

Displays key financial metrics such as total income, total expenses, and net income, offering an immediate snapshot of financial health.

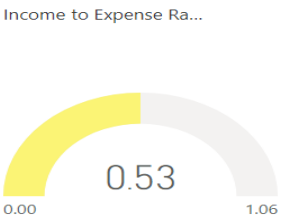
3.4 Income vs. Expenses Over Time (Waterfall Chart)

Illustrates the cumulative impact of income and expenses over time, highlighting key changes and overall financial trends.



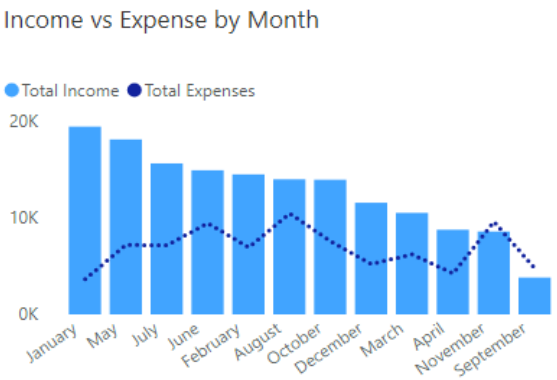
3.5 Expense to Income Ratio (Gauge Chart)

Shows the ratio of expenses to income on a single gauge, indicating financial efficiency and highlighting potential overspending.



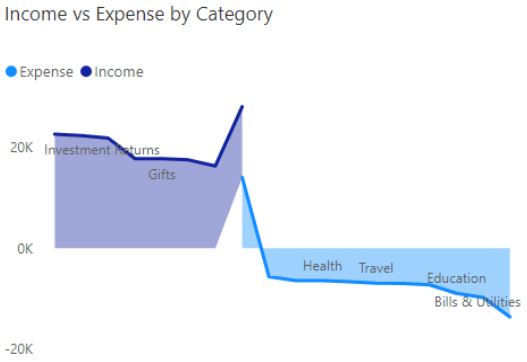
3.6 Expense vs Income by Month (Line and Stacked Chart)

Visualizes monthly income and expenses trends using overlapping lines and stacked areas to compare their progression throughout the year.



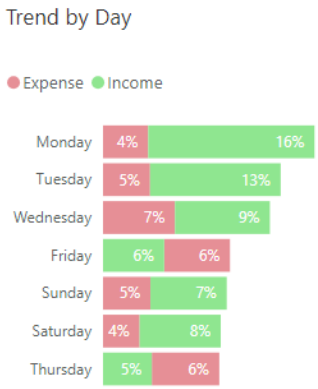
3.7 Expense vs Income by Category (Stacked Area Chart)

Displays the distribution of income and expenses across various categories over time, using stacked areas to illustrate their relative proportions.



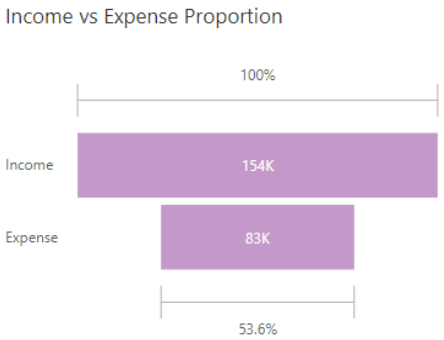
3.8 Trend by Day (Stacked Bar Chart)

Shows daily income and expense trends with stacked bars, enabling the identification of spending patterns and peak activity days.



3.9 Income vs Expense Proportion (Funnel Chart)

Represents the proportion of income to expenses through different stages, illustrating the conversion and allocation of funds



7. Conclusion

The analysis demonstrates effective data management and utilization of SQL for extracting valuable financial insights. Through interactive visualizations in Power BI, Excel, and QuickSight, the data reveals clear spending patterns, such as increased expenses during December and higher spending on weekends. Income remains relatively stable with slight growth in specific months. The strong negative correlation between transaction amounts and month numbers highlights a seasonal trend in financial behavior. These insights empower stakeholders to make informed budgeting decisions and strategize for sustained financial health.

8. References

- **MySQL Documentation:** https://dev.mysql.com/doc/
- **Power BI Tutorials:** https://docs.microsoft.com/en-us/power-bi/
- **AWS QuickSight Guides:** https://docs.aws.amazon.com/quicksight/
- **Python Libraries:**
    - **pandas:** https://pandas.pydata.org/
    - **numpy:** https://numpy.org/
    - **faker:** https://faker.readthedocs.io/
    - **random:** https://docs.python.org/3/library/random.html