Project Name - Daily Transactions (ML _ FA _ DA projects)(Part 1)

Project Type - Data Analysis

Industry - Unified Mentor

Contribution - Individual

Member Name - Hare Krishana Mishra

Task - 1

Project Summary -

Project Description:

The Daily Transactions Analysis project aims to explore and analyze an individual's daily financial records to uncover spending patterns, income trends, and key financial habits. The dataset contains details of various transactions, including the date, payment mode, category, subcategory, amount, and whether the transaction was income or expense. In the first phase of the project, the focus is on data cleaning and preparation—ensuring the dataset is free from missing or invalid entries, correcting data types, and creating a reliable foundation for subsequent exploratory data analysis (EDA) and visualization.

Objective:

- Ensure data quality by handling missing values, removing duplicates, and fixing data types.
- Prepare the dataset for accurate analysis and visualization.
- Establish a clean, consistent, and well-structured financial dataset that reflects real-world daily transactions.
- Set the stage for identifying financial trends, patterns, and anomalies in later stages.

Key Project Details:

Dataset Source: Daily Household Transactions dataset (personal financial records).

Data Cleaning Steps Implemented:

- Filled missing Category values with "Unknown".
- Filled missing Subcategory and Note fields with sensible defaults.
- Dropped rows with missing Date or Amount values.
- Converted Date to datetime format (dayfirst=True) to handle DD/MM/YYYY style entries.
- Converted Amount to numeric format for accurate calculations.
- Removed duplicate records to avoid skewed analysis.
- Reset index for a clean, sequential DataFrame structure.

Tools & Libraries: Python, Pandas, NumPy, Matplotlib, Seaborn.

Outcome of Part 1: A clean, consistent dataset ready for visualizations and in-depth financial analysis.

Let's Begin:-

Import Libraries and Load Data

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
# Load the dataset
df = pd.read_csv('/content/Daily Household Transactions.csv')
```

Data Cleaning

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

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 [ ]:

```python
# Check for missing values
df.isnull().sum()
```

Out[ ]:

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

dtype: int64

In [ ]:

```python
# Fill or drop missing values
df['Category'] = df['Category'].fillna('Unknown')
df.dropna(subset=['Date', 'Amount'], inplace=True)
```

In [ ]:

```python
# Convert Date to datetime (handles mixed formats, day first)
df['Date'] = pd.to_datetime(df['Date'], errors='coerce', dayfirst=True)

