

Project Name - Daily Transactions (ML _ FA _ DA projects)(Part 2)

Project Type - Data Analysis

Industry - Unified Mentor

Contribution - Individual

Member Name - Hare Krishana Mishra

Task - 2

Project Summary -

Project Description:

The Daily Household Transactions project analyzes a dataset of everyday financial transactions, including purchases, subscriptions, transportation, investments, and income sources. By exploring this data, the project aims to uncover spending patterns, highlight areas for cost optimization, and provide actionable insights for personal finance management. The dataset contains attributes such as transaction date, payment mode, category, subcategory, notes, amount, and whether the transaction is an income or expense.

The project involves data cleaning, exploratory data analysis (EDA), visualization, and trend analysis, enabling deeper understanding of financial habits over time.

Objective:

- **Identify Spending Patterns** - Detect trends in expenditure and income over time to understand financial behavior.
- **Category Analysis** - Highlight which categories (e.g., Food, Transportation, Household) consume the most budget.
- **Payment Mode Insights** - Analyze the most frequently used payment methods and their contribution to expenses or income.
- **Time Series Analysis** - Evaluate daily and monthly trends to identify peak spending periods.
- **Support Decision-Making** - Provide data-driven recommendations for budgeting, saving, and financial planning.

Key Project Details:

Domain: Finance Analytics / Personal Finance Management

Difficulty Level: Intermediate

Tools & Technologies:

Python (Pandas, NumPy, Matplotlib, Seaborn)

Jupyter Notebook / Visual Studio Code

Dataset Features:

- Date: Transaction date and time
- Mode: Payment mode (Cash, Bank Account, Credit Card, etc.)
- Category: Main classification of transaction (e.g., Food, Transportation, Investments)
- Subcategory: Detailed transaction type (e.g., Snacks, Train, Netflix)

Note: Short description of the transaction

- Amount: Transaction value (numeric)
- Income/Expense: Indicator whether it's an expense or income
- Currency: All transactions recorded in INR
- Dataset Size: 2,461 transactions, 8 columns

Let's Begin:-

Import Libraries and Load Data

```
In [ ]: import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
In [ ]: import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
In [ ]: import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [ ]: df = pd.read_csv("/content/Daily Household Transactions.csv")
```

Data Handling

```
In [ ]: df.head() #check the first 5 rows of the dataset
```

```
Out[ ]:
```

	Date	Mode	Category	Subcategory	Note	Amount	Incom
0	20/09/2018 12:04:08	Cash	Transportation	Train	2 Place 5 to Place 0	30.0	
1	20/09/2018 12:03:15	Cash	Food	snacks	Idli medu Vada mix 2 plates	60.0	
2	19/09/2018	Saving Bank account 1	subscription	Netflix	1 month subscription	199.0	
3	17/09/2018 23:41:17	Saving Bank account 1	subscription	Mobile Service Provider	Data booster pack	19.0	
4	16/09/2018 17:15:08	Cash	Festivals	Ganesh Pujan	Ganesh idol	251.0	

```
In [ ]: df.shape #get the number of rows and columns in the dataset
```

```
Out[ ]: (2461, 8)
```

```
In [ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 2461 entries, 0 to 2460  
Data columns (total 8 columns):  
#   Column                Non-Null Count  Dtype  
---  -  
0   Date                  2461 non-null  object  
1   Mode                  2461 non-null  object  
2   Category              2461 non-null  object  
3   Subcategory           1826 non-null  object  
4   Note                  1940 non-null  object  
5   Amount                2461 non-null  float64  
6   Income/Expense        2461 non-null  object  
7   Currency              2461 non-null  object  
dtypes: float64(1), object(7)  
memory usage: 153.9+ KB
```

```
In [ ]: df.isnull().sum() #get the null values
```

Out[]:

	0
Date	0
Mode	0
Category	0
Subcategory	635
Note	521
Amount	0
Income/Expense	0
Currency	0

dtype: int64

In []: `df["Mode"].value_counts()`

Out[]:

	count
Mode	
Saving Bank account 1	1223
Cash	1046
Credit Card	162
Equity Mutual Fund B	11
Share Market Trading	5
Saving Bank account 2	5
Recurring Deposit	3
Debit Card	2
Equity Mutual Fund C	1
Equity Mutual Fund A	1
Equity Mutual Fund D	1
Fixed Deposit	1

dtype: int64

Exploratory Data Analysis (EDA)

Weekly Spending Heatmap

In []: `# Ensure Date is datetime`
`df['Date'] = pd.to_datetime(df['Date'], errors='coerce', dayfirst=True)`

```

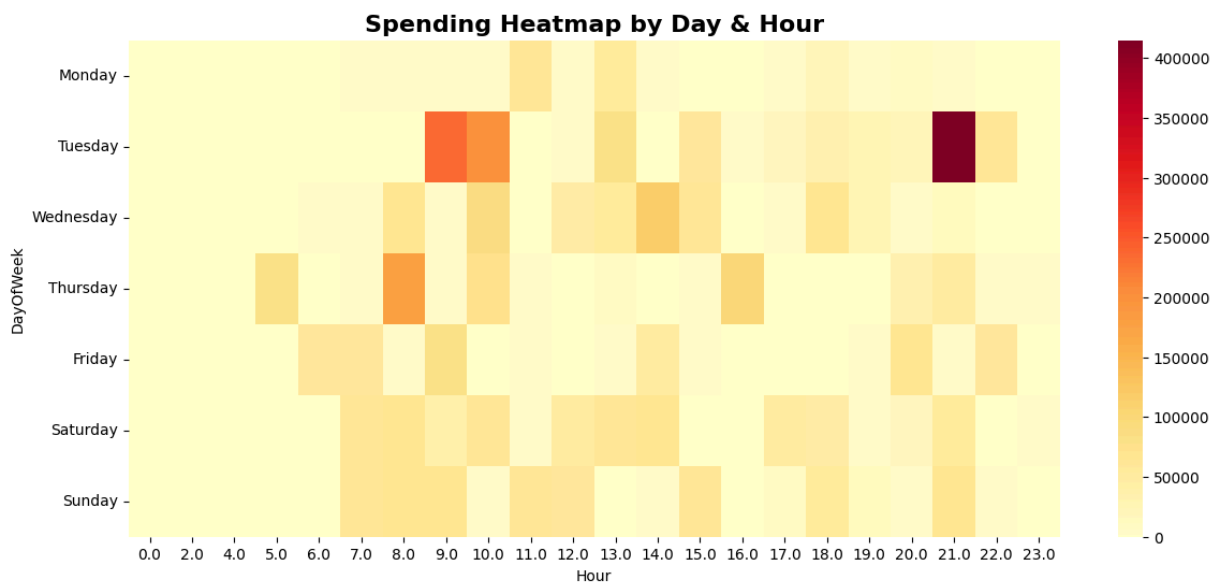
# Extract Day & Hour
df['DayOfWeek'] = df['Date'].dt.day_name()
df['Hour'] = df['Date'].dt.hour

# Group & Pivot
heatmap_data = df.pivot_table(
    index='DayOfWeek',
    columns='Hour',
    values='Amount',
    aggfunc='sum',
    fill_value=0
)

# Reorder days
days_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
heatmap_data = heatmap_data.reindex(days_order)

# Plot Heatmap
plt.figure(figsize=(14,6))
sns.heatmap(heatmap_data, cmap="YlOrRd", annot=False)
plt.title("Spending Heatmap by Day & Hour", fontsize=16, fontweight='bold')
plt.show()

