PERSONAL FINANCE AND HEALTH ANALYZER AND OPTIMIZER

This is one of the finance related MySQL projects which focuses on personal finance analysis and optimization of personal finances. This is one of the niches that is highly relevant on today’s world and is rarely used in depth. The project will be designed using MySQL.

This project is a MySQL based system designed to help individual’s finances. 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.

Prerequisites for this project

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

1. MySQL 8.0 Command Line Client
2. MySQL 8.0 Workbench CE
3. MySQL Statements
4. MySQL Operations
5. MySQL Clauses
6. MySQL Constraints
7. MySQL Subqueries
8. MySQL Joints
9. User permissions (GRANT & REVOKE)
10. Transactions

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

Key features of this project

This project will contain several important features of the real-world finance management such as –

1. Income and Expense Tracking: It is used to track monthly income sources and categorize expenses.
2. Savings and Investment Analysis: Monitoring savings account and investment portfolios along with calculating returns.
3. Debt Management: It is used to track debts (credit cards, loans, etc.) and calculate interest payments along with suggesting debt repayment strategies.
4. Financial Health Scoring: It is used to generate a financial health score based on income, expenses, savings and debts. It is used to asses financial stability.
5. Budget Optimization: It is used to suggest budget allocations that are optimal based on previous data.
6. User Permissions: Used to provide role-based access. It is also used to restrict access on sensitive financial data.

Schema

In order to perform this project, there are various parameters required –

1. Database: Create a new database for this project with the project name “Personal\_Finance\_Health”.
2. Tables required:
3. Users – This table should contain user details (user\_id, user\_name, role, password).
4. Income – Used to store income sources of the user (income\_id, user\_id, source, amount, date).
5. Expenses – It is used to store expense details of the user (expense\_id, user\_id, category, amount, date).
6. Savings – Used to store savings details (saving\_id, user\_id, account\_type, amount, date).
7. Investment – Used to store investment details (investment\_id, user\_id, type, amount, return\_rate, date).
8. Debts – Used to store debt details (debt\_id, user\_id, type, amount, interest\_rate, due\_date).
9. Financial\_Health – Used to store financial health scores ( health\_id, user\_id, score, date).
10. Relationships: With the use of constraint keys, the users table must be linked with all the other tables with the use of primary key and foreign key. This is called one to many relationships.
11. Users:
12. Admin – with admin privileges.
13. User – with user privileges.

Implementation

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

**Step 1: Creating database and creating necessary details.**

>show databases;

>CREATE DATABASE Personal\_Finance\_Health;

>use Personal\_Finance\_Health;

>show tables;

>CREATE TABLE USERS(

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_name VARCHAR(100) NOT NULL,

role ENUM(‘admin’ , ’user’) NOT NULL,

password VARCHAR(50) NOT NULL, UNIQUE(user\_name));

> CREATE TABLE INCOME(

income\_id INT AUTO\_INCREMENT,

user\_id INT,

source VARCHAR(100),

amount DECIMAL(10,2) NOT NULL,

date DATE, PRIMARY KEY(income\_id),

FOREIGN KEY(user\_id) REFERENCES USERS(user\_id));

> CREATE TABLE EXPENSE(

expense\_id INT AUTO\_INCREMENT,

user\_id INT,

category VARCHAR(100),

amount DECIMAL(10,2) NOT NULL,

date DATE, PRIMARY KEY(expense\_id),

FOREIGN KEY(user\_id) REFERENCES USERS(user\_id));

> CREATE TABLE SAVINGS(

saving\_id INT AUTO\_INCREMENT,

user\_id INT,

account\_type VARCHAR(100),

amount DECIMAL(10,2) NOT NULL,

date DATE, PRIMARY KEY(saving\_id),

FOREIGN KEY(user\_id) REFERENCES USERS(user\_id));

