PERSONAL FINANCE HEALTH ANALYZER AND OPTIMIZER

This projects is a MySQL base system design to help individual and optimize the personal . It will include tracking income expenses savings investments and depts. It will provide actional able insights that will help improving financial wellbeing. The system will use MySQL concepts to generate reports, identify financial trends and suggests optimization strategies.

PREQUSITS  
In order to perform this projects there are some pre-requisites that the user needs to have

1. MySQL8.0 command line client
2. MySQL8.0 Workbench CE
3. MySQL statements
4. MySQL operation
5. MYSQL clauses
6. MYSQL constrains MYSQL sub-queries
7. MYSQL joins
8. User permission (grant and revoke)
9. Transactions

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

KEY FEATURES OF THIS PEOECT

This project will contain several important features of real world finance management such as:

1. Income and expense tracking: it is used to track monthly income sources and categorise expenses
2. Savings and investment Analysis: Monitoring savings accounts and investment porfolies along with calculating returns.
3. Dept Management: It is used to trap depts ( Credit Cards, Loans, etc) and calculate intrest payments along with suggesting dept repayment strategies.
4. Finical Health Scoring: it is used to generate a finical health score based on imcome expenses, savings and depts. It is used to asses finical stabilities.
5. Budget Optimization: It is used to suggest budget allocations that are optimal based on previous data.
6. User permisions: Used to provide roll based access. It is also used to restrict access on sensitive finical 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\_FINIACE\_HEALTH”
2. Tables requried: This table should contain user details ( User\_id, username, roll, password).
3. 2nd table income: used to store income sources of user ( income\_id, user\_id, source, amount, date).
4. 3rd table expenses: It store expense detail of the user ( expense\_id, user\_id, category, amount, date).
5. 4th table savings: Used to store savings details ( savings\_id, user\_id, account\_type, amount, date).
6. Investment : Used to store investment details ( investment\_id, user\_id, type, amount, return\_rate, date).
7. Debts: Used to store dept details ( deb.intrest\_id, user\_id, type, amount, intrest\_rate, due\_date).
8. FINICAL\_HEALTH: Used to store finical health score ( health\_id, user\_id , score, date).

SCHEMA RELATIONSHIPS

With the use of constrain keys, the user table must be linked with all the other table with the use of primary key and foregin key, this is called one-to many relationship.

User -1) Admin - with admin privileged

2) User – with select only data privilage

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 the database and the nessesay tables.

Step 2: inserting values more than 100 rows in all 7 coloums to perform the analysis. Apart from the auto increment coloums all the coloums have to be filled with data.

Table 1 Users: There shoukd be 10 user, among those 10 one should me named as admin where as the rest 9 will be users.

Table 2 Income: For each user there are different source of income along with profit generated they have been assigned to a particular user id that matches the user ids from the table users.

Table 3 Expenses: This table covers the expenses of all 9 users and it will classify those expenses into various categories.

Tables 4 Savings: This table provides the values for each and every saving medium for each user.

Table 5 Investment: This tables used to provide value of each investment done by the user and the return they have received .

Table 6 Debts: This table is used to provide values that are loan information of each user and how much amount they have loaned and at what interest.

### REQUIREMENTS:

Calculating monthly net income for each user.in order to fetch details for this requirement 3 tables have to be used – user, income and expenses - all the tables have to be joined.

• Show database. Show databases;

• Create Database: CREATE DATABASE Personal\_Finance\_Health;

USE Personal\_Finance\_Health;

• Use database. use personal\_finance\_health;

• 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 );

• 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);

• 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 );

• 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 );

• 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 );

• 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 );

• 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 );

THE VALUES:

• Insert Users 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');

• Insert Income 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');

• Insert Expenses 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');

• Insert Savings 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');

