Expense Tracker

Introduction:

The "Expense Tracker" project is a Python-based application designed to help usersanalyse their financial transactions. The application allows users to categorize their bank statement transactions monthly, set a budget, and visualize their expenses through bar charts and pie charts.

Features:

1.Categorization of Transactions:

- Users can upload their bank statement in Excel format.

- The application categorizes transactions based on keywords in the transaction descriptions.

- Categories include food, study, cloth, self-care, stock market, income, and others.

2.Budget Setting:

- Users can set a budget for their expenses.

3.Feedback on Budget:

- The application provides feedback on whether the total expenses exceed the set budget.

5.Visualization:

- The application generates a bar chart showing the total amount spent in each expense category.

- Monthly pie charts display the distribution of expenses across different categories for each month.

Implementation Details:

1.Libraries Used:

- ✓ Pandas: Used for data manipulation and analysis.
- ✓ Matplotlib: Used for creating visualizations.
- ✓ Tkinter: Used for creating the graphical user interface.
- ✓ Filedialog: Tkinter submodule for file dialog functionality.
- ✓ TTK: Tkinter submodule for additional styling options.

```python
import pandas as pd
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import filedialog
from tkinter import ttk
# Import ttk for styling

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
```

2.Code Structure:

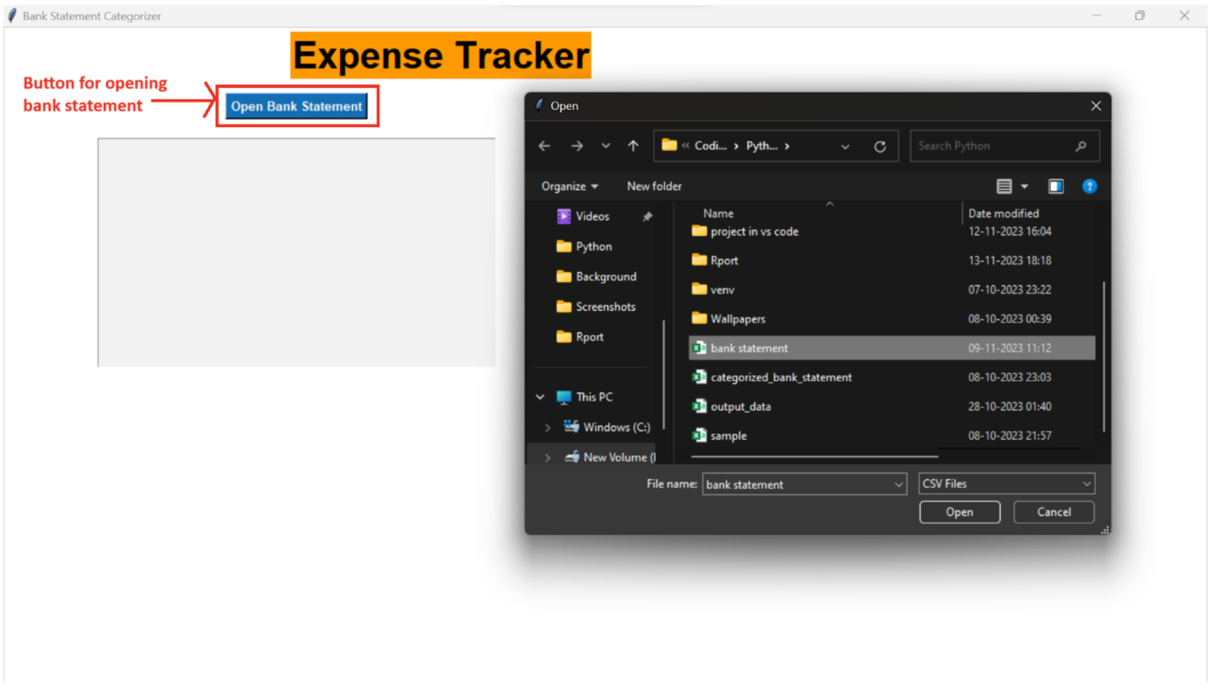
- The project is organized into functions for better modularity and readability.
- Global variables are used for storing the DataFrame (`df`) and budget.
- The Tkinter GUI consists of a main window with buttons, text output area, and a canvas frame for displaying charts.