> CREATE TABLE INVESTMENT(

investment\_id INT AUTO\_INCREMENT,

user\_id INT,

investment\_type VARCHAR(100),

amount DECIMAL(10,2) NOT NULL,

return\_rate DECIMAL(5,2) NOT NULL,

date DATE, PRIMARY KEY(investment\_id),

FOREIGN KEY(user\_id) REFERENCES USERS(user\_id));

> CREATE TABLE DEBTS(

debt\_id INT AUTO\_INCREMENT,

user\_id INT,

debt\_type VARCHAR(100),

amount DECIMAL(10,2) NOT NULL,

interest\_rate DECIMAL(5,2) NOT NULL,

due DATE, PRIMARY KEY(debt\_id),

FOREIGN KEY(user\_id) REFERENCES USERS(user\_id));

> CREATE TABLE FINANCIAL\_HEALTH(

health\_id INT AUTO\_INCREMENT,

user\_id INT,

score DECIMAL(10,2) NOT NULL,

date DATE, PRIMARY KEY(health\_id),

FOREIGN KEY(user\_id) REFERENCES USERS(user\_id));

**Step 2: 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 1 – USERS

There should be 10 users. Among those ten, one should be named as Admin whereas the rest nine will be users.

TABLE 2 – INCOME  
For each user, there are different streams of income along with profit(amount) generated. They have been assigned to a particular user\_id that matches the user\_id from the table USERS.

TABLE 3 – EXPENSE

This table covers the expenses of all nine users and it will classify those expenses into various categories.

TABLE 4 – SAVINGS

This table provides the values for each and every saving medium for each user.

TABLE 5 - INVESTMENT

This table is used to provide the values of each investment done by the users and the return they have received.

TABLE 6 - DEBTS

This table is used to provide values that are related to each user and how much amount they have loaned and at what interest.

**Step 3: Providing the necessary analytics or analysis based on each requirement.**

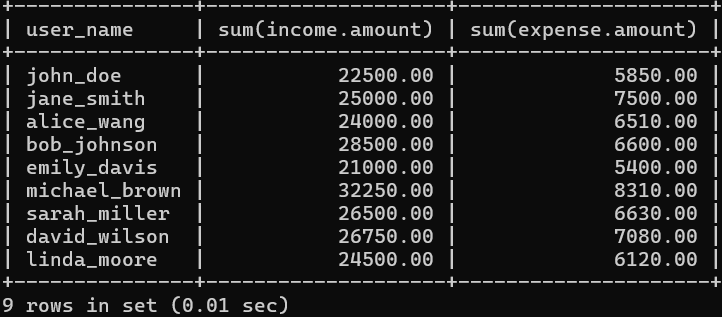
1. Calculating monthly net income for each user: In order to fetch the details for this requirement, three tables have to be used – USERS, INCOME and EXPENSE. All the tables have to be joined.

>select users.user\_name, sum(income.amount), sum(expense.amount) 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;

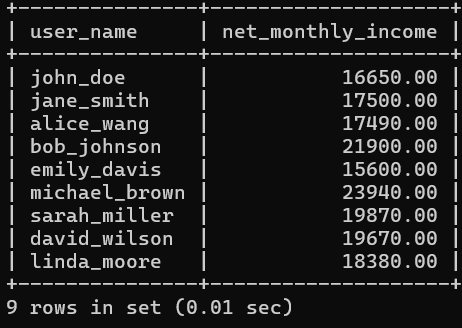


>select users.user\_name, sum(income.amount) - sum(expense.amount) as net\_monthly\_income from users

>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;



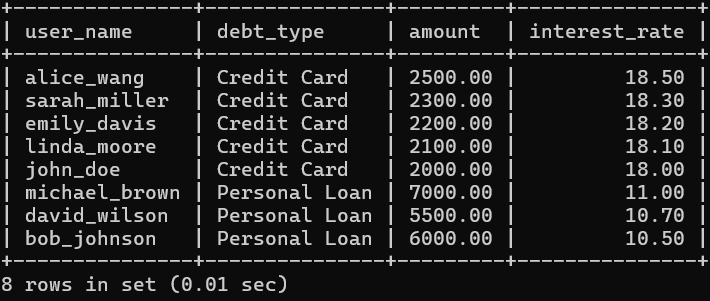
1. Identify high interest debts.