```



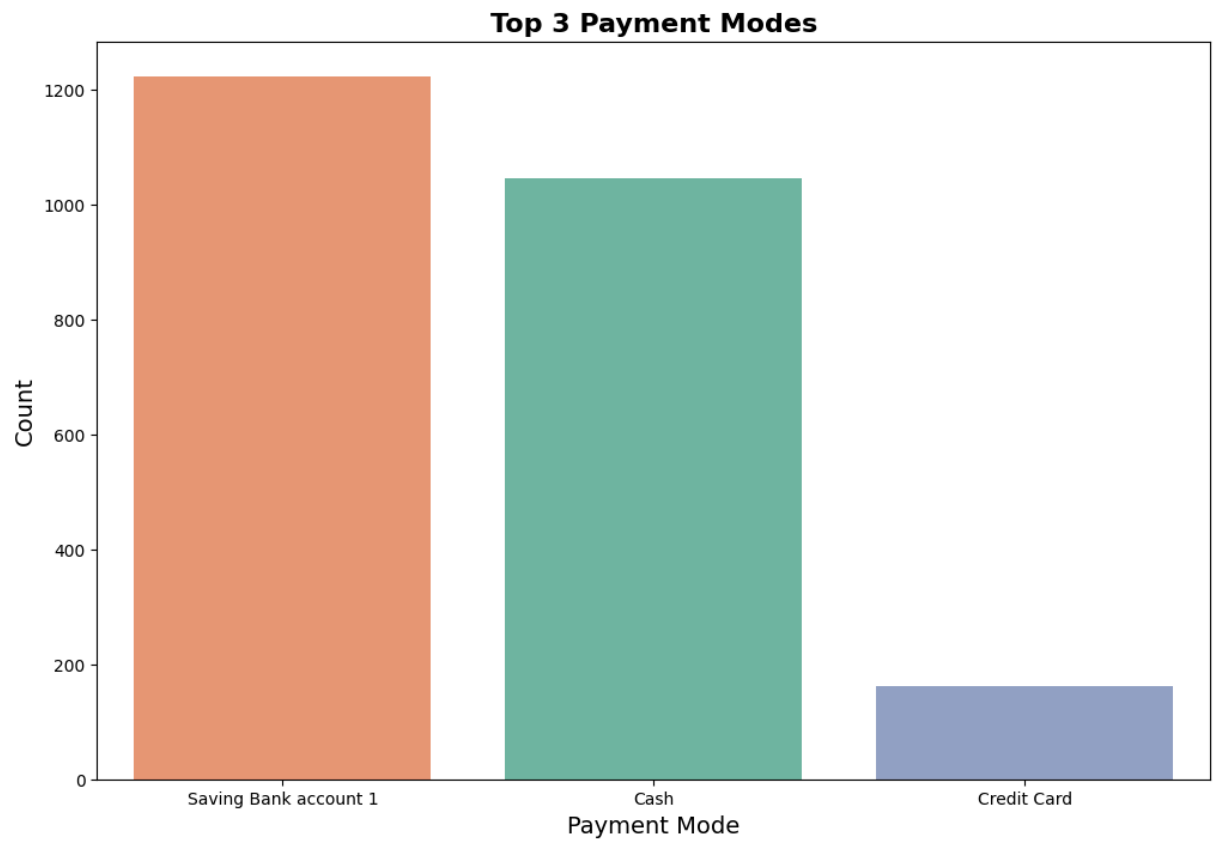
Most Frequently Used Payment Modes

```

In [ ]: plt.figure(figsize=(12,8))
sns.countplot(
    data=df,
    x="Mode",
    hue="Mode", # tells seaborn to color by Mode
    order=df["Mode"].value_counts().iloc[:3].index,
    palette="Set2",
    legend=False
)
plt.title("Top 3 Payment Modes", fontsize=16, fontweight='bold')
plt.xlabel("Payment Mode", fontsize=14)

```

```
plt.ylabel("Count", fontsize=14)  
plt.show()
```



```
In [ ]: df["Category"].value_counts()
```

Out[]:

Category	count
Food	907
Transportation	307
Household	176
subscription	143
Other	126
Investment	103
Health	94
Family	71
Apparel	47
Recurring Deposit	47
Money transfer	43
Salary	43
Gift	30
Public Provident Fund	29
Equity Mutual Fund E	22
Beauty	22
Gpay Reward	21
Education	18
Saving Bank account 1	17
maid	17
Festivals	16
Equity Mutual Fund A	14
Equity Mutual Fund F	13
Dividend earned on Shares	12
Interest	12
Culture	11
Small Cap fund 2	10
Small cap fund 1	10
Share Market	8
Life Insurance	7
Maturity amount	7
Petty cash	6

	count
Category	
Equity Mutual Fund C	6
Bonus	6
Tourism	5
Rent	4
Cook	4
Grooming	4
Saving Bank account 2	3
water (jar /tanker)	3
Self-development	2
Tax refund	2
garbage disposal	2
Documents	2
Amazon pay cashback	2
scrap	2
Fixed Deposit	2
Social Life	1
Equity Mutual Fund D	1
Equity Mutual Fund B	1

dtype: int64

Top 5 Most Frequent Transaction Categories

```
In [ ]: bright_colors = ["#FF0000", "#FF8C00", "#FFD700", "#00CED1", "#1E90FF"]

plt.figure(figsize=(12,8))
sns.countplot(
    data=df,
    x="Category",
    order=df["Category"].value_counts().iloc[:5].index,
    palette=bright_colors
)
plt.title("Top 5 Transaction Categories", fontsize=16, fontweight='bold')
plt.xlabel("Category", fontsize=14)
plt.ylabel("Count", fontsize=14)

# Add value labels
for p in plt.gca().patches:
    plt.gca().annotate(
        f'{int(p.get_height())}',
```

