Birla Institute of Technology and Science

CS F212

Database Systems

HOSPITAL MANAGEMENT SYSTEM DOCUMENTATION

Submitted to: Dr. Amit Dua & Gaurav Kumar

Submitted by:

Jay Goyal (2021A7PS2418P)

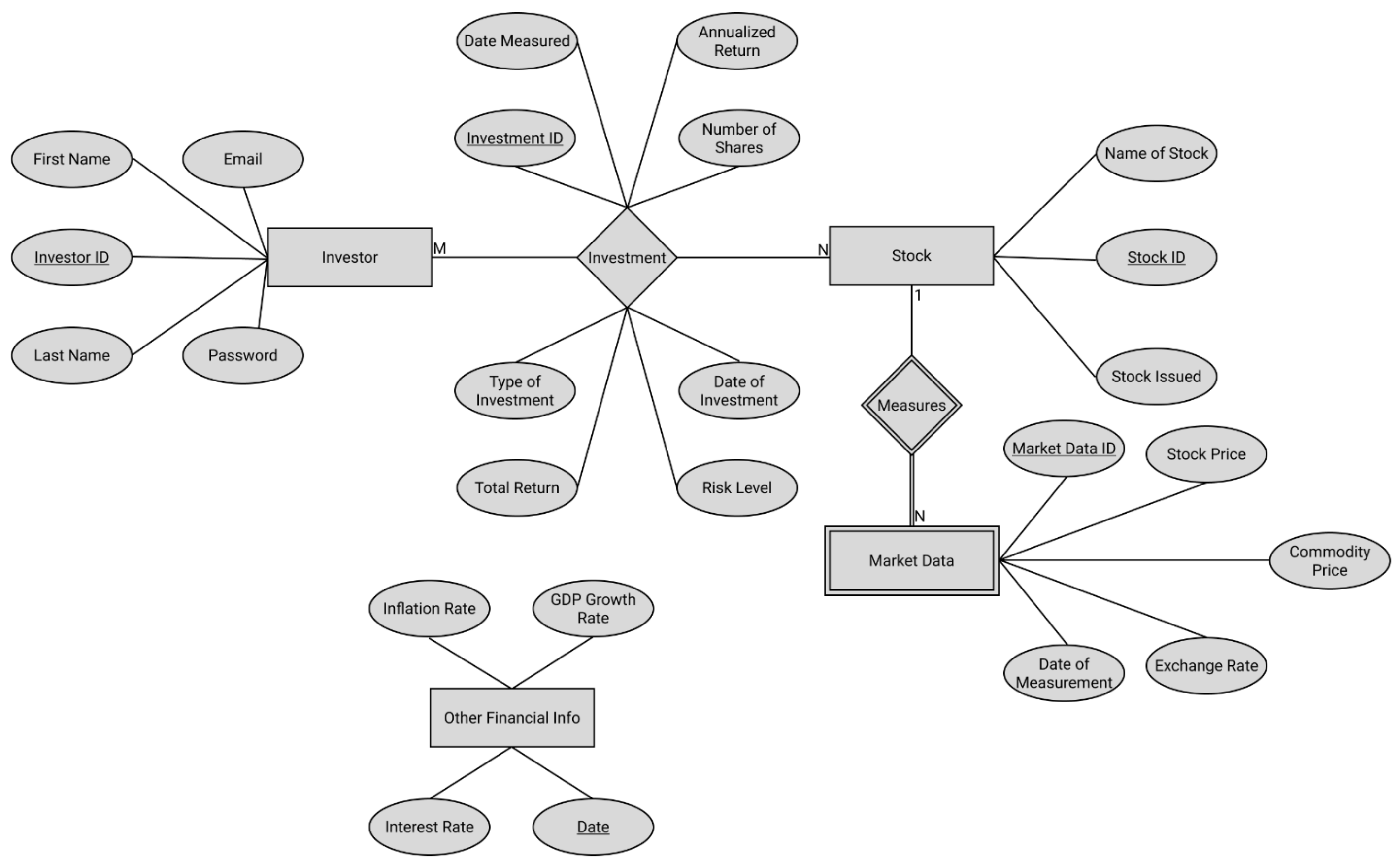
Introduction-

This project aims towards creating a database system for the purpose of hospital management. The schema has several different tables. There are several functionalities like creating/updating/deleting investments, filtering them in multiple ways to get the required information and calculating various metrics based on these metrics. These functions can be directly performed by executing the SQL queries or through the user interface.
There are several assumptions made for designing the schema which are mentioned in the document itself.

The snapshots of ER diagram and relational schema are pasted in the document. These can also be viewed in this figma file.
The drive links for the video recordings are attached at the end of this document.

Note: The only weak entity in this schema is Market Data. This is because market data can exist for a certain stock. In other words, without a stock existing, its market data can not exist. Other than that, apart from MarketDataID, which is an optional field to be included, (as in the data would still be complete without it being a part of the schema) the foreign key of Stock ID is required to uniquely identify an entry.

ER Diagram-



Entities with their attributes -

1) **Investor-** Investor ID(Primary Key-int), Email(varchar(50)), First Name(varchar(50)), Last Name(varchar(50)), Password(varchar(50)).

2) **Stock-** Stock ID(Primary Key-int), Name of Stock(varchar(50)), Stock Issued(int).

3) **Market Data-** Market Data ID(Primary Key-int), Stock Price(float), Commodity Price(float), Exchange Rate(float), Date of Measurement(date).

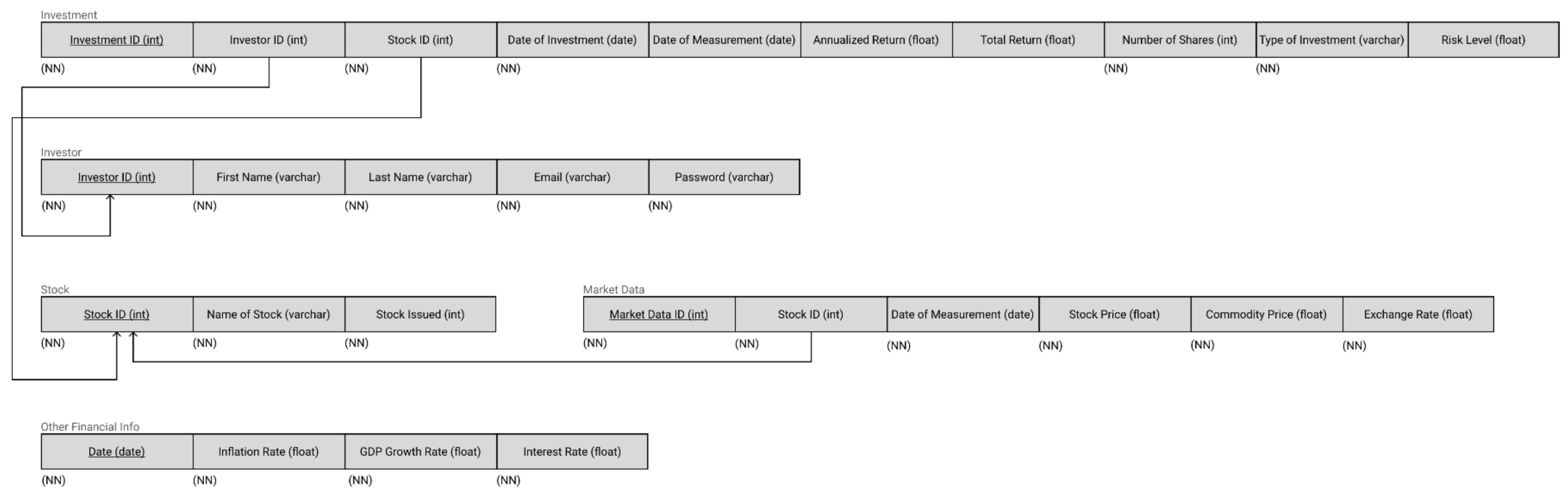
4) **Other Financial Info-** Date(Primary Key-date), Inflation Rate(float), GDP Growth Rate(float), Interest Rate(float).

Relationships-

1) **Investment-** This is a relation between an investor and a stock describing the details which stock he/she has invested in along with some performance metrics of the investment which can be *NULL*. It has a cardinality of M:N meaning that multiple investors can invest in multiple stocks. This relation itself has certain fields, namely:
Investment ID(Primary Key-int), Type of investment(varchar(5)), Date of Investment(date), Number of Shares(int), Date of Measurement(date), Total Return(float), Annualized Return(float), Risk Level(float)

2) **Measures-** This is a relation that tells which market data entry corresponds to which stock item. It has a cardinality of 1:N meaning that one market data entry corresponds to one stock item but one stock item can correspond to multiple market data entries.

<u>RELATIONAL SCHEMA-</u>



<u>Tables-</u>

1) **<u>Investor-</u>** *Investor ID* is the primary key. No foreign key.

2) **<u>Stock-</u>** *Stock ID* is the primary key. No foreign key.

3) **<u>Investor-</u>** *Investment ID* is the primary key. *Investor ID* and *Stock ID* are the foreign keys referencing the *investor* and *stock* tables respectively.

