PERSONAL FINANCE HEALTH

ANALYZER AND OPTIMIZER

Introduction: This is one of the finance related mysql project which focuses on personal finance analysis and optimization of personal finance's. This is one of the niche that is highly relevant in today's world and is rarely used in depth. This project will be designed using mysql.

This project is a mysql based system designed to help individuals analyze and optimize their personal finance's. It will include tracking income expenses savings investments and debts. It will provide actionable insights that will help improve financial well-being. The system will use mysql concepts to generate reports identify financial trends and suggest optimization strategies.

Pre-requisites for this project:

In order to perform this project there are some prerequisites that the user needs to have

- Mysql 8.0 command line client.
- Mysql 8.0 workbench C.E
- Mysql statements
- Mysql operations
- Mysql clauses
- Mysl constraints
- Mysql sub-queries
- Mysql joins
- User permissions(Grant and Revoke)
- Transactions

These are the concepts or technologies which the user needs to be fundamentally strong.

Key features of this projects:

This projects will include several important features of real-world finance management such as

- **Income and expenses tracking:** it is used to track monthly income sources I am categories expenses.
- **Savings and investment analysis:** monitoring savings account and investment portfolios along with calculating returns.
- **Debt management:** it is used to track debts (credit cards and loans etc) and calculate interest payments along with suggesting debt repayment strategies.
- **Financial health scoring:** it is used to generate a financial health call based on income expenses savings and debts it is used to access financial stability.
- **Budget optimization:** it is used to suggest budget allocation that are optimal based on previous data
- **User permissions:** it is used to provide roll based axis it is also used to restrict access on sensitive financial data

Schema

In order to perform this project various parameters required

01. Database: create a new database for this project with the project name personal finance health

02. Tables required:

i. Users: this table should contain user details (user_id, username, roll, password).

ii. Income: used to store income sources of the user (income_id, user_id, source, amount, date).

iii. Expenses: it is used to store expenses details of the user (expense_id, user_id, category, amount, date).

iv. Savings: used to store savings details (saving_id, user_id, account_type, amount, date).

v. Investments: used to store investment details (investment _id , user_id, type, amount, return_rate, due_date).

vi. Debts: used to store debt details (debt_id, user_id, type, amount, Interest_rate, date)

vii. Financial health: used to store financial health score (health_id, user_id, score, date)

03. Relationships: with the use of constraints keys the users table must be linked with all the primary key and foreign key. This is called a one to many relationship.

04. User:

i. **Admin- with admin privilege all privilege.**

ii. **User- with user privilege.**

Implementation

In order to design the structure to this project and generate analysis based on the data, here are the steps of implementation for this project.

Step 01: creating database and creating the necessary tables.

• Show database.

> Show databases;

• Create Database:

> CREATE DATABASE Personal_Finance_Health;

• Use database.

> use personal_finance_health;

Table: 01

• Users Table

CREATE TABLE Users (

User_ID INT PRIMARY KEY AUTO_INCREMENT,

Username VARCHAR(50) NOT NULL UNIQUE,

Role ENUM('Admin', 'User') NOT NULL,

Password VARCHAR(255) NOT NULL,

Unique(username)

);

```
mysql> describe table users;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | users | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.17 sec)
```

Table: 02

• Income Table

CREATE TABLE Income (

Income_ID INT PRIMARY KEY AUTO_INCREMENT,

User_ID INT,

Source VARCHAR(100) NOT NULL,

Amount DECIMAL(10,2) NOT NULL,

Date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Users(User_ID) ON DELETE CASCADE

);

```
mysql> describe table income;
+----+-------------+--------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table  | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+--------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | income | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+--------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.12 sec)
```

Table: 03

• Expenses Table

CREATE TABLE Expenses (

Expense_ID INT PRIMARY KEY AUTO_INCREMENT,

User_ID INT,

Category VARCHAR(100) NOT NULL,

Amount DECIMAL(10,2) NOT NULL,

Date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Users(User_ID) ON DELETE CASCADE

);

```
mysql> describe table expenses;
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | expenses | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

Table: 04

• Savings Table

CREATE TABLE Savings (

Saving_ID INT PRIMARY KEY AUTO_INCREMENT,

User_ID INT,

Account_type VARCHAR(50) NOT NULL,

Amount DECIMAL(10,2) NOT NULL,

Date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Users(User_ID) ON DELETE CASCADE

);

