**Add\_expenses:**

Add expense function is written to collect the details about the expenses, including the date, category, amount and description.

The date is entered in yyyy-mm-dd format so that if the user enters the invalid date, the function will ask to re-enter until the valid one is provided. The datetime.strptime method is used to validate the date format. So, we have import datetime library

The function presents the user with a list of predefined categories (Food, Travel, Entertainment, Utility). The user selects one of these categories. If the user enters an invalid category, they are asked to choose a valid one.

The amount spent and the description of the expenses added at the last.

After collecting all the necessary details, the function returns the dictionary with the keys.

'date': '2025-03-10',

'category': 'Food',

'amount': '50.00',

'description': 'Lunch at a restaurant

**Error Handling**

* The function checks for the correct date format (yyyy-mm-dd) and will prompt the user until a valid date is entered.
* If the user inputs a category that is not in the list, the function will ask the user to select from the available categories.
* The function does not perform validation for the expense amount and description (e.g., no checks for non-numeric values or empty descriptions)

**Budget\_tracker:**

The budget\_tracker function assists users in monitoring their monthly expenses by comparing them against a predefined budget. It calculates the total expenses from a data source, determines the remaining balance, and provides feedback on budget adherence

Functionality:

After the user inputs a value, you can convert it to a floating-point number, This will enable you to perform numerical calculations with the budget.

To retrieve all recorded expenses, you would call the load\_data function

To compute the sum of all expense amounts using the Pandas `.sum()` method, adds all values in each column and return the sum for each column

Calculates the remaining balance by subtracting the total expenses from the budget.

Feeback for the budget status –

1. If the balance is negative, informs the user that they have exceeded their budget.
2. If the balance is positive or zero, displays the remaining amount left for the month

**Load Data:**

The load\_data function is crafted to import expense records from a CSV file called 'Expense Tracker.csv' into a Pandas DataFrame, enabling efficient data manipulation and analysis in Python.

Functionality

Data Loading: Uses Pandas' read\_csv() function to read 'Expense Tracker.csv' and load its contents into a DataFrame. It assumes the CSV is properly structured with headers matching the expense data fields.

This will: -

Import the Pandas library.

Define the load data function to load 'Expense Tracker.csv' into a Data Frame.

Store the data in expenses\_df.

Print the first five rows to the console

**Save Data:**

The save\_data function is vital for managing your finances. It automatically appends new expense entries to 'Expense Tracker.csv' and creates the file if it doesn't exist. This ensures your expense data is always updated and securely stored, making it easy to track your spending habits and simplify your budgeting.

Functionality

1. It accepts an optional parameter called expenses, which should be a dictionary-like object containing the details of the expenses
2. Converts the expenses input into a Pandas DataFrame to facilitate seamless data manipulation.
3. File Handling: The function first attempts to load any existing expense data from 'Expense Tracker.csv' using the load\_data function, which must be defined elsewhere in your code. If the file exists, it combines the new expense data with the current data and writes the updated DataFrame back to the CSV file. Conversely, if the file doesn’t exist, it creates a new DataFrame containing the expense data and saves it as 'Expense Tracker.csv'
4. Error Handling: The function uses a nested try-except block to manage exceptions during data processing. If an error occurs during initial data processing, the function safeguards your existing data by saving it without the new entries, ensuring data integrity. If the file is missing during this attempt, the user is notified that there is no data to save.

Utilizes the Pandas library for data manipulation, specifically the pd.DataFrame constructor, pd.concat() function, and the .to\_csv() method. Ensure that Pandas is installed in your environment:

This will:

Convert new\_expense into a DataFrame.

Attempt to load existing expenses from 'Expense Tracker.csv'.

Append the new expense to the existing data.

Save the updated data back to 'Expense Tracker.csv'.

**View\_expenses**:

The view expenses function reads an existing CSV file named 'Expense Tracker.csv', which contains records of expenses

Nested function is used to ensure that user input for each field is not empty. If the input is empty or consists only of whitespace, it displays an error message and prompts again until valid input is provided.

Need a CSV file to open in a read mode. Hence imported csv libraries. Utilizes the csv.DictReader class to read the CSV file, which allows each row to be accessed as a dictionary with column headers as keys. This approach simplifies handling CSV data, especially when dealing with large files.

For each row, the function checks if any field is missing or empty. If missing data is found, the function prompts the user to provide the missing information using the get\_valid\_input function. It specifically requests input for the following fields:

date

category

amount

description

After collecting all necessary information, the function prints the updated row, reflecting any user-provided data.

When executed, the function will:

* Read the 'Expense Tracker.csv' file.
* For each row with missing data, prompt the user to enter the missing information.
* Display the updated rows with the newly provided data.

**Menu**:

The menu function provides a text interface for users to manage expenses. It displays options, processes input, and calls corresponding functions based on user choices, allowing multiple actions until they exit.

Functionality:

Uses Pandas' read\_csv() to load 'Expense Tracker.csv' into a DataFrame, assuming the CSV has proper headers for the expense data fields.

This will: -

Import Pandas.

Define load\_data to read 'Expense Tracker.csv' into a DataFrame.

Load data into expenses\_df.

Print the first five rows.