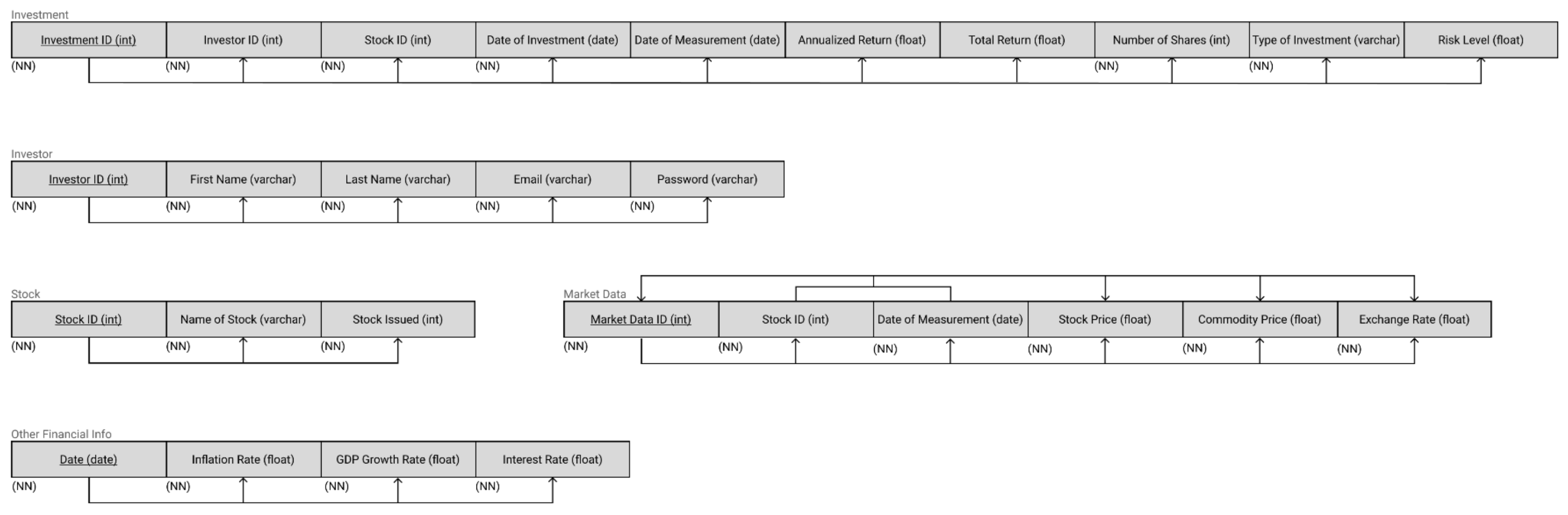
4) **<u>Market Data-</u>** *Market Data ID* is the primary key. *Stock ID* is the foreign key referencing the *stock* table.

5) **<u>Diagnosis-</u>** *Diagnosis ID* is the primary key. *Patient ID* and *Doctor ID* are the foreign keys referencing the *patient* and *doctor* table respectively.

6) **<u>Other Financial Info-</u>** *Date* is the primary key. No foreign key.

<u>Functional Dependencies (3 NF Form)-</u>

The relational schema shown above is directly converted to the 3NF form without any changes because the schema is already designed in such a way that there are no redundancies, no insert/update anomalies. The functional dependencies are demonstrated in the tables below:



The schema is in 3NF as:

1) There are no composite and multivalued attributes in the whole ER diagram. For *market data*, a separate table is made that has *Stock ID* and *Date of Investment* together as a candidate key (A separate field was created for the purpose of making the implementation cleaner). This separate table eliminates the redundancy that could have been generated, if *market data* was part of the *stock* table. Thus, the schema satisfies the first normal form.

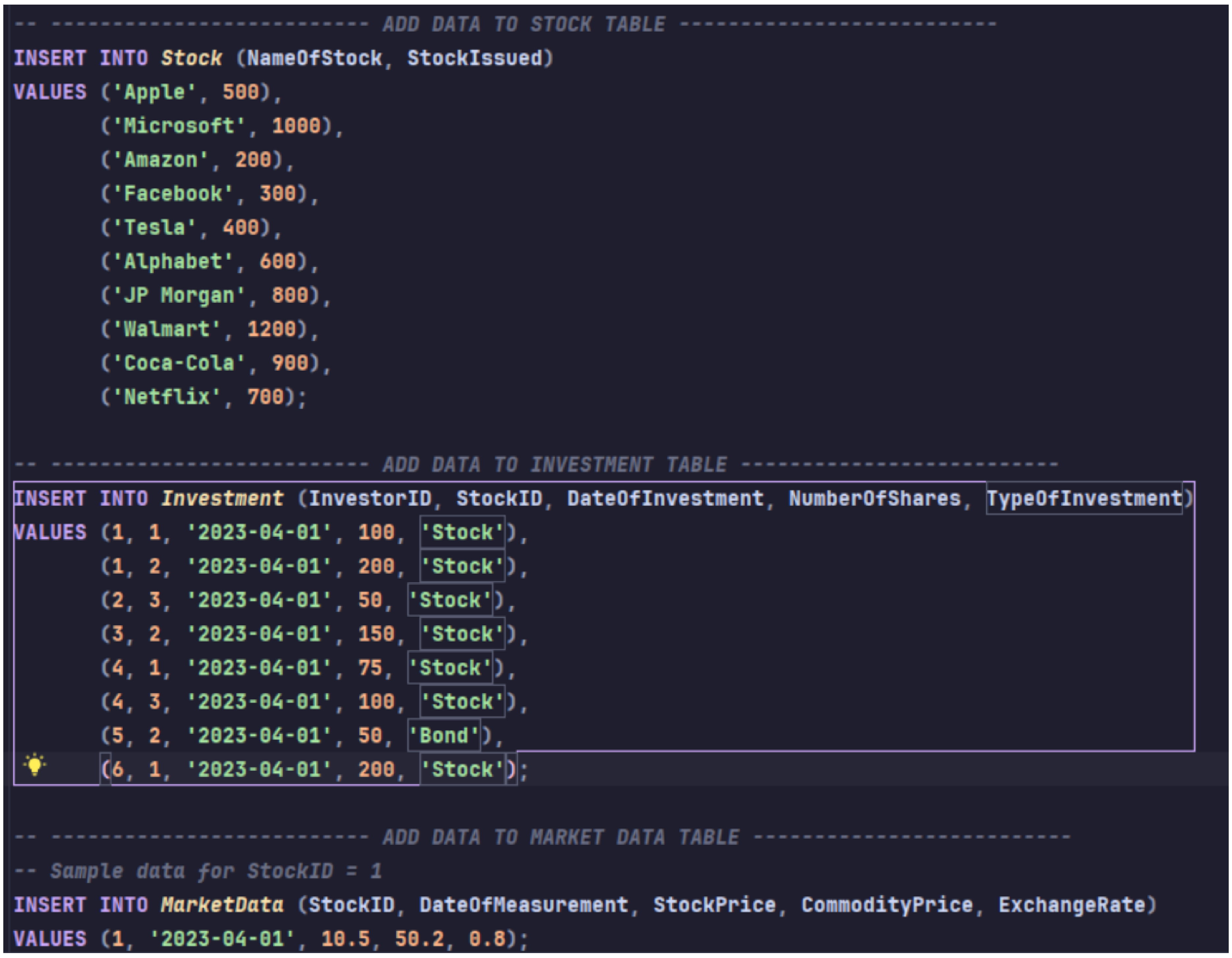
2) All functional dependencies are fully functional as in each table the prime fields (which can be a combination of attributes) can determine the other attributes uniquely. There is no functional dependency that is determined by an attribute which is not the prime fields, but is a part of the prime fields.
Thus, the schema satisfies the second normal form also.

3) There are no transitive functional dependencies in the schema. All the functional dependencies depend on the prime fields only which are the candidate keys.
Thus, the schema satisfies the third normal form also.
Note: The reason that prime fields is being said is because *Stock ID* and *Date* can determine any field in the *Market Data* table but they combined form a candidate key.

SQL Queries-

The script for creating the database and populating the data is in the *schema.sql* file. The required queries for user functionalities are there in the *queries.sql* file.

1) **Basic Insertion Queries-**



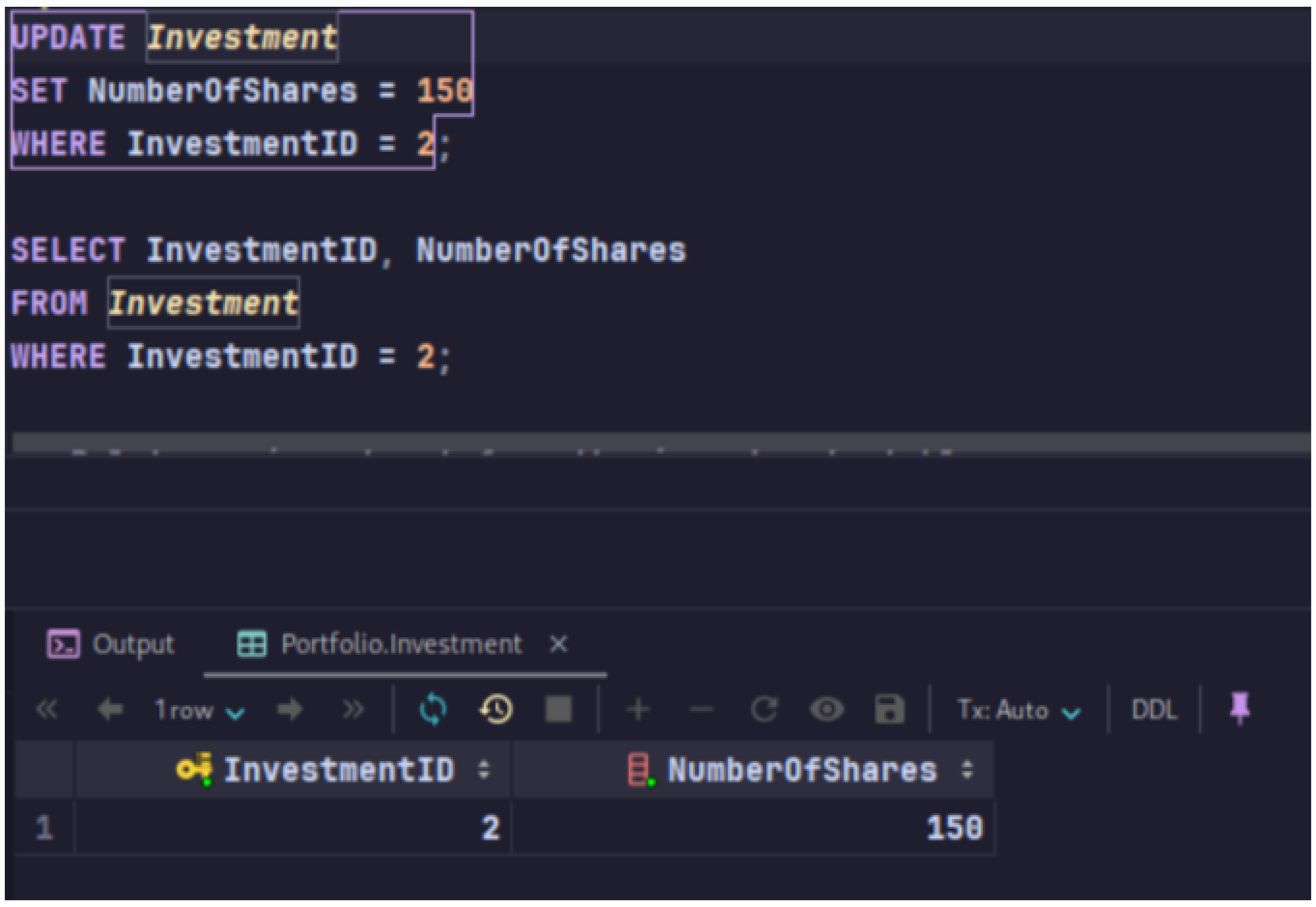
```
-- ------------------------- ADD DATA TO STOCK TABLE -------------------------
INSERT INTO Stock (NameOfStock, StockIssued)
VALUES ('Apple', 500),
       ('Microsoft', 1000),
       ('Amazon', 200),
       ('Facebook', 300),
       ('Tesla', 400),
       ('Alphabet', 600),
       ('JP Morgan', 800),
       ('Walmart', 1200),
       ('Coca-Cola', 900),
       ('Netflix', 700);

-- ------------------------- ADD DATA TO INVESTMENT TABLE -------------------------
INSERT INTO Investment (InvestorID, StockID, DateOfInvestment, NumberOfShares, TypeOfInvestment)
VALUES (1, 1, '2023-04-01', 100, 'Stock'),
       (1, 2, '2023-04-01', 200, 'Stock'),
       (2, 3, '2023-04-01', 50, 'Stock'),
       (3, 2, '2023-04-01', 150, 'Stock'),
       (4, 1, '2023-04-01', 75, 'Stock'),
       (4, 3, '2023-04-01', 100, 'Stock'),
       (5, 2, '2023-04-01', 50, 'Bond'),
       (6, 1, '2023-04-01', 200, 'Stock');

-- ------------------------- ADD DATA TO MARKET DATA TABLE -------------------------
-- Sample data for StockID = 1
INSERT INTO MarketData (StockID, DateOfMeasurement, StockPrice, CommodityPrice, ExchangeRate)
VALUES (1, '2023-04-01', 10.5, 50.2, 0.8);
```