# Convert Amount to float safely
df['Amount'] = pd.to_numeric(df['Amount'], errors='coerce')

# Drop rows where date or amount could not be parsed
df.dropna(subset=['Date', 'Amount'], inplace=True)
```

In [ ]:
```
df.drop_duplicates(inplace=True)
```

In [ ]:
```
# Verify data types
df.dtypes
```

Out[ ]:

	0
Date	datetime64[ns]
Mode	object
Category	object
Subcategory	object
Note	object
Amount	float64
Income/Expense	object
Currency	object

dtype: object

In [ ]:
```
# Summary statistics
df.describe()
```

Out[ ]:

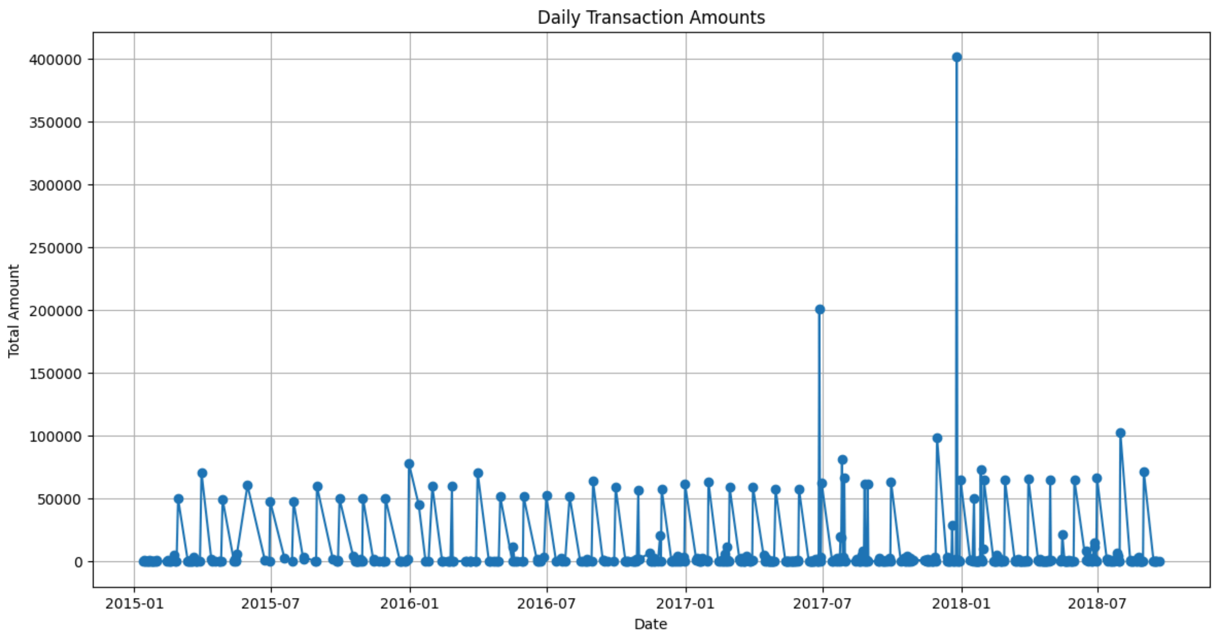
	Date	Amount
count	1303	1303.000000
mean	2017-05-12 20:41:38.546431232	3076.396892
min	2015-01-13 18:52:47	2.000000
25%	2016-12-18 20:18:45.500000	30.000000
50%	2017-07-27 20:05:23	72.000000
75%	2018-01-30 12:09:30.500000	298.500000
max	2018-09-20 12:04:08	250000.000000
std	NaN	14608.948853

Exploratory Data Analysis (EDA)

Time Series Analysis

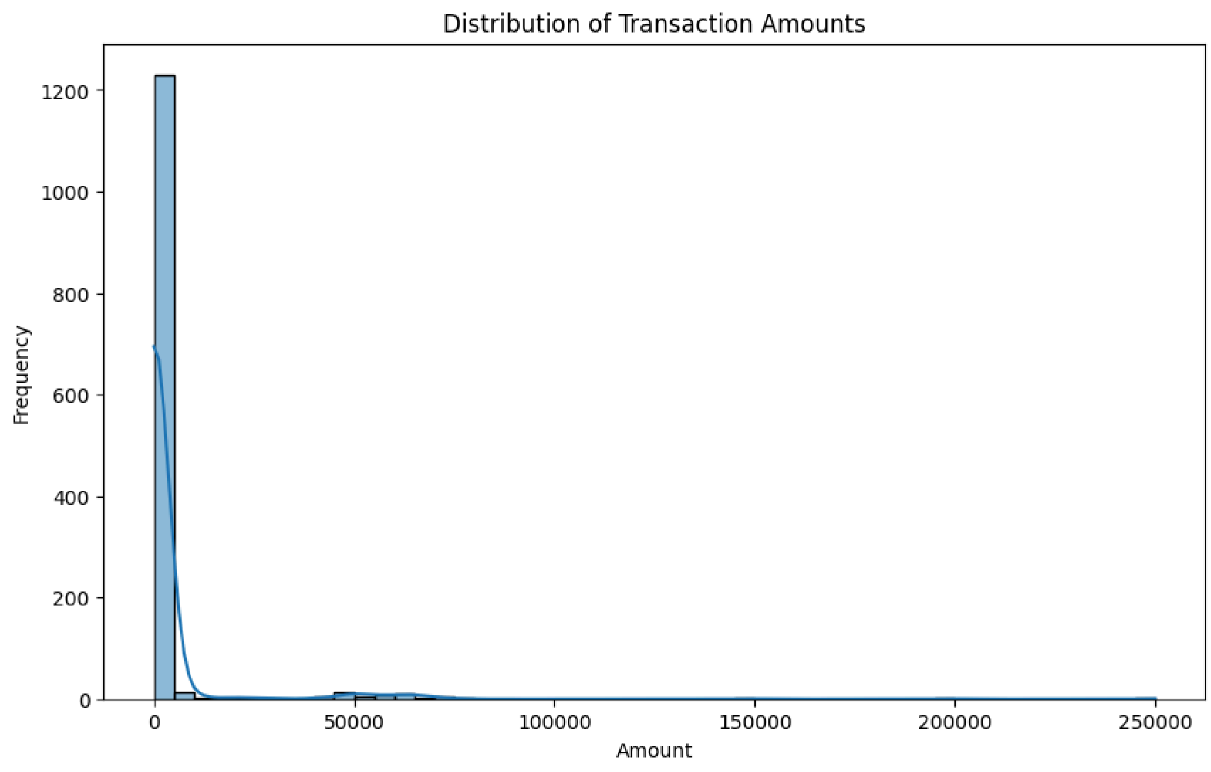
Daily Spending & Income Trend

In [ ]:
```
# Daily trends - sum only numeric columns
daily_data = df.groupby(df['Date'].dt.date)['Amount'].sum()