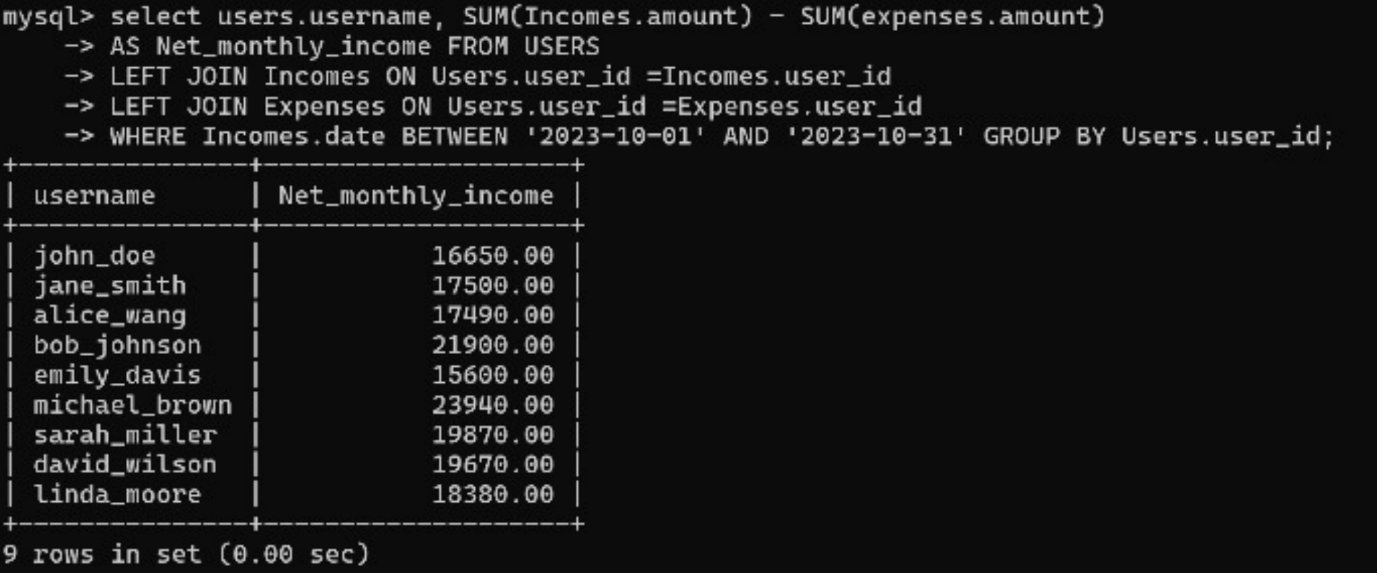
• Insert Investments INSERT INTO Investments (User\_ID, Type, Amount, Return\_rate, Date) VALUES (2, 'Stocks', 5000, 8.5, '2025-01-09'), (3, 'Bonds', 3000, 6.2, '2025-01-16'), (4, 'Real Estate', 10000, 10.0, '2025-01-22'), (5, 'Mutual Funds', 7000, 7.5, '2025-01-27'), (6, 'Stocks', 6000, 9.0, '2025-02-01'), (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');

• Insert Debts 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');

• Insert Financial Health 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');

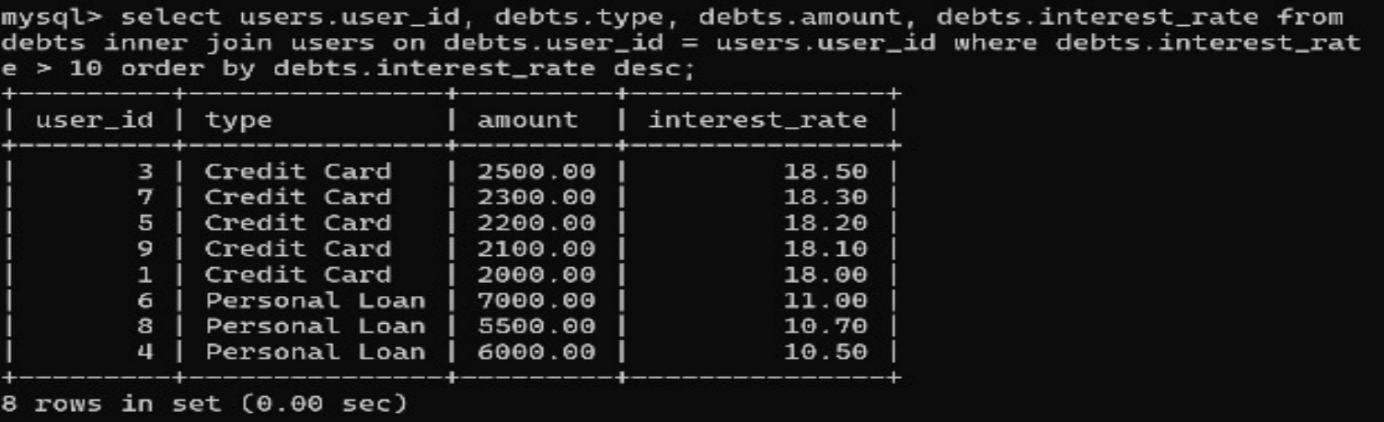
The Net Monthly Income:

