Personal Finance Health Analyser and Optimizer

This project is MySQL base region design to help and analyse and optimize their personal finances. It will include tracking income, expenses, savings, investments and debts. It will provide actionable insights that will help improve financial wellbeing. The system will use MySQL concepts to generate reports, identify financial trends and suggest optimization strategies

Perquisites

In order to perform this project, there are some prerequisites
1.Mysql 8.0 command 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 and Revoke)
10.Transactions
These are the concepts or the technologies which the user needs to be fundamentally strong.

Key features of this project

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 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 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 base access. It is also used to restrict access on sensitive financial data.

Schema1

In order to perform this project, there are various parameters required
Database: Create a new database for this project with the project name PersonalFinanceHealth.

Schema2: Tables Required

Table1: Users
This table should contain user details (user_Id, username, role, password).
Table2: Income

Used to store income sources of the user (income_id, user_id , source, amount, date).
Table3: Expenses
It is used to store expense details of the user (expense_id, user_id, category, amount, date).
Table4: Savings
Used to store savings details (saving_id, user_id, accouut_type, amount, date).
Table5: Investment details
Used to store investment details (investment_id int auto_increment primary key, user_id, type, amount decimal (10,2), return_rate decimal (5,2), date date).
Table6: Debts
Used to debt details (debt_id int auto increment primary, user_id int foreign key , type varchar(50), amount decimal (10,2), intrest_rate decimal(5,2), due_date date).
Table7: Financial health
Used to store financial scores (health_id int auto_increment primary, user_id int, score decimal (5,2), date date).

Schema3: 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.

Schema4: users
1.admin: with admin privilege (all privilege).
2.user: with user privilege (Select).

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

Step1: Creating database and creating the necessary tables
1.Cerating Database
create database PersonalFinanceHealth;

2.Creating a table users
```
create table Users (user_id int auto_increment primary key,
username varchar (100) not null,
role Enum('admin','user') not null,
password varchar (50) not null,
unique(username)
 );
```

3.Creating a table Income
```
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)
```

```
 );
```

4.create table expenses

```
create table expenses (
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)
 );
```

5.Creating table savings

```
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)
 );
```

6.Creating table investment_details

```
create table investment_details(
investment_id int primary key auto_increment,
user_id int,
amount decimal (10,2),
return_rate decimal (5,2),
date date,
foreign key(user_id) references
users(user_id),
Type varchar (50)
);
```

7.Creating table debt_details

```
create table debt_details(
debt_id int primary key auto_increment,
user_id int,
type varchar(50),
amount decimal(10,2),
intrest_rate decimal(5,2),
due_date date,
foreign key(user_id) references
users(user_id)
);
```

8. creating table finacial_scores

```
create table finacial_scores(
health_id int primary key auto_increment,
user_id int,
score decimal(5,2),
date date,
foreign key(user_id) references
users(user_id)
);
```

Step2: Inserting values more than hundred rows in all seven tables to perform the analysis.

Apart from the auto increment columns all the columns have to be filled with data.

Table1 users

There should be 10 users, among those 10, one should be named as admin where as the rest 9 will be users.

```
INSERT INTO Users (username, role, password) VALUES
('john_doe', 'user', 'hashed_password_123'),
('jane_smith', 'user', 'hashed_password_456'),
('alice_wang', 'user', 'hashed_password_789'),
('bob_johnson', 'user', 'hashed_password_101'),
('emily_davis', 'user', 'hashed_password_112'),
('michael_brown', 'user', 'hashed_password_131'),
('sarah_miller', 'user', 'hashed_password_415'),
('david_wilson', 'user', 'hashed_password_161'),
('linda_moore', 'user', 'hashed_password_718'),
('admin_user', 'admin', 'hashed_password_919');
```

Table2: Income

For each user there are different streams of income along with the profit or amount generated. They have been assigned to a particular user id that matches the user ids from the table users.

```
INSERT INTO Income (user_id, source, amount, date) VALUES
(1, 'Salary', 3000.00, '2023-10-01'),
(1, 'Freelance', 500.00, '2023-10-15'),
(1, 'Bonus', 1000.00, '2023-10-30'),
(2, 'Salary', 4000.00, '2023-10-01'),
(2, 'Dividends', 200.00, '2023-10-10'),
(2, 'Bonus', 800.00, '2023-10-25'),
(3, 'Salary', 3500.00, '2023-10-01'),
(3, 'Freelance', 600.00, '2023-10-20'),
(3, 'Bonus', 700.00, '2023-10-31'),
(4, 'Salary', 4500.00, '2023-10-01'),
(4, 'Dividends', 300.00, '2023-10-15'),
(4, 'Bonus', 900.00, '2023-10-30'),
(5, 'Salary', 3200.00, '2023-10-01'),
(5, 'Freelance', 400.00, '2023-10-10'),
(5, 'Bonus', 600.00, '2023-10-25'),
(6, 'Salary', 5000.00, '2023-10-01'),
(6, 'Dividends', 250.00, '2023-10-15'),
(6, 'Bonus', 1200.00, '2023-10-31'),
(7, 'Salary', 3800.00, '2023-10-01'),
(7, 'Freelance', 700.00, '2023-10-20'),
(7, 'Bonus', 800.00, '2023-10-30'),
(8, 'Salary', 4200.00, '2023-10-01'),
(8, 'Dividends', 150.00, '2023-10-10'),
(8, 'Bonus', 1000.00, '2023-10-25'),
(9, 'Salary', 3600.00, '2023-10-01'),
(9, 'Freelance', 550.00, '2023-10-15'),
(9, 'Bonus', 750.00, '2023-10-31');
```