Workflow:

1. Open Bank Statement:

- Users can upload their bank statement, which is read into a Pandas DataFrame.

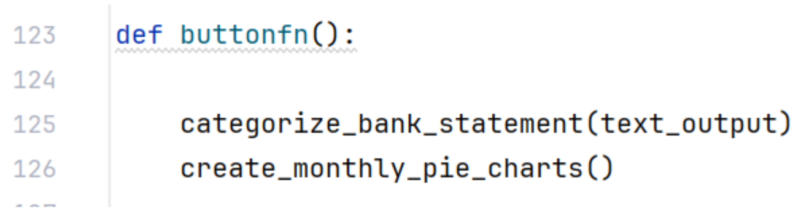
- Transactions are categorized based on the provided keywords.



Name	Date modified
project in vs code	12-11-2023 16:04
Rport	13-11-2023 18:18
venv	07-10-2023 23:22
Wallpapers	08-10-2023 00:39
bank statement	09-11-2023 11:12
categorized_bank_statement	08-10-2023 23:03
output_data	28-10-2023 01:40
sample	08-10-2023 21:57

File name: bank statement

CSV Files



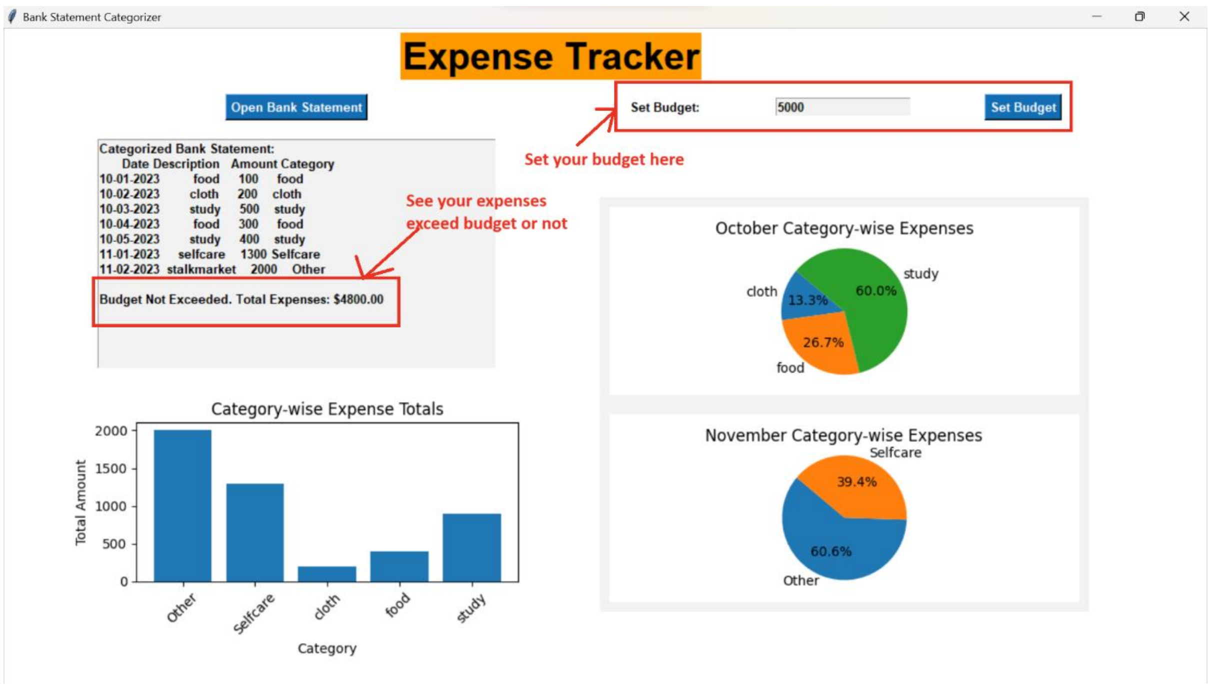
```
def buttonfn():

    categorize_bank_statement(text_output)
    create_monthly_pie_charts()
```

