

# 2024 PERSONAL FINANCE AND INVESTMENT ANALYSIS

(MYSQL EDITION)

**MARK ANTHONY A. BUNA**

*Data Analyst*

## PROBLEM STATEMENT

- Managing personal finances often lacks structure, making it difficult to track where money goes, how investments perform, and whether financial goals are being met. This project uses MySQL to build a personal financial tracking system that organizes income, expenses, investments, savings, and debts — with the goal of uncovering trends and gaining clarity over financial health throughout 2024.

## OBJECTIVE

- Create a MySQL database to track income, expenses, investments, savings, and debts, and analyze financial trends.
- Analyze financial activity to answer key questions for the year 2024:
  - *What is the general financial overview for each month in 2024?*
  - *How are expenses distributed across different categories, and what are the 3 major spending areas?*
  - *How much have investments earned in terms of returns?*
  - *What is the total debt and interest paid?*
  - *What is the total net worth?*

## FINAL PORTFOLIO DELIVERABLES

- Built a MySQL-based financial tracking system with analysis queries, automated views, and full documentation hosted on GitHub.

## LINK:

- Github: <https://github.com/marktheanalyst103/mysql-scripts/blob/main/2024%20Personal%20Finance%20and%20Investment%20Analysis.sql>

# PROJECT STEPS

## Step 1: Set Up MySQL Environment

- Install MySQL and set up the database.

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'personal\_finance' schema, there are tables for debts, expenses, income, investments, and savings. A new database named 'personal\_finance' has been created and selected. The SQL Editor pane contains the following code:

```
# CREATING A DATABASE
create database personal_finance;
use personal_finance;

# CREATING TABLES AND INSERTING SAMPLE DATA

# Income Table
create table income
(
    income_id INT PRIMARY KEY AUTO_INCREMENT,
    income_source VARCHAR(255),
    income_amount DECIMAL (10,2),
    income_date DATE
);

select * from income;
```

The Output pane shows the results of the 'use personal\_finance' command, indicating 0 rows affected. The Result Grid pane at the bottom is empty.

## Step 2: Create Tables & Insert Sample Data

- Define tables for *income*, *expenses*, *investments*, *savings*, and *debts*.
- Populate tables with sample financial data.

The screenshot shows the MySQL Workbench interface. The 'personal\_finance' schema now includes a 'monthly\_financial\_summ' view and two stored procedures, 'GetFinancialHealth' and 'GetFinancialHealth2'. The SQL Editor pane contains the same code as in Step 1, plus an additional 'select \* from income' statement:

```
use personal_finance;

# CREATING TABLES AND INSERTING SAMPLE DATA

# Income Table
create table income
(
    income_id INT PRIMARY KEY AUTO_INCREMENT,
    income_source VARCHAR(255),
    income_amount DECIMAL (10,2),
    income_date DATE
);

select * from income;
```

The Output pane shows the results of the 'create table income' command, indicating 0 rows affected. The Result Grid pane at the bottom displays the following data:

income_id	income_source	income_amount	income_date
NULL	NULL	NULL	NULL

The Output pane also shows the results of the 'select \* from income LIMIT 0, 500' command, indicating 0 rows returned.

MySQL Workbench

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas personal\_finance

Tables debts expenses investments savings

Views monthly\_financial\_summ

Stored Procedures GetFinancialHealth GetFinancialHealth2

Administration Schemas Information

No object selected

MySQL Portfolio

```

15     income_date DATE
16   );
17
18 •   select * from income;
19
20 •   insert into income (income_source, income_amount, income_date)
21   values
22   ('Concentrix Company', 6802, '2024-03-01'),
23   ('Concentrix Company', 8311, '2024-03-15'),
24   ('Concentrix Company', 9044, '2024-03-27'),
25   ('Concentrix Company', 9916, '2024-04-12'),
26   ('Concentrix Company', 10760, '2024-04-26'),
27   ('Concentrix Company', 8096, '2024-05-10'),
28   ('Concentrix Company', 17995, '2024-05-24'),
29   ('Concentrix Company', 18520, '2024-06-07'),
30   ('Concentrix Company', 8151, '2024-06-21'),
31   ('Concentrix Company', 2580, '2024-06-25'),
32   ('Concentrix Company', 6376, '2024-07-05'),
33
34
35
36
37
38
39
40
41

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: income\_id income\_source income\_amount income\_date

income 4 x

Action Output

- Time Action Message Duration / Fetch
- 10 18:43:15 create table income (income\_id INT PRIMARY KEY AUTO\_INCREMENT, income\_source VARCHAR(255), income\_amount DECIMAL(10,2), income\_date DATE) 0 row(s) affected 0.016 sec
- 11 18:43:17 select \* from income LIMIT 0, 500 0 row(s) returned 0.000 sec / 0.000 sec

Object Info Session

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Insertion of data into the "income" table

MySQL Workbench

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator Schemas personal\_finance

Tables debts expenses investments savings

Views monthly\_financial\_summ

Stored Procedures GetFinancialHealth GetFinancialHealth2

Administration Schemas Information

No object selected

MySQL Portfolio

```

27   ('Concentrix Company', 896, '2024-05-10'),
28   ('Concentrix Company', 17995, '2024-05-24'),
29   ('Concentrix Company', 18520, '2024-06-07'),
30   ('Concentrix Company', 8151, '2024-06-21'),
31   ('Concentrix Company', 2580, '2024-06-25'),
32   ('Concentrix Company', 6376, '2024-07-05'),
33   ('Amazon Company', 20761, '2024-07-12'),
34   ('Amazon Company', 10213, '2024-07-31'),
35   ('Concentrix Company', 5119, '2024-08-01'),
36   ('Amazon Company', 8926, '2024-08-28'),
37   ('FNA Company', 4650, '2024-12-18'),
38
39
40 select * from income;
41

```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: income\_id income\_source income\_amount income\_date

income 7 x

Action Output

- Time Action Message Duration / Fetch
- 14 18:45:09 select \* from income LIMIT 0, 500 16 row(s) returned 0.000 sec / 0.000 sec
- 15 18:45:24 select \* from income LIMIT 0, 500 16 row(s) returned 0.000 sec / 0.000 sec

Object Info Session

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Now, data for the "income" table has been inserted.

**MySQL Workbench**

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

personal\_finance

Tables

- debts
- expenses
- investments
- savings

Views

- monthly\_financial\_summ

Stored Procedures

- GetFinancialHealth
- GetFinancialHealth2

No object selected