These queries are only for the purpose of populating data. A lot of data is already populated through the *schema.sql* file.

2) **Update Number of Shares-**



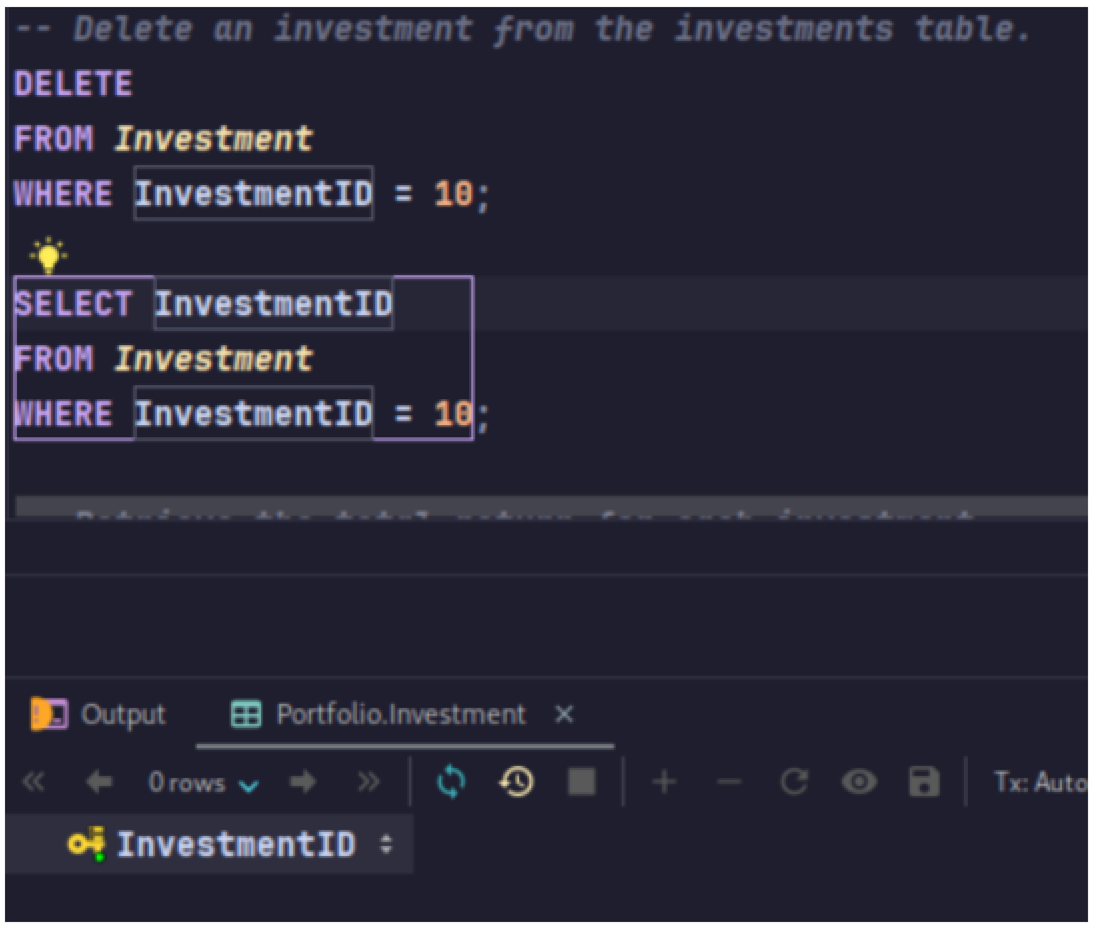
```
UPDATE Investment
SET NumberOfShares = 150
WHERE InvestmentID = 2;

SELECT InvestmentID, NumberOfShares
FROM Investment
WHERE InvestmentID = 2;
```

```
   InvestmentID   NumberOfShares
1             2              150
```

This query updates the number of shares issued in a particular investment

3) **Delete Investment-**



```
-- Delete an investment from the investments table.
DELETE
FROM Investment
WHERE InvestmentID = 10;

SELECT InvestmentID
FROM Investment
WHERE InvestmentID = 10;
```

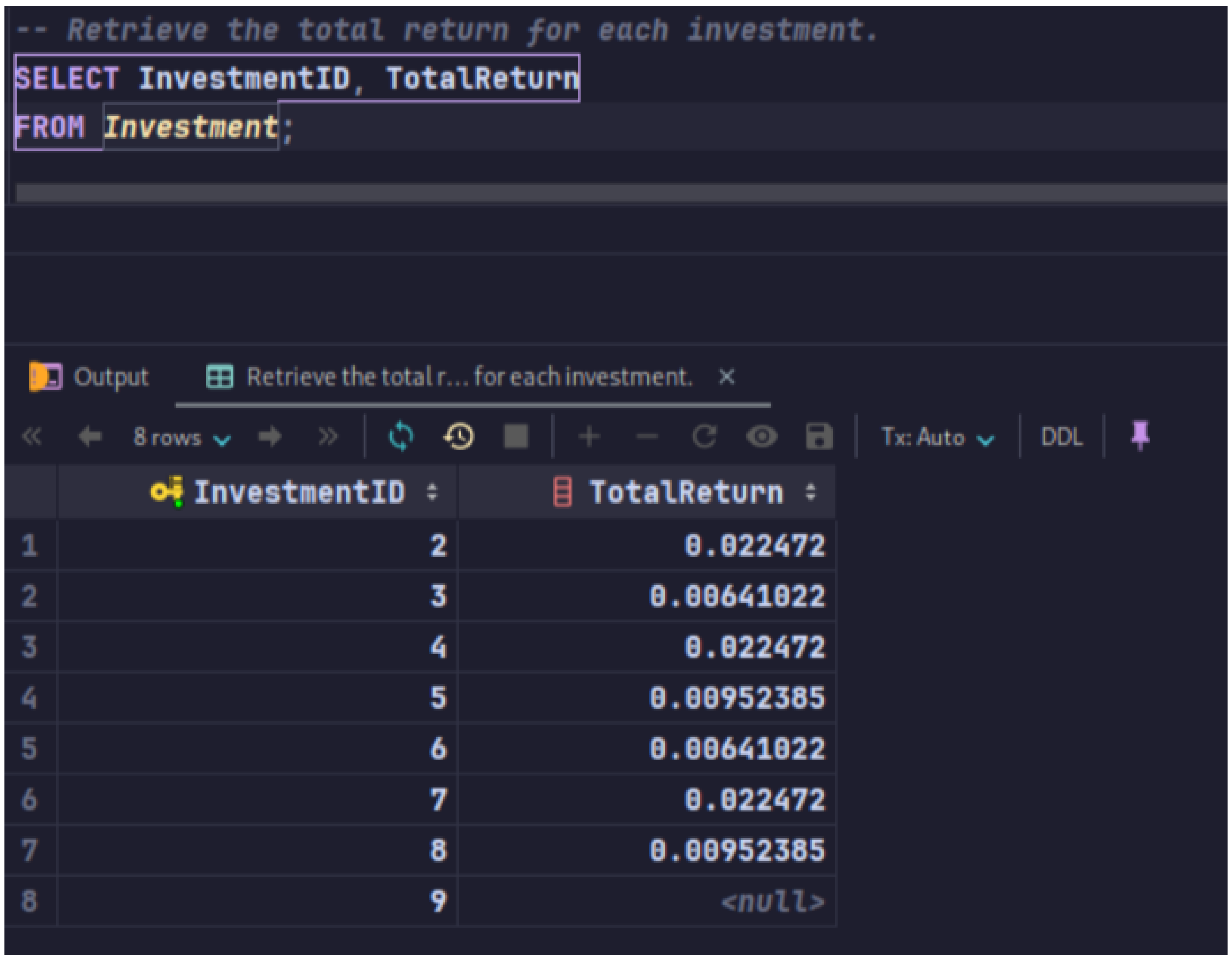
Output | Portfolio.Investment

0 rows | Tx: Auto

```
InvestmentID
```

This query deletes an investment from the table.