2. Set Budget:

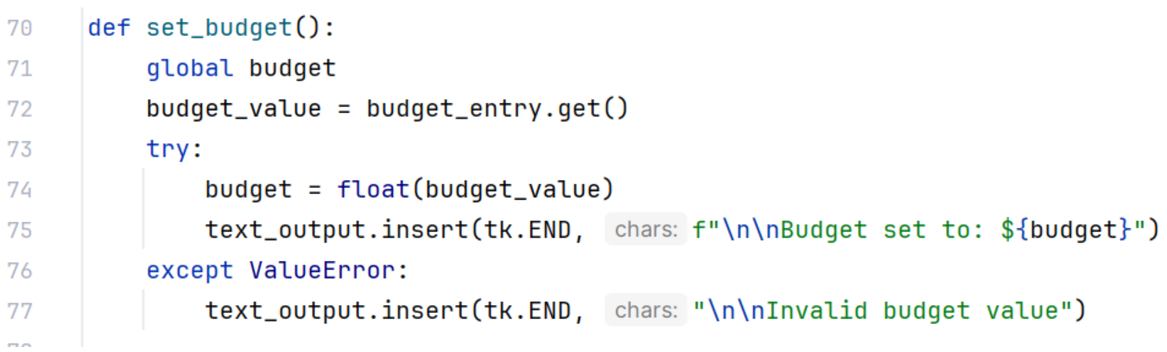
- Users can set a budget for their expenses.

3. Feedback on Budget:

- Provides feedback on whether the total expenses exceed the set budget.

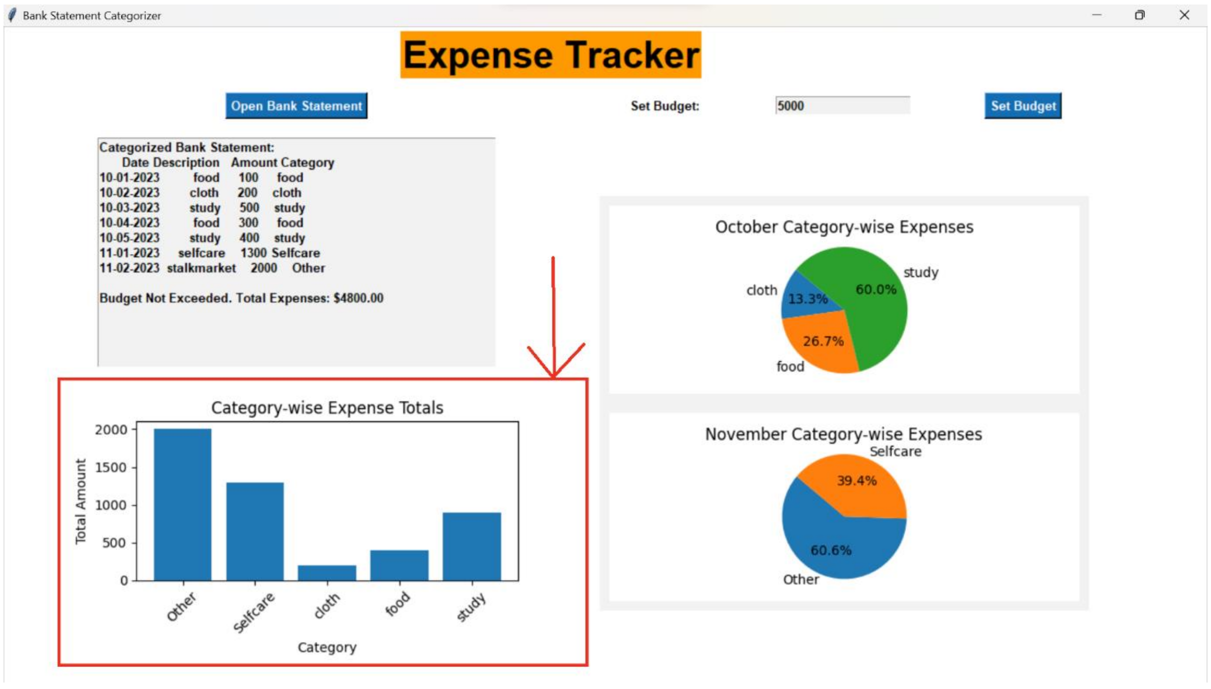


```
Categorized Bank Statement:
     Date Description  Amount Category
10-01-2023        food     100     food
10-02-2023       cloth     200    cloth
10-03-2023       study     500    study
10-04-2023        food     300     food
10-05-2023       study     400    study
11-01-2023    selfcare    1300 Selfcare
11-02-2023 stalkmarket    2000    Other

Budget Not Exceeded. Total Expenses: $4800.00
```