```

1 # Expense Table
2
3 create table expenses
4   (
5     expense_id INT PRIMARY KEY AUTO_INCREMENT,
6     expense_category VARCHAR(255),
7     expense_amount DECIMAL (10,2),
8     expense_date DATE
9   )
10
11
12 select * from expenses;
13
14
15 • insert into expenses (expense_category, expense_amount, expense_date)
16 values
17 ('Other', 99, '2024-01-01'),
18 ('Physical', 60, '2024-01-06'),
19 ('Rent', 3300, '2024-01-06'),
20 ('Physical', 330, '2024-01-06'),
21 ('Physical', 60, '2024-01-09'),
22
23
24

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows: |

expense\_id expense\_category expense\_amount expense\_date

1	Other	99.00	2024-01-01
2	Physical	60.00	2024-01-06
3	Rent	3300.00	2024-01-06
4	Physical	60.00	2024-01-09
5	Physical	60.00	2024-01-10
6	Physical	554.00	2024-01-11
7	Physical	490.00	2024-01-11
8	Physical	60.00	2024-01-12
9	Physical	155.00	2024-01-12
10	Physical	60.00	2024-01-13

expenses 8 x

Output

Object Info Session

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Data for the "expense" table

**MySQL Workbench**

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

personal\_finance

Tables

- debts
- expenses
- investments
- savings

Views

- monthly\_financial\_summ

Stored Procedures

- GetFinancialHealth
- GetFinancialHealth2

No object selected

```

861 # Investment Table
862
863 create table investments
864   (
865     investment_id INT PRIMARY KEY AUTO_INCREMENT,
866     investment_category VARCHAR(255),
867     investment_amount DECIMAL (10,2),
868     investment_return DECIMAL (10,2),
869     investment_date DATE
870   )
871
872
873 select * from investments;
874
875
876 • insert into investments (investment_category, investment_amount, investment_return, investment_date)
877 values
878 ('Pag-Ibig Contribution', 100, 110.25, '2024-01-11'),
879 ('Pag-Ibig Contribution', 1000, 1055.25, '2024-01-11'),
880 ('Banks', 20, 21, '2024-07-11'),
881 ('Banks', 25, 26.25, '2024-07-11'),
882 ('Banks', 10, 10.5, '2024-07-12'),
883 ('Lending', 500, 525.00, '2024-12-25'),
884
885
886

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

investment\_id investment\_category investment\_amount investment\_return investment\_date

1	Pag-Ibig Contribution	100.00	110.25	2024-01-11
2	Pag-Ibig Contribution	1000.00	1055.25	2024-01-11
3	Banks	20.00	21.00	2024-07-11
4	Banks	25.00	26.25	2024-07-11
5	Banks	10.00	10.50	2024-07-12
6	Lending	500.00	525.00	2024-12-25
7	Lending	900.00	915.00	2024-12-25

investments 9 x

Output

Object Info Session

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Data for the "investment" table

MySQL Workbench

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS personal\_finance

Tables debts expenses investments savings

Views monthly\_financial\_summ

Stored Procedures GetFinancialHealth GetFinancialHealth2

No object selected

MySQL Portfolio

```
888 # Savings Table
889 • create table savings
890     (saving_id INT PRIMARY KEY AUTO_INCREMENT,
891      saving_category VARCHAR(255),
892      saving_amount DECIMAL (10,2),
893      saving_return DECIMAL (10,2),
894      saving_date DATE
895 )
896
897
898 • select * from savings
899
900 • insert into savings (saving_category, saving_amount, saving_return, saving_date)
901 values
902 ('Emergency Fund', 28, 29.4, '2024-01-30'),
903 ('Emergency Fund', 20, 21, '2024-01-30'),
904 ('Emergency Fund', 15, 15.75, '2024-02-02'),
905 ('Emergency Fund', 17, 17.85, '2024-02-02'),
906 ('Emergency Fund', 13, 13.65, '2024-02-06'),
907 ('Emergency Fund', 13, 13.65, '2024-02-06')
```

Result Grid

saving_id	saving_category	saving_amount	saving_return	saving_date
1	Emergency Fund	28.00	29.40	2024-01-30
2	Emergency Fund	20.00	21.00	2024-01-30
3	Emergency Fund	15.00	15.75	2024-02-02
4	Emergency Fund	17.00	17.85	2024-02-02
5	Emergency Fund	13.00	13.65	2024-02-06
6	Emergency Fund	13.00	13.65	2024-02-06
7	Emergency Fund	13.00	13.65	2024-02-06

Data for the "savings" table

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

MySQL Workbench

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS personal\_finance

Tables debts expenses investments savings

Views monthly\_financial\_summ

Stored Procedures GetFinancialHealth GetFinancialHealth2

No object selected

MySQL Portfolio