Table3: Expenses

This table covers the expenses of all 9 users and it will classify those expenses into categories.

```
INSERT INTO Expenses (user_id, category, amount, date) VALUES
-- John Doe
(1, 'Rent', 1200.00, '2023-10-01'),
(1, 'Groceries', 300.00, '2023-10-05'),
(1, 'Utilities', 150.00, '2023-10-07'),
(1, 'Entertainment', 200.00, '2023-10-10'),
(1, 'Transportation', 100.00, '2023-10-15'),
-- Jane Smith
(2, 'Rent', 1500.00, '2023-10-01'),
(2, 'Groceries', 400.00, '2023-10-05'),
(2, 'Utilities', 200.00, '2023-10-07'),
(2, 'Entertainment', 250.00, '2023-10-10'),
(2, 'Transportation', 150.00, '2023-10-15'),
-- Alice Wang
(3, 'Rent', 1300.00, '2023-10-01'),
(3, 'Groceries', 350.00, '2023-10-05'),
(3, 'Utilities', 180.00, '2023-10-07'),
(3, 'Entertainment', 220.00, '2023-10-10'),
(3, 'Transportation', 120.00, '2023-10-15'),
-- Bob Johnson
(4, 'Rent', 1400.00, '2023-10-01'),
(4, 'Groceries', 320.00, '2023-10-05'),
(4, 'Utilities', 160.00, '2023-10-07'),
(4, 'Entertainment', 210.00, '2023-10-10'),
(4, 'Transportation', 110.00, '2023-10-15'),
-- Emily Davis
(5, 'Rent', 1100.00, '2023-10-01'),
(5, 'Groceries', 280.00, '2023-10-05'),
(5, 'Utilities', 140.00, '2023-10-07'),
(5, 'Entertainment', 190.00, '2023-10-10'),
(5, 'Transportation', 90.00, '2023-10-15'),
-- Michael Brown
(6, 'Rent', 1600.00, '2023-10-01'),
(6, 'Groceries', 450.00, '2023-10-05'),
(6, 'Utilities', 220.00, '2023-10-07'),
(6, 'Entertainment', 300.00, '2023-10-10'),
(6, 'Transportation', 200.00, '2023-10-15'),
-- Sarah Miller
(7, 'Rent', 1350.00, '2023-10-01'),
(7, 'Groceries', 330.00, '2023-10-05'),
(7, 'Utilities', 170.00, '2023-10-07'),
(7, 'Entertainment', 230.00, '2023-10-10'),
(7, 'Transportation', 130.00, '2023-10-15'),
-- David Wilson
(8, 'Rent', 1450.00, '2023-10-01'),
(8, 'Groceries', 340.00, '2023-10-05'),
(8, 'Utilities', 190.00, '2023-10-07'),
(8, 'Entertainment', 240.00, '2023-10-10'),
(8, 'Transportation', 140.00, '2023-10-15'),
-- Linda Moore
(9, 'Rent', 1250.00, '2023-10-01'),
(9, 'Groceries', 310.00, '2023-10-05'),
(9, 'Utilities', 160.00, '2023-10-07'),
(9, 'Entertainment', 210.00, '2023-10-10'),
(9, 'Transportation', 110.00, '2023-10-15');
```

Table4: Savings

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

```
INSERT INTO Savings (user_id, account_type, amount, date) VALUES
-- John Doe
(1, 'Emergency Fund', 5000.00, '2023-10-01'),
(1, 'Retirement', 2000.00, '2023-10-01'),
-- Jane Smith
(2, 'Emergency Fund', 7000.00, '2023-10-01'),
(2, 'Retirement', 3000.00, '2023-10-01'),
-- Alice Wang
(3, 'Emergency Fund', 6000.00, '2023-10-01'),
(3, 'Retirement', 2500.00, '2023-10-01'),
-- Bob Johnson
(4, 'Emergency Fund', 5500.00, '2023-10-01'),
(4, 'Retirement', 2200.00, '2023-10-01'),
-- Emily Davis
(5, 'Emergency Fund', 4800.00, '2023-10-01'),
(5, 'Retirement', 1800.00, '2023-10-01'),
-- Michael Brown
(6, 'Emergency Fund', 8000.00, '2023-10-01'),
(6, 'Retirement', 4000.00, '2023-10-01'),
-- Sarah Miller
```

```
(7, 'Emergency Fund', 6500.00, '2023-10-01'),
(7, 'Retirement', 2700.00, '2023-10-01'),
-- David Wilson
(8, 'Emergency Fund', 5800.00, '2023-10-01'),
(8, 'Retirement', 2300.00, '2023-10-01'),
-- Linda Moore
(9, 'Emergency Fund', 5200.00, '2023-10-01'),
(9, 'Retirement', 2100.00, '2023-10-01');
```

Table5: Investments
This table is used to provide the values of each investment done by the user and the return they have received.

```
INSERT INTO Investment_details (user_id, type, amount, return_rate, date) VALUES
 John Doe
(1, 'Stocks', 10000.00, 8.50, '2023-10-01'),
(1, 'Bonds', 5000.00, 3.00, '2023-10-01'),
Jane Smith
(2, 'Mutual Funds', 15000.00, 6.00, '2023-10-01'),
(2, 'Real Estate', 20000.00, 5.00, '2023-10-01'),
Alice Wang
(3, 'Stocks', 12000.00, 8.00, '2023-10-01'),
(3, 'Bonds', 6000.00, 3.50, '2023-10-01'),
Bob Johnson
(4, 'Mutual Funds', 18000.00, 6.50, '2023-10-01'),
(4, 'Real Estate', 25000.00, 5.50, '2023-10-01'),
Emily Davis
(5, 'Stocks', 11000.00, 8.20, '2023-10-01'),
(5, 'Bonds', 5500.00, 3.20, '2023-10-01'),
Michael Brown
(6, 'Mutual Funds', 20000.00, 7.00, '2023-10-01'),
(6, 'Real Estate', 30000.00, 6.00, '2023-10-01'),
Sarah Miller
(7, 'Stocks', 13000.00, 8.30, '2023-10-01'),
(7, 'Bonds', 7000.00, 3.30, '2023-10-01'),
David Wilson
(8, 'Mutual Funds', 17000.00, 6.70, '2023-10-01'),
(8, 'Real Estate', 22000.00, 5.70, '2023-10-01'),
Linda Moore
(9, 'Stocks', 10500.00, 8.10, '2023-10-01'),
(9, 'Bonds', 5200.00, 3.10, '2023-10-01');
```

Table6: Debts
This table is used to provide values that are related to the loan information of each user and how much amount they have loaned and at what interest.

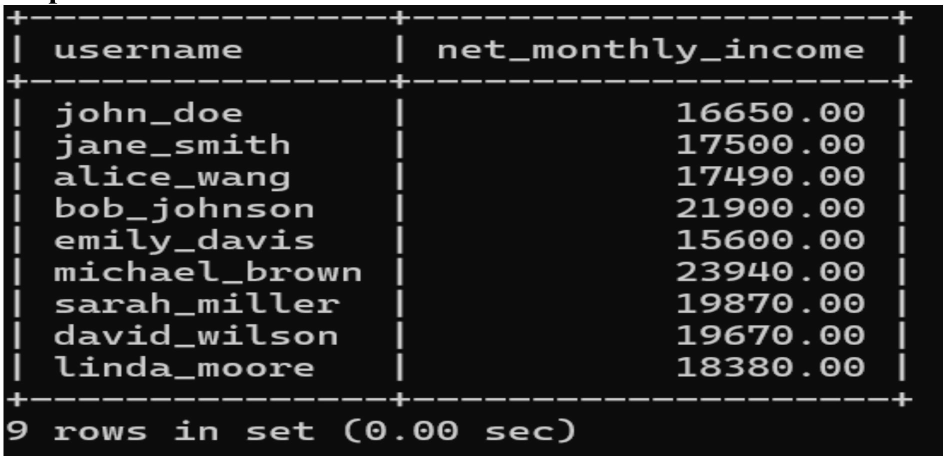
```
INSERT INTO Debt_details (user_id, type, amount, interest_rate, due_date) VALUES
-- John Doe
(1, 'Credit Card', 2000.00, 18.00, '2024-01-01'),
(1, 'Student Loan', 10000.00, 5.00, '2025-01-01'),
-- Jane Smith
(2, 'Car Loan', 15000.00, 6.00, '2024-06-01'),
(2, 'Personal Loan', 5000.00, 10.00, '2023-12-01'),
-- Alice Wang
(3, 'Credit Card', 2500.00, 18.50, '2024-02-01'),
(3, 'Student Loan', 12000.00, 5.50, '2025-02-01'),
-- Bob Johnson
(4, 'Car Loan', 18000.00, 6.50, '2024-07-01'),
(4, 'Personal Loan', 6000.00, 10.50, '2023-12-15'),
-- Emily Davis
(5, 'Credit Card', 2200.00, 18.20, '2024-01-15'),
(5, 'Student Loan', 11000.00, 5.20, '2025-01-15'),
-- Michael Brown
(6, 'Car Loan', 20000.00, 7.00, '2024-08-01'),
(6, 'Personal Loan', 7000.00, 11.00, '2023-12-20'),
-- Sarah Miller
(7, 'Credit Card', 2300.00, 18.30, '2024-02-15'),
(7, 'Student Loan', 13000.00, 5.30, '2025-02-15'),
-- David Wilson
(8, 'Car Loan', 17000.00, 6.70, '2024-07-15'),
(8, 'Personal Loan', 5500.00, 10.70, '2023-12-10'),
-- Linda Moore
(9, 'Credit Card', 2100.00, 18.10, '2024-01-10'),
(9, 'Student Loan', 10500.00, 5.10, '2025-01-10');
```