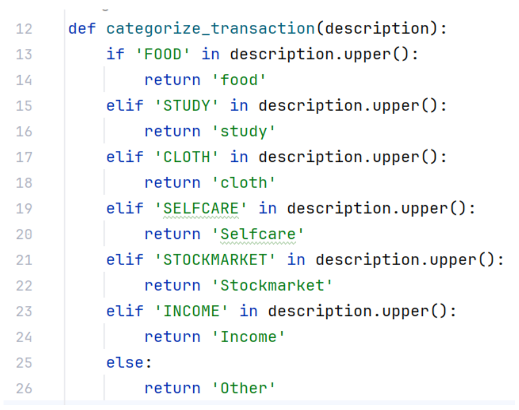


```python
def set_budget():
    global budget
    budget_value = budget_entry.get()
    try:
        budget = float(budget_value)
        text_output.insert(tk.END, chars: f"\n\nBudget set to: ${budget}")
    except ValueError:
        text_output.insert(tk.END, chars: "\n\nInvalid budget value")
```

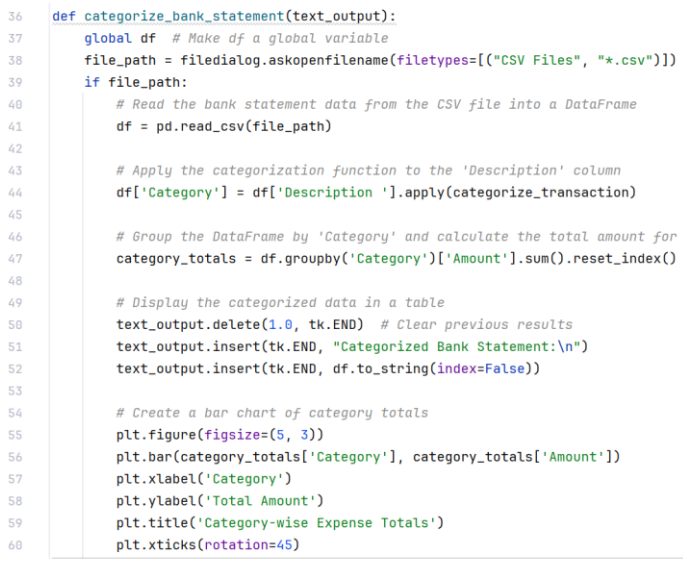
4. Categorization of Transactions:



```
Categorized Bank Statement:
     Date Description  Amount Category
10-01-2023        food     100     food
10-02-2023       cloth     200    cloth
10-03-2023       study     500    study
10-04-2023        food     300     food
10-05-2023       study     400    study
11-01-2023    selfcare    1300 Selfcare
11-02-2023 stalkmarket    2000    Other

Budget Not Exceeded. Total Expenses: $4800.00
```