```
mysql> describe table savings;
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | savings | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.04 sec)
```

Table: 05

• Investments Table

CREATE TABLE Investments (

Investment_ID INT PRIMARY KEY AUTO_INCREMENT,

User_ID INT,

Type VARCHAR(50) NOT NULL,

Amount DECIMAL(10,2) NOT NULL,

Return_rate DECIMAL(5,2) NOT NULL,

Date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Users(User_ID) ON DELETE CASCADE

);

```
mysql> describe table investments;
+----+-------------+-------------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table       | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | investments | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+-------------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.09 sec)
```

Table: 06

• Debts Table

CREATE TABLE Debts (

Debt_ID INT PRIMARY KEY AUTO_INCREMENT,

User_ID INT,

Type VARCHAR(50) NOT NULL,

Amount DECIMAL(10,2) NOT NULL,

Interest_rate DECIMAL(5,2) NOT NULL,

Due_date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Users(User_ID) ON DELETE CASCADE

);

```
mysql> describe table debts;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | debts | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

Table: 07

• Financial Health Table

CREATE TABLE FinancialHealth (

Health_ID INT PRIMARY KEY AUTO_INCREMENT,

User_ID INT,

Score INT NOT NULL CHECK (Score BETWEEN 0 AND 100),

Date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Users(User_ID) ON DELETE CASCADE

);

```
mysql> describe table financialhealth;
+----+-------------+-----------------+------------+------+---------------+------+---------+------+------+----------+-------+
| id | select_type | table           | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra |
+----+-------------+-----------------+------------+------+---------------+------+---------+------+------+----------+-------+
|  1 | SIMPLE      | financialhealth | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    1 |   100.00 | NULL  |
+----+-------------+-----------------+------------+------+---------------+------+---------+------+------+----------+-------+
1 row in set, 1 warning (0.02 sec)
```

Implementation

Step 02: Inserting values more than 100 rows in all 7 tables to perform the analysis. Apart from the auto_increment columns all the columns have to be filled with data.

Table 01: Users - There should be 10 users among those 10 one should be named as admin whereas the rest 9 will be user's.

INSERT INTO Users (Username, Role, Password) VALUES

('Alice', 'Admin', 'password123'),

('Bob', 'User', 'securepass'),

('Charlie', 'User', 'charliepass'),

('David', 'User', 'davidpass'),

('Emma', 'User', 'emmapass'),

('Frank', 'User', 'frankpass'),

('Grace', 'User', 'gracepass'),

('Hannah', 'User', 'hannahpass'),

('Isaac', 'User', 'isaacpass'),

('Jack', 'User', 'jackpass');

User_ID	Username	Role	Password
1	Alice	Admin	password123
2	Bob	User	securepass
3	Charlie	User	charliepass
4	David	User	davidpass
5	Emma	User	emmapass
6	Frank	User	frankpass
7	Grace	User	gracepass
8	Hannah	User	hannahpass
9	Isaac	User	isaacpass
10	Jack	User	jackpass

Table 02: Income - For each user there are different streams of income along with the profit/amount generated. They have been assigned to a particular user ID that matches the user ID from the table user's.

INSERT INTO Income (User_ID, Source, Amount, Date) VALUES

(2, 'Salary', 5000, '2025-01-01'),

(3, 'Freelance', 2000, '2025-01-10'),

(4, 'Business', 8000, '2025-01-15'),

(5, 'Salary', 6000, '2025-01-20'),

(6, 'Freelance', 3000, '2025-01-25'),

(7, 'Business', 9000, '2025-01-28'),

(8, 'Salary', 5500, '2025-02-01'),

(9, 'Freelance', 2500, '2025-02-05'),

(10, 'Business', 7000, '2025-02-10'),

(3, 'Salary', 4800, '2025-02-15');

Income_ID	User_ID	Source	Amount	Date
1	2	Salary	5000.00	2025-01-01
2	3	Freelance	2000.00	2025-01-10
3	4	Business	8000.00	2025-01-15
4	5	Salary	6000.00	2025-01-20
5	6	Freelance	3000.00	2025-01-25
6	7	Business	9000.00	2025-01-28
7	8	Salary	5500.00	2025-02-01
8	9	Freelance	2500.00	2025-02-05
9	10	Business	7000.00	2025-02-10
10	3	Salary	4800.00	2025-02-15

Table 03: Expenses - This table covers the expenses of all 9 users and it will classify those expenses into various categories.