```
986 # Debts Table
987 • create table debts
988     (debt_id INT PRIMARY KEY AUTO_INCREMENT,
989      debt_category VARCHAR(255),
990      debt_amount DECIMAL (10,2),
991      debt_return DECIMAL (10,2),
992      debt_date DATE
993 )
994
995
996 • select * from debts
997
998 • insert into debts (debt_category, debt_amount, debt_return, debt_date)
999 values
1000 ('Credit Card Debt', 250, 262.5, '2024-01-14'),
1001 ('Credit Card Debt', 65, 68.25, '2024-03-13'),
1002 ('Credit Card Debt', 13, 13.65, '2024-03-27'),
1003 ('Debt from Family/Friends', 300, 315.00, '2024-01-12'),
1004 ('Debt from Family/Friends', 500, 525.00, '2024-01-21'),
1005 ('Debt from Family/Friends', 75, 78.75, '2024-03-15')
```

Result Grid

debt_id	debt_category	debt_amount	debt_return	debt_date
1	Credit Card Debt	250.00	262.50	2024-01-14
2	Credit Card Debt	65.00	68.25	2024-03-13
3	Credit Card Debt	13.00	13.65	2024-03-27
4	Debt from Family/Friends	300.00	315.00	2024-01-12
5	Debt from Family/Friends	500.00	525.00	2024-01-21
6	Debt from Family/Friends	75.00	78.75	2024-03-15
7	Debt from Family/Friends	119.00	124.95	2024-01-27

Data for the "debts" table

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

## ⌚ Step 3: Financial Analysis Queries

### 3.1 Total Monthly Income vs. Expenses

- Query to compare total monthly income and expenses.

```
## FINANCIAL ANALYSIS QUERIES
# TOTAL MONTHLY INCOME VS EXPENSES
# Expenses per Month
select
    date_format(expense_date, '%Y-%m') as monthly_report,
    sum(expense_amount) as total_expenses
from expenses
group by monthly_report
;

# Income per month
select
    date_format(income_date, '%Y-%m') as monthly_report,
    sum(income_amount) as total_income
from income
group by monthly_report
;
```

monthly_report	total_expenses
2024-01	13258.00
2024-02	12138.00
2024-03	57316.00
2024-04	36724.00
2024-05	41487.68
2024-06	39895.08
2024-07	40119.74

#### EXPENSES PER MONTH

```
## FINANCIAL ANALYSIS QUERIES
# TOTAL MONTHLY INCOME VS EXPENSES
# Expenses per Month
select
    date_format(expense_date, '%Y-%m') as monthly_report,
    sum(expense_amount) as total_expenses
from expenses
group by monthly_report
;

# Income per month
select
    date_format(income_date, '%Y-%m') as monthly_report,
    sum(income_amount) as total_income
from income
group by monthly_report
;
```

monthly_report	total_income
2024-03	24157.00
2024-04	20676.00
2024-05	26091.00
2024-06	21171.00
2024-07	37350.00
2024-08	14045.00
2024-09	4656.00

#### INCOME PER MONTH

**MySQL Workbench**

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS personal\_finance

Tables debts expenses income investments savings

Views monthly\_financial\_summ GetFinancialHealth

No object selected

Administration Schemas Information

MySQL Portfolio

```

1020 # Monthly Income and Expenses: Joining and Combining Data
1021 SELECT
1022     income.monthly_report,
1023     income.total_income,
1024     expenses.total_expenses
1025
1026     FROM income
1027     GROUP BY monthly_report AS income
1028
1029 LEFT JOIN
1030     (
1031         SELECT
1032             DATE_FORMAT(expense_date, '%Y-%m') AS monthly_report,
1033             SUM(expense_amount) AS total_expenses
1034             FROM expenses
1035             GROUP BY monthly_report AS expenses
1036
1037             ON income.monthly_report = expenses.monthly_report
1038
1039

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Read Only | Context Help | Snippets

monthly_report	total_income	total_expenses
2024-03	24157.00	33159.00
2024-04	20676.00	16048.00
2024-05	26991.00	15396.50
2024-06	21171.00	18724.00
2024-07	37350.00	21761.58
2024-08	14045.00	10992.00
2024-12	4650.00	24216.18

Result 13 x

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

**JOINING MONTHLY INCOME AND EXPENSES**

### 3.2 Expense Breakdown by Category

- SQL query to categorize and summarize expenses.

**MySQL Workbench**

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS personal\_finance

Tables debts expenses income investments savings

Views monthly\_financial\_summ GetFinancialHealth

No object selected

Administration Schemas Information

MySQL Portfolio

```

1032 sum(income_amount) as total_income
1033
1034 from income
1035 group by monthly_report
1036
1037
1038 # EXPENSE BREAKDOWN BY CATEGORY
1039 select
1040     expense_category, sum(expense_amount) as total_expenses
1041
1042     from expenses
1043     group by expense_category
1044
1045
1046 # INVESTMENT RETURNS (ROI CALCULATION)
1047
1048 select
1049     investment_category, investment_amount, investment_return,
1050     FORMAT(((investment_return - investment_amount) / investment_amount)*100,2) as roi_percentage -- TO FORMAT RIGHT AWAY THE RESULT
1051     from investments
1052

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Read Only | Context Help | Snippets

expense_category	total_expenses
Other	15259.58
Physical	84625.00
Rent	39368.50
Social	4305.00
Spiritual	5394.00
Travel	5528.00
Work	14477.75

Result 14 x

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

**EXPENSE BREAKDOWN BY CATEGORY**

### 3.3 Investment Returns (ROI Calculation)

- Query to calculate return on investment (ROI) over time.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** personal\_finance
- Tables:** debts, expenses, investments, savings
- Stored Procedures:** GetFinancialHealth, GetFinancialHealth2
- Query Editor:** The query is titled "Return on Investment (ROI) Calculation". It uses a GROUP BY clause on expense\_category and a SELECT statement to calculate ROI. The output is formatted to show investment\_category, investment\_amount, investment\_return, and roi\_percentage.
- Result Grid:** Shows 16 rows of data. The first few rows are:

investment_category	investment_amount	investment_return	roi_percentage
Pag-ibg Contribution	105.00	110.25	5.00
Pag-ibg Contribution	1005.00	1055.25	5.00
Banks	20.00	21.00	5.00
Banks	25.00	26.25	5.00
Banks	10.00	10.50	5.00
Lending	500.00	525.00	5.00
Lending	300.00	315.00	5.00
Lending	350.00	367.50	5.00
Pag-ibg Contribution	105.00	110.25	5.00
Pag-ibg Contribution	1005.00	1055.25	5.00
Banks	20.00	21.00	5.00
Banks	25.00	26.25	5.00
Others	10.00	10.50	5.00
- Output:** Action Output shows two log entries indicating the execution of the query.

### 3.4 Debt Summary with Interest Calculation

- Query to track debt amounts and interest payments.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** personal\_finance
- Tables:** debts, expenses, investments, savings
- Stored Procedures:** GetFinancialHealth, GetFinancialHealth2
- Query Editor:** The query is titled "Debt Summary with Interest Calculation". It uses a GROUP BY clause on debt\_category and a SELECT statement to calculate debt\_interest. The output is formatted to show debt\_category, debt\_amount, debt\_return, and debt\_interest.
- Result Grid:** Shows 17 rows of data. The first few rows are:

debt_category	debt_amount	debt_return	debt_interest
Credit Card Debt	250.00	262.50	12.50
Credit Card Debt	65.00	68.25	3.25
Credit Card Debt	12.00	12.65	0.65
Debt from Family/Friends	300.00	315.00	15.00
Debt from Family/Friends	500.00	525.00	25.00
Debt from Family/Friends	75.00	78.75	3.75
Debt from Family/Friends	119.00	124.95	5.95
Debt from Family/Friends	75.00	78.75	3.75
Others	80.00	84.00	4.00
Others	62.00	65.10	3.10
Personal Loans	229.00	240.45	11.45
Personal Loans	15.00	15.75	0.75
- Output:** Action Output shows two log entries indicating the execution of the query.

### 3.5 Sum for Each Table

- Aggregate calculations for each financial category.

# SUM FOR EACH TABLE