Step3: Providing the necessary analytics or analysis base on each requirement

Requirement1- Calculation monthly net income for each user. In order to fetch the details for this requirement three tables have to be used users, income, and expenses, all the tables have to be joined.

```
select users.username, sum(income.amount)-sum(expenses.amount) as net_monthly_income from users
left join income on users.user_id =income.user_id
left join expenses on users.user_id =expenses.user_id
where income.date between '2023-10-01' and '2023-10-31'
group by users.user_id;
```

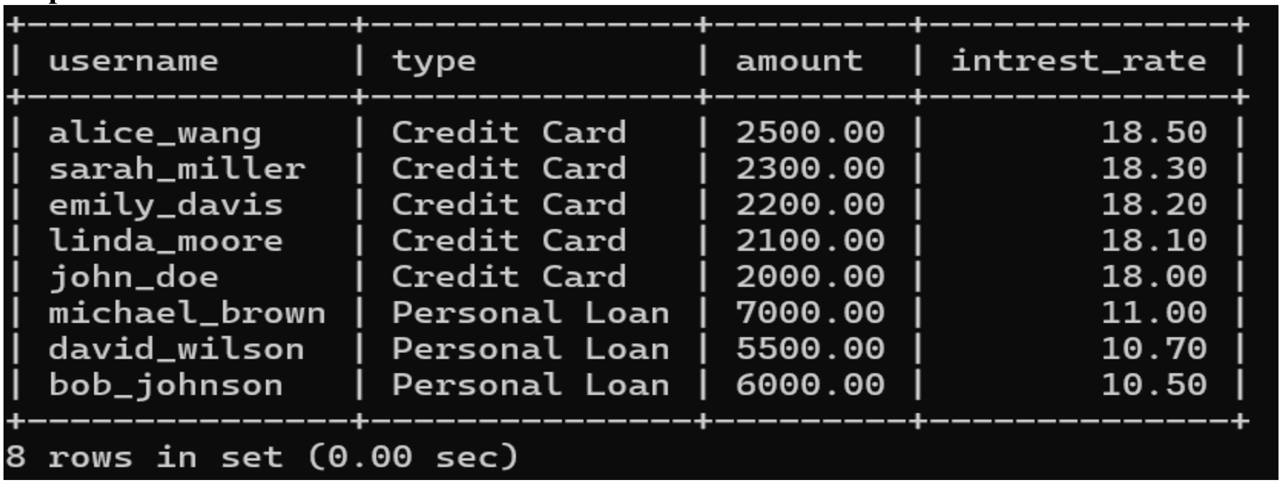
output:



```
+---------------+--------------------+
| username      | net_monthly_income |
+---------------+--------------------+
| john_doe      |           16650.00 |
| jane_smith    |           17500.00 |
| alice_wang    |           17490.00 |
| bob_johnson   |           21900.00 |
| emily_davis   |           15600.00 |
| michael_brown |           23940.00 |
| sarah_miller  |           19870.00 |
| david_wilson  |           19670.00 |
| linda_moore   |           18380.00 |
+---------------+--------------------+
9 rows in set (0.00 sec)
```

Requirement 2- Identifying high interest debts.
select users.username, debt_details.type,debt_details.amount,debt_details.intrest_rate from debt_details
inner join users on debt_details.user_id=users.user_id
where debt_details.intrest_rate >10
order by debt_details.intrest_rate desc;

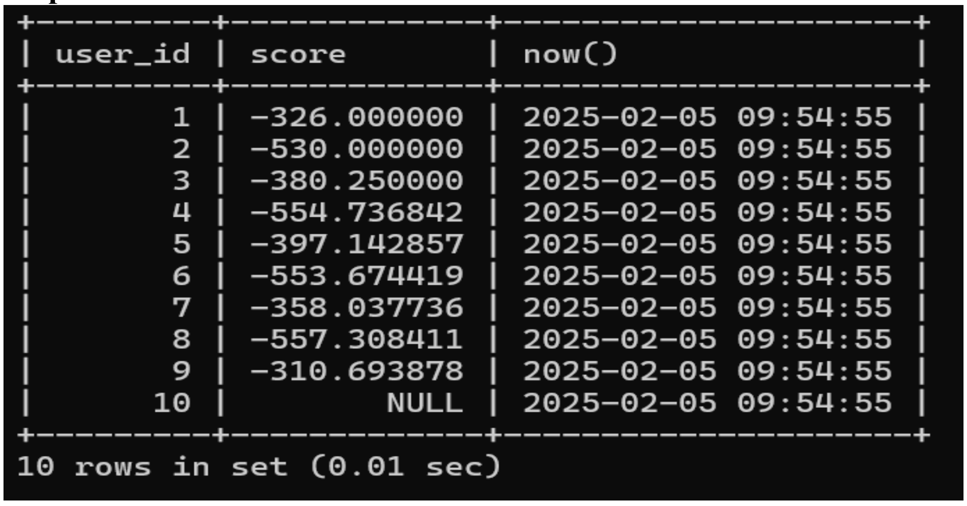
output:



```
+---------------+---------------+---------+---------------+
| username      | type          | amount  | intrest_rate  |
+---------------+---------------+---------+---------------+
| alice_wang    | Credit Card   | 2500.00 |         18.50 |
| sarah_miller  | Credit Card   | 2300.00 |         18.30 |
| emily_davis   | Credit Card   | 2200.00 |         18.20 |
| linda_moore   | Credit Card   | 2100.00 |         18.10 |
| john_doe      | Credit Card   | 2000.00 |         18.00 |
| michael_brown | Personal Loan | 7000.00 |         11.00 |
| david_wilson  | Personal Loan | 5500.00 |         10.70 |
| bob_johnson   | Personal Loan | 6000.00 |         10.50 |
+---------------+---------------+---------+---------------+
8 rows in set (0.00 sec)
```

