Introduction

This application is a web-based expense tracker built with Python. It uses the following technologies and libraries:

- **Faker**: Generate fake data for testing.
- **Streamlit**: For creating an interactive dashboard.
- **MySQL**: Store and query expense data.
- **Pandas**: For data manipulation and analysis.

The application allows users to visualize expenses by distinct categories, payment modes, and timeframes.

Dependencies

Ensure the following Python packages are installed:

pip install faker streamlit pandas mysql-connector-python

Features

1. **Data Generation**:
    - Generates 150 fake expense records using the Faker library.
    - Each record includes:
        - Date
        - Category
        - Payment Mode
        - Description
        - Amount Paid
        - Cashback
2. **Database Management**:
    - Connects to a MySQL database.
    - Creates a database (EXPENSES) and a table (expense) if they do not exist.
    - Inserts generated expense records into the database.
3. **Dashboard**:
    - Displays various visualizations and metrics using Streamlit.
    - Fetches data from the database for analysis.

Code Walkthrough

1. Importing Libraries

```
from faker import Faker
import streamlit as st
import random
import pandas as pd
import mysql.connector
```

 Faker: To create mock data.

 Streamlit: For creating the web dashboard.

 random: To select random values for categories and payment modes.

 Pandas: For data manipulation.

 mysql.connector: For interacting with the MySQL database.

2. Generating Fake Data

```
def gen_expense():
    data = []
    for i in range(150):
        expense = {
            "Date": fake.date_between("-1y", "today"),
            "Category": random.choice(categories),
            "Payment Modes": random.choice(payment_modes),
            "Description": fake.sentence(),
            "Amount Paid": round(random.uniform(50, 2000), 2),
            "Cashback": round(random.uniform(5, 500), 2)
        }
        data.append(expense)
    return pd.DataFrame(data)
```

Generates 150 records with random values for:

- ***Date**: Within the last year.*
- ***Category**: Randomly selected from a predefined list.*
- ***Payment Mode**: Randomly selected from a predefined list.*
- ***Description**: Random sentence generated by Faker.*

- ***Amount Paid**: Random float between 50 and 2000.*
- ***Cashback**: Random float between 5 and 500.*

3. Database Operations

- **Connecting to MySQL**

```
connection = mysql.connector.connect(
   host="localhost",
   port=3306,
   user="root",
   password="1234",
   autocommit=True
)
```

- Connects to MySQL with credentials. Ensure the MySQL server is running.

Database and Table Setup

- Creates the `EXPENSES` database if it does not exist.

```
CREATE DATABASE IF NOT EXISTS EXPENSES;
```

- *Creates the expense table:*

```
CREATE TABLE IF NOT EXISTS expense (
  id INT AUTO_INCREMENT PRIMARY KEY,
  date DATE,
  category VARCHAR(255),
  payment_mode VARCHAR(255),
  description TEXT,
  amount_paid FLOAT,
  cashback FLOAT
);
```

- Inserting Data
  Inserts the generated records into the table:

```
INSERT INTO expense (date, category, payment_mode, description, amount_paid, cashback)
VALUES (%s, %s, %s, %s, %s, %s);
```

4. Fetching Data

```
def fetch_data(query):
  connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="1234",
    database="EXPENSES"
  )
  data = pd.read_sql(query, connection)
  connection.close()
```

return data

- *Executes a SQL query and fetches the results into a Pandas DataFrame.*
-

5. Streamlit Dashboard

Visualizations

- **All Expenses**
  *SELECT * FROM expense ORDER BY Date asc;*

- ***Total Spending**:*
  SELECT SUM(amount_paid) AS total_spent FROM expense;

- ***Monthly Spending**:*
  SELECT DATE_FORMAT(date, '%Y-%m') AS month, SUM(amount_paid) AS total_spent
  FROM expense
  GROUP BY month
  ORDER BY month;

- ***Spending by Category**:*
  SELECT category, SUM(amount_paid) AS total_spent
  FROM expense
  GROUP BY category
  ORDER BY total_spent DESC;

- **Spending by Payment Mode**
  SELECT payment_mode, SUM(round(amount_paid)) AS total_spent FROM expense GROUP BY payment_mode ORDER BY total_spent DESC;

- **Category-Wise cashback**
  SELECT category, SUM(cashback) AS total_cashback FROM expense GROUP BY category ORDER BY total_cashback DESC;

- **Transaction Per Category**
  SELECT category, COUNT(*) AS transaction_count FROM expense GROUP BY category ORDER BY transaction_count DESC;

- **Percentage of Spending by Category**
  *SELECT category, SUM(round(amount_paid)) AS total_spent,round(SUM(round(amount_paid)) / (SELECT SUM(round(amount_paid)) FROM expense) * 100) AS percentage_spent FROM expense GROUP BY category ORDER BY percentage_spent DESC;*

- ***Average Monthly Spending***
  SELECT DATE_FORMAT(date, '%Y-%m') AS month, round(AVG(amount_paid)) AS avg_monthly_spent FROM expense GROUP BY month ORDER BY month;

- **Spending by Day**

SELECT DAYNAME(date) AS day_of_week, SUM(round(amount_paid)) AS total_spent FROM expense GROUP BY day_of_week ORDER BY FIELD(day_of_week, 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday');

- **Payment Mode-wise Cashback**
  SELECT payment_mode, round(SUM(cashback)) AS total_cashback FROM expense GROUP BY payment_mode ORDER BY total_cashback DESC;

- **Spending Distribution by range**
  SELECT round(SUM(CASE WHEN category = 'Investments' THEN amount_paid ELSE 0 END)) AS total_investments,round(SUM(CASE WHEN category != 'Investments' THEN amount_paid ELSE 0 END)) AS total_other_spent FROM expense;

- **Daily Spending Trend**
  SELECT date, SUM(amount_paid) AS daily_spent FROM expense GROUP BY date ORDER BY date;