**PERSONAL FINANCIAL HEALTH ANALYZER AND OPTIMIZER**

**Introduction:**

This project is a MySQL based system design to help individuals analyse and optimise their personal finances. It will include tracking income expenses, savings, investments and debts. It will provide actionable insights that will help improving financial wellbeing. 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 pre requisites

1. MySQL 8.0 commend line clint
2. MySQL 8.0 workbench

Knowledge of these MySQL concepts

1. MySQL statements
2. MySQL operations
3. MySQL clauses
4. MySQL constraints
5. MySQL subqueries
6. MySQL joins
7. User permissions (grant and revoke)
8. Transactions

These are the concepts or 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 use to track monthly income sources and categorize expenses
2. **Savings and investment analysis:** monitoring savings accounts 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:** use 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 data base for this project with the project name personal\_finance\_health
2. **Tables required**

**Users:** this table should contain user details (User\_id, user.name, role, password).

**Income:** use to store income sources of the user (income\_id, user\_id, source, amount, date).

**Expenses:** it is used to store expense details of the user (expense\_id, user\_id, category, amount, date).

**Savings:** use to store savings details (saving\_id, user\_id, account\_type, amount, date).

**Investments:** use to store investment details (investment\_id, user\_id, type, amount, return\_rate, date).

**Debts:** use to store debt details (debt\_id, user\_id, type, amount, interest\_rate, due\_date).

1. **Relationships:** with the use of constraints keys the users table must be linked with all the other table with the use of primary key and foreign key. This is called one to many relationships.
2. **Users:**
3. Admin-with admin privileges (ALL PRIVILEGES)
4. User – with user privileges (SELECT)

**Implementation:**

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

**Step 1:** creating data base and creating the necessary tables

1. show databases;

2. create database PersonalFinanceHealth;

3. use PersonalFinanceHealth;

4.

5. show tables;

6. CREATE TABLE Users(

7. user\_id int auto\_increment,

8. username varchar(225) not null,

9. role enum('admin','user') not null,

10. password varchar(50) not null,

11. primary key(user\_id),

12. unique(username)

13. );

14.

15. CREATE TABLE Income(

16. income\_id int auto\_increment,

17. user\_id int,

18. source varchar(100),

19. amount decimal(10,2) not null,

20. date DATE,

21. primary key(income\_id),

22. foreign key(user\_id) references

23. users(user\_id)

24. );

25. CREATE TABLE Expenses(

26. expense\_id int auto\_increment,

27. user\_id int,

28. category varchar(100),

29. amount decimal(10,2) not null,

30. date DATE,

31. PRIMARY KEY(expense\_id),

32. FOREIGN KEY(user\_id) references

33. users(user\_id)

34. );

35. CREATE TABLE Savings(

36. saving\_id int auto\_increment,

37. user\_id int,

38. account\_type varchar(100),

39. amount decimal(10,2) not null,

40. date DATE,

41. PRIMARY KEY(saving\_id),

42. FOREIGN KEY(user\_id) references

43. users(user\_id)

44. );

45. CREATE TABLE Investments(

46. investment\_id int auto\_increment,

47. user\_id int,

48. type varchar(50),

49. amount decimal(10,2),

50. return\_rate decimal(5,2),

51. date date,

52. PRIMARY KEY(investment\_id),

53. FOREIGN KEY(user\_id) references

54. users(user\_id)

55. );

56. CREATE TABLE Debts(

57. debts\_id int auto\_increment,

58. user\_id int,

59. type varchar(50),

60. amount decimal(10,2),

61. interest\_rate decimal(5,2),

62. due\_date date,

63. PRIMARY KEY(debts\_id),

64. FOREIGN KEY(user\_id) references

65. users(user\_id)

66. );

67.

**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 must be filled with data.

**Users:** There should be 10 users among those 10 ones should be named as admin whereas the rest 9 will be users.

1. INSERT INTO Users (username, role, password) VALUES

2. ('john\_doe', 'user', 'hashed\_password\_123'),

3. ('jane\_smith', 'user', 'hashed\_password\_456'),

4. ('alice\_wang', 'user', 'hashed\_password\_789'),

5. ('bob\_johnson', 'user', 'hashed\_password\_101'),

6. ('emily\_davis', 'user', 'hashed\_password\_112'),

7. ('michael\_brown', 'user', 'hashed\_password\_131'),

8. ('sarah\_miller', 'user', 'hashed\_password\_415'),

9. ('david\_wilson', 'user', 'hashed\_password\_161'),

10. ('linda\_moore', 'user', 'hashed\_password\_718'),

11. ('admin\_user', 'admin', 'hashed\_password\_919');

12.