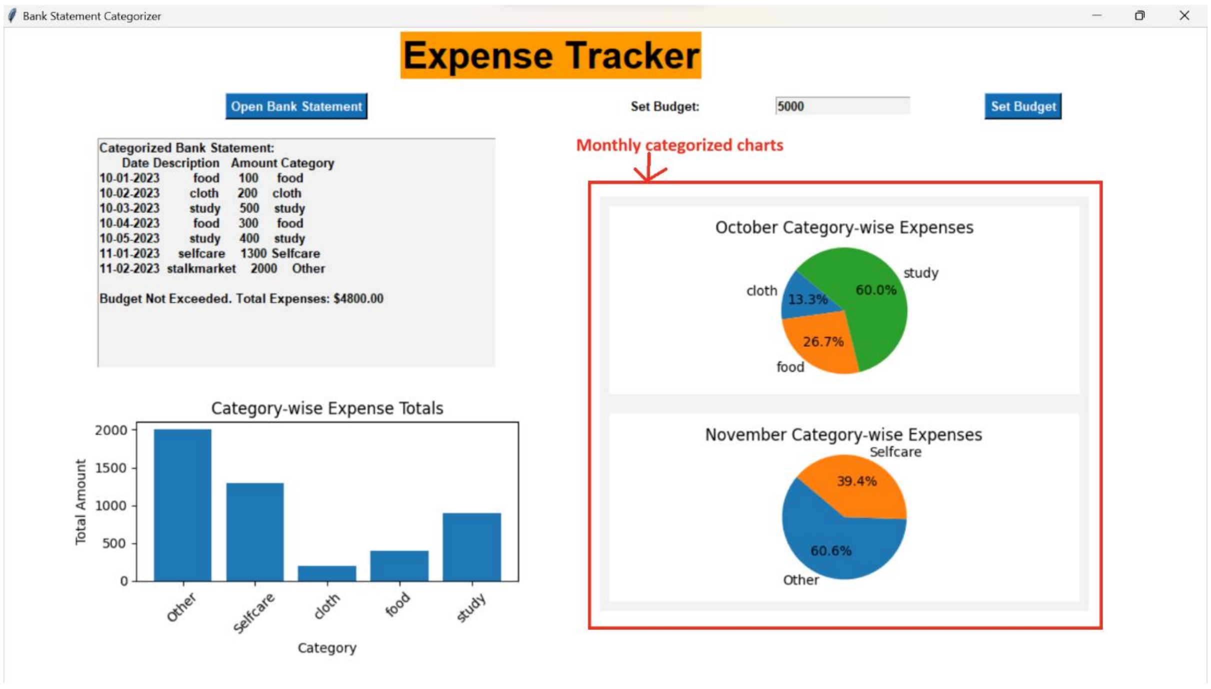


```python
def categorize_transaction(description):
    if 'FOOD' in description.upper():
        return 'food'
    elif 'STUDY' in description.upper():
        return 'study'
    elif 'CLOTH' in description.upper():
        return 'cloth'
    elif 'SELFCARE' in description.upper():
        return 'Selfcare'
    elif 'STOCKMARKET' in description.upper():
        return 'Stockmarket'
    elif 'INCOME' in description.upper():
        return 'Income'
    else:
        return 'Other'
```

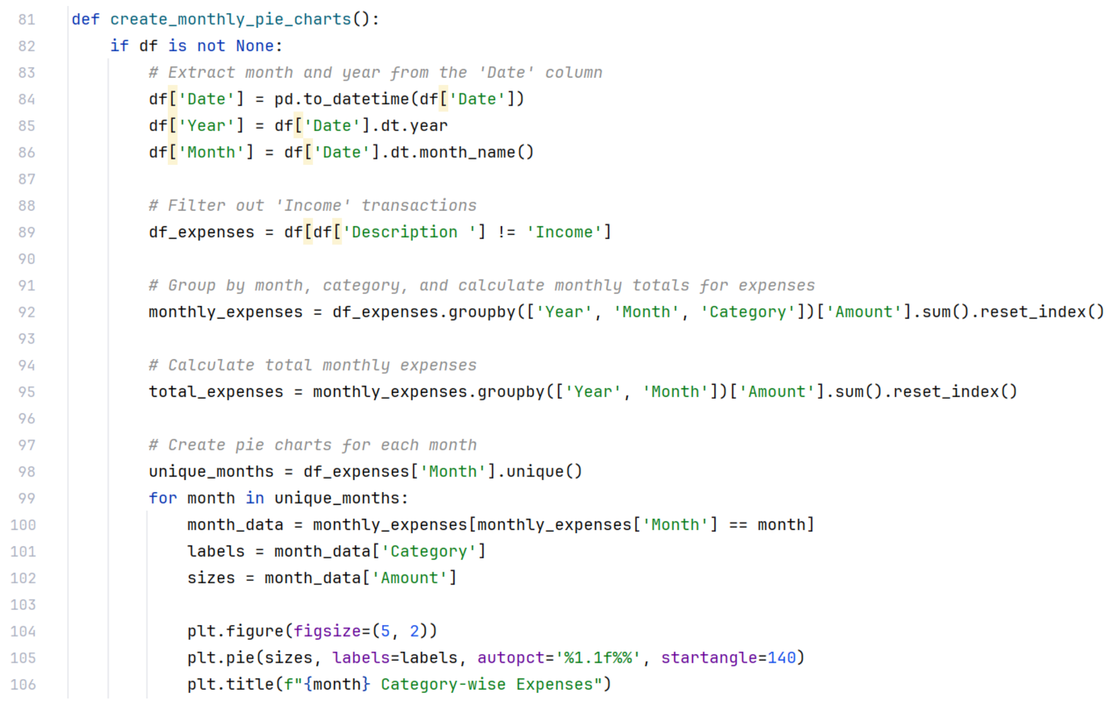


```python
def categorize_bank_statement(text_output):
    global df  # Make df a global variable
    file_path = filedialog.askopenfilename(filetypes=[("CSV Files", "*.csv")])
    if file_path:
        # Read the bank statement data from the CSV file into a DataFrame
        df = pd.read_csv(file_path)

        # Apply the categorization function to the 'Description' column
        df['Category'] = df['Description '].apply(categorize_transaction)

        # Group the DataFrame by 'Category' and calculate the total amount for
        category_totals = df.groupby('Category')['Amount'].sum().reset_index()

        # Display the categorized data in a table
        text_output.delete(1.0, tk.END)  # Clear previous results
        text_output.insert(tk.END, "Categorized Bank Statement:\n")
        text_output.insert(tk.END, df.to_string(index=False))

        # Create a bar chart of category totals
        plt.figure(figsize=(5, 3))
        plt.bar(category_totals['Category'], category_totals['Amount'])
        plt.xlabel('Category')
        plt.ylabel('Total Amount')
        plt.title('Category-wise Expense Totals')
        plt.xticks(rotation=45)
```