4) **Retrieve Total Return-**



```
-- Retrieve the total return for each investment.
SELECT InvestmentID, TotalReturn
FROM Investment;
```

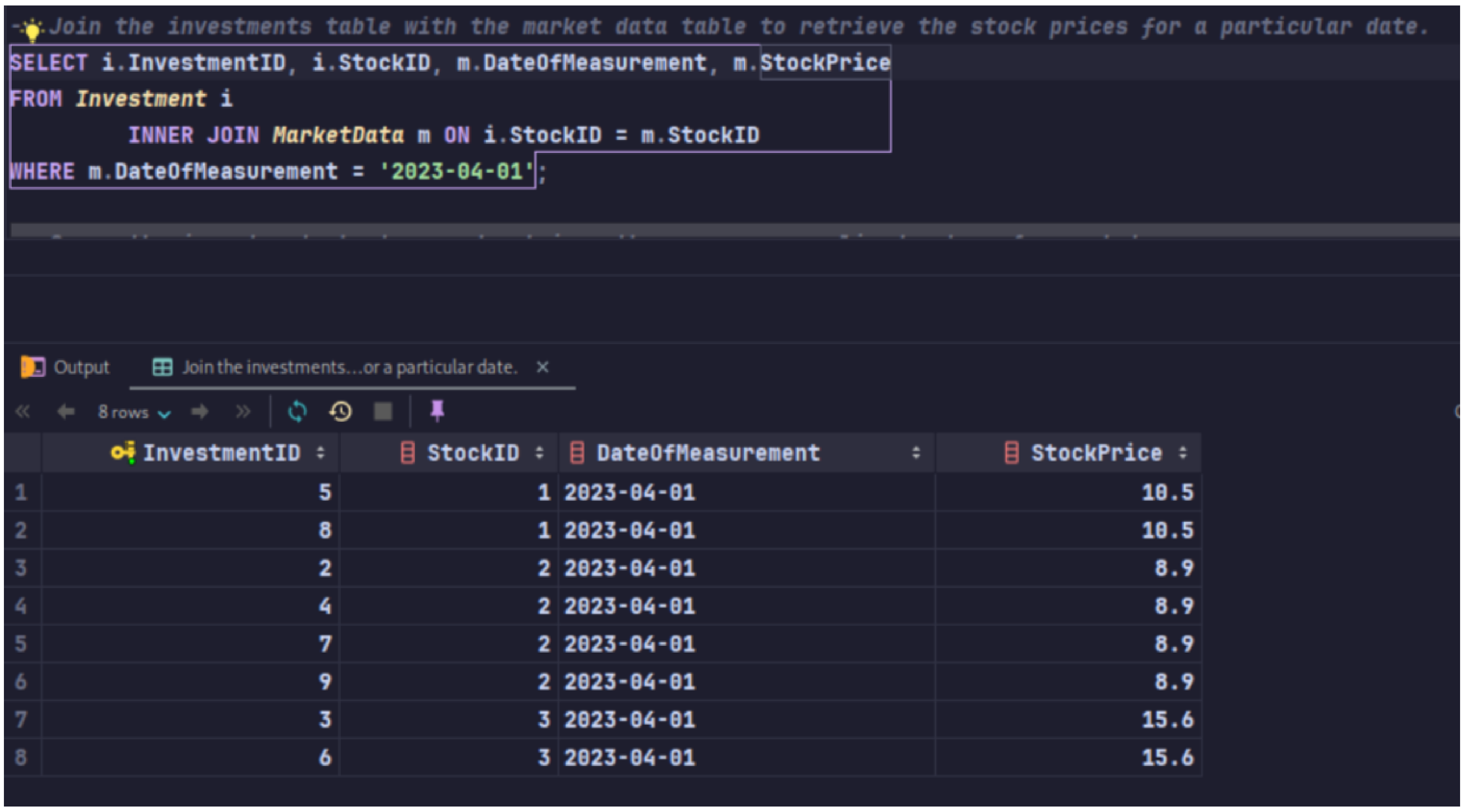
Output | Retrieve the total r… for each investment.

8 rows | Tx: Auto | DDL

```
   InvestmentID   TotalReturn
1             2      0.022472
2             3    0.00641022
3             4      0.022472
4             5    0.00952385
5             6    0.00641022
6             7      0.022472
7             8    0.00952385
8             9        <null>
```

This query retrieves the total return for each investment

5) **Retrive Stock Prices for a Date-**

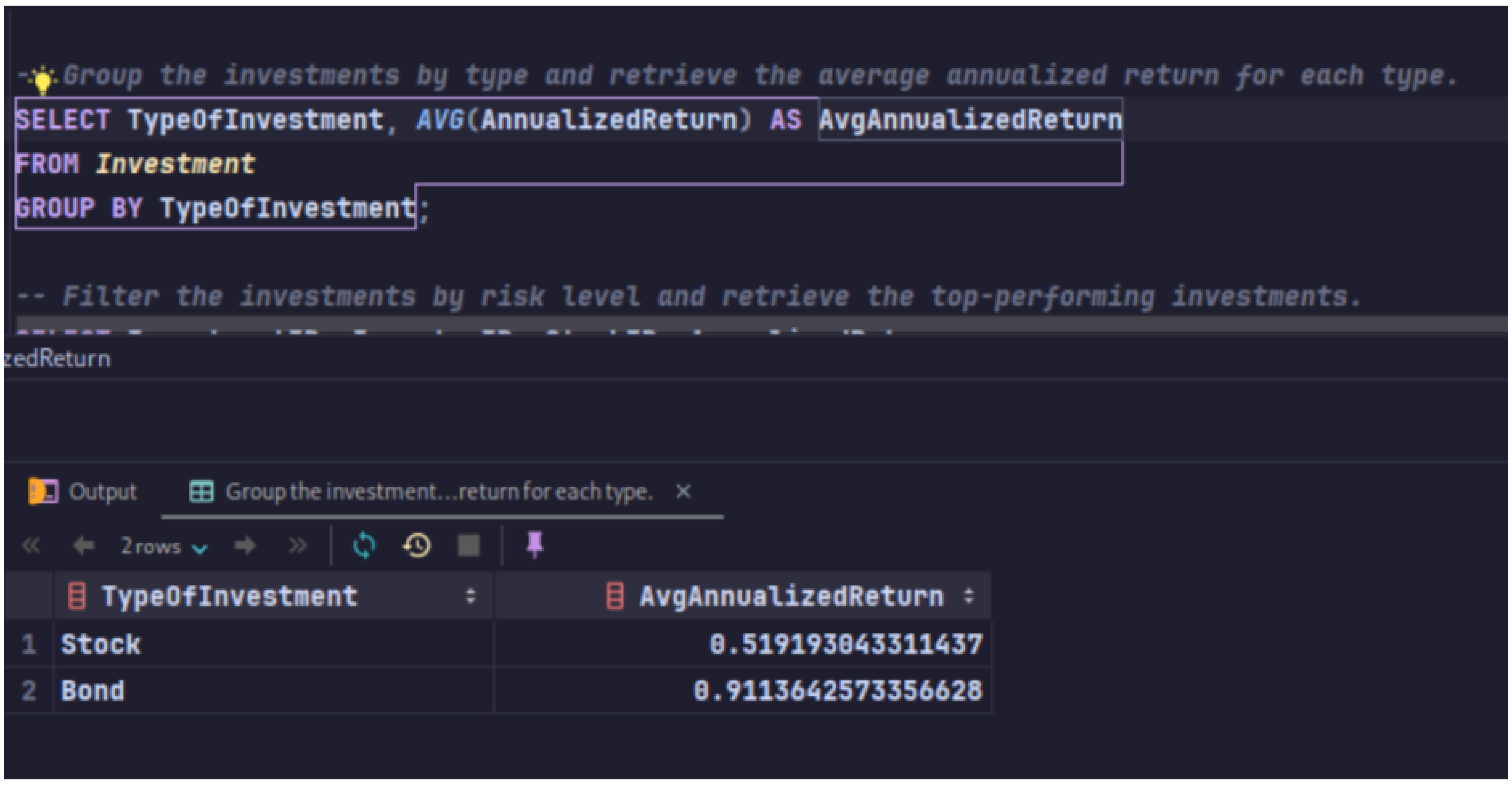


```
-- Join the investments table with the market data table to retrieve the stock prices for a particular date.
SELECT i.InvestmentID, i.StockID, m.DateOfMeasurement, m.StockPrice
FROM Investment i
         INNER JOIN MarketData m ON i.StockID = m.StockID
WHERE m.DateOfMeasurement = '2023-04-01';
```

	InvestmentID	StockID	DateOfMeasurement	StockPrice
1	5	1	2023-04-01	10.5
2	8	1	2023-04-01	10.5
3	2	2	2023-04-01	8.9
4	4	2	2023-04-01	8.9
5	7	2	2023-04-01	8.9
6	9	2	2023-04-01	8.9
7	3	3	2023-04-01	15.6
8	6	3	2023-04-01	15.6

This retrieves prices for the stocks invested in for a particular date for each stock.