```
1661
1662
1663 select
1664     (select SUM(income_amount) from income) as total_income,
1665     (select SUM(expense_amount) from expenses) as total_expense,
1666     (select SUM(investment_amount) from investments) as total_investment,
1667     (select SUM(saving_amount) from savings) as total_saving,
1668     (select SUM(debt_amount) from debts) as total_debt
1669
1670
```

SUM for each table made

	total_income	total_expense	total_investment	total_saving	total_debt
	148140.00	349663.04	4630.00	22939.00	3796.00

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

No object selected

Object Info Session

Result 18 x

Action Output

#	Time	Action	Message	Duration / Fetch
25	19:15:17	select debt_category, debt_amount, debt_return, (debt_return - debt_amount) as debt_interest from debts	28 row(s) returned	0.000 sec / 0.000 sec
26	19:17:38	select (select SUM(income_amount) from income) as total_income, (select SUM(expense_amount) from expenses) as total_expense, (select SUM(investment_amount) from investments) as total_investment, (select SUM(saving_amount) from savings) as total_saving, (select SUM(debt_amount) from debts) as total_debt	1 row(s) returned	0.000 sec / 0.000 sec

SQLAdditions | < | > | ⌂ | ⌂ | Jump to

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

### 3.6 Total Net Worth Calculation

- Query to calculate overall financial standing.

# TOTAL NETWORTH CALCULATION

```
1670
1671
1672 select
1673     (select SUM(income_amount) from income)
1674     - (select SUM(expense_amount) from expenses)
1675     + (select SUM(investment_amount) from investments)
1676     + (select SUM(saving_amount) from savings)
1677     - (select SUM(debt_amount) from debts) as net_worth
1678
1679
```

TOTAL NETWORTH Calculation

net_worth
-177750.04

Result 19 x

Action Output

#	Time	Action	Message	Duration / Fetch
26	19:17:38	select (select SUM(income_amount) from income) as total_income, (select SUM(expense_amount) from expenses) as total_expense, (select SUM(investment_amount) from investments) as total_investment, (select SUM(saving_amount) from savings) as total_saving, (select SUM(debt_amount) from debts) as total_debt	1 row(s) returned	0.000 sec / 0.000 sec
27	19:19:42	select (select SUM(income_amount) from income) - (select SUM(expense_amount) from expenses) + (select SUM(investment_amount) from investments) + (select SUM(saving_amount) from savings) - (select SUM(debt_amount) from debts) as net_worth	1 row(s) returned	0.000 sec / 0.000 sec

SQLAdditions | < | > | ⌂ | ⌂ | Jump to

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

## Step 4: Automate Reports with Views & Stored Procedures

### 4.1 Create a View for Monthly Financial Overview

- View to display key financial metrics.

```
## AUTOMATE REPORTS WITH VIEWS & STORED PROCEDURES
# Creating a VIEW for the Monthly Financial Overview
CREATE VIEW monthly_financial_summary AS
SELECT
    monthly_report, -- [IT WILL EXIST BECAUSE OF THE SUBQUERIES BELOW]
    SUM(total_income) AS total_income, -- [THIS ALSO WORKS BECAUSE IT SUMS UP THE VALUES FROM THE DIFFERENT UNION ALL RESULTS BELOW]
    SUM(total_expense) AS total_expense, -- [...THESE ARE CALLED OUTER QUERIES]
    SUM(total_investment) AS total_investment, -- [...THEY ARE ALIASES DEFINED INSIDE EACH SELECT STATEMENT OF THE UNION ALL QUERY]
    SUM(total_saving) AS total_saving,
    SUM(total_debt) AS total_debt
FROM (
    -- Income
    SELECT DATE_FORMAT(income_date, 'YY-MM') AS monthly_report,
        SUM(income_amount) AS total_income,
        0 AS total_expense,
        0 AS total_investment,
        0 AS total_saving,
        0 AS total_debt
    FROM income
    GROUP BY monthly_report
    UNION ALL
    -- Expenses
    SELECT DATE_FORMAT(expense_date, 'YY-MM') AS monthly_report,
        0 AS total_income,
        SUM(expense_amount) AS total_expense,
        0 AS total_investment,
        0 AS total_saving,
        0 AS total_debt
    FROM expense
    GROUP BY monthly_report
    UNION ALL
    -- Debts
    SELECT DATE_FORMAT(debt_date, 'YY-MM') AS monthly_report,
        0 AS total_income,
        0 AS total_expense,
        0 AS total_investment,
        0 AS total_saving,
        SUM(debt_amount) AS total_debt
    FROM debts
    GROUP BY monthly_report
) AS financial_data
GROUP BY monthly_report
;
select * from monthly_financial_summary;
```

CREATING A VIEW for the Monthly Financial Overview

monthly_report	total_income	total_expense	total_investment	total_saving	total_debt
2024-03	24157.00	5716.00	0.00	2545.00	804.00
2024-04	20676.00	36724.00	0.00	779.00	150.00
2024-05	26991.00	41487.68	0.00	70.00	0.00
2024-06	21171.00	38893.08	0.00	93.00	0.00
2024-07	37350.00	59110.74	110.00	679.00	0.00
2024-08	14045.00	25042.20	0.00	71.00	0.00
2024-12	4650.00	28864.09	2300.00	1157.00	0.00
2024-01	0.00	12358.00	2220.00	308.00	2384.00
2024-02	0.00	12138.00	0.00	398.00	458.00
2024-09	0.00	18313.25	0.00	14254.00	0.00

## 4.2 Stored Procedure for Financial Health by Month

- Automated query to analyze finances month by month.