INSERT INTO Expenses (User_ID, Category, Amount, Date) VALUES

-> (2, 'Rent', 1200, '2025-01-05'),

-> (3, 'Groceries', 500, '2025-01-08'),

-> (4, 'Transport', 300, '2025-01-12'),

-> (5, 'Shopping', 800, '2025-01-18'),

-> (6, 'Healthcare', 400, '2025-01-22'),

-> (7, 'Entertainment', 600, '2025-01-26'),

-> (8, 'Rent', 1300, '2025-02-02'),

-> (9, 'Groceries', 550, '2025-02-06'),

-> (10, 'Transport', 350, '2025-02-11'),

-> (3, 'Shopping', 900, '2025-02-16');

Expense_ID	User_ID	Category	Amount	Date
1	2	Rent	1200.00	2025-01-05
2	3	Groceries	500.00	2025-01-08
3	4	Transport	300.00	2025-01-12
4	5	Shopping	800.00	2025-01-18
5	6	Healthcare	400.00	2025-01-22
6	7	Entertainment	600.00	2025-01-26
7	8	Rent	1300.00	2025-02-02
8	9	Groceries	550.00	2025-02-06
9	10	Transport	350.00	2025-02-11
10	3	Shopping	900.00	2025-02-16

Table 04: Savings - This table provides the values for each and every saving medium for each user.

INSERT INTO Savings (User_ID, Account_type, Amount, Date) VALUES

-> (2, 'Bank Savings', 5000, '2025-01-07'),

-> (3, 'Fixed Deposit', 7000, '2025-01-14'),

-> (4, 'Bank Savings', 6000, '2025-01-20'),

-> (5, 'Mutual Funds', 8000, '2025-01-25'),

-> (6, 'Bank Savings', 7500, '2025-01-30'),

-> (7, 'Fixed Deposit', 9000, '2025-02-03'),

-> (8, 'Mutual Funds', 8500, '2025-02-07'),

-> (9, 'Bank Savings', 7000, '2025-02-10'),

-> (10, 'Fixed Deposit', 9500, '2025-02-15'),

-> (3, 'Mutual Funds', 8800, '2025-02-18');

Saving_ID	User_ID	Account_type	Amount	Date
1	2	Bank Savings	5000.00	2025-01-07
2	3	Fixed Deposit	7000.00	2025-01-14
3	4	Bank Savings	6000.00	2025-01-20
4	5	Mutual Funds	8000.00	2025-01-25
5	6	Bank Savings	7500.00	2025-01-30
6	7	Fixed Deposit	9000.00	2025-02-03
7	8	Mutual Funds	8500.00	2025-02-07
8	9	Bank Savings	7000.00	2025-02-10
9	10	Fixed Deposit	9500.00	2025-02-15
10	3	Mutual Funds	8800.00	2025-02-18

Table 05: Investments - This table is used to provide the values of each investment done by the user and the return they have received.

INSERT INTO Investments (User_ID, Type, Amount, Return_rate, Date) VALUES

-> (2, 'Stocks', 5000, 8.5, '2025-01-09'),

-> (3, 'Bonds', 3000, 6.2, '2025-01-16'),

-> (4, 'Real Estate', 10000, 10.0, '2025-01-22'),

-> (5, 'Mutual Funds', 7000, 7.5, '2025-01-27'),

-> (6, 'Stocks', 6000, 9.0, '2025-02-01'),

-> (7, 'Bonds', 4000, 5.8, '2025-02-04'),

-> (8, 'Real Estate', 12000, 11.2, '2025-02-08'),

-> (9, 'Mutual Funds', 7500, 7.0, '2025-02-12'),

-> (10, 'Stocks', 5500, 8.0, '2025-02-17'),

-> (3, 'Bonds', 3500, 6.5, '2025-02-19');

Investment_ID	User_ID	Type	Amount	Return_rate	Date
1	2	Stocks	5000.00	8.50	2025-01-09
2	3	Bonds	3000.00	6.20	2025-01-16
3	4	Real Estate	10000.00	10.00	2025-01-22
4	5	Mutual Funds	7000.00	7.50	2025-01-27
5	6	Stocks	6000.00	9.00	2025-02-01
6	7	Bonds	4000.00	5.80	2025-02-04
7	8	Real Estate	12000.00	11.20	2025-02-08
8	9	Mutual Funds	7500.00	7.00	2025-02-12
9	10	Stocks	5500.00	8.00	2025-02-17
10	3	Bonds	3500.00	6.50	2025-02-19

Table 06: Debts - This table is used to provide values that are related to the loan informtion of each user and how much amount they have loaned and at what interest.