6) **Get Average Annualized Return for Each Type of Investment-**



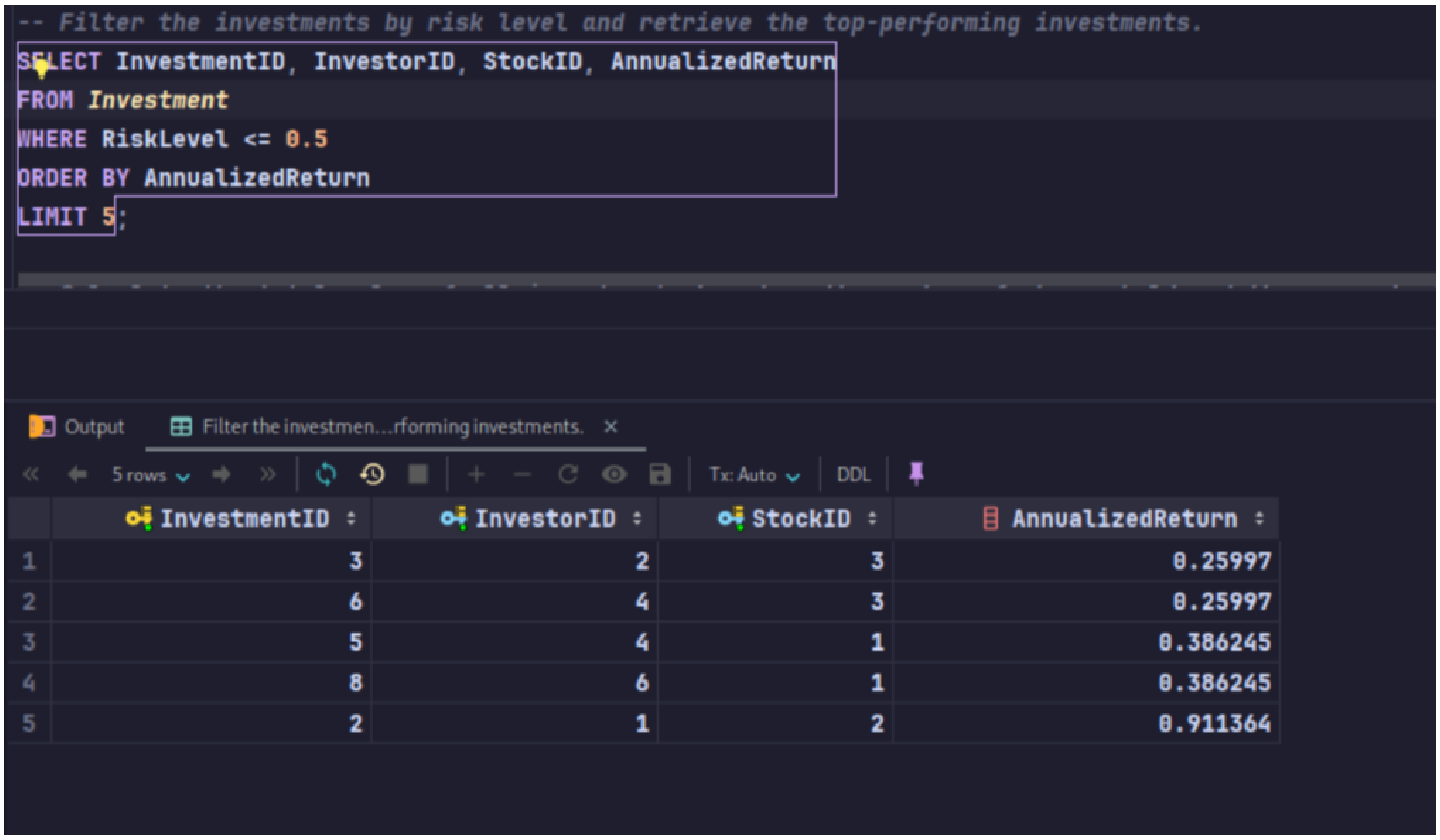
```
-- Group the investments by type and retrieve the average annualized return for each type.
SELECT TypeOfInvestment, AVG(AnnualizedReturn) AS AvgAnnualizedReturn
FROM Investment
GROUP BY TypeOfInvestment;

-- Filter the investments by risk level and retrieve the top-performing investments.
```

	TypeOfInvestment	AvgAnnualizedReturn
1	Stock	0.519193043311437
2	Bond	0.9113642573356628

This query groups investments by type and calculates the average annualized return for each type.

7) **Get top performing investments in terms of risk levels-**



```sql
-- Filter the investments by risk level and retrieve the top-performing investments.
SELECT InvestmentID, InvestorID, StockID, AnnualizedReturn
FROM Investment
WHERE RiskLevel <= 0.5
ORDER BY AnnualizedReturn
LIMIT 5;
```

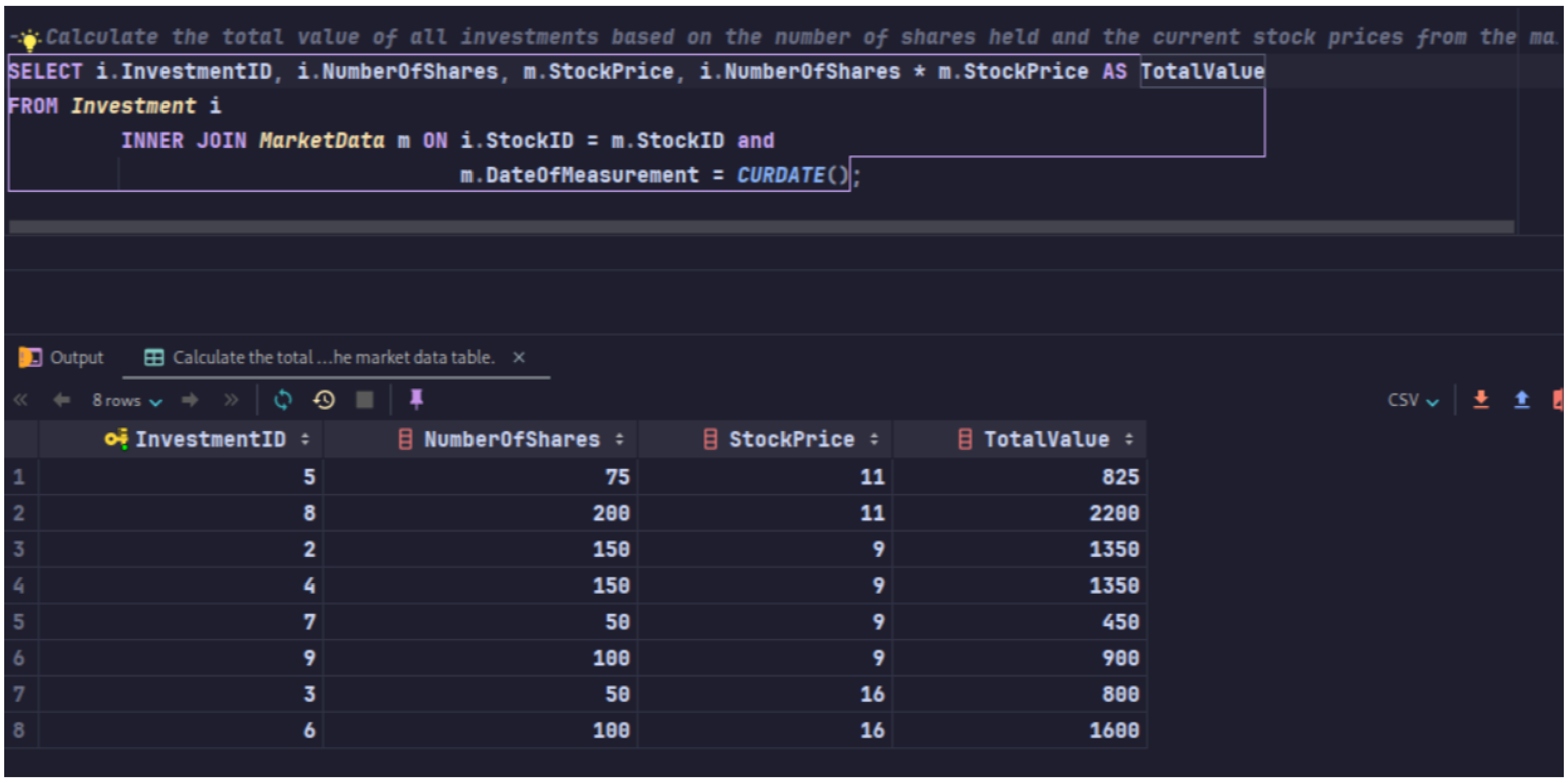
Output | Filter the investmen...rforming investments.

5 rows | Tx: Auto | DDL

	InvestmentID	InvestorID	StockID	AnnualizedReturn
1	3	2	3	0.25997
2	6	4	3	0.25997
3	5	4	1	0.386245
4	8	6	1	0.386245
5	2	1	2	0.911364

This query filters out the best investments in terms of the risk levels

8) **Calculate Total Value of Investments-**



```sql
-Calculate the total value of all investments based on the number of shares held and the current stock prices from the ma
SELECT i.InvestmentID, i.NumberOfShares, m.StockPrice, i.NumberOfShares * m.StockPrice AS TotalValue
FROM Investment i
        INNER JOIN MarketData m ON i.StockID = m.StockID and
                                   m.DateOfMeasurement = CURDATE();
```

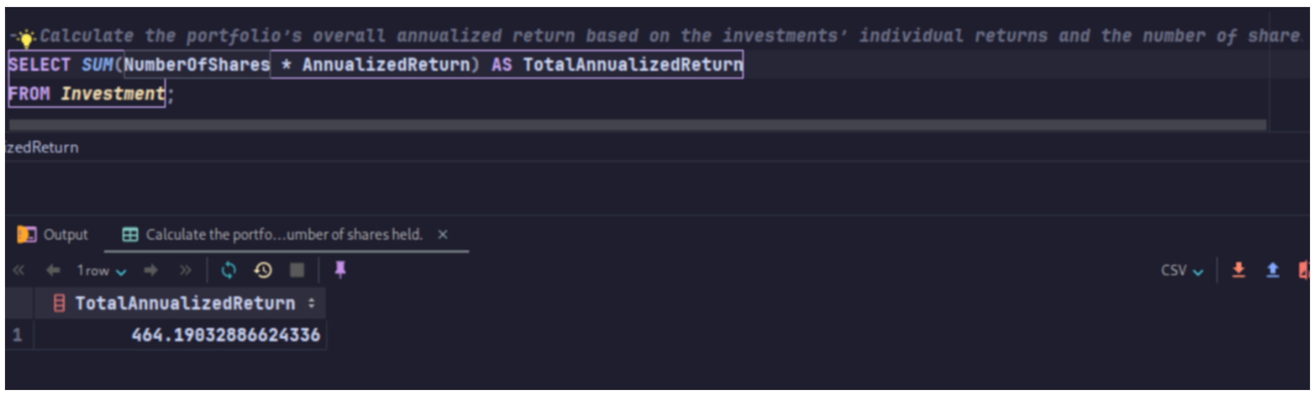
Output | Calculate the total ...he market data table.