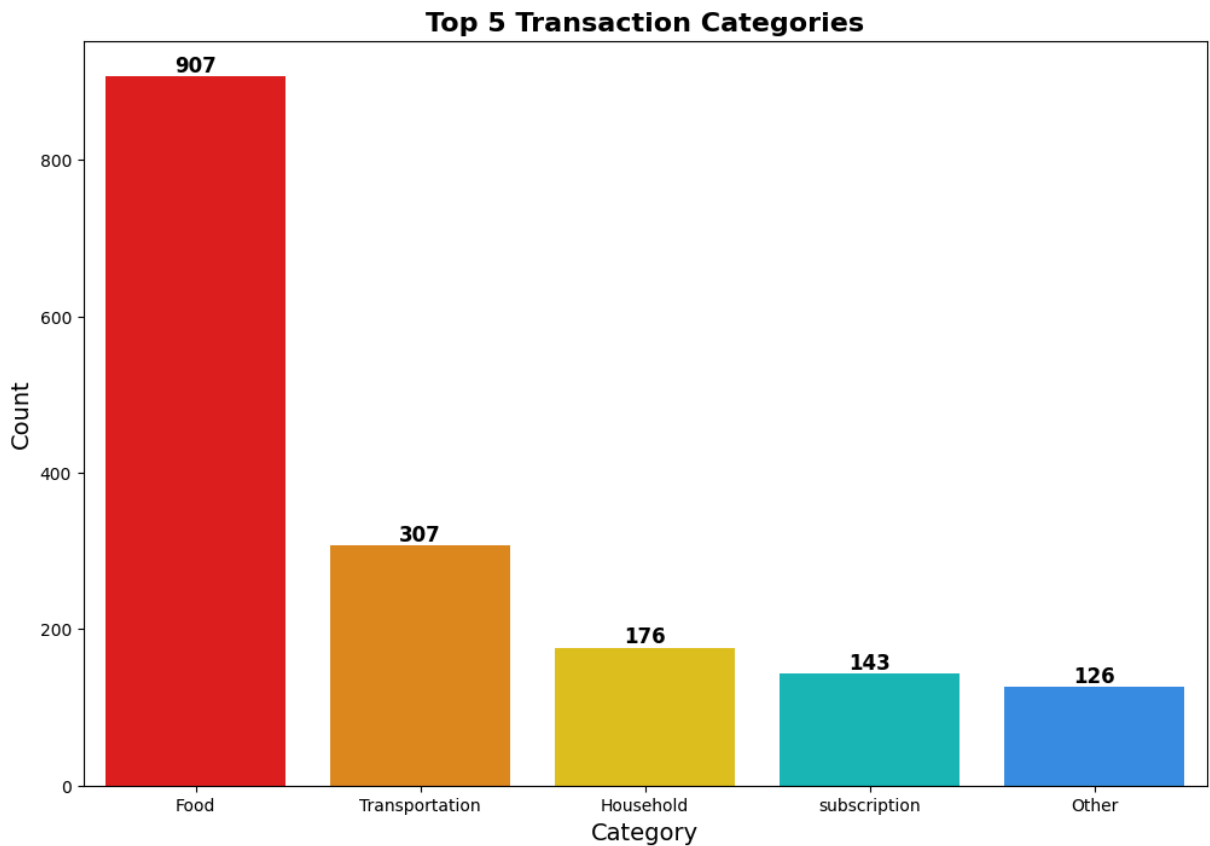


```

        (p.get_x() + p.get_width() / 2., p.get_height()),
        ha='center', va='bottom',
        fontsize=12, fontweight='bold'
    )

plt.show()

```



```
In [ ]: df["Subcategory"].unique()
```

```

Out[ ]: array(['Train', 'snacks', 'Netflix', 'Mobile Service Provider',
               'Ganesh Pujan', 'Tata Sky', 'auto', nan, 'Grocery', 'Lunch',
               'Milk', 'Pocket money', 'Laundry', 'breakfast', 'Dinner', 'Sweets',
               'Kirana', 'Ice cream', 'curd', 'Biscuits', 'Rajgira ladu',
               'Navratri', 'train', 'Tea', 'flour mill', 'Appliances',
               'home decor', 'grooming', 'Health', 'Clothing', 'clothes', 'Home',
               'chocolate', 'Medicine', 'Eating out', 'Movie', 'vegetables',
               'fruits', 'Potato', 'Onions', 'Taxi', 'Hardware', 'Eggs', 'Bread',
               'Petrol', 'Hospital', 'Mahanagar Gas', 'Lab Tests', 'Bus',
               'Travels', 'Kitchen', 'Footwear', 'Entry Fees', 'gadgets',
               'Accessories', 'misc', 'Stationary', 'Newspaper', 'Toiletries',
               'Bike', 'beverage', 'makeup', 'Books', 'Holi', 'Courier',
               'Leisure', 'Updation', 'Amazon Prime', 'Edtech Course', 'Hotstar',
               'Diwali', 'Wifi Internet Service', 'Trip', 'Furniture', 'Water',
               'Cable TV', 'medicine', 'Mutual fund', 'Public Provident Fund',
               'ropeway', 'RD', 'LIC', 'Saloon', 'gift', 'Rakshabandhan',
               'exam fee', 'Kindle unlimited', 'OTT Platform', 'School supplies',
               'Audible', 'Makeup'], dtype=object)