```
# STORED PROCEDURE FOR FINANCIAL HEALTH BY ONE MONTH
DELIMITER //
-- Used to change the default delimiter ";" to "//"
CREATE PROCEDURE GetFinancialHealth(IN month_year VARCHAR(7))
BEGIN
    SELECT
        (SELECT SUM(income_amount) FROM income WHERE DATE_FORMAT(income_date, '%Y-%m') = month_year) AS total_income,
        (SELECT SUM(expense_amount) FROM expenses WHERE DATE_FORMAT(expense_date, '%Y-%m') = month_year) AS total_expenses,
        (SELECT SUM(investment_amount) FROM investments WHERE DATE_FORMAT(investment_date, '%Y-%m') = month_year) AS total_investments,
        (SELECT SUM(saving_amount) FROM savings WHERE DATE_FORMAT(saving_date, '%Y-%m') = month_year) AS total_savings,
        (SELECT SUM(debt_amount) FROM debts WHERE DATE_FORMAT(debt_date, '%Y-%m') = month_year) AS total_debts;
END //
DELIMITER ;
-- changing the delimiter back to its default ";" again

CALL GetFinancialHealth('2024-01'); -- Checking if the code works
CALL GetFinancialHealth('2024-03');


```

Action	Time	Message	Duration / Fetch
28	19:22:25	CREATE VIEW monthly_financial_summary AS SELECT monthly_report. -- IT WILL EXIST BECAUSE O... Error Code: 1050. Table 'monthly_financial_summary' already exists	0.015 sec
29	19:26:49	select * from monthly_financial_summary LIMIT 0, 500 13 row(s) returned	0.000 sec / 0.000 sec
30	19:27:03	select * from monthly_financial_summary LIMIT 0, 500 13 row(s) returned	0.000 sec / 0.000 sec
31	19:28:00	CREATE PROCEDURE GetFinancialHealth(IN month_year VARCHAR(7)) BEGIN SELECT (SELEC... Error Code: 1304. PROCEDURE GetFinancialHealth already exists	0.000 sec
32	19:29:44	CALL GetFinancialHealth('2024-03') 1 row(s) returned	0.016 sec / 0.000 sec
33	19:29:48	CALL GetFinancialHealth('2024-01') 1 row(s) returned	0.000 sec / 0.000 sec
34	19:30:13	drop procedure if exists GetFinancialHealth 0 rows affected	0.000 sec
35	19:30:20	CREATE PROCEDURE GetFinancialHealth(IN month_year VARCHAR(7)) BEGIN SELECT (SELEC... 0 rows affected	0.000 sec

## 4.3 Stored Procedure for Financial Health Over Multiple Months

- Extended analysis over a chosen period.

```
# STORED PROCEDURE FOR FINANCIAL HEALTH BY ONE MONTH
DELIMITER //
-- Used to change the default delimiter ";" to "//"
CREATE PROCEDURE GetFinancialHealth(IN month_year VARCHAR(7))
BEGIN
    SELECT
        (SELECT SUM(income_amount) FROM income WHERE DATE_FORMAT(income_date, '%Y-%m') = month_year) AS total_income,
        (SELECT SUM(expense_amount) FROM expenses WHERE DATE_FORMAT(expense_date, '%Y-%m') = month_year) AS total_expenses,
        (SELECT SUM(investment_amount) FROM investments WHERE DATE_FORMAT(investment_date, '%Y-%m') = month_year) AS total_investments,
        (SELECT SUM(saving_amount) FROM savings WHERE DATE_FORMAT(saving_date, '%Y-%m') = month_year) AS total_savings,
        (SELECT SUM(debt_amount) FROM debts WHERE DATE_FORMAT(debt_date, '%Y-%m') = month_year) AS total_debts;
END //
DELIMITER ;
-- changing the delimiter back to its default ";" again

CALL GetFinancialHealth('2024-01'); -- Checking if the code works
CALL GetFinancialHealth('2024-03');


```

	total_income	total_expenses	total_investments	total_savings	total_debts
	24157.00	57316.00	2545.00	804.00	

Now, a working stored procedure.

## 4.4 Stored procedure for financial health over multiple months

MySQL Workbench

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- savings
- Views
- monthly\_financial\_summ
- Stored Procedures
- GetFinancialHealth2**
- Functions
- sakila
- Tables
- Views
- Stored Procedures

Administration Schemas Information

No object selected