8 rows | CSV

	InvestmentID	NumberOfShares	StockPrice	TotalValue
1	5	75	11	825
2	8	200	11	2200
3	2	150	9	1350
4	4	150	9	1350
5	7	50	9	450
6	9	100	9	900
7	3	50	16	800
8	6	100	16	1600

This query calculates the total values of all investments based on the current market data.

9) **Total Annualized Returns-**



```sql
-Calculate the portfolio's overall annualized return based on the investments' individual returns and the number of share
SELECT SUM(NumberOfShares * AnnualizedReturn) AS TotalAnnualizedReturn
FROM Investment;
```

zedReturn

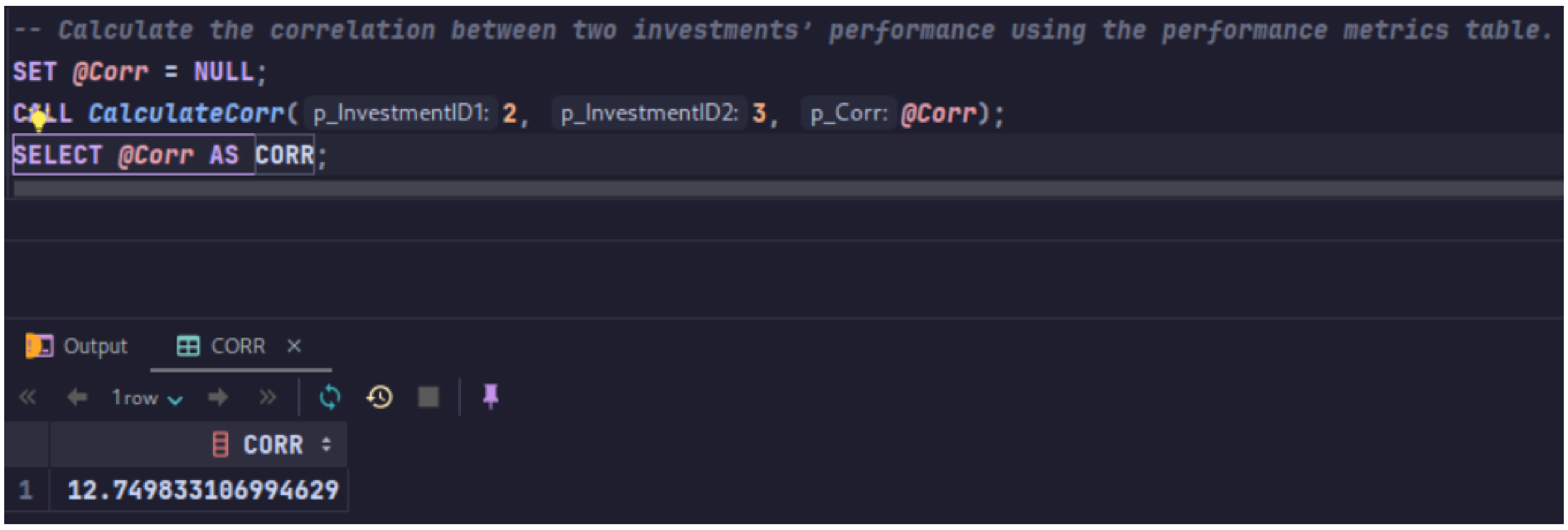
Output | Calculate the portfo...umber of shares held.

1 row | CSV

	TotalAnnualizedReturn
1	464.19032886624336

This query calculates the total annualized returns of all the investments combined.

10) Correlation-

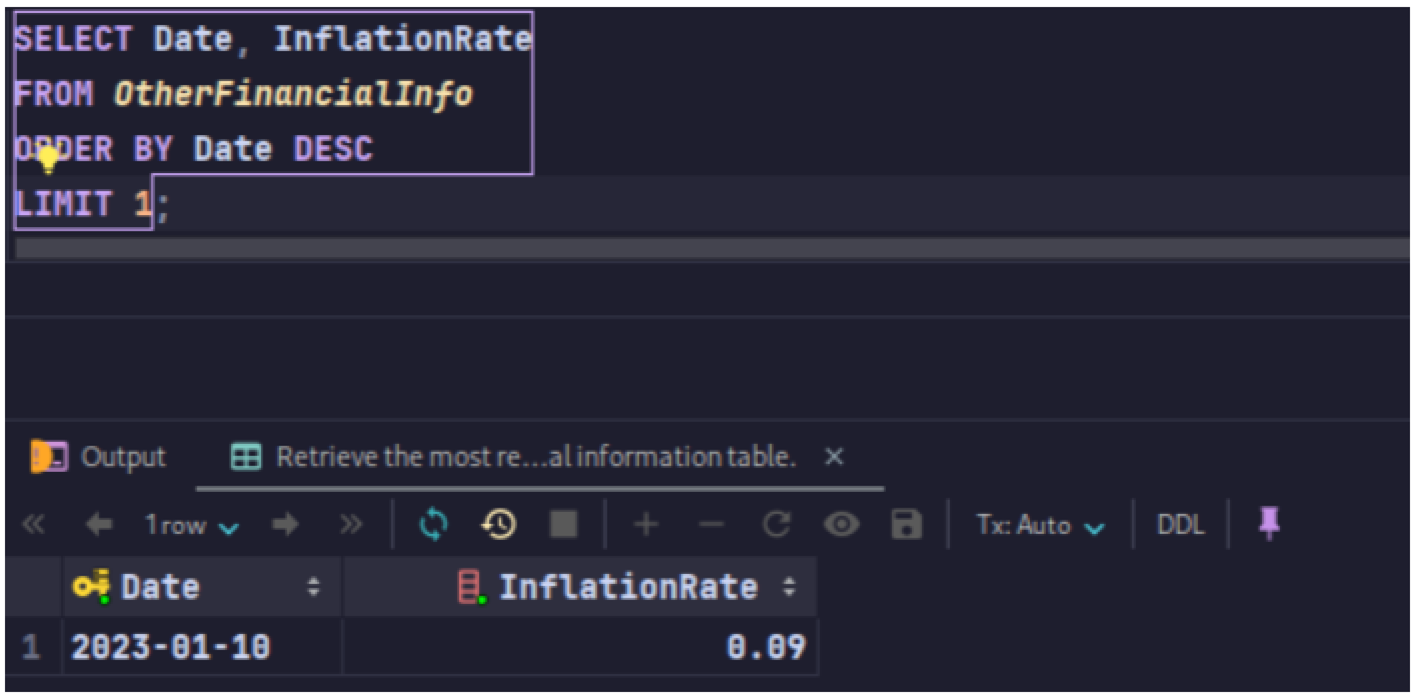


```
-- Calculate the correlation between two investments' performance using the performance metrics table.
SET @Corr = NULL;
CALL CalculateCorr( p_InvestmentID1: 2, p_InvestmentID2: 3, p_Corr: @Corr);
SELECT @Corr AS CORR;
```

```
  CORR
1 12.749833106994629
```

This query calculates the correlation between two relations. The procedure is defined at the end of *schema.sql*.
Note: It should be noted that while this query works, it is not present in the frontend. This is because this query generated syntax errors despite being correct.

11) Inflation Rate-

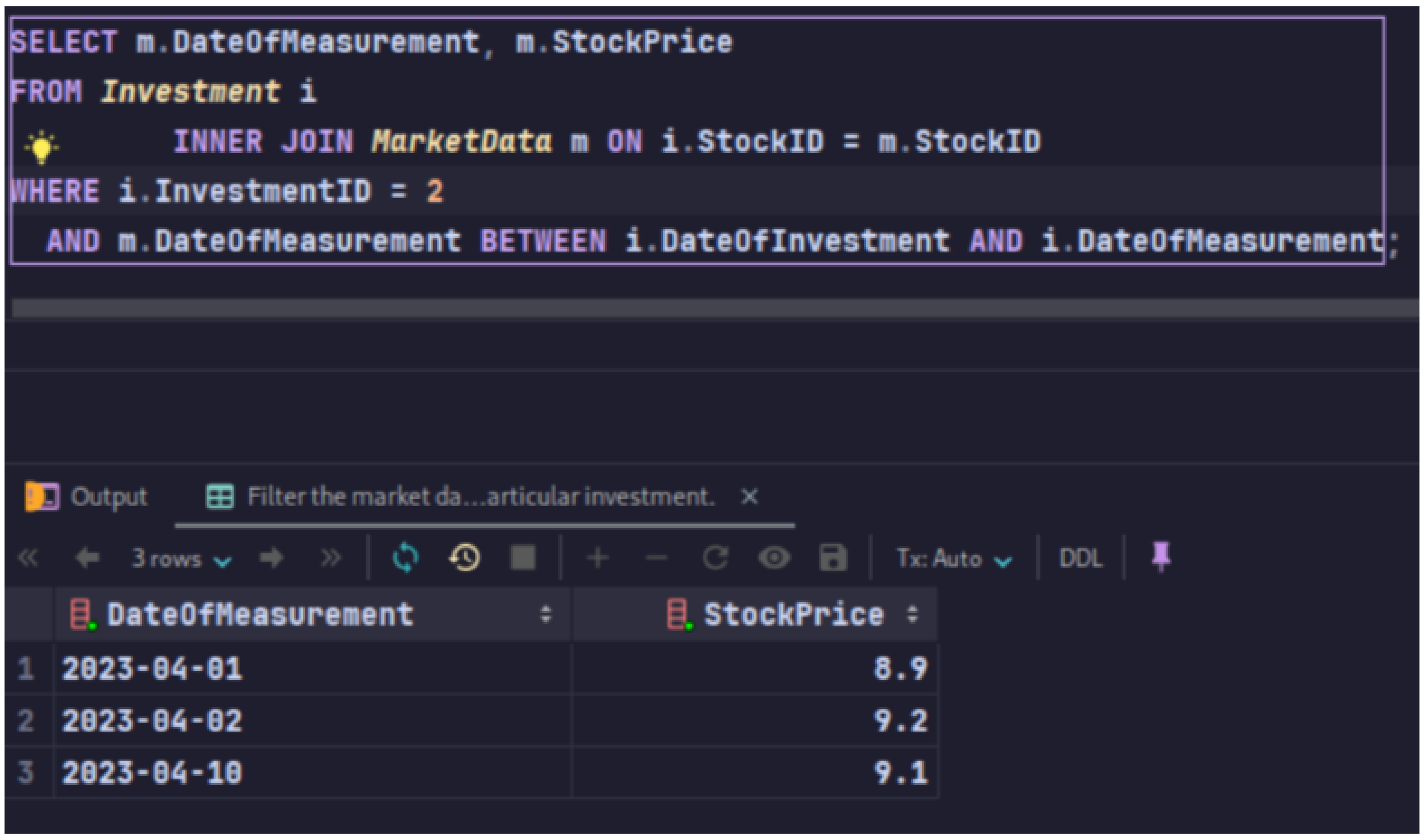


```
SELECT Date, InflationRate
FROM OtherFinancialInfo
ORDER BY Date DESC
LIMIT 1;
```

```
  Date        InflationRate
1 2023-01-10           0.09
```

This query retrieves the most recent inflation rate.