plt.figure(figsize=(14, 7))
plt.plot(daily_data.index, daily_data.values, marker='o')
plt.title('Daily Transaction Amounts')
plt.xlabel('Date')
plt.ylabel('Total Amount')
plt.grid(True)
plt.show()
```



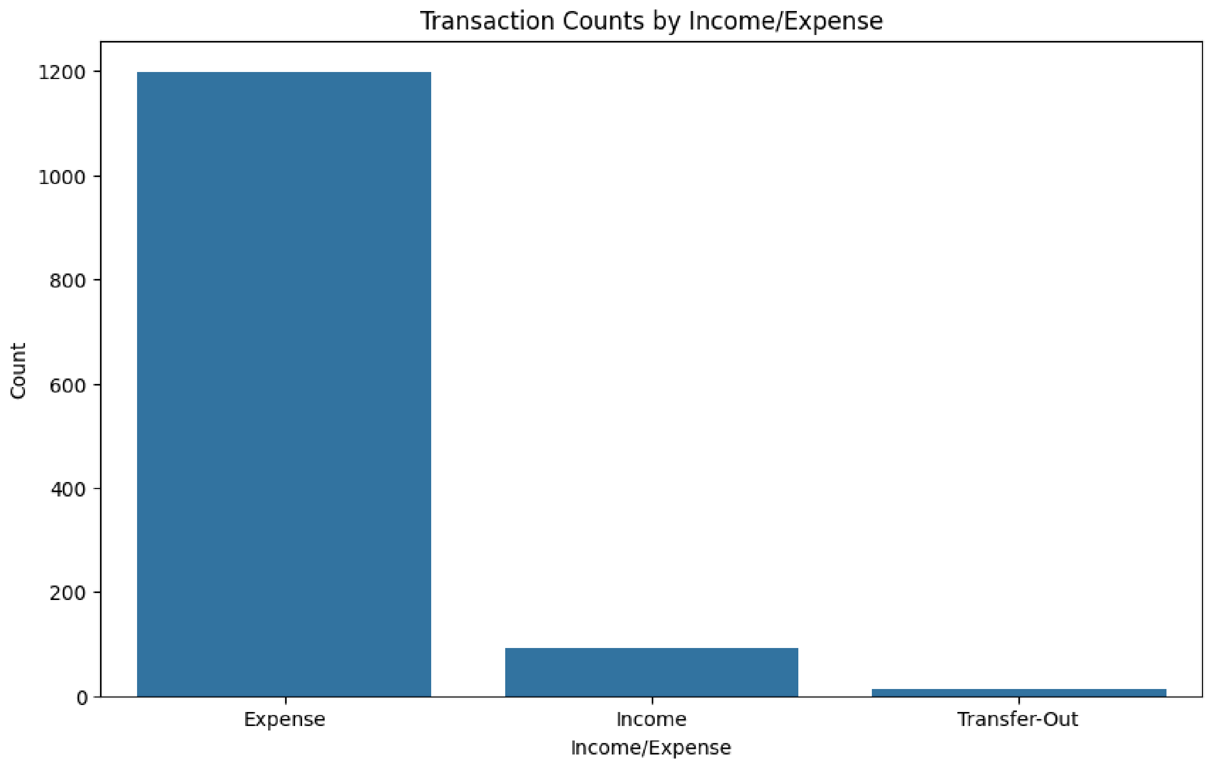
Distribution of Daily Transaction Amounts

In [ ]:
```
# Distribution of transaction amounts
plt.figure(figsize=(10, 6))
sns.histplot(df['Amount'], bins=50, kde=True)
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.show()
```



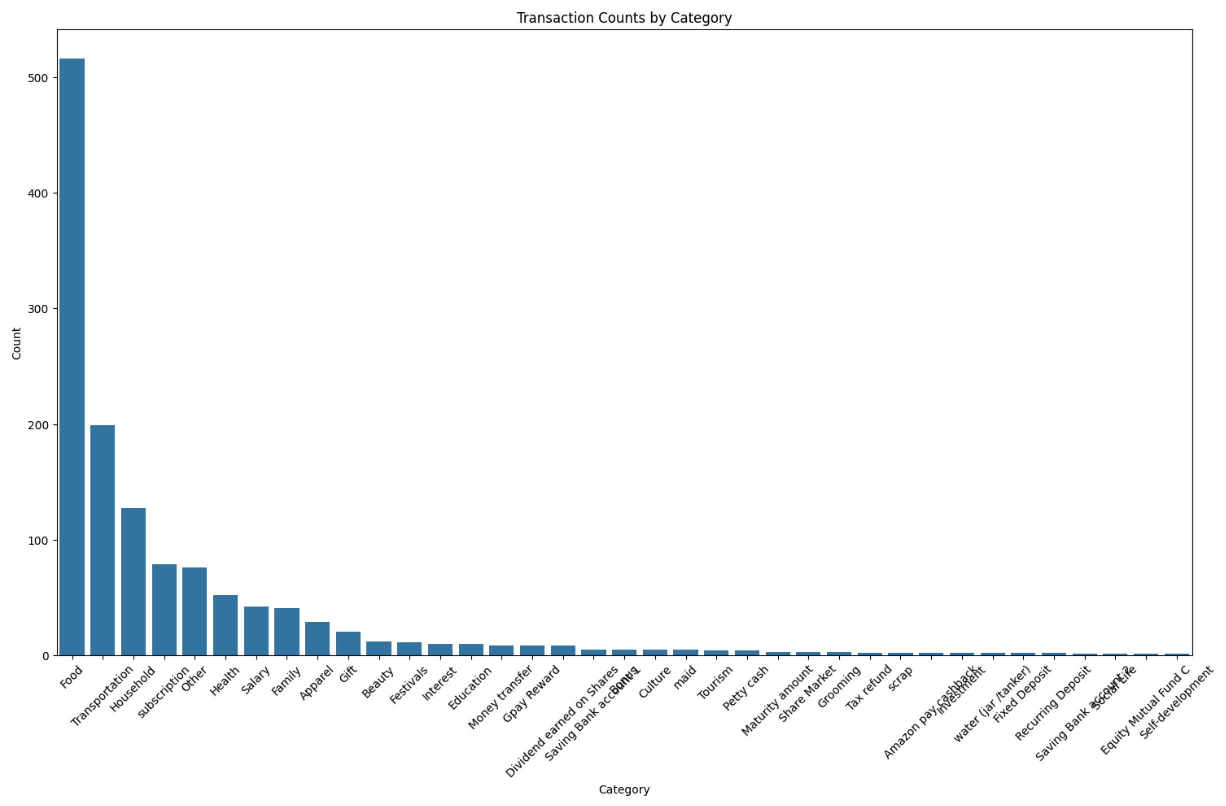
Income vs Expense Transaction Counts