Requirement 3: Generating financial health score.
In this requirement the user will generate the financial health score with help of four tables users, income, expenses and debts
select users.user_id,(sum(income.amount)-sum(expenses.amount)-sum(debt_details.amount))/sum(Income.amount)*100 as score, now()
from users
left join income on users.user_id=income.user_id
left join expenses on users.user_id=expenses.user_id
left join debt_details on users.user_id=debt_details.user_id
group by users.user_id;

output:



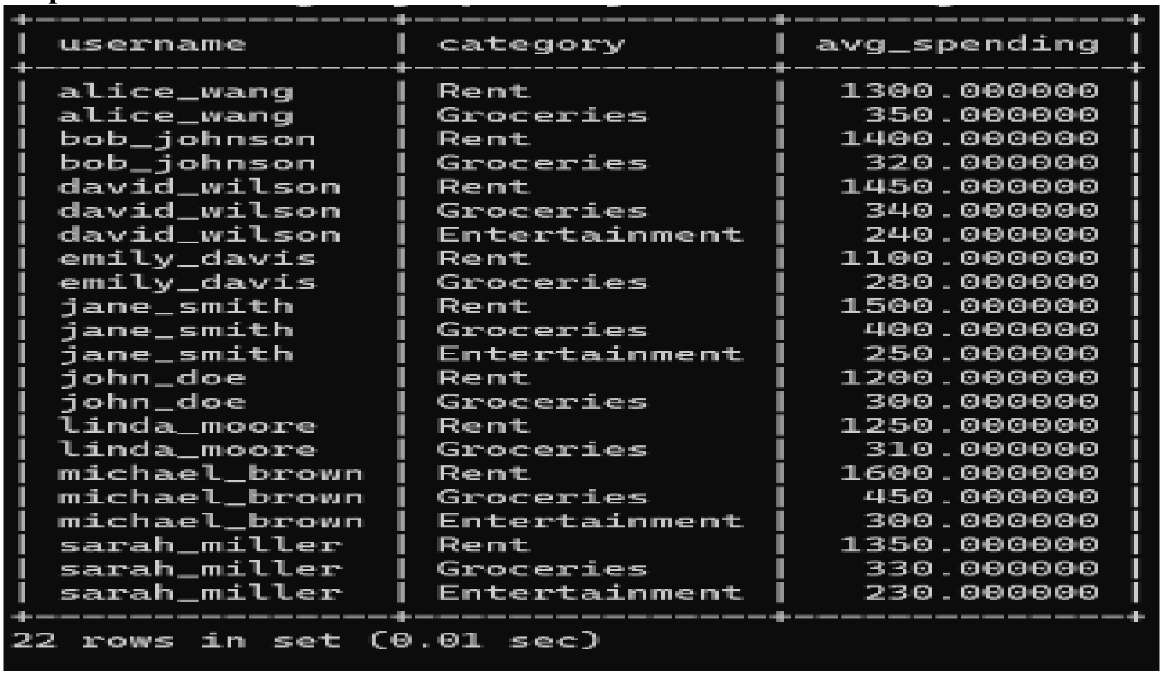
```
+---------+-------------+---------------------+
| user_id | score       | now()               |
+---------+-------------+---------------------+
|       1 | -326.000000 | 2025-02-05 09:54:55 |
|       2 | -530.000000 | 2025-02-05 09:54:55 |
|       3 | -380.250000 | 2025-02-05 09:54:55 |
|       4 | -554.736842 | 2025-02-05 09:54:55 |
|       5 | -397.142857 | 2025-02-05 09:54:55 |
|       6 | -553.674419 | 2025-02-05 09:54:55 |
|       7 | -358.037736 | 2025-02-05 09:54:55 |
|       8 | -557.308411 | 2025-02-05 09:54:55 |
|       9 | -310.693878 | 2025-02-05 09:54:55 |
|      10 |        NULL | 2025-02-05 09:54:55 |
+---------+-------------+---------------------+
10 rows in set (0.01 sec)
```

Requirement 4: Budget expenses

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

select users.username,expenses.category,avg(expenses.amount)
as avg_spending
from expenses
inner join users on expenses.user_id=users.user_id
group by users.user_id,expenses.category
having avg_spending > (select avg(amount) from expenses where category='entertainment');

output:

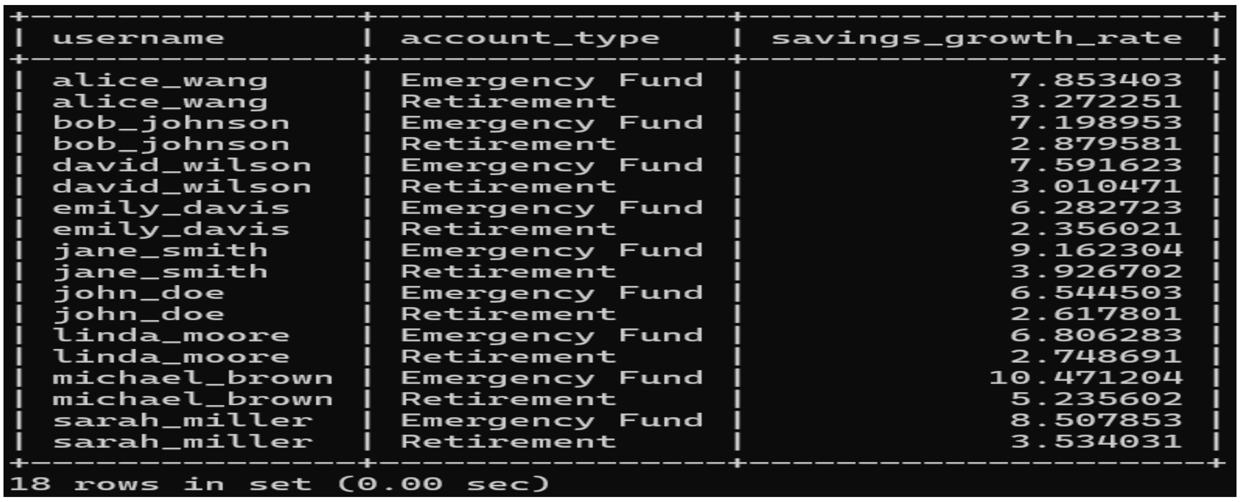


```
+----------------+----------------+--------------+
| username       | category       | avg_spending |
+----------------+----------------+--------------+
| alice_wang     | Rent           |  1300.000000 |
| alice_wang     | Groceries      |   350.000000 |
| bob_johnson    | Rent           |  1400.000000 |
| bob_johnson    | Groceries      |   320.000000 |
| david_wilson   | Rent           |  1450.000000 |
| david_wilson   | Groceries      |   340.000000 |
| david_wilson   | Entertainment  |   240.000000 |
| emily_davis    | Rent           |  1100.000000 |
| emily_davis    | Groceries      |   280.000000 |
| jane_smith     | Rent           |  1500.000000 |
| jane_smith     | Groceries      |   400.000000 |
| jane_smith     | Entertainment  |   250.000000 |
| john_doe       | Rent           |  1200.000000 |
| john_doe       | Groceries      |   300.000000 |
| linda_moore    | Rent           |  1250.000000 |
| linda_moore    | Groceries      |   310.000000 |
| michael_brown  | Rent           |  1600.000000 |
| michael_brown  | Groceries      |   450.000000 |
| michael_brown  | Entertainment  |   300.000000 |
| sarah_miller   | Rent           |  1350.000000 |
| sarah_miller   | Groceries      |   330.000000 |
| sarah_miller   | Entertainment  |   230.000000 |
+----------------+----------------+--------------+
22 rows in set (0.01 sec)
```

