Expenses Tracking

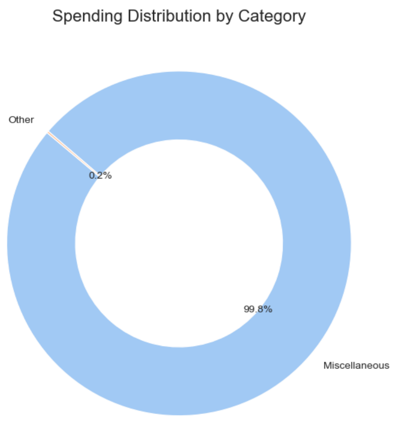
August 5, 2025

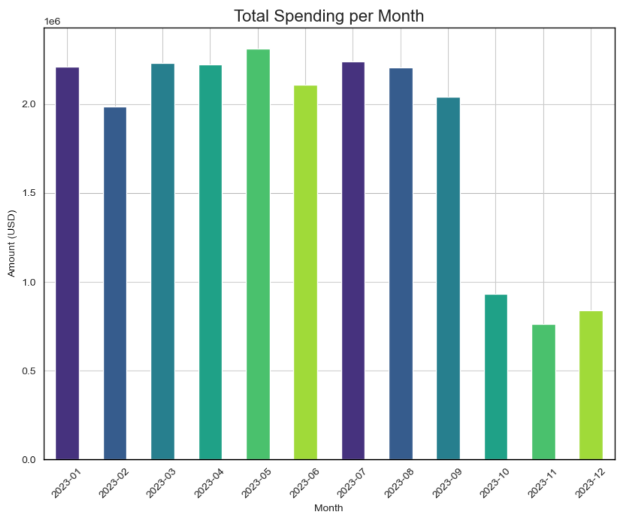
[24]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate

def categorize_transaction(merchant_name):
    """
    Categorizes a transaction based on keywords in the merchant's name.
    This is a simple rule-based classifier for a general dataset.

    Args:
        merchant_name (str): The merchant name from the transaction.

    Returns:
        str: The category of the transaction.
    """
    # Ensure merchant_name is a string before calling .lower()
    if not isinstance(merchant_name, str):
        return 'Miscellaneous'

    merchant_name = merchant_name.lower() # Convert to lowercase for easier␣
↪matching

    # Define keywords for each category - expanded for a more general dataset
    categories = {
        'Groceries': ['walmart', 'costco', 'trader joe\'s', 'safeway',␣
↪'kroger', 'whole foods'],
        'Transport': ['uber', 'lyft', 'gas station', 'exxonmobil', 'shell',␣
↪'amtrak'],
        'Food & Dining': ['starbucks', 'mcdonald\'s', 'subway', 'chipotle',␣
↪'pizza hut', 'restaurant'],
        'Utilities': ['verizon', 'at&t', 'comcast', 'utility', 'electric',␣
↪'water'],
        'Shopping': ['amazon', 'target', 'best buy', 'home depot', 'macys',␣
↪'online purchase'],
        'Health': ['cvs', 'walgreens', 'pharmacy', 'hospital', 'clinic'],
        'Entertainment': ['netflix', 'spotify', 'hulu', 'disney plus', 'movie␣
↪theater', 'ticketmaster'],
        'Transfers': ['venmo', 'paypal', 'zelle', 'bank transfer']
    }

    for category, keywords in categories.items():
        for keyword in keywords:
            if keyword in merchant_name:
                return category

    return 'Miscellaneous' # Default category if no keyword is matched

def analyze_expenses(file_path='credit_card_transactions.csv'):
    """
    Loads transaction data from the Kaggle dataset, categorizes expenses,
    and generates reports.

    Args:
        file_path (str): The path to the transaction CSV file.
    """
    try:
        # Load the dataset from the CSV file
        # The filename must be a string, which is why it's passed in quotes.
        df = pd.read_csv(file_path)
    except FileNotFoundError:
        # Correctly use the file_path variable in the error message
        print(f"Error: The file '{file_path}' was not found.")
        print("Please download the dataset from Kaggle and save it with this␣
↪name,")
        print("or update the file_path variable.")
        return

    # --- 1. Data Cleaning and Preparation ---

    # Rename columns for easier use, handling potential extra spaces
    df.columns = df.columns.str.strip()
    df.rename(columns={
        'Transaction Amount': 'Amount',
        'Merchant Name': 'Description',
        'Date': 'Transaction Date' # Renaming original Date to avoid confusion
    }, inplace=True)