```
In [ ]: # Transaction counts by Income/Expense type
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Income/Expense')
plt.title('Transaction Counts by Income/Expense')
plt.xlabel('Income/Expense')
plt.ylabel('Count')
plt.show()
```



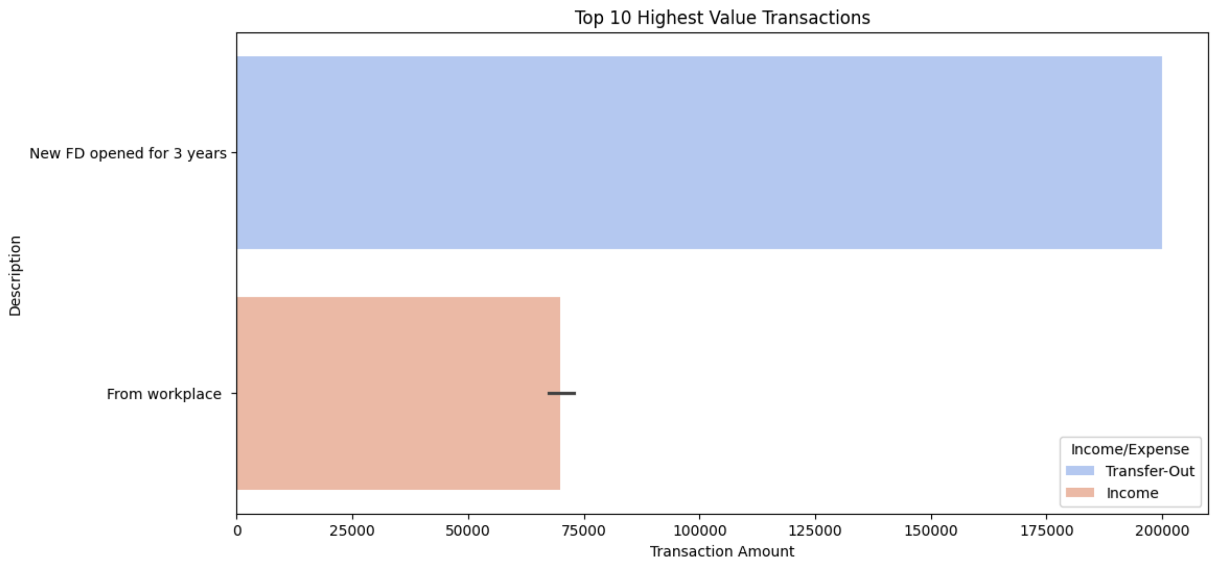
Number of Transactions per Category