```

Top 10 Most Frequent Transaction Subcategories

```

In [ ]: # Decent, muted color palette
decent_colors = sns.color_palette("Set2", 10) # soft but distinct

plt.figure(figsize=(12,8))
sns.countplot(
    data=df,
    x="Subcategory",
    hue="Subcategory", # avoids FutureWarning
    order=df["Subcategory"].value_counts().iloc[:10].index,
    palette=decent_colors,
    legend=False # hides duplicate legend
)
plt.xticks(rotation=90)

plt.title("Top 10 Transaction Subcategories", fontsize=16, fontweight='bold')
plt.xlabel("Subcategory", fontsize=14)
plt.ylabel("Count", fontsize=14)

# Add value labels
for p in plt.gca().patches:
    plt.gca().annotate(
        f'{int(p.get_height())}',
        (p.get_x() + p.get_width() / 2., p.get_height()),
        ha='center', va='bottom',
        fontsize=11, fontweight='bold'
    )

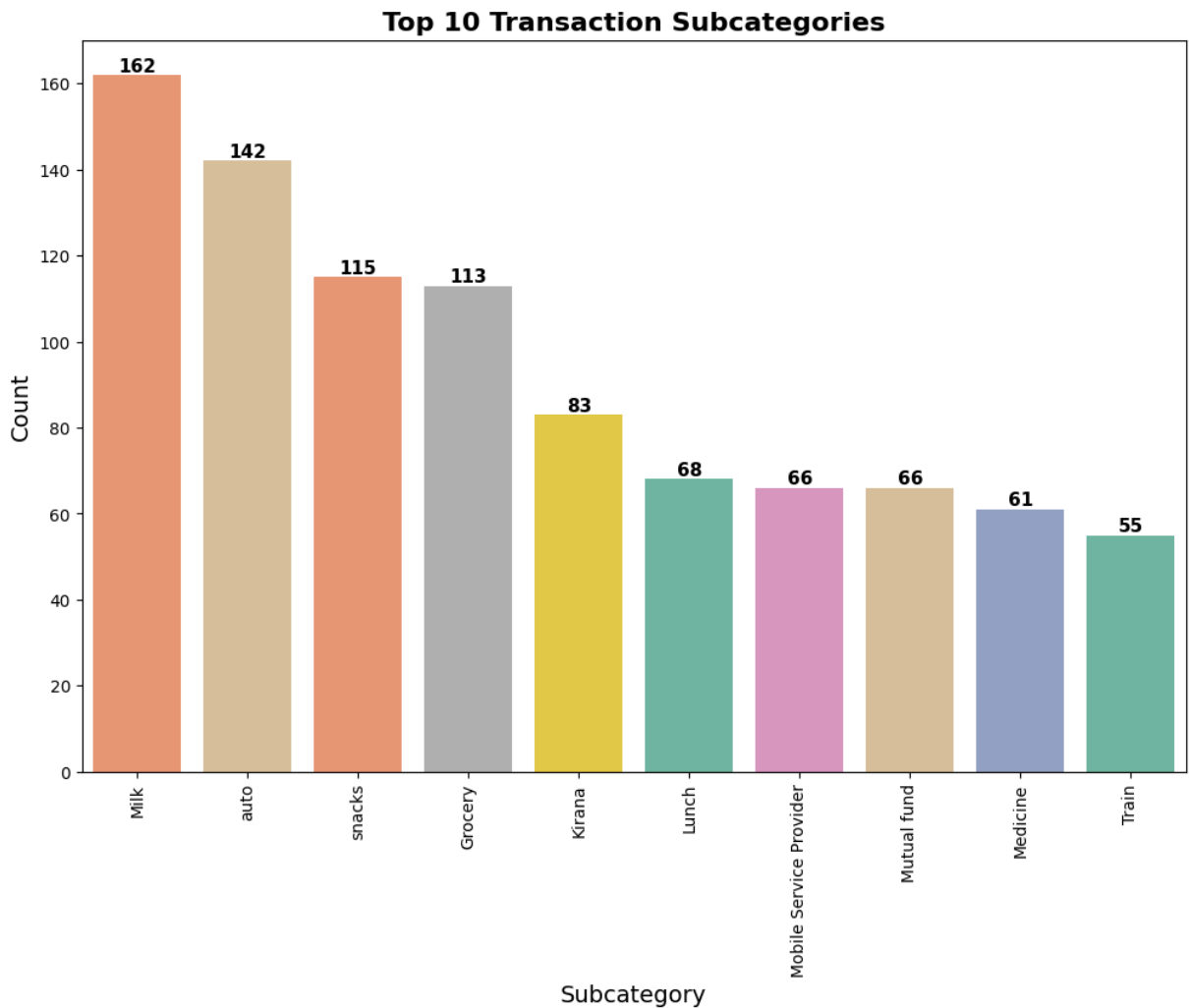
plt.show()

```

```

/tmp/ipython-input-3823772236.py:5: UserWarning:
The palette list has fewer values (10) than needed (90) and will cycle, which
may produce an uninterpretable plot.
sns.countplot(