**Income:** For each user there are different streams of income along with the profit generated or amount generated they have been assigned to a particular user id that matches the user id’s from the table USERS.

1. INSERT INTO Income (user\_id, source, amount, date) VALUES

2. (1, 'Salary', 3000.00, '2023-10-01'),

3. (1, 'Freelance', 500.00, '2023-10-15'),

4. (1, 'Bonus', 1000.00, '2023-10-30'),

5. (2, 'Salary', 4000.00, '2023-10-01'),

6. (2, 'Dividends', 200.00, '2023-10-10'),

7. (2, 'Bonus', 800.00, '2023-10-25'),

8. (3, 'Salary', 3500.00, '2023-10-01'),

9. (3, 'Freelance', 600.00, '2023-10-20'),

10. (3, 'Bonus', 700.00, '2023-10-31'),

11. (4, 'Salary', 4500.00, '2023-10-01'),

12. (4, 'Dividends', 300.00, '2023-10-15'),

13. (4, 'Bonus', 900.00, '2023-10-30'),

14. (5, 'Salary', 3200.00, '2023-10-01'),

15. (5, 'Freelance', 400.00, '2023-10-10'),

16. (5, 'Bonus', 600.00, '2023-10-25'),

17. (6, 'Salary', 5000.00, '2023-10-01'),

18. (6, 'Dividends', 250.00, '2023-10-15'),

19. (6, 'Bonus', 1200.00, '2023-10-31'),

20. (7, 'Salary', 3800.00, '2023-10-01'),

21. (7, 'Freelance', 700.00, '2023-10-20'),

22. (7, 'Bonus', 800.00, '2023-10-30'),

23. (8, 'Salary', 4200.00, '2023-10-01'),

24. (8, 'Dividends', 150.00, '2023-10-10'),

25. (8, 'Bonus', 1000.00, '2023-10-25'),

26. (9, 'Salary', 3600.00, '2023-10-01'),

27. (9, 'Freelance', 550.00, '2023-10-15'),

28. (9, 'Bonus', 750.00, '2023-10-31');

29.

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

1. INSERT INTO Expenses (user\_id, category, amount, date) VALUES

2. -- John Doe

3. (1, 'Rent', 1200.00, '2023-10-01'),

4. (1, 'Groceries', 300.00, '2023-10-05'),

5. (1, 'Utilities', 150.00, '2023-10-07'),

6. (1, 'Entertainment', 200.00, '2023-10-10'),

7. (1, 'Transportation', 100.00, '2023-10-15'),

8.

9. -- Jane Smith

10. (2, 'Rent', 1500.00, '2023-10-01'),

11. (2, 'Groceries', 400.00, '2023-10-05'),

12. (2, 'Utilities', 200.00, '2023-10-07'),

13. (2, 'Entertainment', 250.00, '2023-10-10'),

14. (2, 'Transportation', 150.00, '2023-10-15'),

15.

16. -- Alice Wang

17. (3, 'Rent', 1300.00, '2023-10-01'),

18. (3, 'Groceries', 350.00, '2023-10-05'),

19. (3, 'Utilities', 180.00, '2023-10-07'),

20. (3, 'Entertainment', 220.00, '2023-10-10'),

21. (3, 'Transportation', 120.00, '2023-10-15'),

22.

23. -- Bob Johnson

24. (4, 'Rent', 1400.00, '2023-10-01'),

25. (4, 'Groceries', 320.00, '2023-10-05'),

26. (4, 'Utilities', 160.00, '2023-10-07'),

27. (4, 'Entertainment', 210.00, '2023-10-10'),

28. (4, 'Transportation', 110.00, '2023-10-15'),

29.

30. -- Emily Davis

31. (5, 'Rent', 1100.00, '2023-10-01'),

32. (5, 'Groceries', 280.00, '2023-10-05'),

33. (5, 'Utilities', 140.00, '2023-10-07'),

34. (5, 'Entertainment', 190.00, '2023-10-10'),

35. (5, 'Transportation', 90.00, '2023-10-15'),

36.