Requirement 5: Calculating savings growth rate

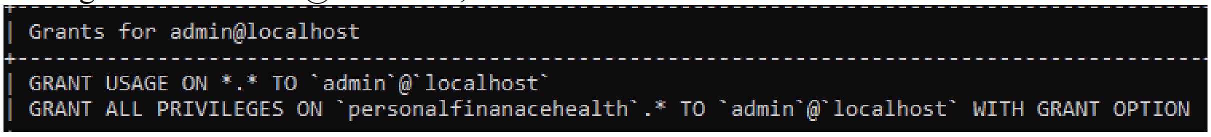
select users.username,savings.account_type,(savings.amount/(select sum(amount) from savings where
user_id = savings.user_id))*100 as savings_growth_rate
from savings
inner join users on savings.user_id=users.user_id;

Output:



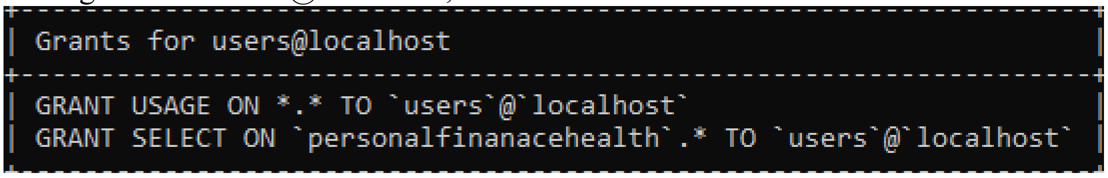
```
+----------------+----------------+---------------------+
| username       | account_type   | savings_growth_rate |
+----------------+----------------+---------------------+
| alice_wang     | Emergency Fund |            7.853403 |
| alice_wang     | Retirement     |            3.272251 |
| bob_johnson    | Emergency Fund |            7.198953 |
| bob_johnson    | Retirement     |            2.879581 |
| david_wilson   | Emergency Fund |            7.591623 |
| david_wilson   | Retirement     |            3.010471 |
| emily_davis    | Emergency Fund |            6.282723 |
| emily_davis    | Retirement     |            2.356021 |
| jane_smith     | Emergency Fund |            9.162304 |
| jane_smith     | Retirement     |            3.926702 |
| john_doe       | Emergency Fund |            6.544503 |
| john_doe       | Retirement     |            2.617801 |
| linda_moore    | Emergency Fund |            6.806283 |
| linda_moore    | Retirement     |            2.748691 |
| michael_brown  | Emergency Fund |           10.471204 |
| michael_brown  | Retirement     |            5.235602 |
| sarah_miller   | Emergency Fund |            8.507853 |
| sarah_miller   | Retirement     |            3.534031 |
+----------------+----------------+---------------------+
18 rows in set (0.00 sec)
```