INSERT INTO Debts (User_ID, Type, Amount, Interest_rate, Due_date) VALUES

-> (2, 'Credit Card', 2000, 15.0, '2025-02-01'),

-> (3, 'Student Loan', 5000, 5.5, '2025-06-01'),

-> (4, 'Home Loan', 15000, 7.0, '2025-12-01'),

-> (5, 'Car Loan', 10000, 6.5, '2025-09-01'),

-> (6, 'Credit Card', 2500, 14.0, '2025-02-20'),

-> (7, 'Student Loan', 6000, 5.8, '2025-07-01'),

-> (8, 'Home Loan', 14000, 7.2, '2025-11-01'),

-> (9, 'Car Loan', 9500, 6.0, '2025-08-01'),

-> (10, 'Credit Card', 1800, 16.0, '2025-03-01'),

-> (3, 'Personal Loan', 4000, 9.0, '2025-04-01');

Debt_ID	User_ID	Type	Amount	Interest_rate	Due_date
1	2	Credit Card	2000.00	15.00	2025-02-01
2	3	Student Loan	5000.00	5.50	2025-06-01
3	4	Home Loan	15000.00	7.00	2025-12-01
4	5	Car Loan	10000.00	6.50	2025-09-01
5	6	Credit Card	2500.00	14.00	2025-02-20
6	7	Student Loan	6000.00	5.80	2025-07-01
7	8	Home Loan	14000.00	7.20	2025-11-01
8	9	Car Loan	9500.00	6.00	2025-08-01
9	10	Credit Card	1800.00	16.00	2025-03-01
10	3	Personal Loan	4000.00	9.00	2025-04-01

Table 07: FinancialHealth

INSERT INTO FinancialHealth (User_ID, Score, Date) VALUES

(2, 85, '2025-01-31'),

(3, 75, '2025-01-31'),

(4, 90, '2025-01-31'),

(5, 80, '2025-01-31'),

(6, 70, '2025-01-31'),

(7, 95, '2025-01-31'),

(8, 78, '2025-01-31'),

(9, 88, '2025-01-31'),

(10, 83, '2025-01-31'),

(3, 76, '2025-02-01');

Health_ID	User_ID	Score	Date
1	2	85	2025-01-31
2	3	75	2025-01-31
3	4	90	2025-01-31
4	5	80	2025-01-31
5	6	70	2025-01-31
6	7	95	2025-01-31
7	8	78	2025-01-31
8	9	88	2025-01-31
9	10	83	2025-01-31
10	3	76	2025-02-01

Implementation

Step 03: Providing the necessary analytics/analysis based on each requirement.

Requirement 01: Calculating monthly net income for each user. In order to fetch the details for this requirement 3 tables have to be used (user, income and expenses). All the tables have to be joined.

MySQL> **select users.User_name, sum(Income.amount) - sum(Expense.amount) as Net_monthly_Income from users left join income on Users.User_id = Income.User_id left join Expense on Users.User_id = Expense.User_id where Income.date between '2023-10-01' and '2023-10-31' group by Users.User_id;**

Requirment 02: Identify high interest debts

Mysql>**select users.User_name, debts.type, debts.amount, debts.interest_rate from debts inner join users on debts.user_id = users.user_id where debts.interest_rate > 10 order by debts.interest_rate desc;**

Requirement 03: Generate financial health score - using the data generating financial health score. In this requirement the user will calculate the finance health score with use of 4 tables user, income, expenses and debts.