    # Convert 'Transaction Date' column to datetime objects
    # Using format='mixed' allows pandas to infer different date formats.
    df['Date'] = pd.to_datetime(df['Transaction Date'], format='mixed',␣
↪errors='coerce')
```

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

    # Ensure 'Amount' is a numeric type
    df['Amount'] = pd.to_numeric(df['Amount'])

    # The Kaggle dataset amounts are all positive expenses, so we don't need to␣
 ↪filter
    expenses_df = df.copy()

    # Extract month and year for monthly analysis
    expenses_df['Month'] = expenses_df['Date'].dt.to_period('M')

    # --- 2. Categorization ---

    # Apply the categorization function to create a 'Category' column
    expenses_df['Category'] = expenses_df['Description'].
 ↪apply(categorize_transaction)

    # --- 3. Analysis ---

    # Calculate total spending per category
    category_spending = expenses_df.groupby('Category')['Amount'].sum().
 ↪sort_values(ascending=False)

    # Calculate total spending per month
    monthly_spending = expenses_df.groupby('Month')['Amount'].sum()

# --- 4. Reporting and Visualization ---

    print("--- Personal Expense Analysis ---")

    # --- Format and print Category Spending Table ---
    category_spending_df = category_spending.reset_index()
    category_spending_df.columns = ['Category', 'Total Spending']
    category_spending_df['Total Spending'] = category_spending_df['Total␣
 ↪Spending'].map('${:,.2f}'.format)

    print("\nTotal Spending per Category:")
    print(tabulate(category_spending_df, headers='keys', tablefmt='grid',␣
 ↪showindex=False))

    # --- Format and print Monthly Spending Table ---
    monthly_spending_df = monthly_spending.reset_index()
    monthly_spending_df.columns = ['Month', 'Total Spending']
    monthly_spending_df['Total Spending'] = monthly_spending_df['Total␣
 ↪Spending'].map('${:,.2f}'.format)
```

```
    # Convert Period object to string for cleaner display
    monthly_spending_df['Month'] = monthly_spending_df['Month'].astype(str)

    print("\nTotal Spending per Month:")
    print(tabulate(monthly_spending_df, headers='keys', tablefmt='grid',␣
↪showindex=False))

    # --- Plotting ---
    sns.set_style("whitegrid")
    plt.rcParams['figure.figsize'] = (16, 7)

    # --- Prepare data for Pie Chart to avoid label overlap ---
    # Group small slices (e.g., less than 2% of total) into an 'Other' category
    plot_data = category_spending.copy()
    threshold_percent = 2.0
    min_value = (threshold_percent / 100) * plot_data.sum()

    # Identify small slices
    small_slices = plot_data[plot_data < min_value]

    if not small_slices.empty:
        # Sum small slices into 'Other'
        other_sum = small_slices.sum()
        # Remove small slices from main data
        plot_data = plot_data[plot_data >= min_value]
        # Add the 'Other' category
        plot_data['Other'] = other_sum

    # Plot 1: Spending by Category (Pie Chart)
    ax1 = plt.subplot(1, 2, 1)
    plot_data.plot(kind='pie', autopct='%1.1f%%', startangle=140,
                           wedgeprops=dict(width=0.4, edgecolor='w'),
                           colors=sns.color_palette('pastel'), ax=ax1)
    plt.title('Spending Distribution by Category', fontsize=16)
    plt.ylabel('')
    # Add a border around the pie chart
    for spine in ax1.spines.values():
        spine.set_edgecolor('black')
        spine.set_linewidth(1)
        spine.set_visible(True)

    # Plot 2: Monthly Spending (Bar Chart)
    ax2 = plt.subplot(1, 2, 2)
    monthly_spending.plot(kind='bar', color=sns.color_palette('viridis'),␣
↪ax=ax2)
    plt.title('Total Spending per Month', fontsize=16)
    plt.xlabel('Month')
    plt.ylabel('Amount (USD)')
    plt.xticks(rotation=45)
    # Add a border around the bar chart
    for spine in ax2.spines.values():
        spine.set_edgecolor('black')
        spine.set_linewidth(1)
        spine.set_visible(True)

    plt.tight_layout()
    plt.show()


# --- Run the analysis ---
if __name__ == "__main__":
    analyze_expenses(file_path="credit_card_transaction_flow.csv")
```

```
--- Personal Expense Analysis ---

Total Spending per Category:
+---------------+------------------+
| Category      | Total Spending   |
+===============+==================+
| Miscellaneous | $22,067,867.05   |
+---------------+------------------+
| Utilities     | $21,562.20       |
+---------------+------------------+
| Transport     | $16,532.72       |
+---------------+------------------+

Total Spending per Month:
+---------+------------------+
| Month   | Total Spending   |
+=========+==================+
| 2023-01 | $2,210,193.73    |
+---------+------------------+
| 2023-02 | $1,988,572.39    |
+---------+------------------+
| 2023-03 | $2,233,291.59    |
+---------+------------------+
| 2023-04 | $2,224,289.16    |
+---------+------------------+
| 2023-05 | $2,314,428.12    |
+---------+------------------+
| 2023-06 | $2,109,355.54    |
+---------+------------------+
| 2023-07 | $2,241,160.38    |
+---------+------------------+
| 2023-08 | $2,206,324.72    |
+----------+------------------+
| 2023-09  | $2,042,671.23    |
+----------+------------------+
| 2023-10  | $934,387.60      |
+----------+------------------+
| 2023-11  | $763,150.62      |
+----------+------------------+
| 2023-12  | $838,136.89      |
+----------+------------------+
```





```
[ ]:
```