Step 4: User Permissions
create user 'admin'@'localhost' identified by 'admin123';
show grants for 'admin'@'localhost';
grant all privileges on personalfinancehealth.* to 'admin'@'localhost' with grant option;
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
```

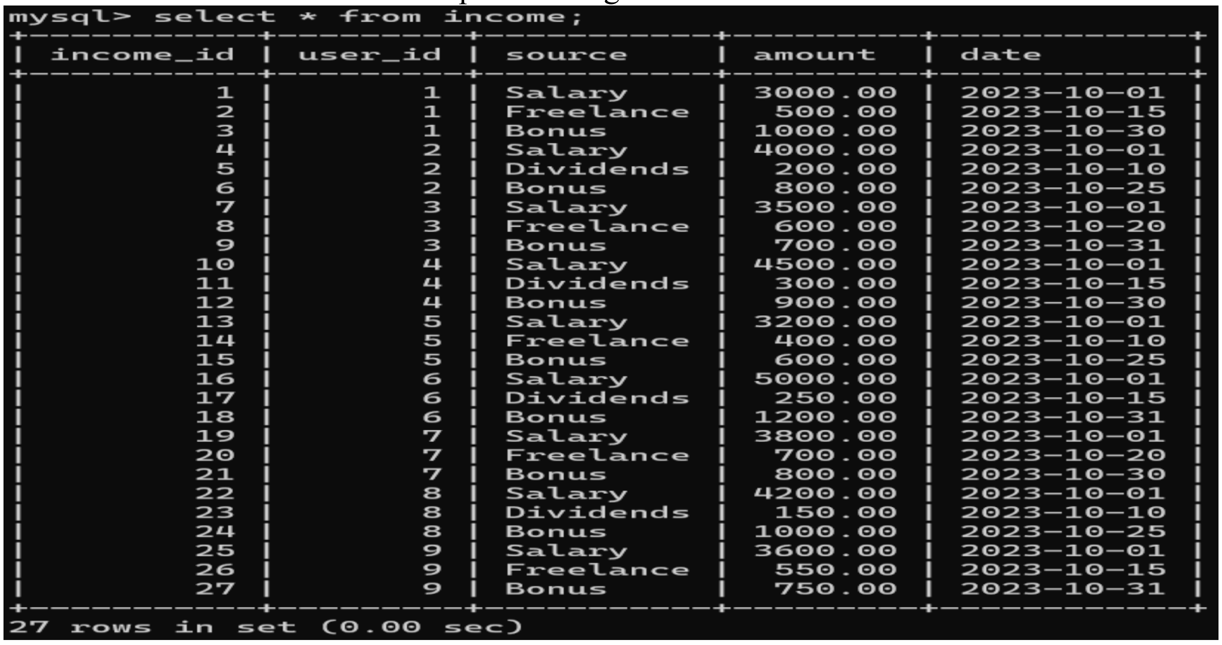
select user,host from mysql.user;
create user 'users'@'localhost' identified by 'users123';
grant select on personalfinancehealth.* to 'users'@'localhost';
show grants for 'users'@'localhost';



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

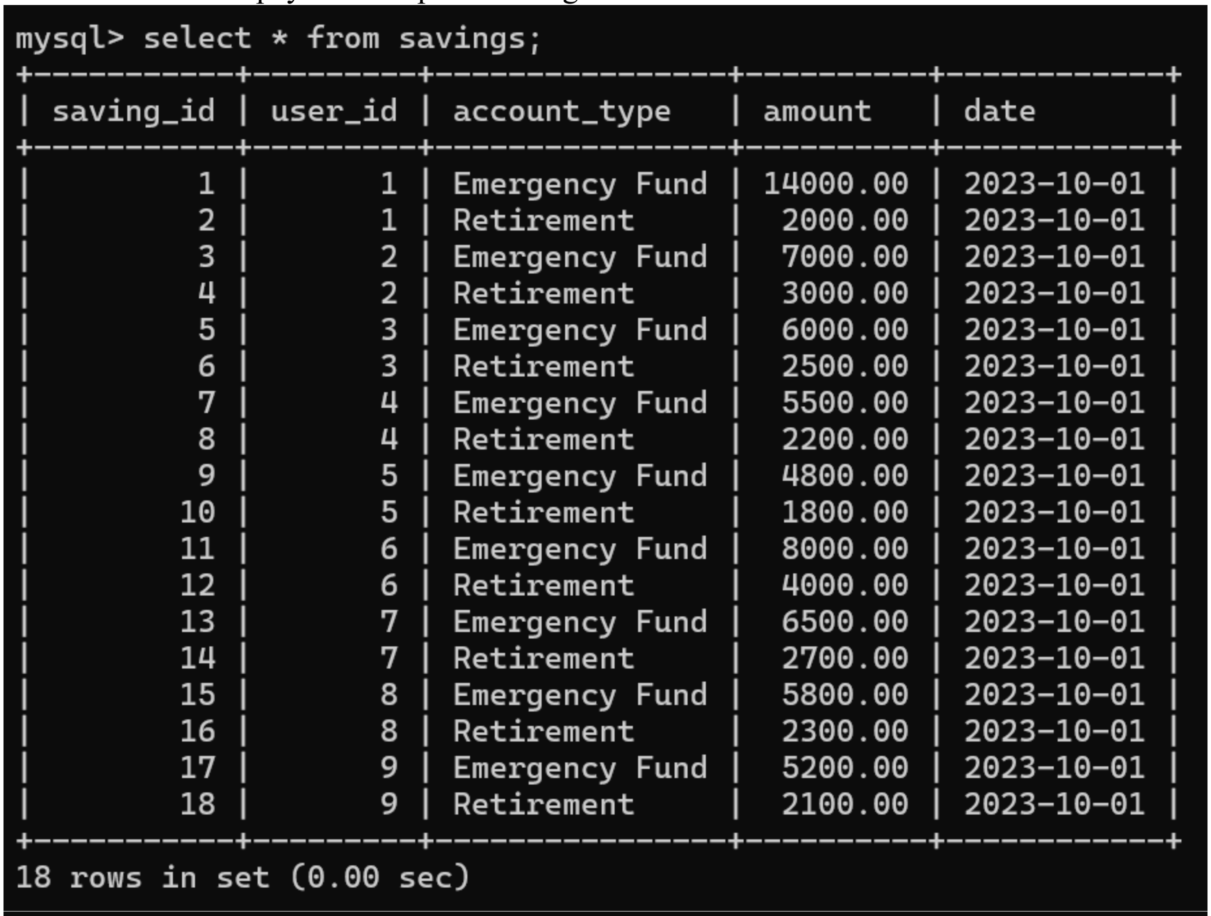
select user,host from mysql.user;

Step 5: Transactions -Over the course at the period in the data there are several updates when it comes to income, savings, investment, debts and expenses. In order to update the values, it is safer to update the values inside of transaction. There are 4 transactions to be performed on the project
Transaction1: Record income and update savings

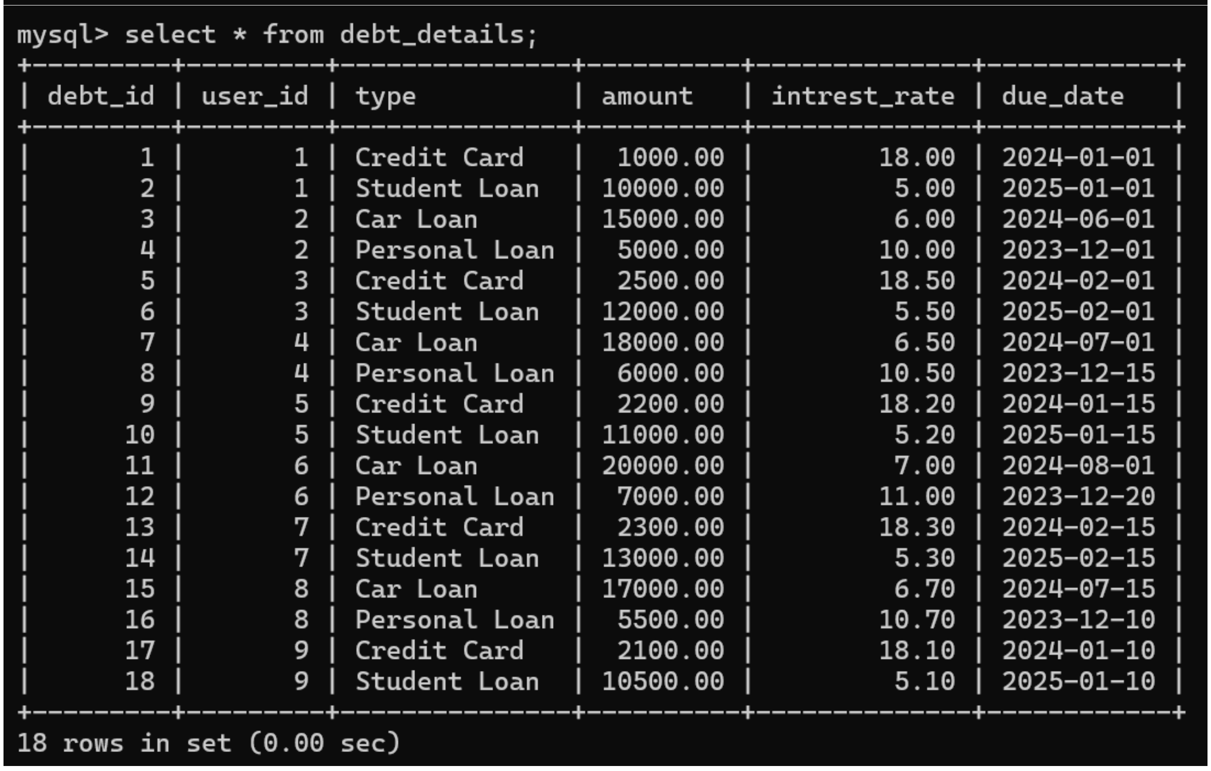