```



Comparison of Income and Expense Frequency

```
In [ ]: # Color palette for all unique Income/Expense values
unique_vals = df["Income/Expense"].nunique()
colors = sns.color_palette("Set1", unique_vals) # bright but decent

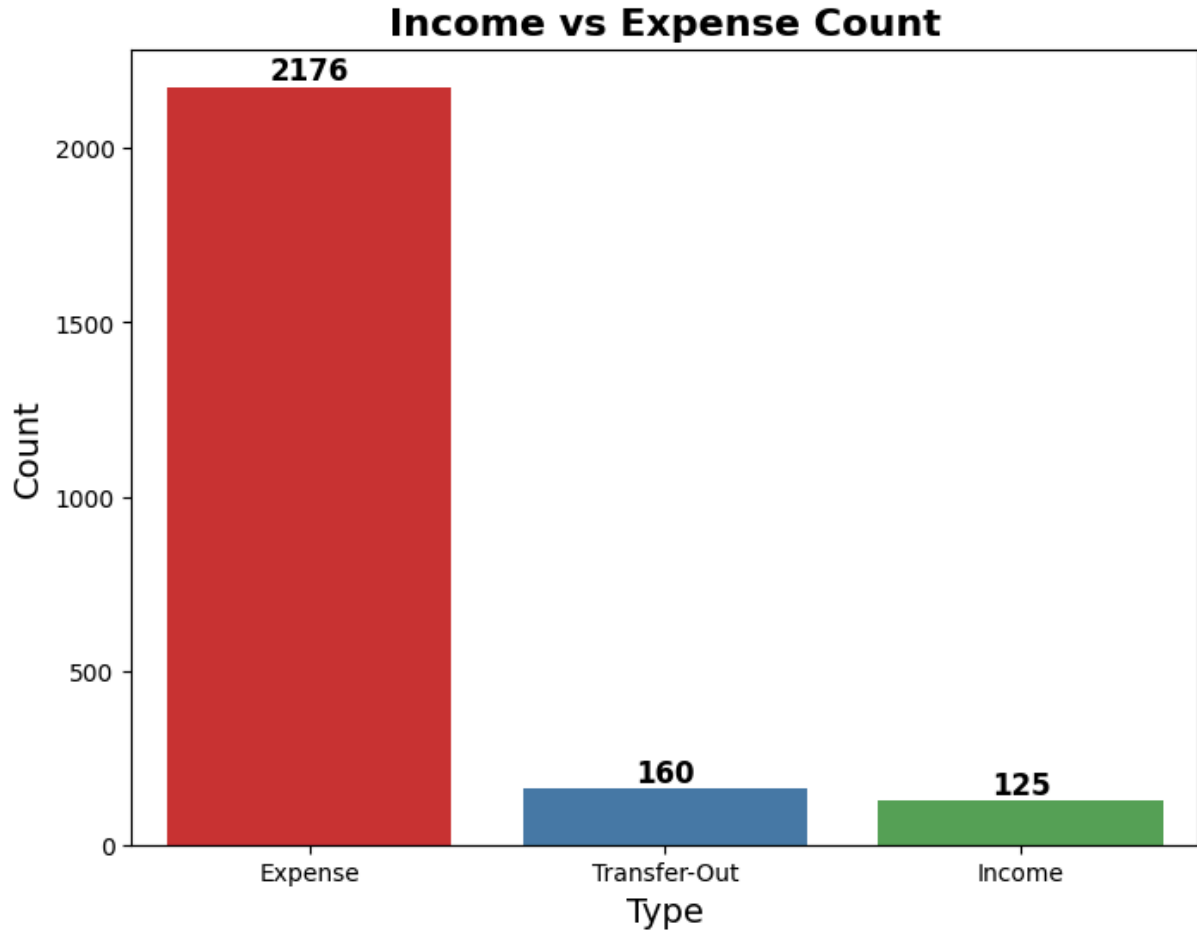
plt.figure(figsize=(8,6))
sns.countplot(
    data=df,
    x="Income/Expense",
    hue="Income/Expense", # avoids FutureWarning
    palette=colors,
    legend=False         # remove extra legend
)
plt.title("Income vs Expense Count", fontsize=16, fontweight='bold')
plt.xlabel("Type", fontsize=14)
plt.ylabel("Count", fontsize=14)

# Add value labels
for p in plt.gca().patches:
    plt.gca().annotate(
        f'{int(p.get_height())}',
        (p.get_x() + p.get_width() / 2., p.get_height()),
        ha='center', va='bottom',
```

```

        fontsize=12, fontweight='bold'
    )
plt.show()

```



```
In [ ]: df["Note"].nunique()
```

Out[]: 1057

```
In [ ]: df["Currency"].value_counts()
```

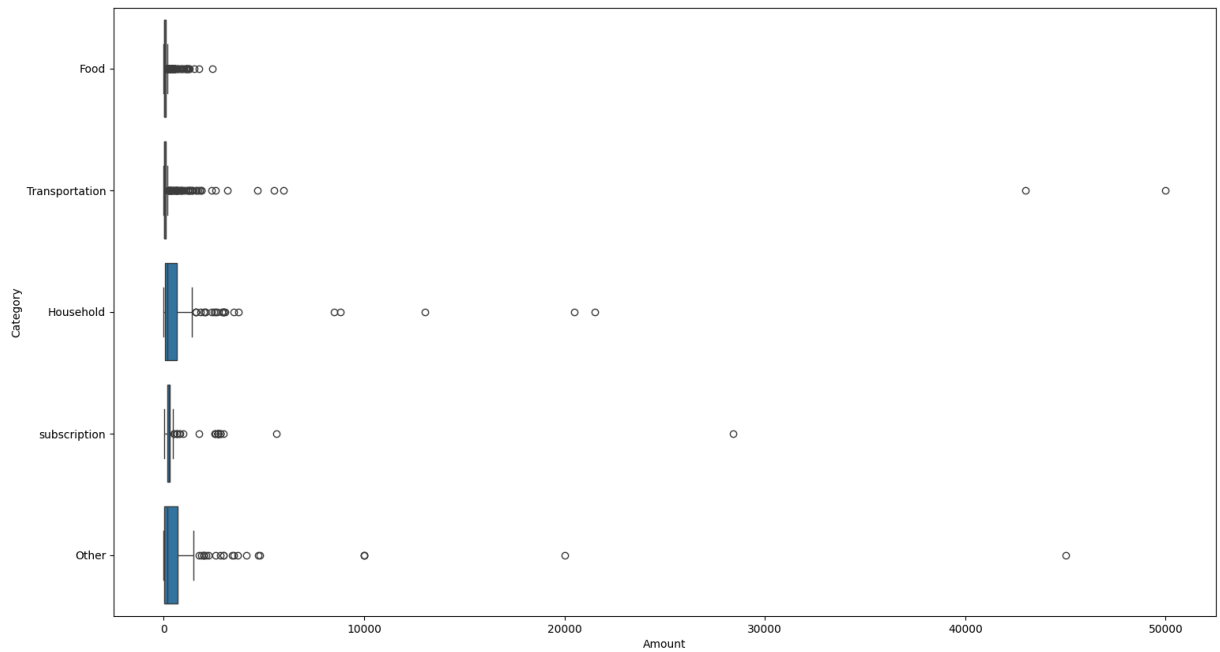
```
Out[ ]:
      count
Currency
INR      2461
```

dtype: int64

Transaction Amount Distribution Across Top 5 Categories

```
In [ ]: plt.figure(figsize = (18,10))
sns.boxplot(data = df, x = "Amount", y = "Category", order =
df["Category"].value_counts().iloc[:5].index)
plt.show()