• Net monthly Income SELECT Users.Username, (COALESCE(SUM(Income.Amount), 0) - COALESCE(SUM(Expenses.Amount), 0)) AS Net\_Monthly\_Income FROM Users LEFT JOIN Income ON Users.User\_ID = Income.User\_ID AND Income.Date BETWEEN '2025-01-01' AND '2025-01-31' LEFT JOIN Expenses ON Users.User\_ID = Expenses.User\_ID AND Expenses.Date BETWEEN '2025-01-01' AND '2025-01-31' GROUP BY Users.User\_ID, Users.Username;



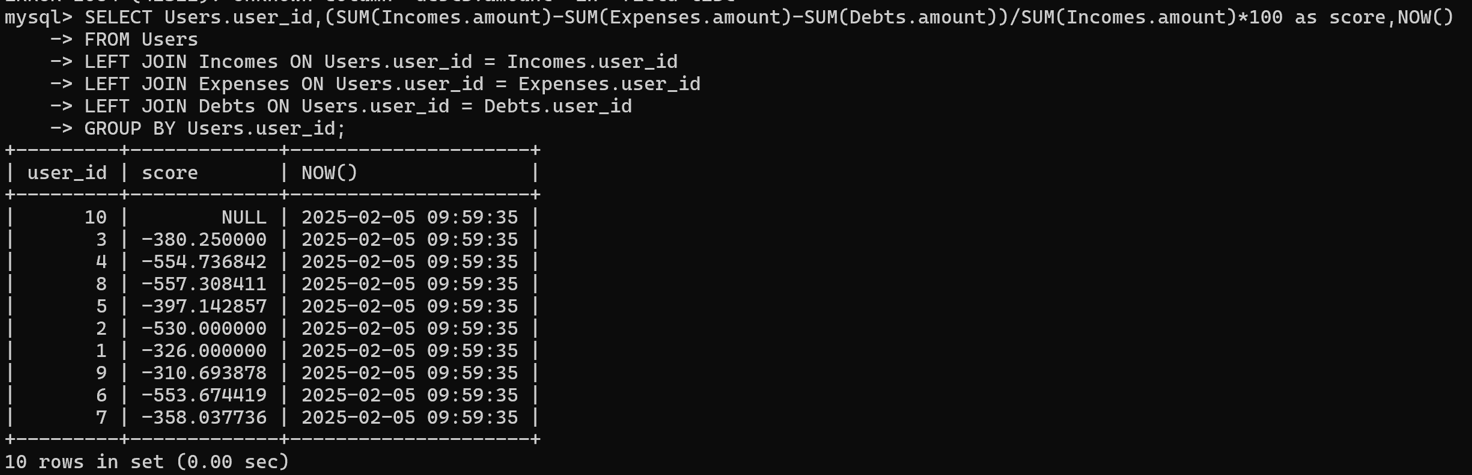
Identifying high interest debts:

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;



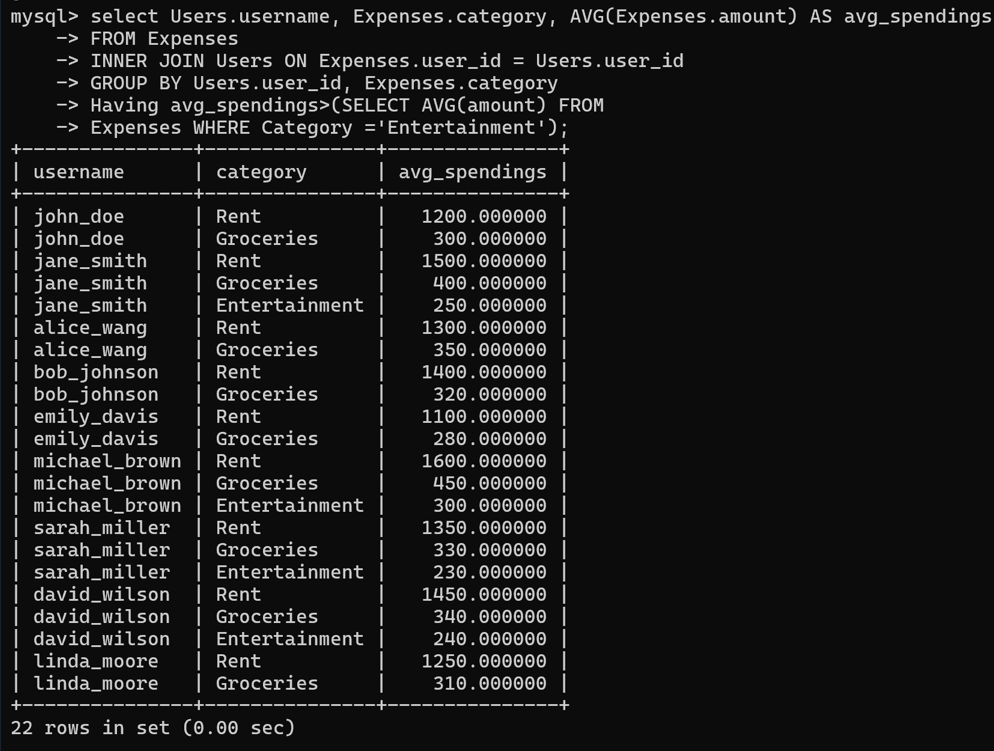
By Using the data in this requirement the user will calculate financial health score with the help of 4 table – user income, expense and debts.