```
mysql> select * from income;
+-----------+---------+-----------+---------+------------+
| income_id | user_id | source    | amount  | date       |
+-----------+---------+-----------+---------+------------+
|         1 |       1 | Salary    | 3000.00 | 2023-10-01 |
|         2 |       1 | Freelance |  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 |
|        26 |       9 | Freelance |  550.00 | 2023-10-15 |
|        27 |       9 | Bonus     |  750.00 | 2023-10-31 |
+-----------+---------+-----------+---------+------------+
27 rows in set (0.00 sec)
```

Transaction2: Debt payoff and update savings

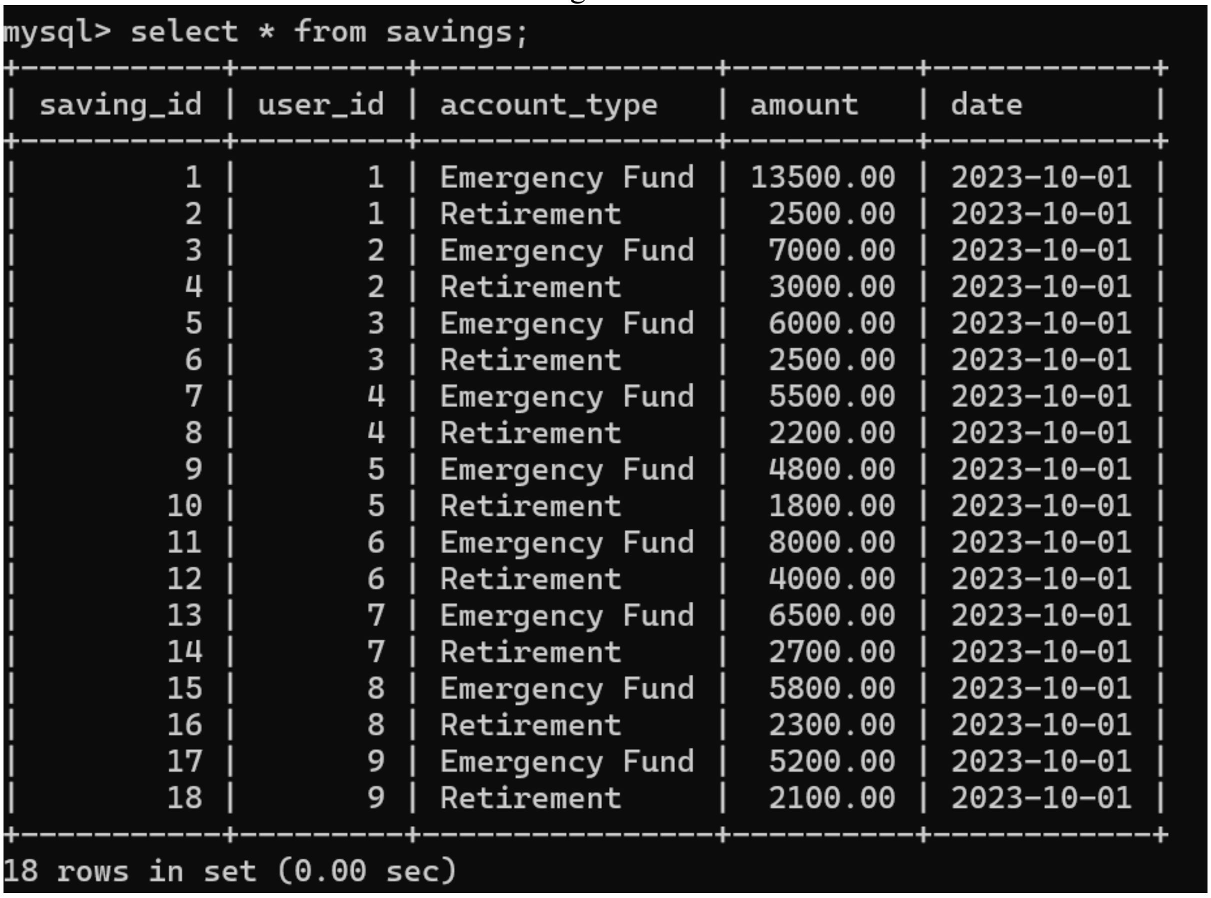