MYSQL> Select users.user_id, (sum(income.amount) - sum(expense.amount) - sum(debts.amount)) / sum(income.amount) * 100 as Score, Now() from users left join Income on users.user_id = income.user_id left join Expense on users.user_id = expense.user_id left join debts on users.user_id = debts.user_id group by users.user_id

Requirment 04: Budget Optimization - In this requirement the users will be provided the average spending in based on expenses

MYSQL> **SELECT users.user_name, expense.category, avg(expense.amount) as AVG_Spending from expense inner join Users on users.user_id = expense.user_id group by users.user_id Expense.category having AVG_Spending > (Select avg(amount) from expense where category = 'Entertainment');**

Requirment 05: Calculating savings growth rate

MYSQL> **select users.User_name, savings.account_type, (savings.amount) (select sum(amount) from savings where user_id = savings.user_id) * 100 as SAVINGS_GROWTH_RATE from savings inner join users on savings.user_id = users.user_id;**

sys

Implementation

Step 04: User Permissions

- Create user 'admin'@'localhost' identified by 'admin123';
- Show grants for 'admin'@'localhost';

```
+-----------------------------------------------------------------------------------------------+
| Grants for admin@localhost                                                                    |
+-----------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `admin`@`localhost`                                                     |
| GRANT ALL PRIVILEGES ON `personalfinanacehealth`.* TO `admin`@`localhost` WITH GRANT OPTION   |
```

- Grant all privileges on personalfinancehealth .* to 'admin'@'localhost' with grant option;
- Select users, host from mysql user;
- create user 'user'@'localhost' identified by'user123';
- grant select on personalfinancehealth.* to 'user'@'localhost';
- Show grants for 'users'@'localhost';

```
+-----------------------------------------------------------------------+
| Grants for users@localhost                                            |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `users`@`localhost`                             |
| GRANT SELECT ON `personalfinanacehealth`.* TO `users`@`localhost`     |
+-----------------------------------------------------------------------+
```

Implementation

Step 05: Transaction

Over the course of the period in the data there are several updates when it comes to income, savings, investments, debts and expenses. In order to update the values, it is safer to update the values inside a transaction. There are 4 transactions to be performed on this project

- Record income and update savings.
- Debt pay off and update savings.
- Transport funds between savings account.
- Record investments and update savings.

Demonstrations:

- **RECORD INCOME AND UPDATE SAVINGS**:

  Mysql> start transaction;

  Mysql> select * from savings;

```
mysql> select * from savings;
+-----------+---------+----------------+---------+------------+
| Saving_id | User_id | Account_type   | Amount  | Date       |
+-----------+---------+----------------+---------+------------+
|         1 |       1 | Emergency Fund | 5000.00 | 2023-10-01 |
|         2 |       1 | Retirement     | 2000.00 | 2023-10-01 |
|         3 |       2 | Emergency Fund | 7000.00 | 2023-10-01 |
|         4 |       2 | Retirement     | 3000.00 | 2023-10-01 |
|         5 |       3 | Emergency Fund | 6000.00 | 2023-10-01 |
|         6 |       3 | Retirement     | 2500.00 | 2023-10-01 |
|         7 |       4 | Emergency Fund | 5500.00 | 2023-10-01 |
|         8 |       4 | Retirement     | 2200.00 | 2023-10-01 |
|         9 |       5 | Emergency Fund | 4800.00 | 2023-10-01 |
|        10 |       5 | Retirement     | 1800.00 | 2023-10-01 |
|        11 |       6 | Emergency Fund | 8000.00 | 2023-10-01 |
|        12 |       6 | Retirement     | 4000.00 | 2023-10-01 |
|        13 |       7 | Emergency Fund | 6500.00 | 2023-10-01 |
|        14 |       7 | Retirement     | 2700.00 | 2023-10-01 |
|        15 |       8 | Emergency Fund | 5800.00 | 2023-10-01 |
|        16 |       8 | Retirement     | 2300.00 | 2023-10-01 |
|        17 |       9 | Emergency Fund | 5200.00 | 2023-10-01 |
|        18 |       9 | Retirement     | 2100.00 | 2023-10-01 |
+-----------+---------+----------------+---------+------------+
```

  Mysql> select * from income;

  Mysql> SAVEPOINT s1;

  Mysql> insert into income (User_id, source, amount, date) values (1, ‘Side-Hustle’, 9000, ‘2023-10-09’):

  Mysql> SAVEPOINT s2;