* 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



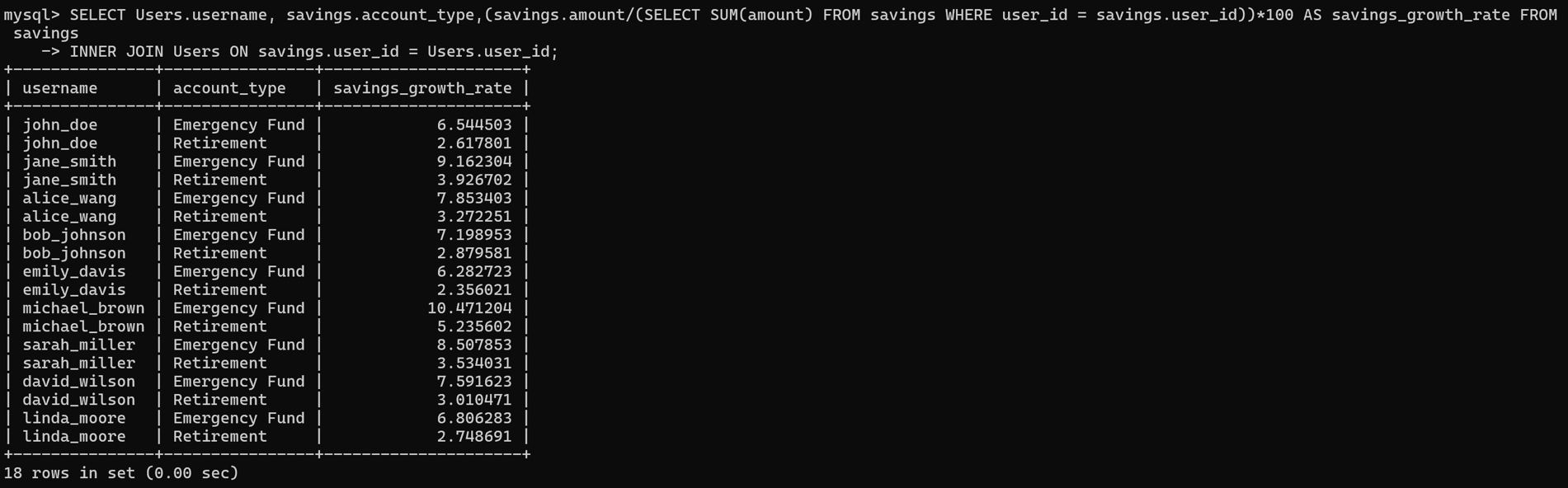
Budget Optimization From the salary Of the users:

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’)



**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;



**USER PERMISIONS:**

* Create user ‘admin’@’localhost’ identified by ‘admin123’;
* Show grants for ‘admin’@’localhost’;
* 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';

**TRANSACTIONS**

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

1. Record income and update savings.
2. Debt pay off and update savings.
3. Transport funds between savings account.
4. Record investments and update savings.

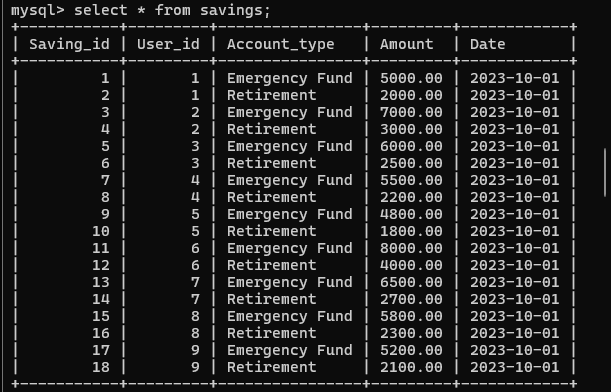
**Demonstrations:**

**RECORD INCOME AND UPDATE SAVINGS:**

Mysql> start transaction;

Mysql> select \* from savings;