```
In [ ]: # Transaction counts by category
plt.figure(figsize=(18, 10))
sns.countplot(data=df, x='Category', order=df['Category'].value_counts().inc
plt.title('Transaction Counts by Category')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



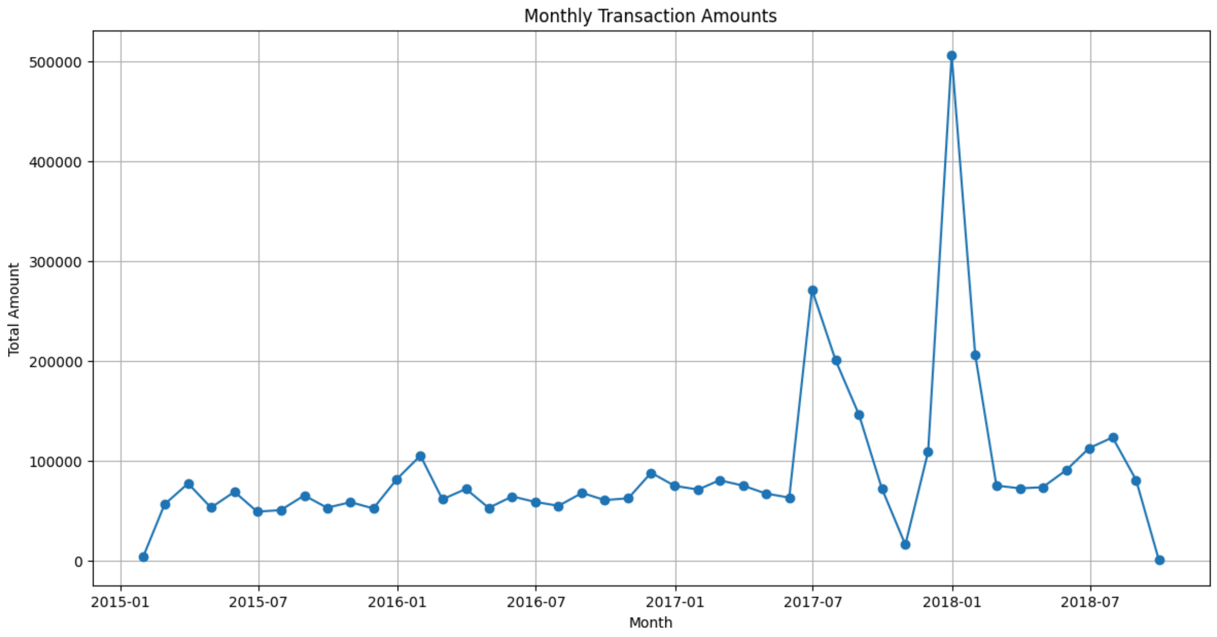
Top 10 Most Expensive Transactions

```
In [ ]: top_transactions = df.nlargest(10, 'Amount')

        plt.figure(figsize=(12, 6))
        sns.barplot(data=top_transactions, x='Amount', y='Note', hue='Income/Expense
        plt.title('Top 10 Highest Value Transactions')
        plt.xlabel('Transaction Amount')
        plt.ylabel('Description')
        plt.show()
```