5. Monthly categorized charts:



```
Categorized Bank Statement:
     Date Description  Amount Category
10-01-2023        food     100     food
10-02-2023       cloth     200    cloth
10-03-2023       study     500    study
10-04-2023        food     300     food
10-05-2023       study     400    study
11-01-2023    selfcare    1300 Selfcare
11-02-2023 stalkmarket    2000    Other

Budget Not Exceeded. Total Expenses: $4800.00
```

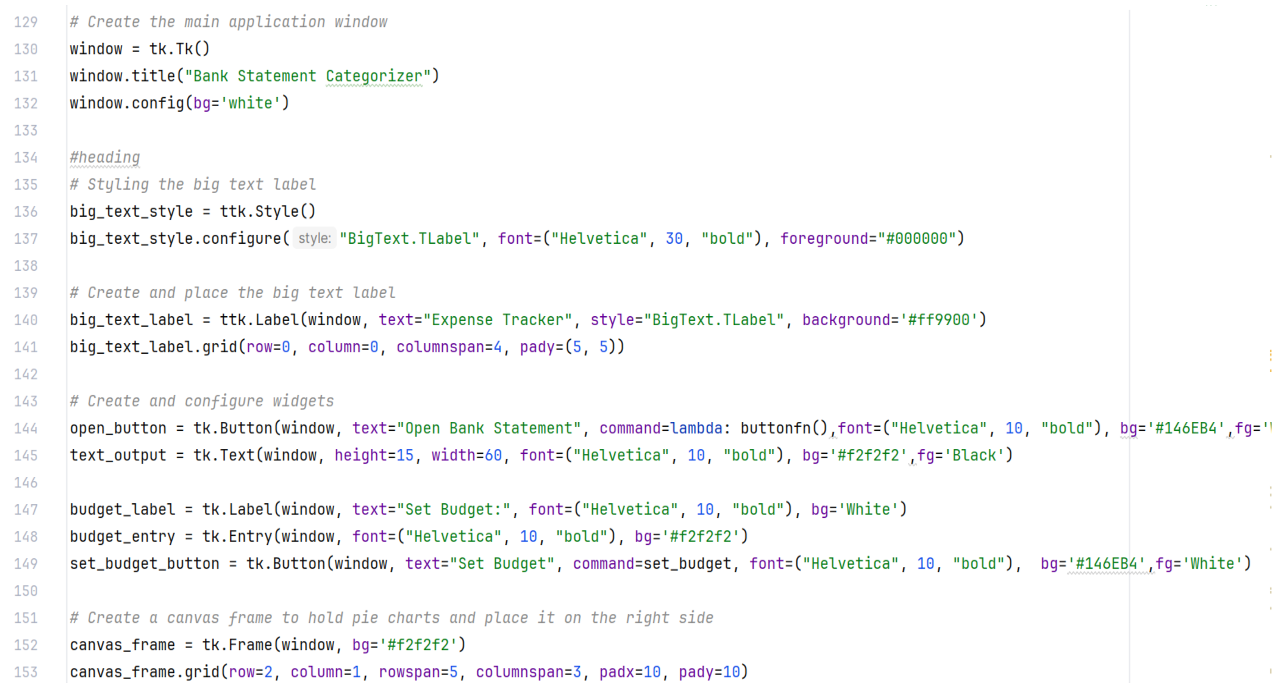


```python
def create_monthly_pie_charts():
    if df is not None:
        # Extract month and year from the 'Date' column
        df['Date'] = pd.to_datetime(df['Date'])
        df['Year'] = df['Date'].dt.year
        df['Month'] = df['Date'].dt.month_name()

        # Filter out 'Income' transactions
        df_expenses = df[df['Description '] != 'Income']

        # Group by month, category, and calculate monthly totals for expenses
        monthly_expenses = df_expenses.groupby(['Year', 'Month', 'Category'])['Amount'].sum().reset_index()

        # Calculate total monthly expenses
        total_expenses = monthly_expenses.groupby(['Year', 'Month'])['Amount'].sum().reset_index()

        # Create pie charts for each month
        unique_months = df_expenses['Month'].unique()
        for month in unique_months:
            month_data = monthly_expenses[monthly_expenses['Month'] == month]
            labels = month_data['Category']
            sizes = month_data['Amount']

            plt.figure(figsize=(5, 2))
            plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
            plt.title(f"{month} Category-wise Expenses")
```