37. -- Michael Brown

38. (6, 'Rent', 1600.00, '2023-10-01'),

39. (6, 'Groceries', 450.00, '2023-10-05'),

40. (6, 'Utilities', 220.00, '2023-10-07'),

41. (6, 'Entertainment', 300.00, '2023-10-10'),

42. (6, 'Transportation', 200.00, '2023-10-15'),

43.

44. -- Sarah Miller

45. (7, 'Rent', 1350.00, '2023-10-01'),

46. (7, 'Groceries', 330.00, '2023-10-05'),

47. (7, 'Utilities', 170.00, '2023-10-07'),

48. (7, 'Entertainment', 230.00, '2023-10-10'),

49. (7, 'Transportation', 130.00, '2023-10-15'),

50.

51. -- David Wilson

52. (8, 'Rent', 1450.00, '2023-10-01'),

53. (8, 'Groceries', 340.00, '2023-10-05'),

54. (8, 'Utilities', 190.00, '2023-10-07'),

55. (8, 'Entertainment', 240.00, '2023-10-10'),

56. (8, 'Transportation', 140.00, '2023-10-15'),

57.

58. -- Linda Moore

59. (9, 'Rent', 1250.00, '2023-10-01'),

60. (9, 'Groceries', 310.00, '2023-10-05'),

61. (9, 'Utilities', 160.00, '2023-10-07'),

62. (9, 'Entertainment', 210.00, '2023-10-10'),

63. (9, 'Transportation', 110.00, '2023-10-15');

64.

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

1. INSERT INTO Savings (user\_id, account\_type, amount, date) VALUES

2. -- John Doe

3. (1, 'Emergency Fund', 5000.00, '2023-10-01'),

4. (1, 'Retirement', 2000.00, '2023-10-01'),

5.

6. -- Jane Smith

7. (2, 'Emergency Fund', 7000.00, '2023-10-01'),

8. (2, 'Retirement', 3000.00, '2023-10-01'),

9.

10. -- Alice Wang

11. (3, 'Emergency Fund', 6000.00, '2023-10-01'),

12. (3, 'Retirement', 2500.00, '2023-10-01'),

13.

14. -- Bob Johnson

15. (4, 'Emergency Fund', 5500.00, '2023-10-01'),

16. (4, 'Retirement', 2200.00, '2023-10-01'),

17.

18. -- Emily Davis

19. (5, 'Emergency Fund', 4800.00, '2023-10-01'),

20. (5, 'Retirement', 1800.00, '2023-10-01'),

21.

22. -- Michael Brown

23. (6, 'Emergency Fund', 8000.00, '2023-10-01'),

24. (6, 'Retirement', 4000.00, '2023-10-01'),

25.

26. -- Sarah Miller

27. (7, 'Emergency Fund', 6500.00, '2023-10-01'),

28. (7, 'Retirement', 2700.00, '2023-10-01'),

29.

30. -- David Wilson

31. (8, 'Emergency Fund', 5800.00, '2023-10-01'),

32. (8, 'Retirement', 2300.00, '2023-10-01'),

33.

34. -- Linda Moore

35. (9, 'Emergency Fund', 5200.00, '2023-10-01'),

36. (9, 'Retirement', 2100.00, '2023-10-01');

37.

**Investment:** this table is used to provide the values of each investment done by the user and the return they have received.

1. INSERT INTO Investments (user\_id, type, amount, return\_rate, date) VALUES

2. -- John Doe

3. (1, 'Stocks', 10000.00, 8.50, '2023-10-01'),

4. (1, 'Bonds', 5000.00, 3.00, '2023-10-01'),

5.

6. -- Jane Smith

7. (2, 'Mutual Funds', 15000.00, 6.00, '2023-10-01'),

8. (2, 'Real Estate', 20000.00, 5.00, '2023-10-01'),

9.

10. -- Alice Wang

11. (3, 'Stocks', 12000.00, 8.00, '2023-10-01'),

12. (3, 'Bonds', 6000.00, 3.50, '2023-10-01'),

13.

14. -- Bob Johnson

15. (4, 'Mutual Funds', 18000.00, 6.50, '2023-10-01'),

16. (4, 'Real Estate', 25000.00, 5.50, '2023-10-01'),

17.