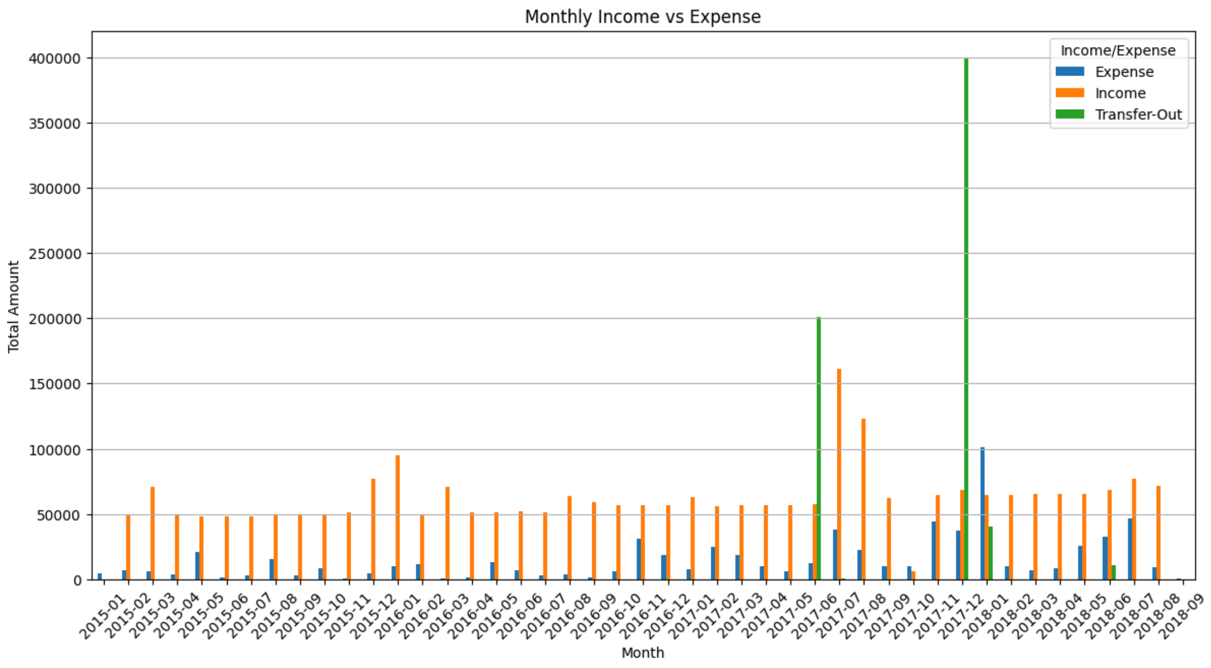
Total Amount of Transactions per Month

In [ ]:
```python
# Resample data to month-end frequency
monthly_data = df.resample('ME', on='Date').sum()

plt.figure(figsize=(14, 7))
plt.plot(monthly_data.index, monthly_data['Amount'], marker='o')
plt.title('Monthly Transaction Amounts')
plt.xlabel('Month')
plt.ylabel('Total Amount')
plt.grid(True)
plt.show()
```