>select users.user\_name, debt\_type, amount, interest\_rate from debt

>inner join users on debt.user\_id = users.user\_id

>where interest\_rate > 10

>order by interest\_rate desc;



1. Generate the financial health score using the data generating financial health score. In this requirement, the user will calculate the finance health score with use of four tables USER, INCOME, EXPENSE and DEBT.

> select users.user\_id,(sum(income.amount) - sum(expense.amount) - sum(debt.amount))/sum(income.amount)\*100

> as score , now()

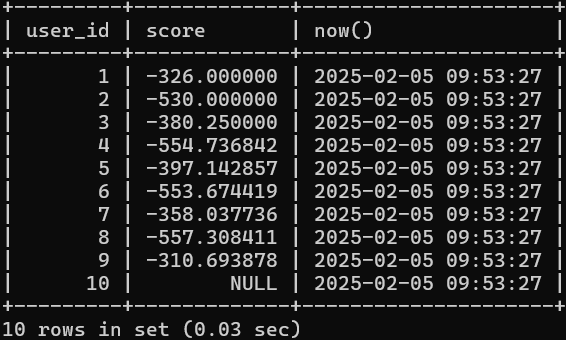
>from users

>left join income on users.user\_id = income.user\_id

>left join expense on users.user\_id = expense.user\_id

>left join debt on users.user\_id = debt.user\_id

>group by users.user\_id;



1. In this requirement, the users will be provided with their average spending based on each expense.

> SELECT users.user\_name, expense.category, avg(expense.amount)

>as avg\_spending

>from expense inner join users on expense.user\_id = users.user\_id

>group by users.user\_id, expense.category

>having avg\_spending>(select avg(amount) from expense where category = 'Entertainment');



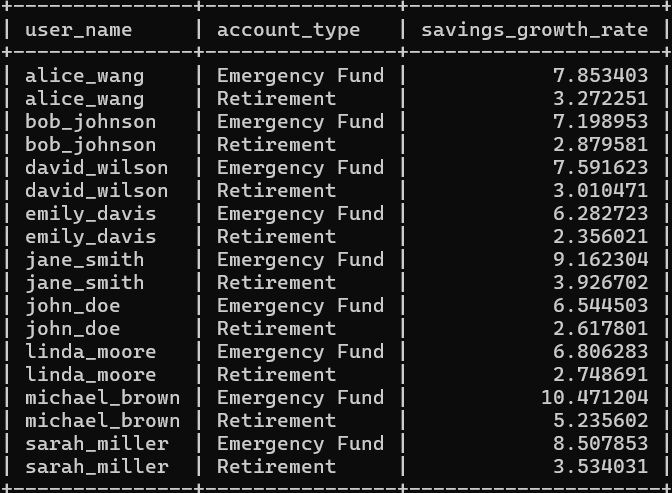
1. Calculating savings growth rate.

>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;



**Step 4: User Permissions.**

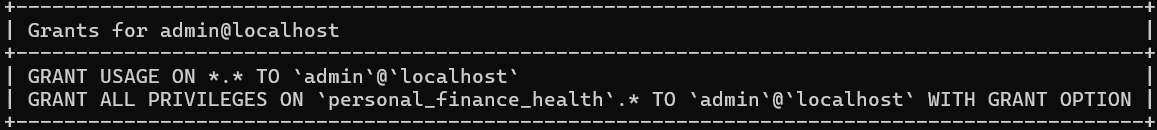
FOR ADMIN-

>create user 'admin'@'localhost' identified by 'admin123';

