**PERSONAL FINANCE HEALTH ANALYSIS AND OPTIMIZATION**

**INTRODUCTION**

This is MySQL project which focuses on personal finance analysis and optimization of personal finances. This is one of the niches that is highly relevant and is rarely used. this will be designed using MySQL. This project is a MySQL based system designed to help individuals analyze and optimize their finances. It will include cracking income expenses, savings, investments and debts. It will provide actionable help that will improve financial health. The system will use MySQL concepts to generate reports, identify financial threats and suggest optimization strategies.

**PREREQUISITES FOR PROJECT**

In order to perform we need some prerequisites that the user needs to have the following

\* MySQL 8.0 Command Line client

\* MySQL 8.0 workbench CE

\* MySQL statements

\* MySQL clauses

\* MySQL operations

\* MySQL constraints

\* MySQL subqueries

\* MySQL joins

\* User permissions (grant and revoke)

\* Transactions

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

**KEY FEATURES**

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

\* **Income and expense tracking** - It is used to track monthly income sources and categorize expenses.

\* **Savings and investment analysis** - Monitoring saving accounts and investment portfolio with calculating returns.

\* **Debt management** - It is used to track debts and calculate interest payments with suggesting debt repayment strategies.

\* **Financial health scoring** - It is used to generate a financial health score based on income, expenses, savings, and debts it is used to access financial stability.

\* **Budget optimization** - It is used to suggest budget allocations that are optimal based on previous data.

\* **User permissions** – It is used to provide role-based access it is also used to restrict access on sensitive data.

**SCHEMA**

In order to perform project various parameters are required. They are: -

\* **Database**- Create a new database for this project with project name personal\_financial\_health.

\* **Tables**-

1.**Users** – It contains user’s details (name, id, role, password)

2.**Income** – It contains users income sources (income\_id, id, source, amount, date)

3.**Expenses** – It contains users expense details (expense\_id, id, amount, date)

4.**Savings** – It contains user’s savings details (saving\_id, id, amount, date)

5.**Investments** – It contains users investments details (investment\_id, id, type, amount, return\_rate, date)

6.**Debts** – It contains users debt details (debt\_id, id, type, amount, interest, date)

7.**FinancialHealth** – It contains user financial health score (health\_id, id, score, date)

\* **Relationships**- With use of constraint keys the users table must be linked with all other tables with use of primary key and foreign key.

\* **Users**- There are 2 types of users

i. Admin (with all permissions given)

ii. User (with select privilege).

**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 create tables.

show databases;

create database personal\_financial\_health;

use personal\_financial\_health;

show tables;

create table users(id int primary key auto\_increment, username varchar(20) unique not null, role enum('admin','user'), password varchar(25) not null);

create table income(income\_id int primary key auto\_increment, id int, source varchar(100), amount decimal(10,2) not null, date date, foreign key(id) references users(id));

create table expenses(expense\_id int primary key auto\_increment, id int, category varchar(40), amount decimal(10,2) not null, date date, foreign key(id) references users(id));

create table savings(saving\_id int primary key auto\_increment, id int, account\_type varchar(50) not null, amount decimal(10,2) not null, date date, foreign key(id) references users(id));

create table investments(invetsment\_id int primary key auto\_increment, id int, type varchar(50), amount decimal(10,2) not null, return\_rate decimal(5,2), date date, foreign key(id) references users(id));

create table debts(debt\_id int primary key auto\_increment, id int, type varchar(50), amount decimal(10,2) not null, interest\_rate decimal(5,2), due\_date date, foreign key(id) references users(id));

create table financial\_health(health\_id int primary key auto\_increment, id int, score decimal(5,2), date date, foreign key(id) references users(id));

**Step 2**- Insertion

Inserting values more than 100 rows in all seven tables to perform the analysis. apart from the auto increment columns all the columns have to be filled with data.

**Users table** - There should be 10 users among those 10 1 should be admin whereas the rest are users.

CODE: -

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');

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

CODE: -

INSERT INTO Income (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');

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

CODE: -

INSERT INTO Expenses (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');

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

CODE: -

INSERT INTO Savings (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');

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

CODE: -

INSERT INTO Investments (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');

**Debts table** - This table is used to provide values that are related to the loan information of each user and how much amount they have loaned.

CODE: -

INSERT INTO Debts (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');

**Step 3**- Requirements

Providing the necessary analytics or analysis based on each requirement.

8 Requirements: -

1. **Calculating monthly net income for each user**

In order to fetch the details for this requirement 3 tables are used and all the tables should be joined.

a. users

b. income

c. expenses.