```
-- STORED PROCEDURE FOR FINANCIAL HEALTH OVER MULTIPLE MONTHS
DELIMITER //
CREATE PROCEDURE GetFinancialHealth2(IN start_month VARCHAR(7), IN end_month VARCHAR(7))
BEGIN
    SELECT
        month_year,
        SUM(total_income) AS total_income,
        SUM(total_expenses) AS total_expenses,
        SUM(total_investments) AS total_investments,
        SUM(total_savings) AS total_savings,
        SUM(total_debts) AS total_debts
    FROM
        income
    WHERE DATE_FORMAT(income_date, '%Y-%m') BETWEEN start_month AND end_month;
END //
DELIMITER ;
```

Stored Procedure for Financial Health over MULTIPLE MONTHS

Output

#	Time	Action	Message	Duration / Fetch
36	19:33:41	CALL GetFinancialHealth('2024-03')	1 row(s) returned	0.000 sec / 0.000 sec
37	19:33:46	CALL GetFinancialHealth('2024-01')	1 row(s) returned	0.000 sec / 0.000 sec
38	19:33:49	CALL GetFinancialHealth('2024-03')	1 row(s) returned	0.000 sec / 0.000 sec
39	19:33:58	CALL GetFinancialHealth('2024-03')	1 row(s) returned	0.000 sec / 0.000 sec
40	19:37:04	drop procedure if exists GetFinancialHealth2	0 row(s) affected	0.000 sec
41	19:37:17	CREATE PROCEDURE GetFinancialHealth2(IN start_month VARCHAR(7), IN end_month VARCHAR(7)) BE	0 row(s) affected	0.000 sec

Action Output

Object Info Session

MySQL Workbench

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- savings
- Views
- monthly\_financial\_summ
- Stored Procedures
- GetFinancialHealth
- GetFinancialHealth2**
- Functions
- sakila
- Tables
- Views
- Stored Procedures

Administration Schemas Information

No object selected

```
SELECT
    DATE_FORMAT(debt_date, '%Y-%m') AS month_year,
    SUM(debt_amount) AS total_debts
FROM debts
WHERE DATE_FORMAT(debt_date, '%Y-%m') BETWEEN start_month AND end_month
GROUP BY month_year
ORDER BY month_year;
```

DELIMITER ;