Mysql> update savings set amount=amount + 9000.00 where User_id =1 and Account_type = ‘Emergency Funds’;

Mysql> COMMIT;

```
mysql> select * from income;
+-----------+---------+-------------+---------+------------+
| Income_id | User_id | Source      | Amount  | Date       |
+-----------+---------+-------------+---------+------------+
|         1 |       1 | Salary      | 3000.00 | 2023-10-01 |
|         2 |       1 | Freelance   |  500.00 | 2023-10-15 |
|         3 |       1 | Bonus       | 1000.00 | 2023-10-30 |
|         4 |       2 | Salary      | 4000.00 | 2023-10-01 |
|         5 |       2 | Dividends   |  200.00 | 2023-10-10 |
|         6 |       2 | Bonus       |  800.00 | 2023-10-25 |
|         7 |       3 | Salary      | 3500.00 | 2023-10-01 |
|         8 |       3 | Freelance   |  600.00 | 2023-10-20 |
|         9 |       3 | Bonus       |  700.00 | 2023-10-31 |
|        10 |       4 | Salary      | 4500.00 | 2023-10-01 |
|        11 |       4 | Dividends   |  300.00 | 2023-10-15 |
|        12 |       4 | Bonus       |  900.00 | 2023-10-30 |
|        13 |       5 | Salary      | 3200.00 | 2023-10-01 |
|        14 |       5 | Freelance   |  400.00 | 2023-10-10 |
|        15 |       5 | Bonus       |  600.00 | 2023-10-25 |
|        16 |       6 | Salary      | 5000.00 | 2023-10-01 |
|        17 |       6 | Dividends   |  250.00 | 2023-10-15 |
|        18 |       6 | Bonus       | 1200.00 | 2023-10-31 |
|        19 |       7 | Salary      | 3800.00 | 2023-10-01 |
|        20 |       7 | Freelance   |  700.00 | 2023-10-20 |
|        21 |       7 | Bonus       |  800.00 | 2023-10-30 |
|        22 |       8 | Salary      | 4200.00 | 2023-10-01 |
|        23 |       8 | Dividends   |  150.00 | 2023-10-10 |
|        24 |       8 | Bonus       | 1000.00 | 2023-10-25 |
|        25 |       9 | Salary      | 3600.00 | 2023-10-01 |
```

- **DEBT PAY OFF AND UPDATE SAVINGS.**

Mysql> start transaction;

Mysql> select * from savings;

```
mysql> select * from savings;
+-----------+---------+----------------+---------+------------+
| Saving_id | User_id | Account_type   | Amount  | Date       |
+-----------+---------+----------------+---------+------------+
|         1 |       1 | Emergency Fund | 6000.00 | 2023-10-01 |
|         2 |       1 | Retirement     | 7000.00 | 2023-10-01 |
|         3 |       2 | Emergency Fund | 7000.00 | 2023-10-01 |
|         4 |       2 | Retirement     | 3000.00 | 2023-10-01 |
|         5 |       3 | Emergency Fund | 6000.00 | 2023-10-01 |
|         6 |       3 | Retirement     | 2500.00 | 2023-10-01 |
|         7 |       4 | Emergency Fund | 5500.00 | 2023-10-01 |
|         8 |       4 | Retirement     | 2200.00 | 2023-10-01 |
|         9 |       5 | Emergency Fund | 4800.00 | 2023-10-01 |
|        10 |       5 | Retirement     | 1800.00 | 2023-10-01 |
|        11 |       6 | Emergency Fund | 8000.00 | 2023-10-01 |
|        12 |       6 | Retirement     | 4000.00 | 2023-10-01 |
|        13 |       7 | Emergency Fund | 6500.00 | 2023-10-01 |
|        14 |       7 | Retirement     | 2700.00 | 2023-10-01 |
|        15 |       8 | Emergency Fund | 5800.00 | 2023-10-01 |
|        16 |       8 | Retirement     | 2300.00 | 2023-10-01 |
|        17 |       9 | Emergency Fund | 5200.00 | 2023-10-01 |
|        18 |       9 | Retirement     | 2100.00 | 2023-10-01 |
+-----------+---------+----------------+---------+------------+
18 rows in set (0.00 sec)
```

Mysql> select * from debts;

Mysql> SAVEPOINT s3;