6. Visualization:

- Bar chart: Shows total expenses in each category.

- Monthly pie charts: Display the distribution of expenses for each category in a given month.



```python
# Create the main application window
window = tk.Tk()
window.title("Bank Statement Categorizer")
window.config(bg='white')

#heading
# Styling the big text label
big_text_style = ttk.Style()
big_text_style.configure( style: "BigText.TLabel", font=("Helvetica", 30, "bold"), foreground="#000000")

# Create and place the big text label
big_text_label = ttk.Label(window, text="Expense Tracker", style="BigText.TLabel", background='#ff9900')
big_text_label.grid(row=0, column=0, columnspan=4, pady=(5, 5))

# Create and configure widgets
open_button = tk.Button(window, text="Open Bank Statement", command=lambda: buttonfn(),font=("Helvetica", 10, "bold"), bg='#146EB4',fg='
text_output = tk.Text(window, height=15, width=60, font=("Helvetica", 10, "bold"), bg='#f2f2f2',fg='Black')

budget_label = tk.Label(window, text="Set Budget:", font=("Helvetica", 10, "bold"), bg='White')
budget_entry = tk.Entry(window, font=("Helvetica", 10, "bold"), bg='#f2f2f2')
set_budget_button = tk.Button(window, text="Set Budget", command=set_budget, font=("Helvetica", 10, "bold"),  bg='#146EB4',fg='White')

# Create a canvas frame to hold pie charts and place it on the right side
canvas_frame = tk.Frame(window, bg='#f2f2f2')
canvas_frame.grid(row=2, column=1, rowspan=5, columnspan=3, padx=10, pady=10)
```

Usage Instructions:

1. Launch the application.

2. Click "Open Bank Statement" to upload a CSV file containing your bank statement.

3. View the categorized bank statement and the bar chart of category-wise expense totals.

4. Set a budget using the "Set Budget" button.

5. Monthly pie charts will be generated to visualize the distribution of expenses.

6. Receive feedback on whether your expenses exceed the budget.

Conclusion:

The "Expense Tracker" project provides users with a convenient tool to analyse their finances. The categorization, budget setting, and visualization features empower users to make informed decisions about their spending habits. The user-friendly interface enhances the overall experience, making financial tracking more accessible and efficient.