```
CALL GetFinancialHealth2('2024-01', '2024-01');
CALL GetFinancialHealth2('2024-01', '2024-03');
```

Result Grid

month_year	total_income	total_expenses	total_investments	total_savings	total_debts
2024-01	0.00	13258.00	2220.00	208.00	2384.00

Result 30

Output

Object Info Session

Result Grid Form Editor

Read Only Context Help Snippets

CHECKING...

MySQL Workbench

unconnected Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

sakila

monthly\_financial\_summ

Stored Procedures

GetFinancialHealth

GetFinancialHealth2

Functions

Tables

Views

Stored Procedures

No object selected

MySQL Portfolio\*

SELECT  
DATE\_FORMAT(debt\_date, '%Y-%m') AS month\_year,  
0 AS total\_income,  
0 AS total\_expenses,  
0 AS total\_investments,  
0 AS total\_savings,  
SUM(debt\_amount) AS total\_debts  
FROM debts  
WHERE DATE\_FORMAT(debt\_date, '%Y-%m') BETWEEN start\_month AND end\_month  
GROUP BY month\_year  
) AS financial\_data  
GROUP BY month\_year  
ORDER BY month\_year;  
END //  
DELIMITER ;  
  
1256 • CALL GetFinancialHealth2('2024-01', '2024-01');  
1257 • CALL GetFinancialHealth2('2024-01', '2024-03');

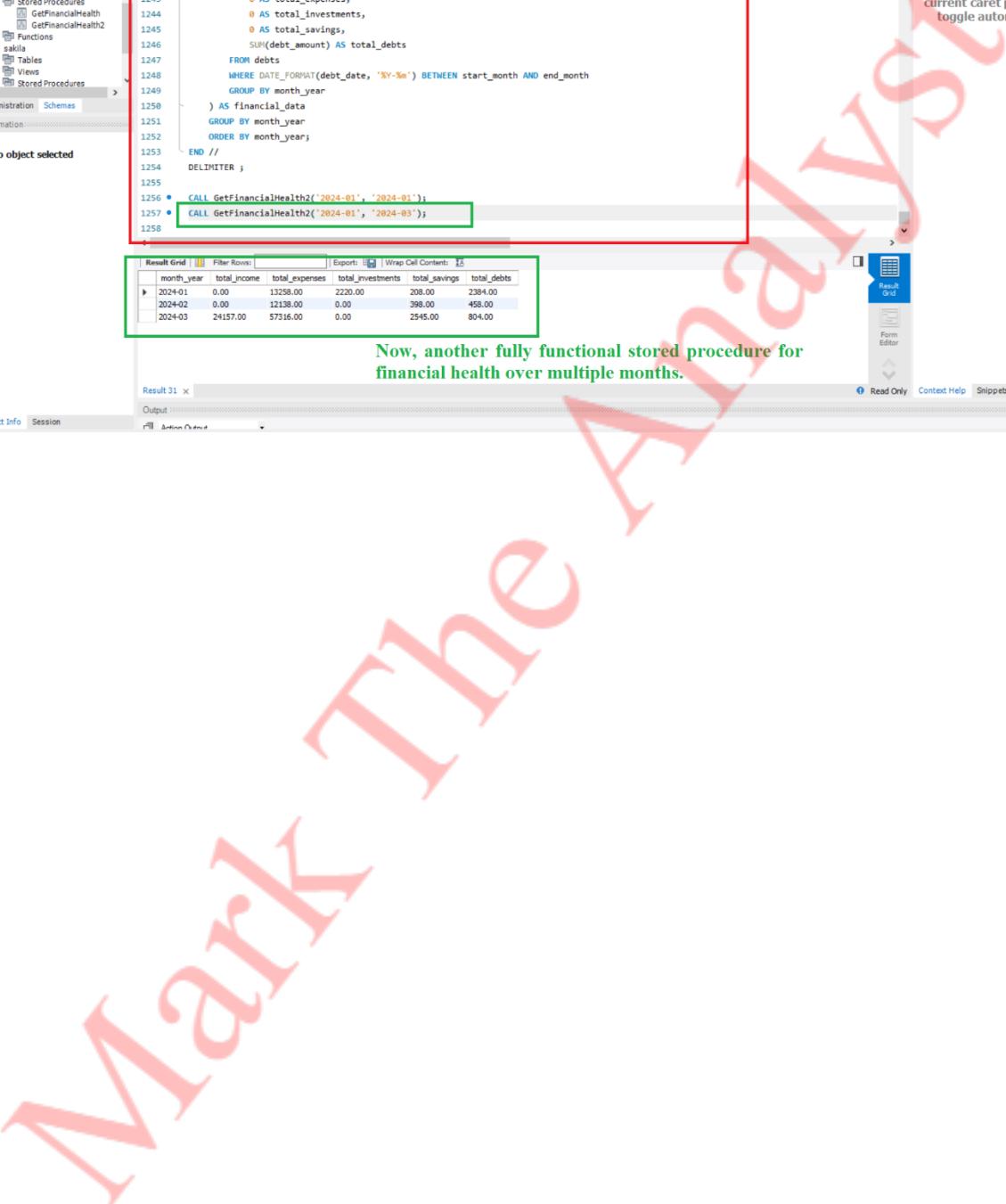
Result Grid | Filter Rows: Export: Wrap Cell Content: Result 31 x Output: Read Only Context Help Snippets

month\_year total\_income total\_expenses total\_investments total\_savings total\_debts

month_year	total_income	total_expenses	total_investments	total_savings	total_debts
2024-01	0.00	13258.00	2220.00	208.00	2384.00
2024-02	0.00	12138.00	0.00	398.00	458.00
2024-03	24157.00	57316.00	0.00	2545.00	804.00

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

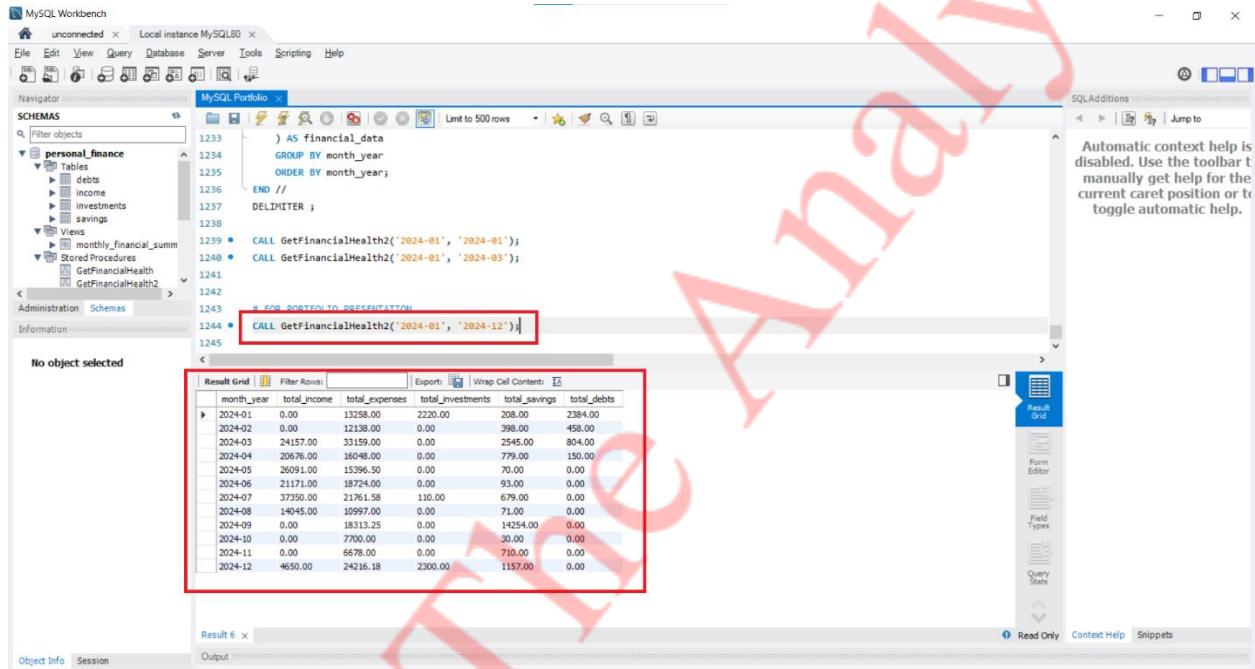
Now, another fully functional stored procedure for financial health over multiple months.



## DATA INSIGHTS

### 1. What is the general financial overview for each month in 2024?

- The monthly financial overview for 2024 highlights trends in income, expenses, investments, savings, and debts. It provides a clear snapshot of financial performance, showing growth, stability, or areas needing adjustment. This helps track progress and make informed financial decisions throughout the year.



The screenshot shows the MySQL Workbench interface with a query editor window. The code in the editor is:

```
1233 ) AS financial_data
1234 GROUP BY month_year
1235 ORDER BY month_year;
1236 END //
1237 DELIMITER ;
1238
1239 • CALL GetFinancialHealth2('2024-01', '2024-01');
1240 • CALL GetFinancialHealth2('2024-01', '2024-03');
1241
1242
1243 -- END REPORT TO PRESENTATION
1244 • CALL GetFinancialHealth2('2024-01', '2024-12');
1245
```

The result grid displays the following data:

month_year	total_income	total_expenses	total_investments	total_savings	total_debts
2024-01	0.00	13258.00	2220.00	208.00	2384.00
2024-02	0.00	12138.00	0.00	398.00	458.00
2024-03	24157.00	33159.00	0.00	2545.00	804.00
2024-04	20676.00	16048.00	0.00	779.00	150.00
2024-05	26911.00	15396.50	0.00	70.00	0.00
2024-06	21171.00	18724.00	0.00	93.00	0.00
2024-07	37350.00	21761.58	110.00	679.00	0.00
2024-08	14045.00	10997.00	0.00	71.00	0.00
2024-09	0.00	18313.25	0.00	14254.00	0.00
2024-10	0.00	7700.00	0.00	30.00	0.00
2024-11	0.00	6678.00	0.00	710.00	0.00
2024-12	46500.00	24216.18	2300.00	1157.00	0.00

### 2. How are expenses distributed across different categories, and what are the 3 major spending areas?

- Expenses are categorized into distinct groups (e.g., physical, travel, social, etc.), and as shown in the results, the top three spending categories are '**Physical**', which ranks highest with a total of ₱84,625 spent, followed by '**Rent**' at ₱39,368, and '**Family**' with ₱28,416.18, with the remaining categories following in order.
  - By breaking down spending in these categories, we can identify which areas consume the most resources and explore opportunities to reduce costs, providing valuable insights for optimizing budgeting strategies.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
personal_finance
Tables
expenses
income
investments
savings
Views
monthly_financial_summ
Stored Procedures
GetFinancialHealth
GetFinancialHealth2
Administration Schemas
Information
No object selected
Result Grid Filter Rows: Export: Wrap Cell Content: 
expense_category total_expenses
Physical 84625.00
Rent 39368.50
Family 28416.18
Jhea 18627.25
Other 15259.58
Travel 5528.00
Spiritual 5394.00
Social 4305.00
Result 12 x
Object Info Session Output