18. -- Emily Davis

19. (5, 'Stocks', 11000.00, 8.20, '2023-10-01'),

20. (5, 'Bonds', 5500.00, 3.20, '2023-10-01'),

21.

22. -- Michael Brown

23. (6, 'Mutual Funds', 20000.00, 7.00, '2023-10-01'),

24. (6, 'Real Estate', 30000.00, 6.00, '2023-10-01'),

25.

26. -- Sarah Miller

27. (7, 'Stocks', 13000.00, 8.30, '2023-10-01'),

28. (7, 'Bonds', 7000.00, 3.30, '2023-10-01'),

29.

30. -- David Wilson

31. (8, 'Mutual Funds', 17000.00, 6.70, '2023-10-01'),

32. (8, 'Real Estate', 22000.00, 5.70, '2023-10-01'),

33.

34. -- Linda Moore

35. (9, 'Stocks', 10500.00, 8.10, '2023-10-01'),

36. (9, 'Bonds', 5200.00, 3.10, '2023-10-01');

37.

**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.

1. INSERT INTO Debts (user\_id, type, amount, interest\_rate, due\_date) VALUES

2. -- John Doe

3. (1, 'Credit Card', 2000.00, 18.00, '2024-01-01'),

4. (1, 'Student Loan', 10000.00, 5.00, '2025-01-01'),

5.

6. -- Jane Smith

7. (2, 'Car Loan', 15000.00, 6.00, '2024-06-01'),

8. (2, 'Personal Loan', 5000.00, 10.00, '2023-12-01'),

9.

10. -- Alice Wang

11. (3, 'Credit Card', 2500.00, 18.50, '2024-02-01'),

12. (3, 'Student Loan', 12000.00, 5.50, '2025-02-01'),

13.

14. -- Bob Johnson

15. (4, 'Car Loan', 18000.00, 6.50, '2024-07-01'),

16. (4, 'Personal Loan', 6000.00, 10.50, '2023-12-15'),

17.

18. -- Emily Davis

19. (5, 'Credit Card', 2200.00, 18.20, '2024-01-15'),

20. (5, 'Student Loan', 11000.00, 5.20, '2025-01-15'),

21.

22. -- Michael Brown

23. (6, 'Car Loan', 20000.00, 7.00, '2024-08-01'),

24. (6, 'Personal Loan', 7000.00, 11.00, '2023-12-20'),

25.

26. -- Sarah Miller

27. (7, 'Credit Card', 2300.00, 18.30, '2024-02-15'),

28. (7, 'Student Loan', 13000.00, 5.30, '2025-02-15'),

29.

30. -- David Wilson

31. (8, 'Car Loan', 17000.00, 6.70, '2024-07-15'),

32. (8, 'Personal Loan', 5500.00, 10.70, '2023-12-10'),

33.

34. -- Linda Moore

35. (9, 'Credit Card', 2100.00, 18.10, '2024-01-10'),

36. (9, 'Student Loan', 10500.00, 5.10, '2025-01-10');

37.

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

**Requirement 1:** 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.

1. SELECT users.username,SUM(income.amount),SUM(expenses.amount)

2. FROM users

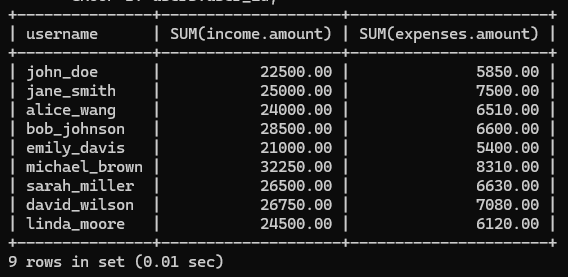
3. LEFT JOIN income on users.user\_id = income.user\_id

4. LEFT JOIN expenses on users.user\_id = expenses.user\_id

5. WHERE income.date between '2023-10-01' AND '2023-10-31'

6. GROUP BY users.user\_id;

7.

**Output:**

1. select users.username,SUM(income.amount)-SUM(expenses.amount)

2. as net\_monthly\_income from users

3. LEFT JOIN income on users.user\_id = income.user\_id

4. LEFT JOIN expenses on users.user\_id = expenses.user\_id

5. WHERE income.date BETWEEN '2023-10-01' AND '2023-10-31'

6. GROUP BY users.user\_id;

7.