```



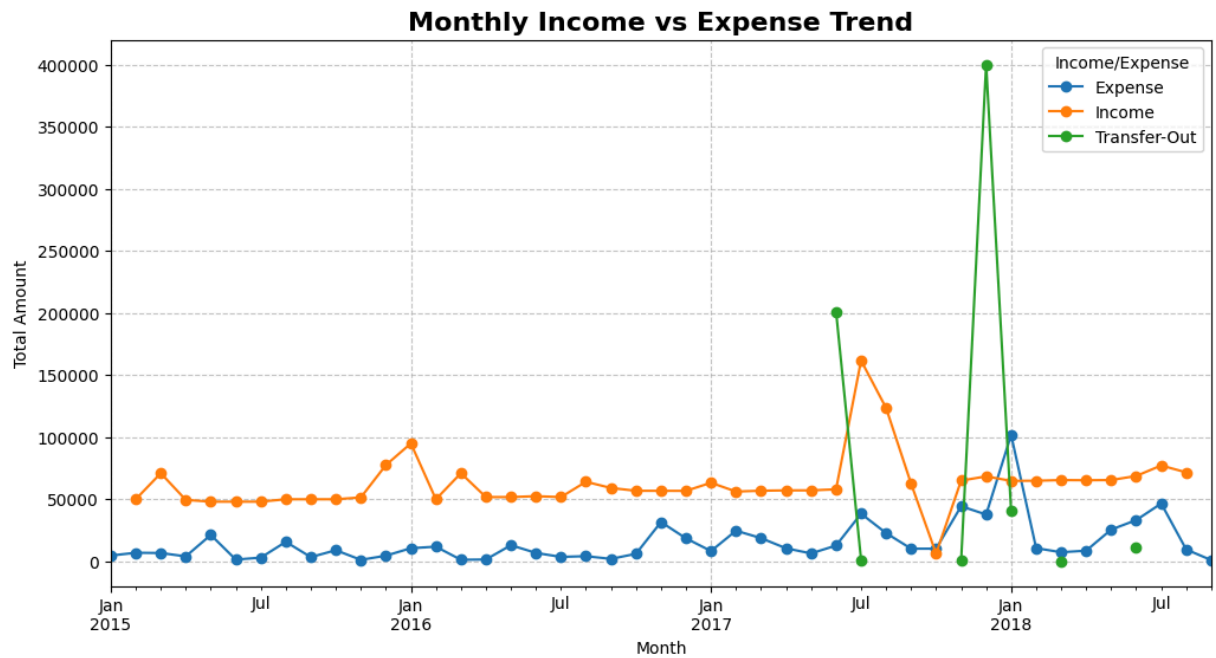
Monthly Trend of Income vs Expense

```
In [ ]: # Convert Date to datetime
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Extract Year-Month
df['YearMonth'] = df['Date'].dt.to_period('M')

# Group by Month & Income/Expense
monthly_trend = df.groupby(['YearMonth', 'Income/Expense'])['Amount'].sum()

# Plot
monthly_trend.plot(kind='line', marker='o', figsize=(12,6))
plt.title("Monthly Income vs Expense Trend", fontsize=16, fontweight='bold')
plt.xlabel("Month")
plt.ylabel("Total Amount")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```



Transaction Amount Distribution Across Top 10 Spending Subcategories

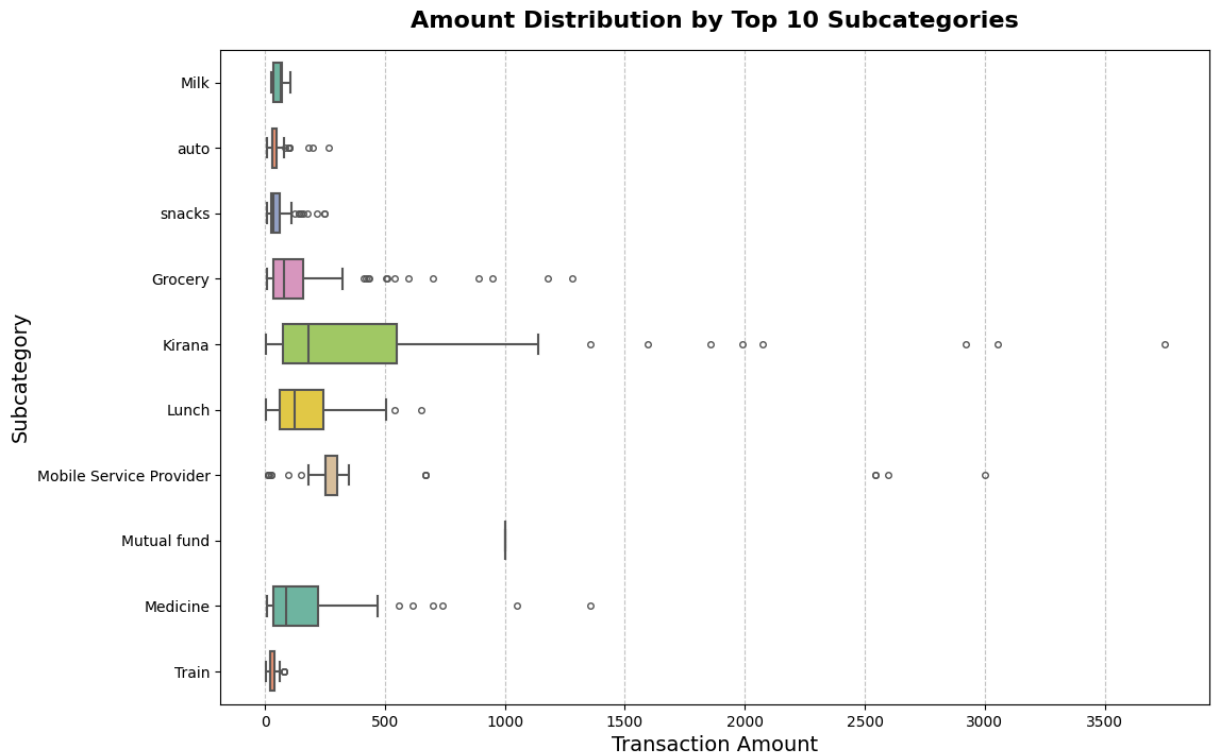
```
In [ ]: import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

plt.figure(figsize=(12,8))

palette_colors = sns.color_palette("Set2", 10)

sns.boxplot(
    data=df,
    x="Amount",
    y="Subcategory",
    order=df["Subcategory"].value_counts().iloc[:10].index,
    palette=palette_colors,
    width=0.6,
    fliersize=4,
    linewidth=1.5
)

plt.title("Amount Distribution by Top 10 Subcategories", fontsize=16, fontwe
plt.xlabel("Transaction Amount", fontsize=14)
plt.ylabel("Subcategory", fontsize=14)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