>show grants for 'admin'@'localhost';

>grant all privileges on Personal\_Finance\_Health.\* to 'admin'@'localhost' with grant option; >show grants for 'admin'@'localhost'

>select user, host from mysql.user;



FOR USERS-

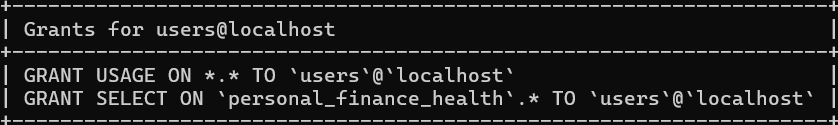
>create user 'users'@'localhost' identified by 'users123';

>show grants for 'users'@'localhost';

>grant select on Personal\_Finance\_Health.\* to 'users'@'localhost';

>show grants for 'users'@'localhost';

>select user, host from mysql.user;



**Step 5: Transactions.**

Over the course of 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 four transactions to be performed on this project.

1. Record income and update savings

> start transaction;

>select \* from savings;

>select \* from income;

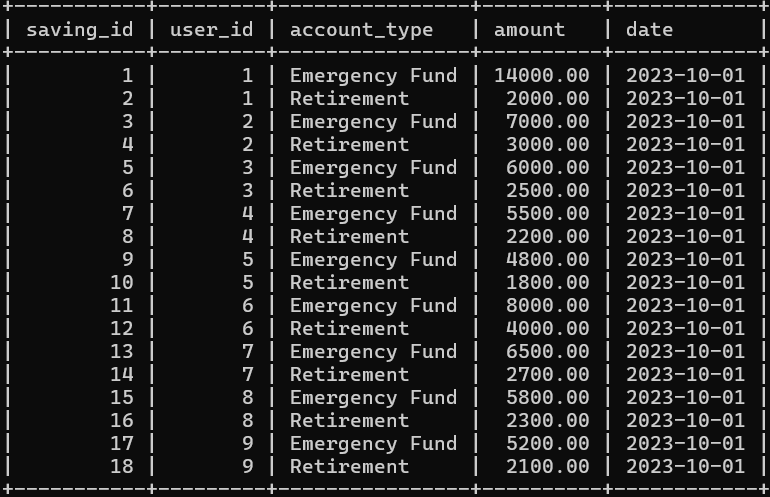
>savepoint s1;

>Insert into income(user\_id,source,amount,date) values(1,'side hustle', 9000.00,'2023-10-11');

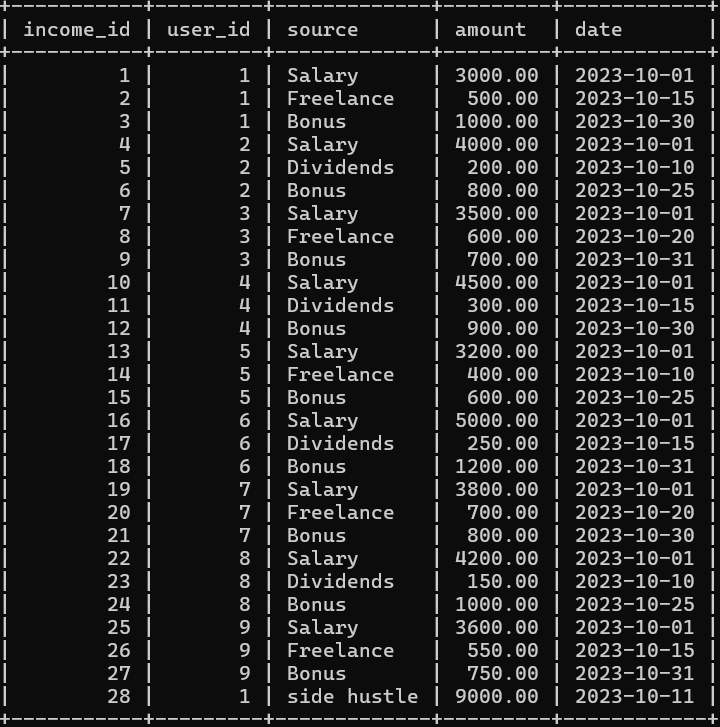
>savepoint s2;