 **Output:**

**Requirement 2:** identifying high interest debts

1. select users.username, debts.type, debts.amount, debts.interest\_rate

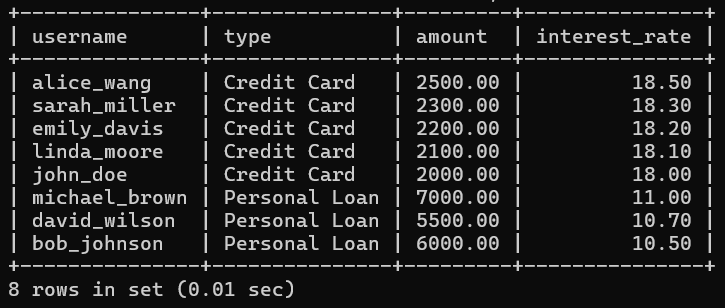
2. from debts

3. INNER JOIN users ON debts.user\_id = users.user\_id

4. WHERE debts.interest\_rate > 10

5. ORDER BY debts.interest\_rate DESC;

6.

**Output:**

**Requirement 3:** Generate financial health score

In this requirment the user will calculate the financial health score with the use of 4 tables (users, income, expenses and debts)

1. select users.user\_id, (SUM(income.amount) - SUM(expenses.amount) - SUM(debts.amount)) / SUM(income.amount) \* 100

2. as score, NOW()

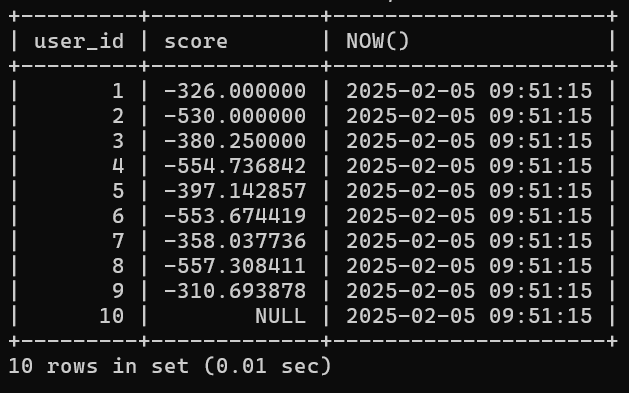
3. FROM users

4. LEFT JOIN income ON users.user\_id = income.user\_id

5. LEFT JOIN expenses ON users.user\_id = expenses.user\_id

6. LEFT JOIN debts ON users.user\_id = debts.user\_id

7. GROUP BY users.user\_id;

8.  

**Requirement 4:** Budget optimization

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

1. select users.username, expenses.category, AVG(expenses.amount)

2. AS avg\_spending

3. FROM expenses

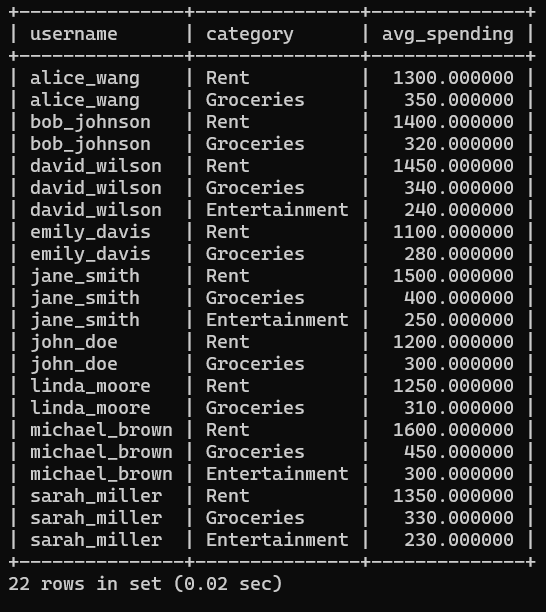
4. INNER JOIN users ON expenses.user\_id = users.user\_id

5. GROUP BY users.user\_id, expenses.category

6. HAVING avg\_spending>(SELECT AVG(amount) FROM expenses WHERE category = 'entertainment');

7.

Instead of entertainment we can use anything in the category column like rent, groceries.