**CODE**: - select users.username, sum(income.amount) - sum(expenses.amount) as net\_monthly\_income from users left join income on users.id = income.id left join expenses on users.id = expenses.id where income.date between '2023-10-01' and '2023-10-31' group by users.id;

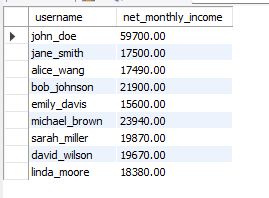


Fig:1

1. **Identify high interest debts**

**CODE**: - select users.username, debts.type, debts.amount, debts.interest\_rate from debts inner join users on debts.id = users.id where debts.interest\_rate > 10 order by debts.interest\_rate desc;

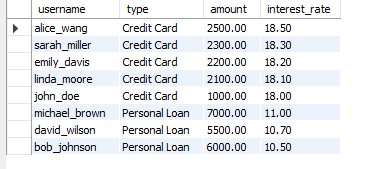


Fig:2

1. **Generating financial health score**

In this requirement the user will generate the financial health score using users, income, expenses, debts tables.

**CODE**: - select users.id, (sum(income.amount) - sum(expenses.amount) - sum(debts.amount))/sum(income.amount)\*100 as score, now() from users left join income on users.id = income.id left join expenses on users.id = expenses.id left join debts on users.id = debts.id group by users.id;

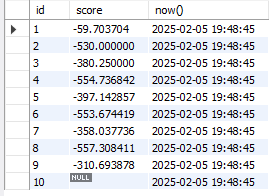


Fig:3

1. **Budget optimization**

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

**CODE**: - select users.username, expenses.category, avg(expenses.amount) as avg\_spending from expenses inner join users on expenses.id = users.id group by users.id, expenses.category having avg\_spending > (select avg(amount) from expenses where category = 'entertainment');

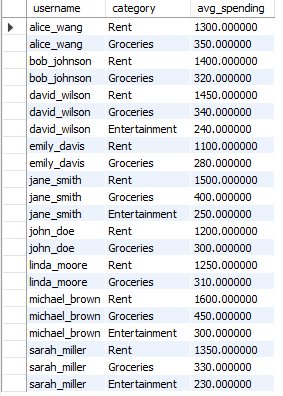


Fig:4

1. **Calculating savings growth rate**

In this requirement the users will be provided with their savings growth rate.

**CODE**: - select users.id, savings.account\_type, (savings.amount/(select sum(amount) from savings where id = savings.id))\*100 as growth\_rate from savings inner join users on savings.id = users.id;

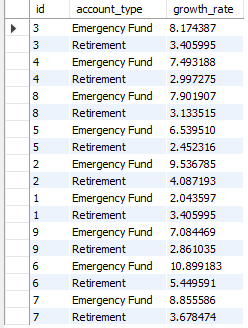


Fig:5

**Step 4**- User permissions

In this step we create user and admin then provide required permissions to them.

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

show grants for 'admin'@'localhost';

grant all on personal\_financial\_health.\* to 'admin'@'localhost' with grant option;

create user 'user'@'localhost' identified by 'user123';

grant select on personal\_financial\_health.\* to 'user'@'localhost';

show grants for 'user'@'localhost';

select user, host from mysql.user;

**Step5**- Transactions

Over the course of the period in the data there are several updates when it comes to income, savings, investments, debts, 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. They are: -

1. record income and update savings

CODE: -

start transaction;

select \* from savings;

select \* from income;

savepoint s1;

insert into income(id, source, amount, date) values(1,'side hustle',9000.00,'2023-10-11');

savepoint s2;

update savings set amount = amount + 9000.00 where id = 1 and account\_type = 'retirement fund';

commit;

1. debt payoff and update savings

CODE: -

start transaction;

select \* from debts;

select \* from savings;

savepoint s3;

update savings set amount = amount - 1000.00 where id = 1 and account\_type = 'emergency fund';

savepoint s4;

update debts set amount = amount - 1000.00 where id = 1;

commit;

1. transfer funds between savings account

CODE: -

start transaction;

select \* from savings;

savepoint s5;

update savings set amount = amount - 500.00 where id = 1 and account\_type = 'emergency fund';

savepoint s6;

update savings set amount = amount + 500.00 where id = 1 and account\_type = 'retirement';

commit;

1. record investments and update savings

CODE: -

start transaction;

select \* from investments;

select \* from savings;

savepoint s7;

update savings set amount = amount - 2000 where id = 1 and account\_type = 'emergency fund';

savepoint s8;

insert into investments(id, type, amount, return\_rate, date) values(1,'mutual funds', 2000.00, 6.00, '2023-10-03');

commit;

**CONCLUSION**

Using this project the users can maintain a good personal finance score here by helping them while making future financial decisions and maintain a good track of all the 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 the scores of the personal finance. Many analytics were produced apart from that many other analytics can be performed on the data such as finding the debt-to-income ratio (debt burden analysis, investment performance analysis, monthly expense trends and so more).

Ihe code has to be stored in .SQL file. Prepare a report of findings along with the SQL file and it will be posted in GitHub.

**FUTURE ENHANCEMENTS**

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

1. **Integrating with API: -** To fetch real life data.

2. **Data Visualization: -** To create interactive dashboards and reports.

3. **Machine Learning: -** Implementing ml models to predict financial health using certain algorithms and provide some recommendations.

**IMPACT OF PROJECT**

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