>update savings set amount = amount+9000.00 where user\_id = 1 AND account\_type = "Emergency Fund";

>commit;



Savings table



Investment table

1. Debt payoff and update savings

>start transaction;

>select \* from savings;

>select \* from debt;

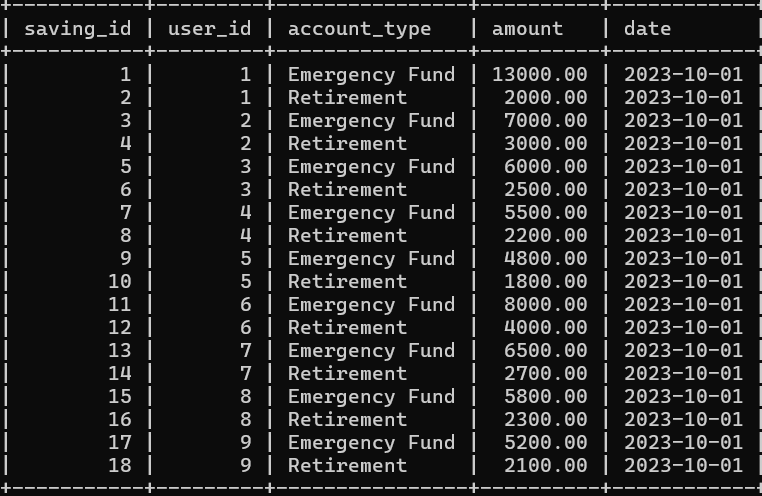
>savepoint s3;

>update savings set amount = amount-1000.00 where user\_id = 1 AND account\_type = "Emergency Fund";

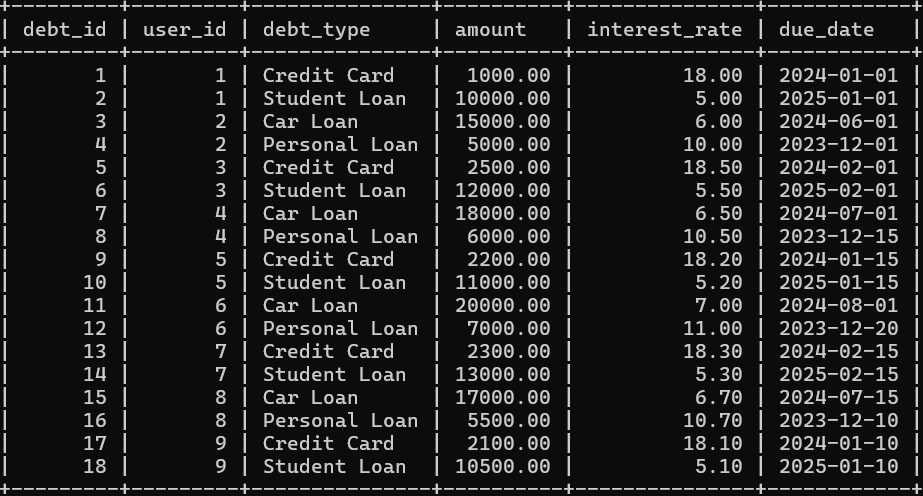
>savepoint s4;

>update debt set amount = amount-1000.00 where debt\_id = 1;

>commit;



Savings table



Debt table

1. Transfer funds between savings account

>start transaction;

>select \* from savings;