Income vs Expense: Transaction Amount Distribution

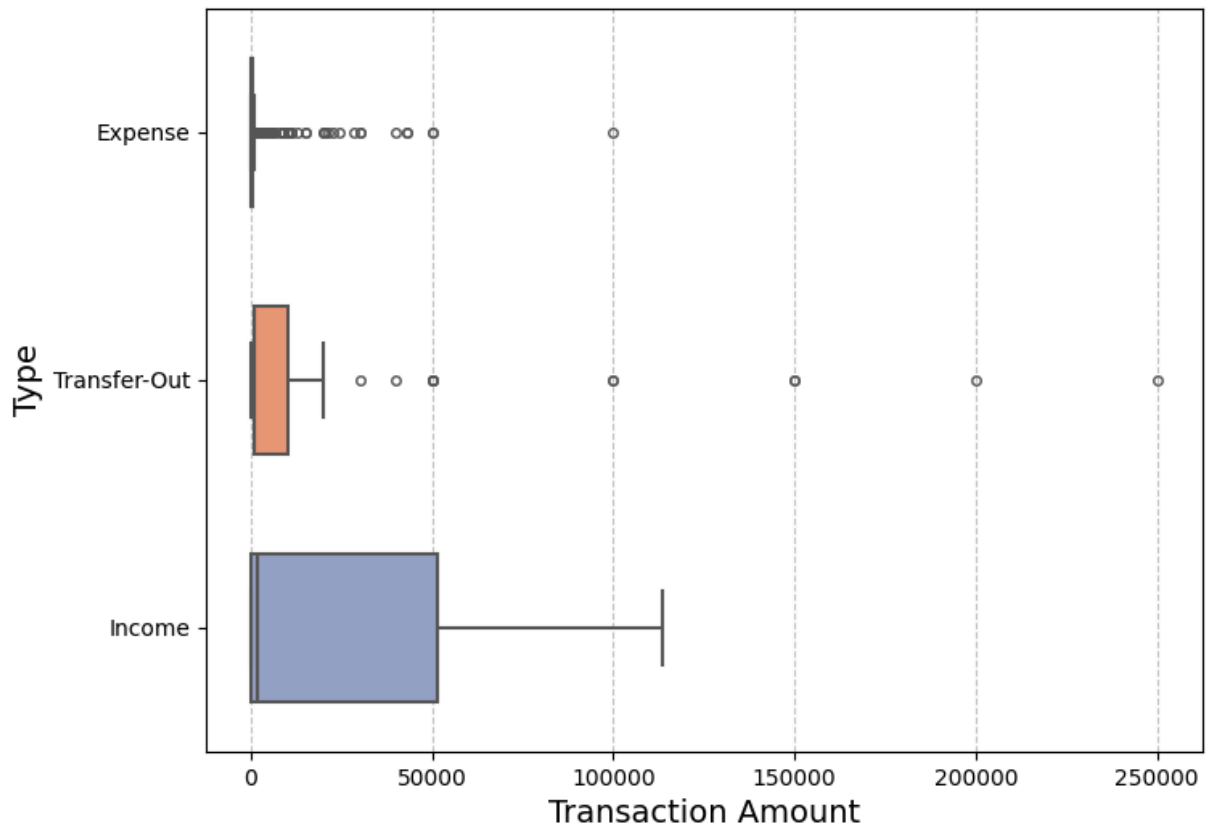
```
In [ ]: # Match colors to unique values
unique_vals = df["Income/Expense"].nunique()
palette_colors = sns.color_palette("Set2", unique_vals) # auto-generate end

plt.figure(figsize=(8,6))
sns.boxplot(
    data=df,
    x="Amount",
    y="Income/Expense",
    hue="Income/Expense", # avoids FutureWarning
    palette=palette_colors,
    width=0.6,
    fliersize=4,
    linewidth=1.5,
    legend=False # remove duplicate legend
)

plt.title("Amount Distribution: Income vs Expense", fontsize=16, fontweight=
plt.xlabel("Transaction Amount", fontsize=14)
plt.ylabel("Type", fontsize=14)
plt.grid(axis='x', linestyle='--', alpha=0.7)

plt.show()
```