Mysql> update savings set amount = amount - 1000.00 where user_id = 1 and Account_type = 'emergency fund';

Mysql> SAVEPOINT s4;

Mysql> update debts set amount = amount - 1000.00 where user_id = 1;

Mysql> COMMIT;

```
mysql> select * from debts;
+---------+---------+---------------+----------+---------------+------------+
| Debt_id | User_id | Type          | Amount   | Interest_rate | Due_Date   |
+---------+---------+---------------+----------+---------------+------------+
|       1 |       1 | Credit Card   |  1000.00 | 18.00         | 2024-01-01 |
|       2 |       1 | Student Loan  |  9000.00 | 5.00          | 2025-01-01 |
|       3 |       2 | Car Loan      | 15000.00 | 6.00          | 2024-06-01 |
|       4 |       2 | Personal Loan |  5000.00 | 10.00         | 2023-12-01 |
|       5 |       3 | Credit Card   |  2500.00 | 18.50         | 2024-02-01 |
|       6 |       3 | Student Loan  | 12000.00 | 5.50          | 2025-02-01 |
|       7 |       4 | Car Loan      | 18000.00 | 6.50          | 2024-07-01 |
|       8 |       4 | Personal Loan |  6000.00 | 10.50         | 2023-12-15 |
|       9 |       5 | Credit Card   |  2200.00 | 18.20         | 2024-01-15 |
|      10 |       5 | Student Loan  | 11000.00 | 5.20          | 2025-01-15 |
|      11 |       6 | Car Loan      | 20000.00 | 7.00          | 2024-08-01 |
|      12 |       6 | Personal Loan |  7000.00 | 11.00         | 2023-12-20 |
|      13 |       7 | Credit Card   |  2300.00 | 18.30         | 2024-02-15 |
|      14 |       7 | Student Loan  | 13000.00 | 5.30          | 2025-02-15 |
|      15 |       8 | Car Loan      | 17000.00 | 6.70          | 2024-07-15 |
|      16 |       8 | Personal Loan |  5500.00 | 10.70         | 2023-12-10 |
|      17 |       9 | Credit Card   |  2100.00 | 18.10         | 2024-01-10 |
|      18 |       9 | Student Loan  | 10500.00 | 5.10          | 2025-01-10 |
+---------+---------+---------------+----------+---------------+------------+
18 rows in set (0.00 sec)
```

- **TRANSPORT FUNDS BETWEEN SAVINGS ACCOUNT**

  Mysql> start transaction;

  Mysql> select * from savings;

  Mysql> SAVEPOINT s5;

  Mysql> update savings set amount = amount - 5000.00 where user_id = 1 and Account_type = 'emergency fund';

  Mysql> SAVEPOINT s6;

  Mysql> update savings set amount = amount + 5000.00 where user_id = 1 and Account_type = 'Retirement';

  Mysql> COMMIT;

```
mysql> select * from savings;
+-----------+---------+----------------+---------+------------+
| Saving_id | User_id | Account_type   | Amount  | Date       |
+-----------+---------+----------------+---------+------------+
|         1 |       1 | Emergency Fund | 6000.00 | 2023-10-01 |
|         2 |       1 | Retirement     | 7000.00 | 2023-10-01 |
|         3 |       2 | Emergency Fund | 7000.00 | 2023-10-01 |
|         4 |       2 | Retirement     | 3000.00 | 2023-10-01 |
|         5 |       3 | Emergency Fund | 6000.00 | 2023-10-01 |
|         6 |       3 | Retirement     | 2500.00 | 2023-10-01 |
|         7 |       4 | Emergency Fund | 5500.00 | 2023-10-01 |
|         8 |       4 | Retirement     | 2200.00 | 2023-10-01 |
|         9 |       5 | Emergency Fund | 4800.00 | 2023-10-01 |
|        10 |       5 | Retirement     | 1800.00 | 2023-10-01 |
|        11 |       6 | Emergency Fund | 8000.00 | 2023-10-01 |
|        12 |       6 | Retirement     | 4000.00 | 2023-10-01 |
|        13 |       7 | Emergency Fund | 6500.00 | 2023-10-01 |
|        14 |       7 | Retirement     | 2700.00 | 2023-10-01 |
|        15 |       8 | Emergency Fund | 5800.00 | 2023-10-01 |
|        16 |       8 | Retirement     | 2300.00 | 2023-10-01 |
|        17 |       9 | Emergency Fund | 5200.00 | 2023-10-01 |
|        18 |       9 | Retirement     | 2100.00 | 2023-10-01 |
+-----------+---------+----------------+---------+------------+
18 rows in set (0.00 sec)
```