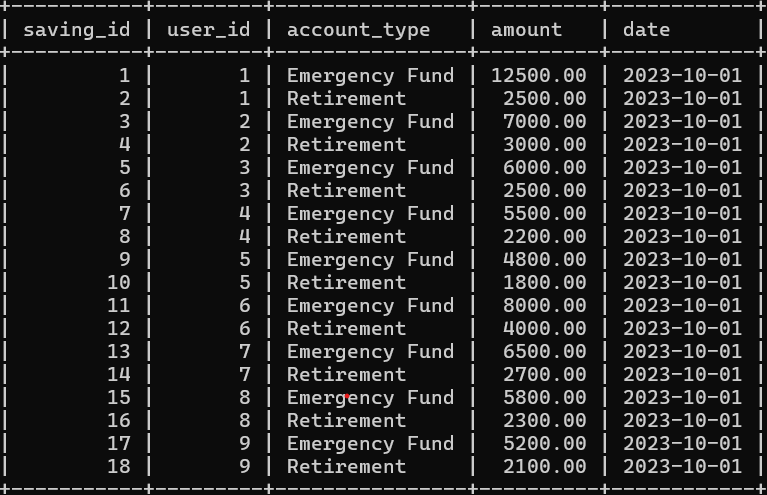
>savepoint s5;

>update savings set amount = amount - 500.00 where user\_id = 1 and account\_type = "Emergency fund";

>savepoint s6;

>update savings set amount = amount+500.00 where user\_id = 1 and account\_type = "Retirement";

>commit;



Savings table

1. Record investments and update savings

>start transaction;

>select \* from savings;

>select \* from investment;

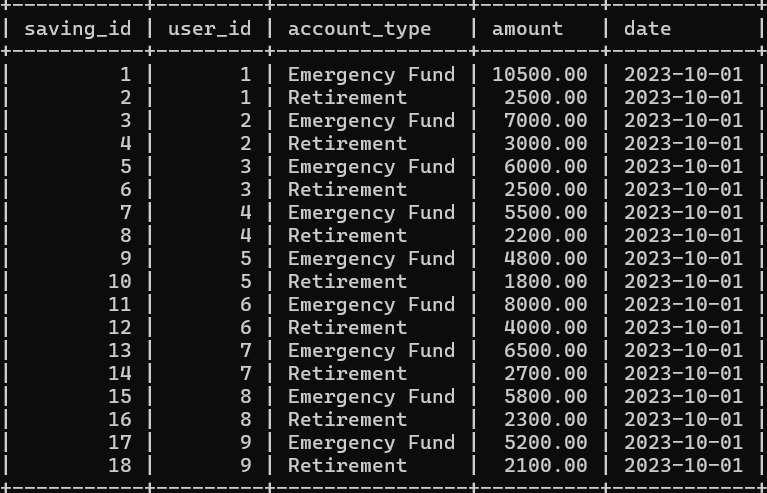
>savepoint s7;

>update savings set amount = amount-2000.00 where user\_id=1 and account\_type = "Emergency fund";

>savepoint s8;

>insert into investment(user\_id, investment\_type, amount, return\_rate, date) values(1,'mutual Funds', 2000.00, 6.00, '2023-10-03');

>commit;



Savings table



Investment table

CONCLUSION

Using this project, the users can maintain a good personal finance score hereby helping them by making future financial decisions and maintain a good track of all their future assets and liabilities. With the use tables such as INCOME, EXPENSE, DEBT, SAVINGS, INVESTMENT, the user generated the financial health table which provided them the scores of their personal finance. Many analytics were produced apart from that many other analytics can be performed based on the data such as finding the debt-to-income ratio (debt burden analysis, investment performance analysis, monthly expense trends and so forth). The code has to be stored in a .SQL file. Prepare a report of the findings along with the SQL file an it will be hosted in GitHub.

FUTURE ENHANCEMENT

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

1. Integrating with API to fetch real life data.
2. Data Visualization: To create interactive dashboards and reports.
3. Machine Learning: Implementing machine learning models to predict financial health using certain algorithms and provide some recommendations.

IMPACT OF PROJECT

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