**Output:**

**Requirement 5:** Calculating savings growth rate

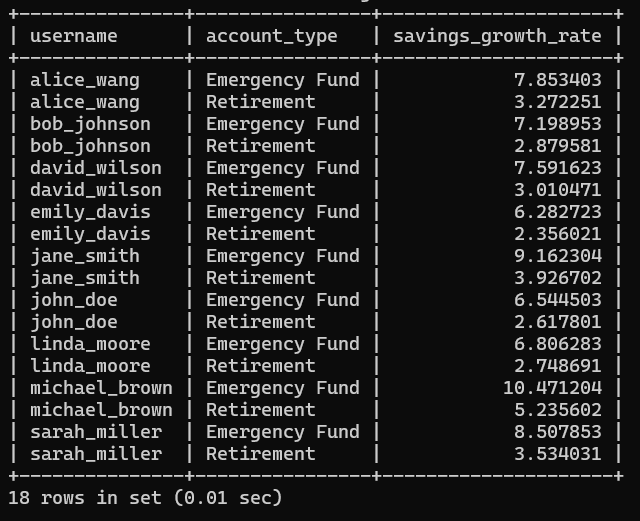
1. select users.username, savings.account\_type, (savings.amount / (select SUM(amount) FROM savings WHERE

2. user\_id = savings.user\_id))\*100 as savings\_growth\_rate

3. FROM savings

4. INNER JOIN users ON savings.user\_id = users.user\_id;

5.

**Output:**

**Requirement 6: Debt Burden Analysis (Debt to Income Ratio)**

1. SELECT

2. u.username,

3. SUM(d.amount) AS total\_debt,

4. SUM(i.amount) AS total\_income,

5. (SUM(d.amount) / SUM(i.amount)) \* 100 AS debt\_to\_income\_ratio

6. FROM Users u

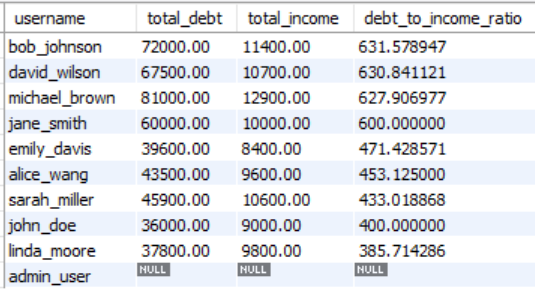
7. LEFT JOIN Debts d ON u.user\_id = d.user\_id

8. LEFT JOIN Income i ON u.user\_id = i.user\_id

9. GROUP BY u.user\_id

10. ORDER BY debt\_to\_income\_ratio DESC;

11.

****

**Requirement 7: Investment Performance Analysis**

1. SELECT

2. u.username,

3. i.type AS investment\_type,

4. i.amount AS investment\_amount,

5. i.return\_rate,

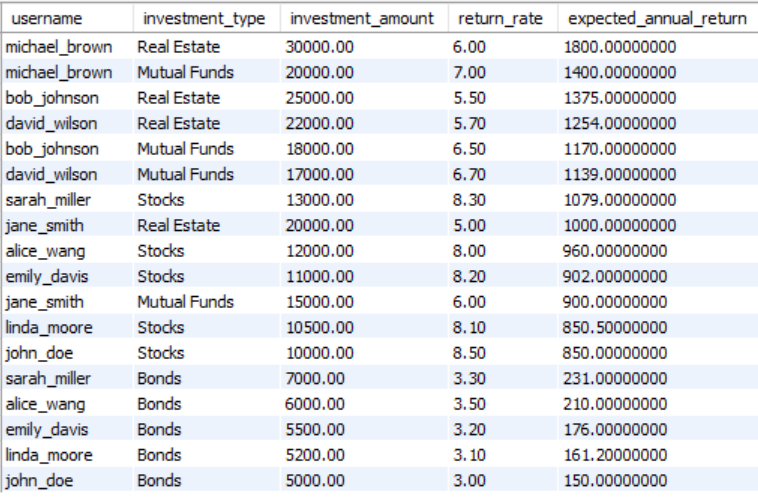
6. (i.amount \* i.return\_rate / 100) AS expected\_annual\_return

7. FROM Users u

8. JOIN Investments i ON u.user\_id = i.user\_id

9. ORDER BY expected\_annual\_return DESC;

10.