```
mysql> select * from savings;
+-----------+---------+----------------+----------+------------+
| saving_id | user_id | account_type   | amount   | date       |
+-----------+---------+----------------+----------+------------+
|         1 |       1 | Emergency Fund | 14000.00 | 2023-10-01 |
|         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 |
+-----------+---------+----------------+----------+------------+
18 rows in set (0.00 sec)
```



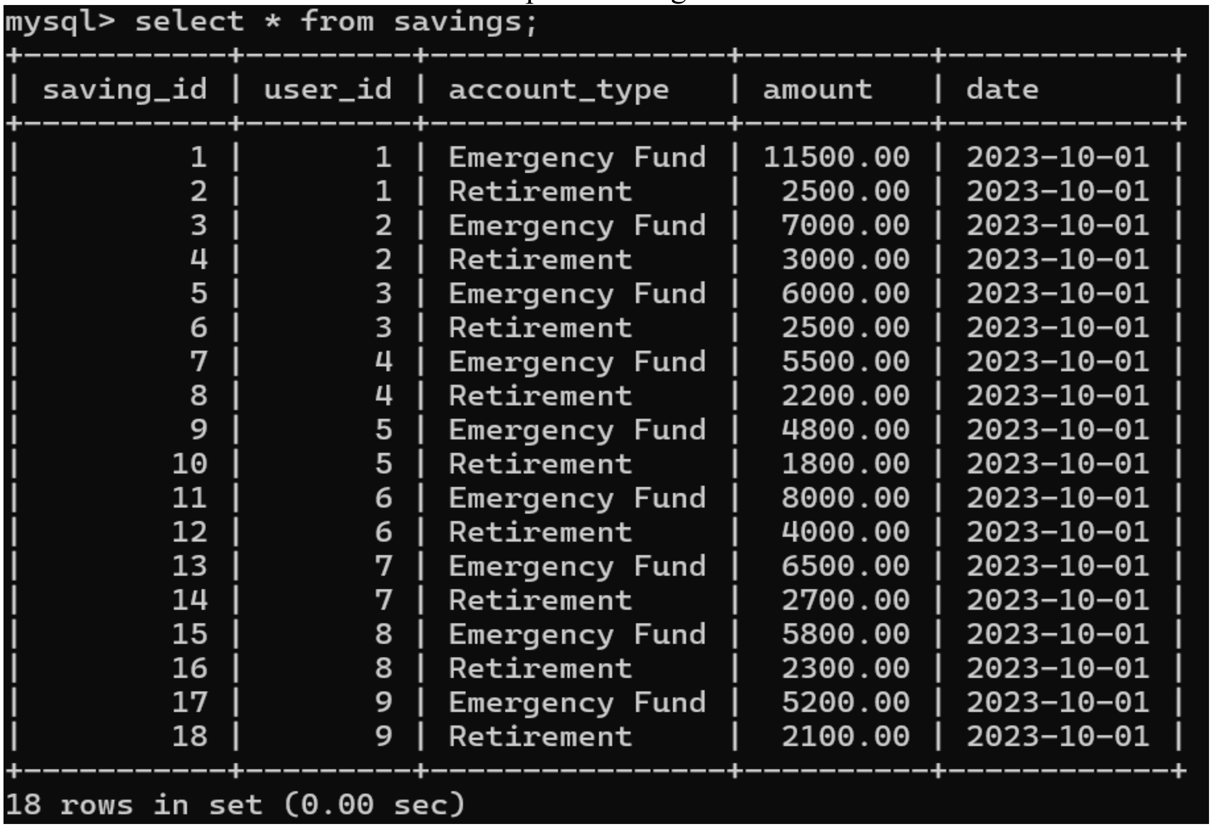
```
mysql> select * from debt_details;
+---------+---------+---------------+----------+--------------+------------+
| debt_id | user_id | type          | amount   | intrest_rate | due_date   |
+---------+---------+---------------+----------+--------------+------------+
|       1 |       1 | Credit Card   |  1000.00 |        18.00 | 2024-01-01 |
|       2 |       1 | Student Loan  | 10000.00 |         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)
```

Transaction 3: Transfer funds between savings account

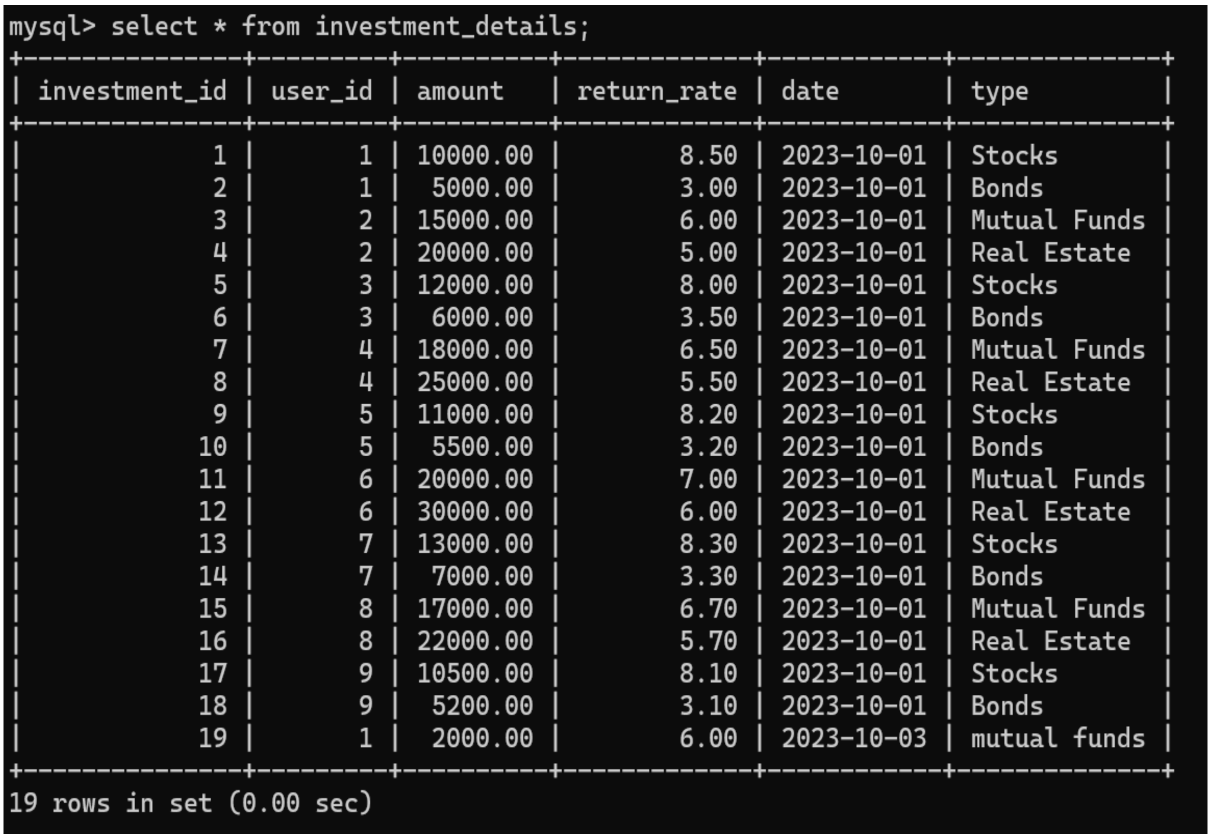


```
mysql> select * from savings;
+-----------+---------+----------------+----------+------------+
| saving_id | user_id | account_type   | amount   | date       |
+-----------+---------+----------------+----------+------------+
|         1 |       1 | Emergency Fund | 13500.00 | 2023-10-01 |
|         2 |       1 | Retirement     |  2500.00 | 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)
```

Transaction 4: Record investments and update savings



```
mysql> select * from savings;
+-----------+---------+----------------+----------+------------+
| saving_id | user_id | account_type   | amount   | date       |
+-----------+---------+----------------+----------+------------+
|         1 |       1 | Emergency Fund | 11500.00 | 2023-10-01 |
|         2 |       1 | Retirement     |  2500.00 | 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 investment_details;
+---------------+---------+----------+-------------+------------+--------------+
| investment_id | user_id | amount   | return_rate | date       | type         |
+---------------+---------+----------+-------------+------------+--------------+
|             1 |       1 | 10000.00 |        8.50 | 2023-10-01 | Stocks       |
|             2 |       1 |  5000.00 |        3.00 | 2023-10-01 | Bonds        |
|             3 |       2 | 15000.00 |        6.00 | 2023-10-01 | Mutual Funds |
|             4 |       2 | 20000.00 |        5.00 | 2023-10-01 | Real Estate  |
|             5 |       3 | 12000.00 |        8.00 | 2023-10-01 | Stocks       |
|             6 |       3 |  6000.00 |        3.50 | 2023-10-01 | Bonds        |
|             7 |       4 | 18000.00 |        6.50 | 2023-10-01 | Mutual Funds |
|             8 |       4 | 25000.00 |        5.50 | 2023-10-01 | Real Estate  |
|             9 |       5 | 11000.00 |        8.20 | 2023-10-01 | Stocks       |
|            10 |       5 |  5500.00 |        3.20 | 2023-10-01 | Bonds        |
|            11 |       6 | 20000.00 |        7.00 | 2023-10-01 | Mutual Funds |
|            12 |       6 | 30000.00 |        6.00 | 2023-10-01 | Real Estate  |
|            13 |       7 | 13000.00 |        8.30 | 2023-10-01 | Stocks       |
|            14 |       7 |  7000.00 |        3.30 | 2023-10-01 | Bonds        |
|            15 |       8 | 17000.00 |        6.70 | 2023-10-01 | Mutual Funds |
|            16 |       8 | 22000.00 |        5.70 | 2023-10-01 | Real Estate  |
|            17 |       9 | 10500.00 |        8.10 | 2023-10-01 | Stocks       |
|            18 |       9 |  5200.00 |        3.10 | 2023-10-01 | Bonds        |
|            19 |       1 |  2000.00 |        6.00 | 2023-10-03 | mutual funds |
+---------------+---------+----------+-------------+------------+--------------+
19 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 Enhancement:
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.