```

The screenshot shows the MySQL Workbench interface. In the SQL editor, a query is run against the 'personal\_finance' schema to calculate total monthly expenses by category. The results are displayed in a grid, with the entire grid highlighted by a red box.

expense_category	total_expenses
Physical	84625.00
Rent	39368.50
Family	28416.18
Jhea	18627.25
Other	15259.58
Travel	5528.00
Spiritual	5394.00
Social	4305.00

### 3. How much have investments earned in terms of returns?

- The *ROI formula* is used to calculate investment returns by comparing the total amount invested to the returns generated. According to the query, the total ROI for 2024 is **P231.50**, which accounts for **5% of the total investment**.
  - This calculation helps assess the performance of various investments over the year, providing valuable insights for making strategic decisions on where to allocate future funds.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas
personal_finance
Tables
expenses
income
investments
savings
Views
monthly_financial_summ
Stored Procedures
GetFinancialHealth
GetFinancialHealth2
Administration Schemas
Information
No object selected
Result Grid Filter Rows: Export: Wrap Cell Content: 
total_return total_investment ROI ROI_percentage
4861.50 4630.00 231.50 5.00
Result 16 x
Object Info Session Output

```

The screenshot shows the MySQL Workbench interface. In the SQL editor, a query is run against the 'personal\_finance' schema to calculate the total return, total investment, ROI, and ROI percentage for investments. The results are displayed in a grid, with the entire grid highlighted by a red box.

total_return	total_investment	ROI	ROI_percentage
4861.50	4630.00	231.50	5.00

#### 4. What is the total debt and interest paid?

- The debt summary query calculates the total outstanding debt, while interest calculations provide insights into the cost of borrowing over time. This helps in managing repayment strategies effectively. For 2024, the total debt stands at ₦3,796, with ₦189.90 paid in interest, representing 5% of the total debt.

A screenshot of MySQL Workbench showing a query in the SQL editor and its results in the Result Grid. The query calculates total debt return, total debt amount, interest, and interest percentage. The result shows a total debt of 3985.80, interest of 189.80, and an interest percentage of 5.00.

```
1283     From investments
1284
1285
1286 • select
1287     SUM(debt_return) as total_debt_return,
1288     SUM(debt_amount) as total_debt_amount,
1289     SUM(debt_return) - SUM(debt_amount) as interest,
1290     FORMAT(((SUM(debt_return) - SUM(debt_amount)) / SUM(debt_amount)) * 100,2) as interest_percentage
1291   from debts
1292
1293
1294 • select
1295
1296
1297
1298
1299
1300
1301
```

total_debt_return	total_debt_amount	interest	interest_percentage
3985.80	3796.00	189.80	5.00

#### 5. What is the total net worth?

- Total net worth is calculated by adding up assets (e.g., savings, investments, etc.) and subtracting liabilities (e.g., expenses, debts). As shown in the query, the total net worth for the year 2024 is ₦-29,610.51, indicating poor financial management during this period.
  - By tracking net worth over time, we can evaluate overall financial progress and assess the impact of income, spending, and investment decisions.

A screenshot of MySQL Workbench showing a query in the SQL editor and its results in the Result Grid. The query calculates net worth by summing income, expenses, investments, savings, and debts. The result shows a net worth of -29610.51.

```
1291     From debts
1292
1293
1294 • select
1295     (select SUM(income_amount) from income)
1296     - (select SUM(expense_amount) from expenses)
1297     + (select SUM(investment_amount) from investments)
1298     + (select SUM(saving_amount) from savings)
1299     - (select SUM(debt_amount) from debts) as net_worth
1300
1301
```

net_worth
-29610.51

## KEY INSIGHTS

- Monthly financial tracking in 2024 revealed varying trends in income, spending, savings, and debt, providing a clearer picture of overall financial flow and stability.
- The top 3 spending categories were **Physical** (₱84,625), **Rent** (₱39,368), and **Family** (₱28,416.18), indicating where most resources were allocated.
- Total investment returns reached **₱231.50**, representing a modest **5% ROI**, which helps evaluate the effectiveness of investment decisions.
- The total debt stood at **₱3,796**, with **₱189.90** in interest paid — highlighting the cost of borrowing and areas for improving repayment strategies.
- The **net worth for 2024 was negative ₱29,610.51**, pointing to overspending or underperformance in savings and asset growth throughout the year.

## RECOMMENDATION

- To improve overall financial health, consider reviewing and adjusting high-expense categories like Physical and Rent, while increasing focus on consistent savings and better-performing investments. Monitoring financial data monthly and reallocating funds strategically could help shift net worth into a positive trajectory.

## REFLECTION & NEXT STEPS

- This project strengthened my ability to write complex SQL queries and structure a database from scratch, helping me turn raw personal finance data into meaningful insights. It gave me hands-on experience applying SQL to real-world questions. Moving forward, I'd like to automate the data entry process using tools like Google Sheets and integrate a dashboard for live tracking and more dynamic analysis.