12) Filter Prices by Date Range-

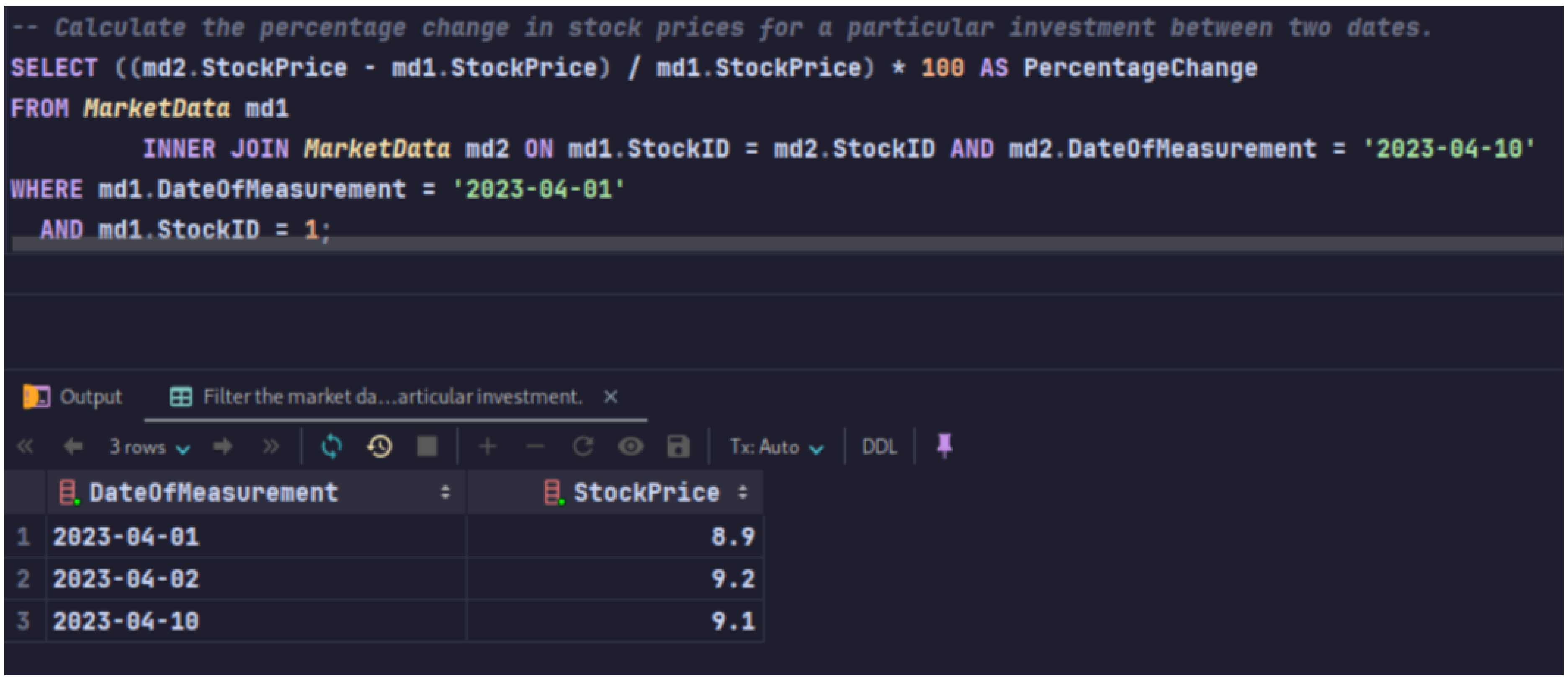


```
SELECT m.DateOfMeasurement, m.StockPrice
FROM Investment i
         INNER JOIN MarketData m ON i.StockID = m.StockID
WHERE i.InvestmentID = 2
  AND m.DateOfMeasurement BETWEEN i.DateOfInvestment AND i.DateOfMeasurement;
```

```
  DateOfMeasurement   StockPrice
1 2023-04-01                 8.9
2 2023-04-02                 9.2
3 2023-04-10                 9.1
```

This filters the prices of stock for a particular investment based on when the investment was made and when its metrics were last calculated.

13) <u>**Percent Change in Stock Prices-**</u>

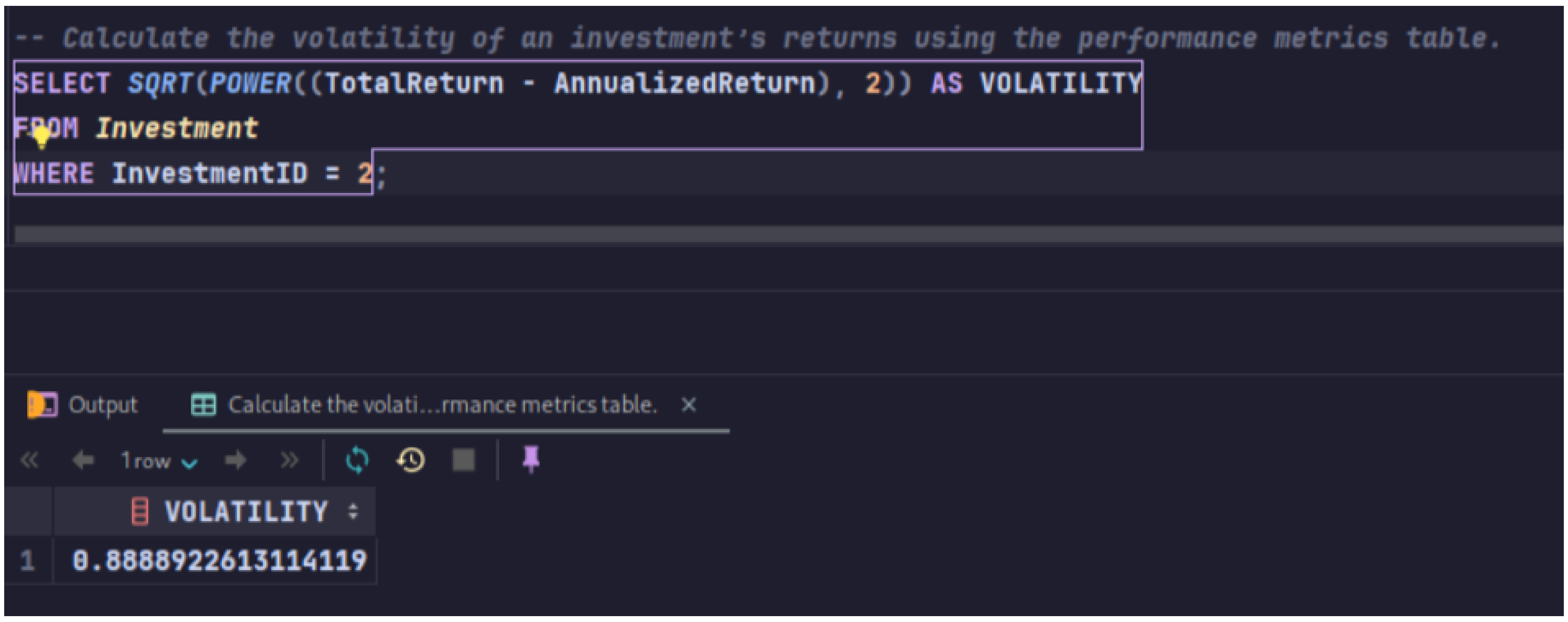


```sql
-- Calculate the percentage change in stock prices for a particular investment between two dates.
SELECT ((md2.StockPrice - md1.StockPrice) / md1.StockPrice) * 100 AS PercentageChange
FROM MarketData md1
         INNER JOIN MarketData md2 ON md1.StockID = md2.StockID AND md2.DateOfMeasurement = '2023-04-10'
WHERE md1.DateOfMeasurement = '2023-04-01'
  AND md1.StockID = 1;
```

	DateOfMeasurement	StockPrice
1	2023-04-01	8.9
2	2023-04-02	9.2
3	2023-04-10	9.1

Given two dates, this query calculates the percent change in the stock price.