- **RECORD INVESTMENTS AND UPDATE SAVINGS.**

  Mysql> start transaction;

  Mysql> select * from savings;

  Mysql> select * from investment;

  Mysql> SAVEPOINT s7;

  Mysql> update savings set amount = amount - 2000.00 where user_id = 1 and Account_type = 'Emergency fund';

  Mysql> SAVEPOINT s8;

  Mysql> insert into investments (user_id, type, amount, Return_rate, date) values (1,'Mutual Funds',2000.00, 600, '2023-10-13');

  Mysql> COMMIT;

```
mysql> select * from investments;
ERROR 1146 (42S02): Table 'personalfinancehealth.investments' doesn't exist
mysql> select * from investment;
+---------------+---------+--------------+----------+-------------+------------+
| Investment_id | User_id | Type         | Amount   | Return_rate | Date       |
+---------------+---------+--------------+----------+-------------+------------+
|             1 |       1 | Stocks       | 10000.00 |        8.50 | 2023-10-01 |
|             2 |       1 | Bonds        |  5000.00 |        3.00 | 2023-10-01 |
|             3 |       2 | Mutual Funds | 15000.00 |        6.00 | 2023-10-01 |
|             4 |       2 | Real Estate  | 20000.00 |        5.00 | 2023-10-01 |
|             5 |       3 | Stocks       | 12000.00 |        8.00 | 2023-10-01 |
|             6 |       3 | Bonds        |  6000.00 |        3.50 | 2023-10-01 |
|             7 |       4 | Mutual Funds | 18000.00 |        6.50 | 2023-10-01 |
|             8 |       4 | Real Estate  | 25000.00 |        5.50 | 2023-10-01 |
|             9 |       5 | Stocks       | 11000.00 |        8.20 | 2023-10-01 |
|            10 |       5 | Bonds        |  5500.00 |        3.20 | 2023-10-01 |
|            11 |       6 | Mutual Funds | 20000.00 |        7.00 | 2023-10-01 |
|            12 |       6 | Real Estate  | 30000.00 |        6.00 | 2023-10-01 |
|            13 |       7 | Stocks       | 13000.00 |        8.30 | 2023-10-01 |
|            14 |       7 | Bonds        |  7000.00 |        3.30 | 2023-10-01 |
|            15 |       8 | Mutual Funds | 17000.00 |        6.70 | 2023-10-01 |
|            16 |       8 | Real Estate  | 22000.00 |        5.70 | 2023-10-01 |
|            17 |       9 | Stocks       | 10500.00 |        8.10 | 2023-10-01 |
|            18 |       9 | Bonds        |  5200.00 |        3.10 | 2023-10-01 |
|            19 |       1 | Mutual Funds |  2000.00 |      600.00 | 2023-10-13 |
|            20 |       1 | Mutual Funds |  2000.00 |      600.00 | 2023-10-13 |
+---------------+---------+--------------+----------+-------------+------------+
20 rows in set (0.00 sec)
```

Conclusion:

Using this project the users can maintain a good personal finance score hereby helping them while making future financial decisions and maintain a good track of all their future assets and liabilities. With the use of table such as income, expenses, debts, savings, investments the user generated the financial health table which provided them their score of there personal finance. Many analytics were produced apart from that many other analytics also perform based on the data such as finding the debt to ratio (debt burden analysis), investment performance analysis, monthly expense trends and so forth.

The cod has to be stored in .SQL file. Prepare a report of the findings along with the SQL file and it will be hosted in GitHub.

Future Enhancements:

The project can be updated with some more features which are external in nature compared to sql such as

1. Integrating with Api- To fetch real life data.

2. Data visualization- To create interactive dash boards and reports.

3. Machine learning -Implementing Machine learning models to predict financial health using certain algorithms and provide some recommendations.

Impact of the project:

This project demonstrates proficiency in MySQL and the ability to solve real world problems using data base systems. It is a unique and practical addition for a portfolio and will help the user advance in the field of database systems.