****

**Requirement 8: Calculate Monthly Expenses Trends**

1. SELECT

2. u.username,

3. e.category,

4. MONTH(e.date) AS month,

5. SUM(e.amount) AS total\_spending

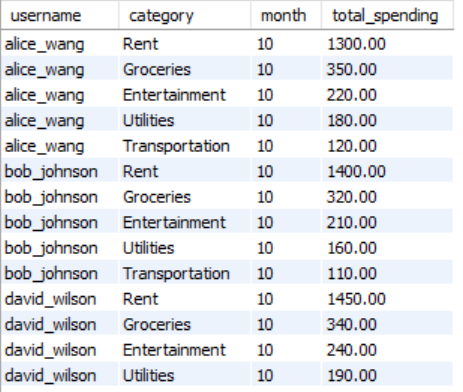
6. FROM Users u

7. JOIN Expenses e ON u.user\_id = e.user\_id

8. GROUP BY u.user\_id, e.category, MONTH(e.date)

9. ORDER BY u.username, month, total\_spending DESC;

10.

****

**STEP 4:** User permission

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

2. SHOW GRANTS for 'admin'@'localhost';

3. GRANT ALL PRIVILEGES ON PersonalFinanceHealth.\* TO 'admin'@'localhost' WITH GRANT OPTION;

4. SHOW GRANTS FOR 'admin'@'localhost';

5. select user, host from mysql.user;

6. CREATE USER 'users'@'localhost' identified by 'user123';

7. GRANT SELECT ON PersonalFinanceHealth.\* to 'users'@'localhost';

8.

**STEP 5:** 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 saver to update the values inside a transaction. There are 4 transactions to be performed on this project.

Transaction 1: record income and update savings.

1. START TRANSACTION;

2. SELECT \* FROM savings;

3. SELECT \* FROM income;

4. SAVEPOINT S1;

5. INSERT INTO income(user\_id,source,amount,date)

6. VALUES(1,'side hustle',9000.00,'2023-10-11');

7. SAVEPOINT S2;

8. UPDATE savings SET amount=amount+9000.00

9. WHERE user\_id = 1 AND account\_type = 'emergency fund';

10. COMMIT;

11.

Transaction 2: Transaction to debt payoff and update savings.

1. START TRANSACTION;

2. SELECT \* FROM savings;

3. SELECT \* FROM DEBTS;

4. SAVEPOINT S3;

5. UPDATE SAVINGS SET amount = amount-1000.00

6. WHERE user\_id = 1 AND account\_type = 'emergency fund';

7. SAVEPOINT S4;

8. UPDATE debts SET amount = amount-1000.00

9. WHERE debts\_id = 1;

10. COMMIT;

11.

Transaction 3: Transfer funds between savings account.

1. START TRANSACTION;

2. SELECT \* FROM savings;

3. SAVEPOINT S5;

4. UPDATE savings SET amount = amount-500.00

5. WHERE user\_id = 1 AND account\_type = 'emergency fund';

6. SAVEPOINT S6;

7. UPDATE savings SET amount = amount+500.00

8. WHERE user\_id = 1 AND account\_type = 'retirement';

9. COMMIT;

10.

Transaction 4: Record investments and update savings.

1. START TRANSACTION;

2. SELECT \* FROM savings;

3. SELECT \* FROM investments;

4. SAVEPOINT S7;

5. UPDATE savings SET amount = amount-2000.00

6. WHERE user\_id = 1 AND account\_type = 'emergenct fund';

7. SAVEPOINT S8;

8. INSERT INTO investments(user\_id,type,amount,return\_rate,date)

9. VALUES(1,'Mutual Fund',2000.00,6.00,'2023-10-03');

10. COMMIT;

11.

**Conclusion**

Using this project the users can maintain a good personal finance score hereby helping them future financial decisions and maintain a good track of all their future assets and liabilities. With the use of tables such as income expenses debts, savings investments 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 also be performed bases on their data such as finding the debt to income ratio (debt burden analysis, investment performance analysis of monthly expense trends and so forth. The code has to stored in .sql file. Prepare a report of the findings along with the sql file and it will hosted in github.

**Future Enhancements**

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

1. Integrating with api to fetch real life data.
2. Data visualization to create an interactive dashboards and reports
3. Implementing machine learning models to predict financial health using certain algorithms and provide some recommendations.

This project provides proficiency in my sql 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