Mysql> select \* from income



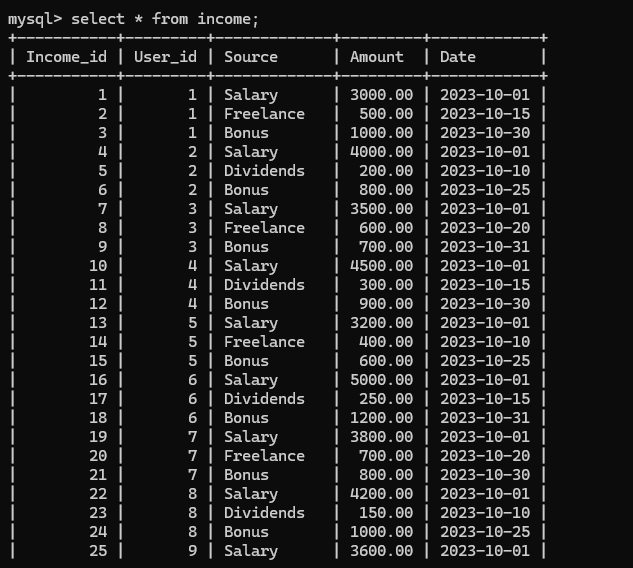
Mysql> savepoint s1;

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;

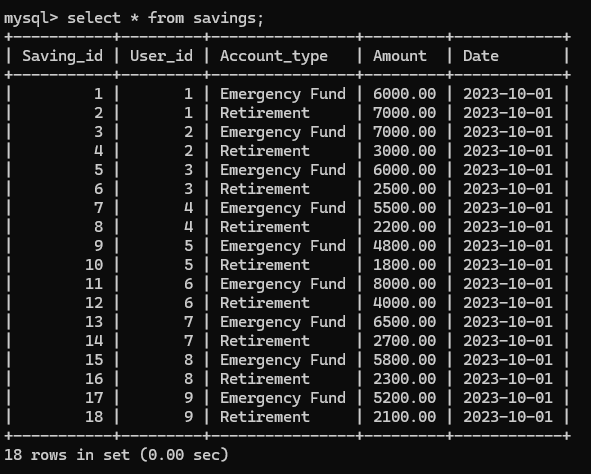


**DEBT PAY OFF AND UPDATE SAVINGS.**

Mysql> start transaction;

Mysql> select \* from savings;

Mysql> select \* from debts;



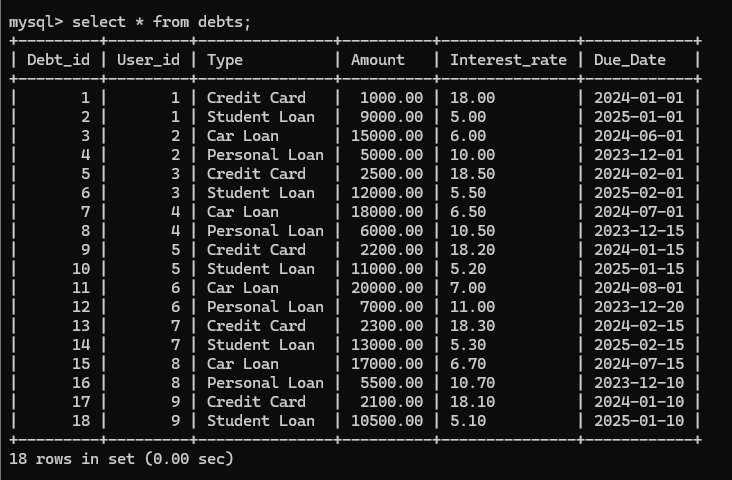
Mysql> savepoint s3;

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;



**TRANSPORT FUNDS BETWEEN SAVINGS ACCOUNT**

Mysql> start transaction;

Mysql> select \* from savings;

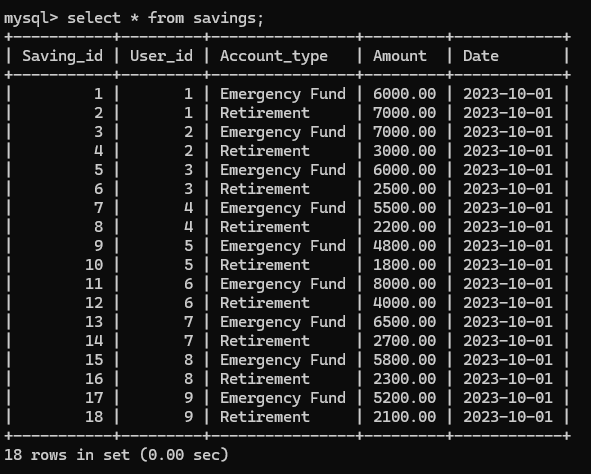
Mysql> savepoint s5;

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;



**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;