Monthly Income vs Expense Comparison

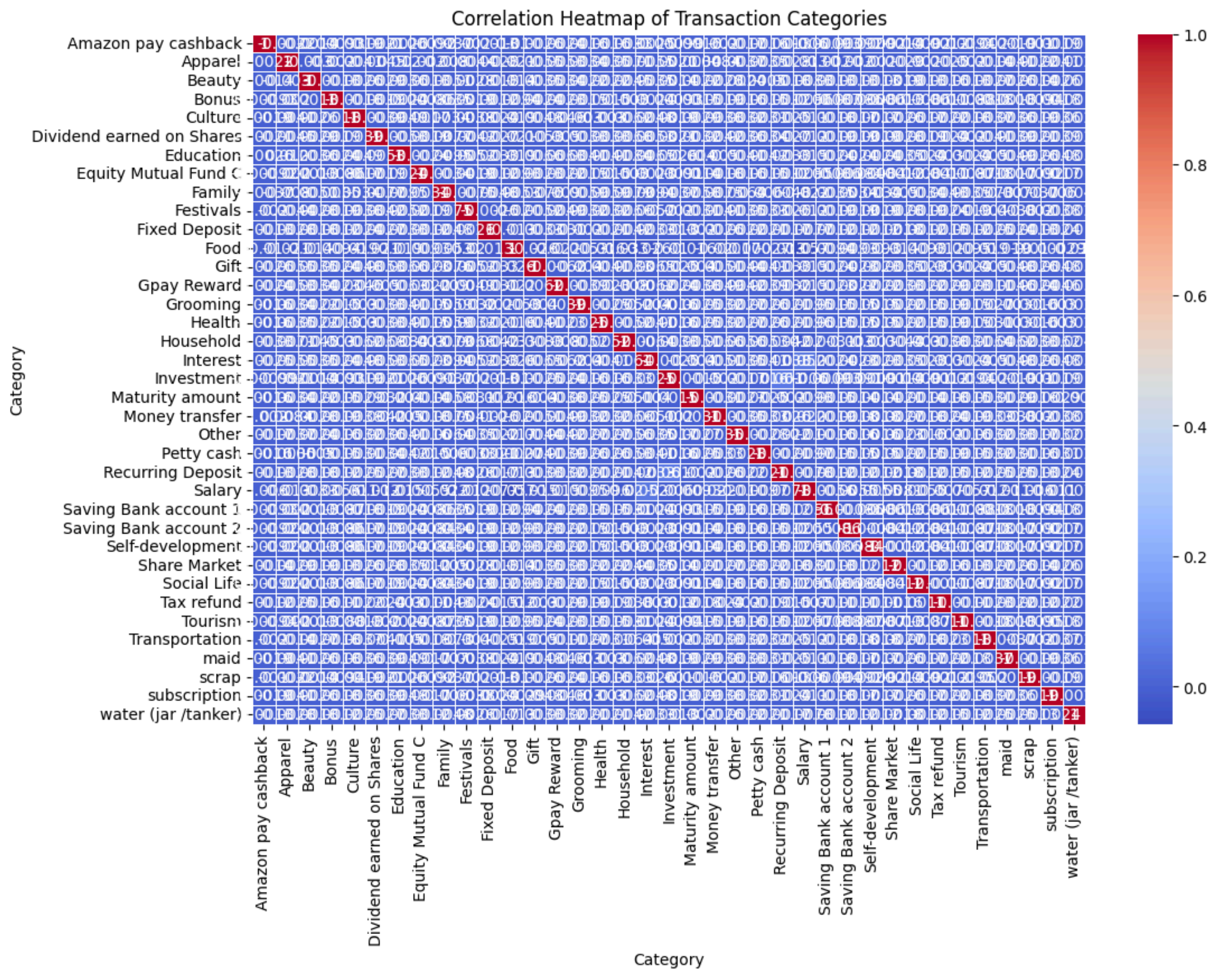
In [ ]:
```python
# Create Month-Year column
df['Month'] = df['Date'].dt.to_period('M')

monthly_income_expense = df.groupby(['Month', 'Income/Expense'])['Amount'].s

monthly_income_expense.plot(kind='bar', figsize=(14, 7), stacked=False)
plt.title('Monthly Income vs Expense')
plt.xlabel('Month')
plt.ylabel('Total Amount')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



Correlation Analysis

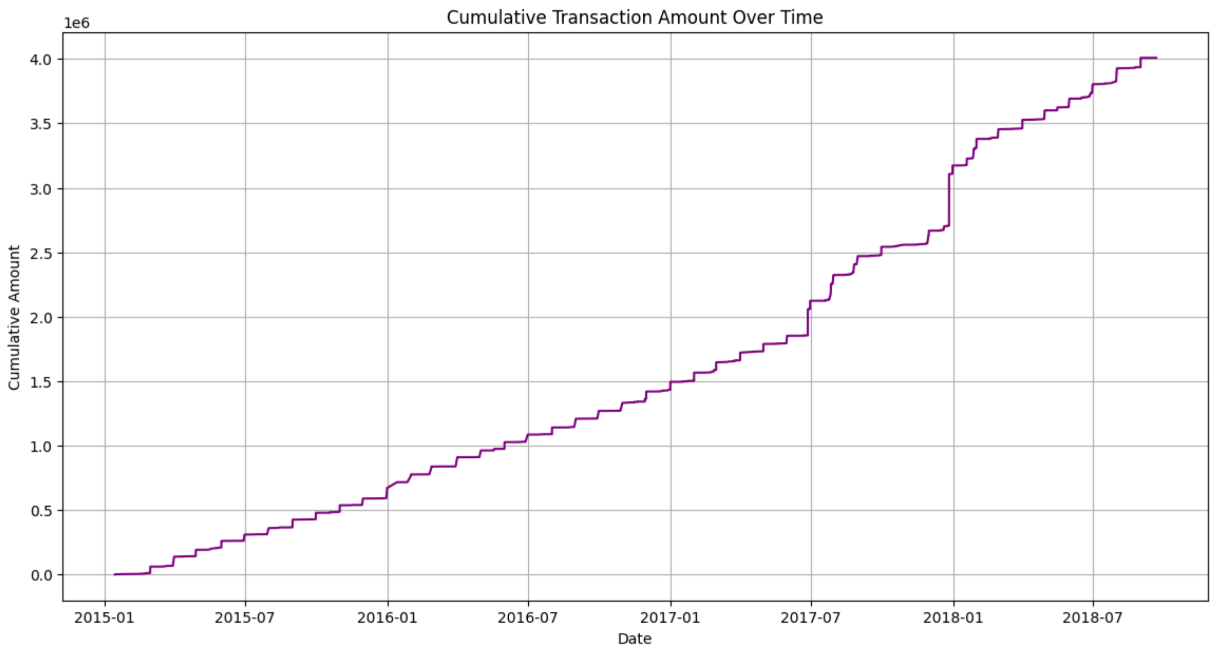
In [ ]:
```
# Create a pivot table for correlation analysis
pivot_table = df.pivot_table(index='Date', columns='Category', values='Amour
aggfunc='sum', fill_value=0)

# Calculate correlation matrix
correlation_matrix = pivot_table.corr()
```

In [ ]:
```
# Plot correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Transaction Categories')
plt.show()
```



Cumulative Spending Over Time

```
In [ ]: df_sorted = df.sort_values('Date')
        df_sorted['Cumulative_Amount'] = df_sorted['Amount'].cumsum()

        plt.figure(figsize=(14, 7))
        plt.plot(df_sorted['Date'], df_sorted['Cumulative_Amount'], color='purple')
        plt.title('Cumulative Transaction Amount Over Time')
        plt.xlabel('Date')
        plt.ylabel('Cumulative Amount')
        plt.grid(True)
        plt.show()
```


Cumulative Transaction Amount Over Time
1e6
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Cumulative Amount
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
Date