Amount Distribution: Income vs Expense

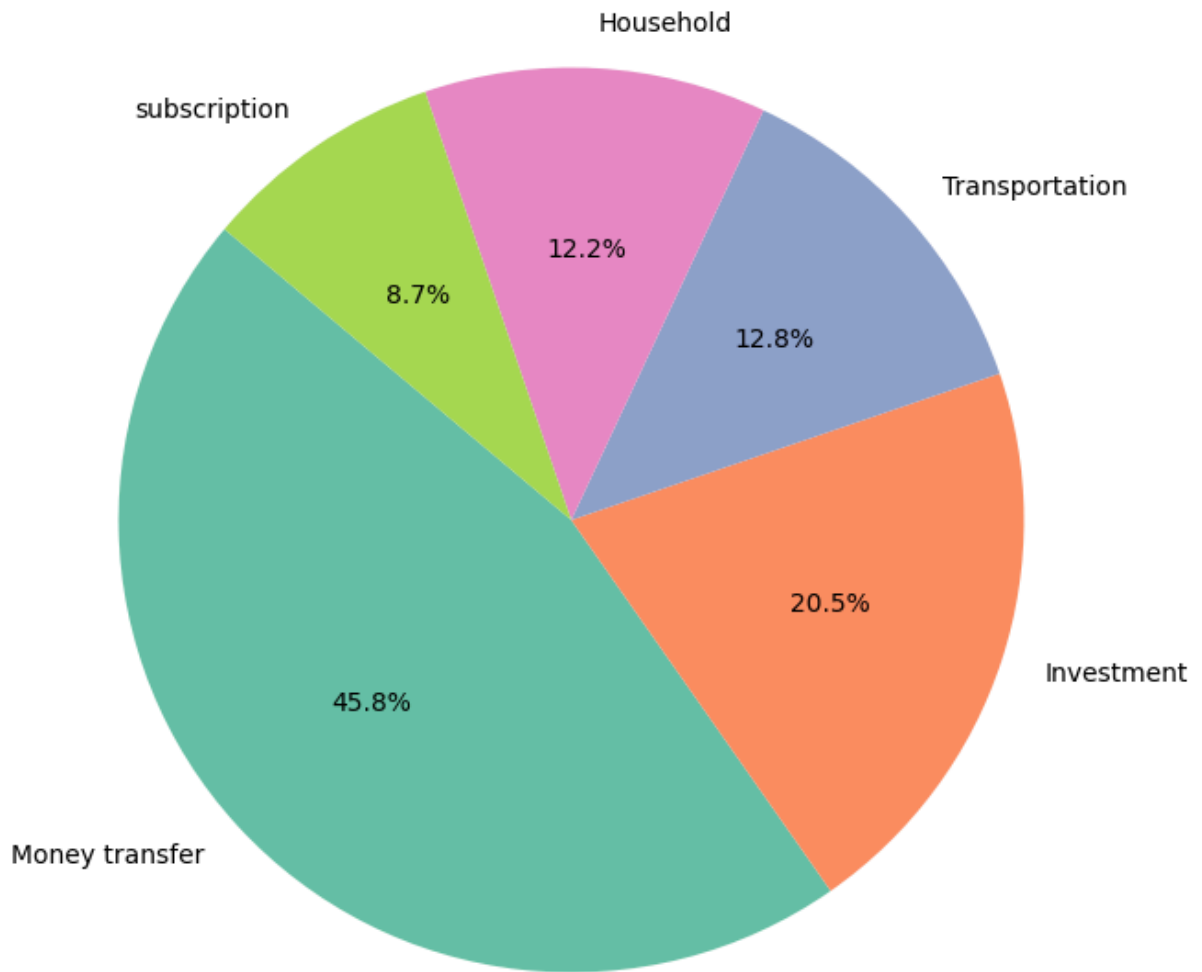


Share of Top 5 Spending Categories

```
In [ ]: expense_df = df[df['Income/Expense'] == 'Expense']
category_expense = expense_df.groupby('Category')['Amount'].sum().nlargest(5)

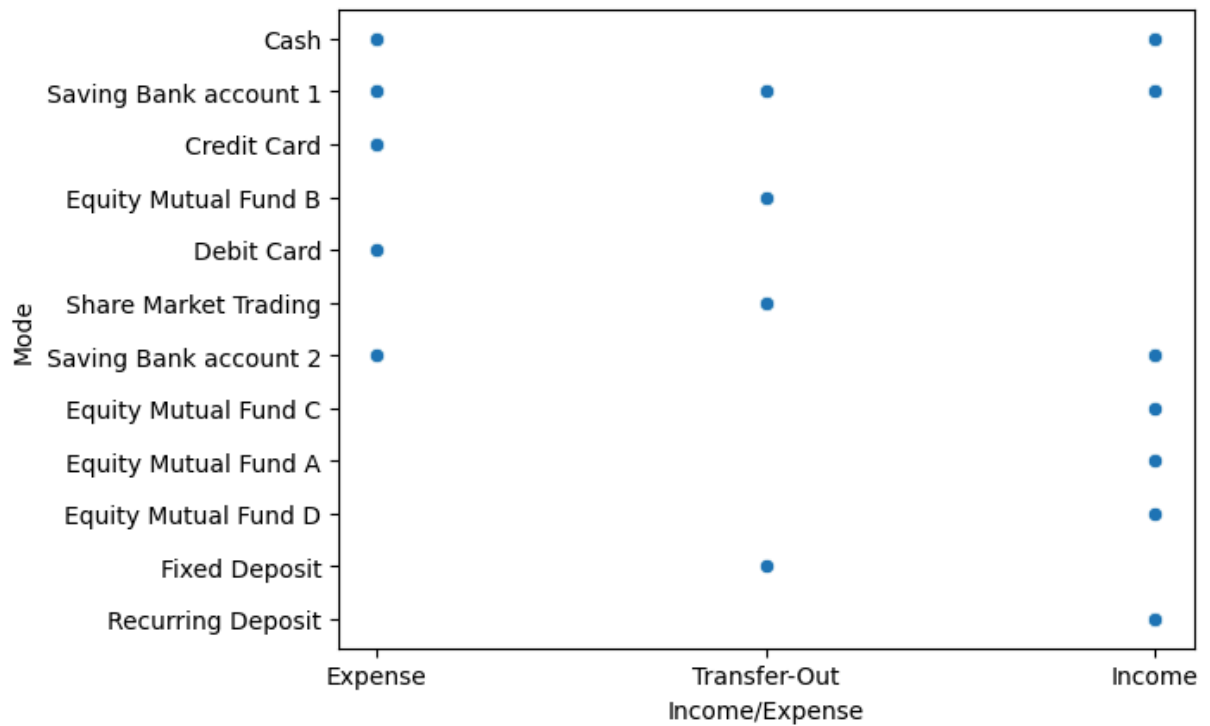
plt.figure(figsize=(8,8))
category_expense.plot.pie(
    autopct='%1.1f%%',
    startangle=140,
    colors=sns.color_palette("Set2", 5)
)
plt.title("Top 5 Expense Categories Share", fontsize=16, fontweight='bold')
plt.ylabel("")
plt.show()
```


Top 5 Expense Categories Share



Transaction Modes by Income and Expense

```
In [ ]: sns.scatterplot(data=df, x="Income/Expense", y="Mode", );
```



Growth of Cumulative Savings Over Time

```
In [ ]: # Income as positive, Expense as negative
df['NetAmount'] = df.apply(lambda row: row['Amount'] if row['Income/Expense']

# Cumulative sum
df = df.sort_values('Date')
df['CumulativeSavings'] = df['NetAmount'].cumsum()

plt.figure(figsize=(12,6))
plt.plot(df['Date'], df['CumulativeSavings'], color="green", linewidth=2)
plt.title("Cumulative Savings Over Time", fontsize=16, fontweight='bold')
plt.xlabel("Date")
plt.ylabel("Total Savings")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