14) <u>**Volatility-**</u>

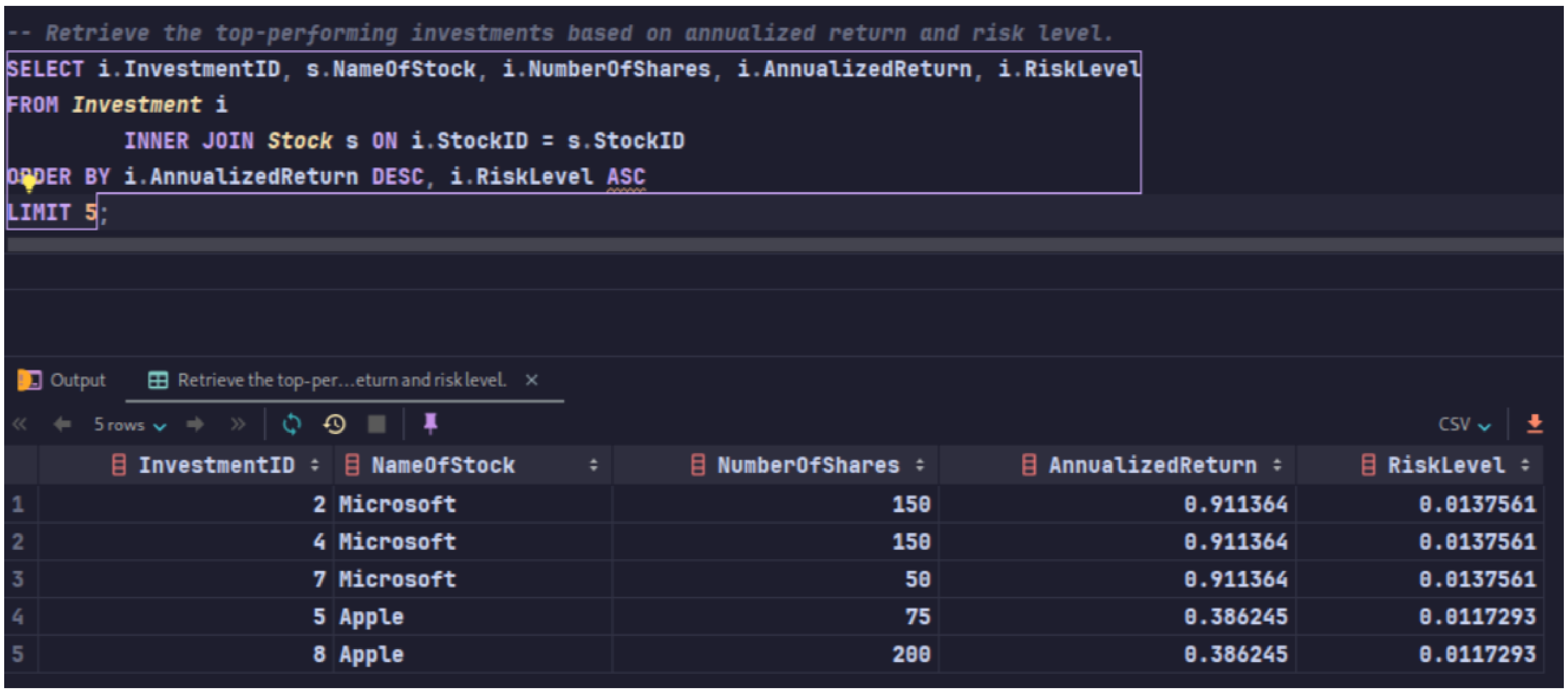


```sql
-- Calculate the volatility of an investment's returns using the performance metrics table.
SELECT SQRT(POWER((TotalReturn - AnnualizedReturn), 2)) AS VOLATILITY
FROM Investment
WHERE InvestmentID = 2;
```

	VOLATILITY
1	0.8888922613114119

This calculates the volatility between two investments.

15) <u>**Top Performing Investments-**</u>

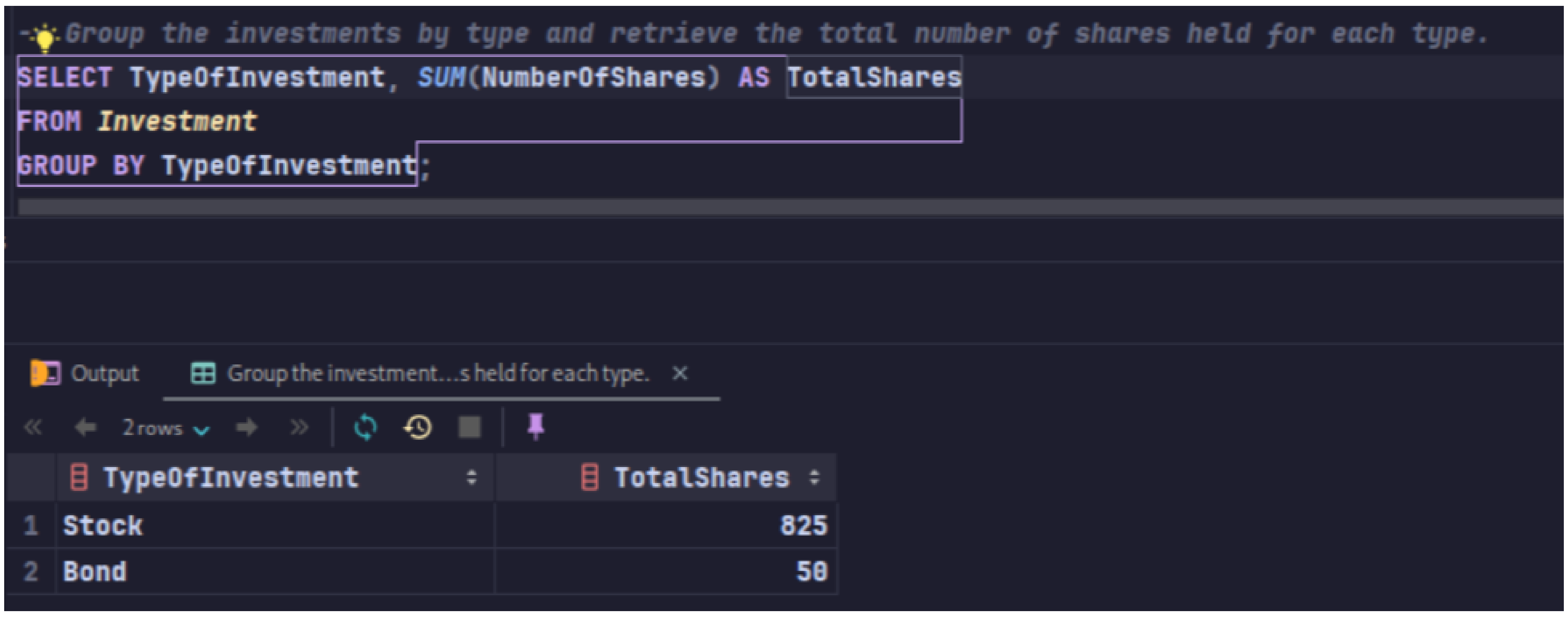


```sql
-- Retrieve the top-performing investments based on annualized return and risk level.
SELECT i.InvestmentID, s.NameOfStock, i.NumberOfShares, i.AnnualizedReturn, i.RiskLevel
FROM Investment i
       INNER JOIN Stock s ON i.StockID = s.StockID
ORDER BY i.AnnualizedReturn DESC, i.RiskLevel ASC
LIMIT 5;
```

	InvestmentID	NameOfStock	NumberOfShares	AnnualizedReturn	RiskLevel
1	2	Microsoft	150	0.911364	0.0137561
2	4	Microsoft	150	0.911364	0.0137561
3	7	Microsoft	50	0.911364	0.0137561
4	5	Apple	75	0.386245	0.0117293
5	8	Apple	200	0.386245	0.0117293

This is another top performing investments query. Only this time, it prioritizes annualized return over risk level.

16) **Number of Shares for each Type of Investment-**



```
-- Group the investments by type and retrieve the total number of shares held for each type.
SELECT TypeOfInvestment, SUM(NumberOfShares) AS TotalShares
FROM Investment
GROUP BY TypeOfInvestment;
```

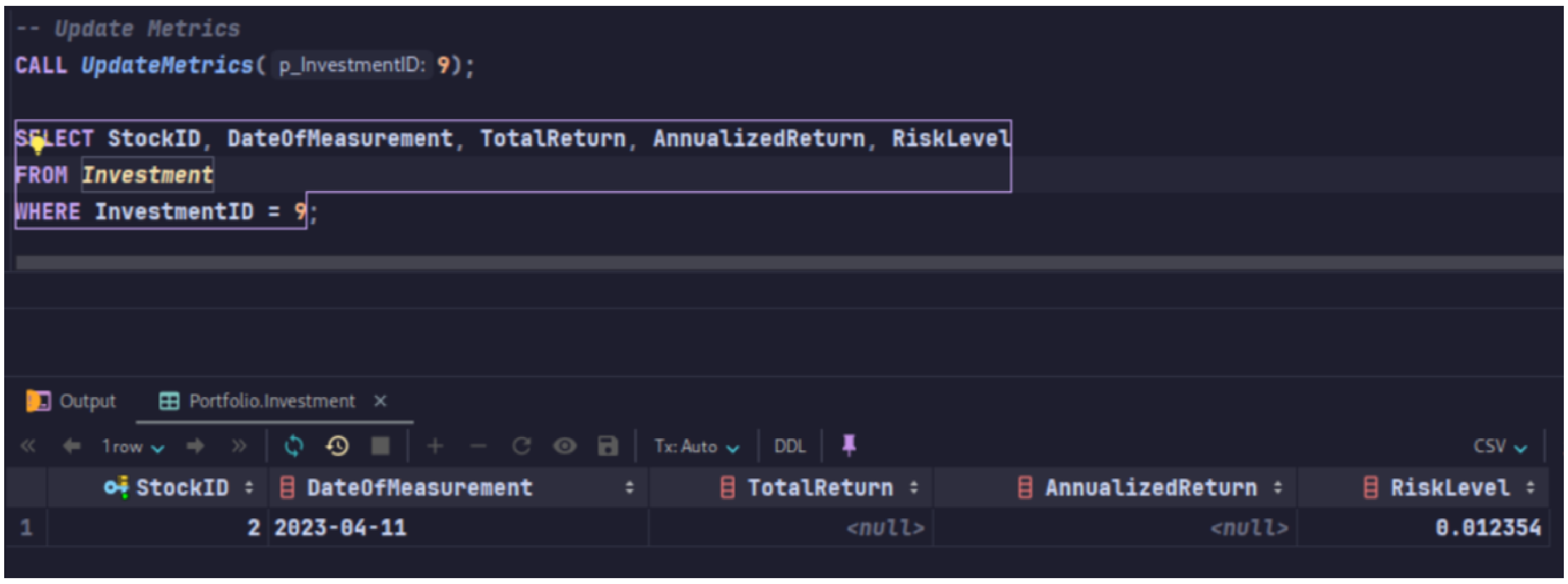
Output | Group the investment...s held for each type.

2 rows

	TypeOfInvestment	TotalShares
1	Stock	825
2	Bond	50

This query groups investments based on type and then calculates the total number of shares for each type.

17) **Update Performance Metrics-**



```
-- Update Metrics
CALL UpdateMetrics( p_InvestmentID: 9);

SELECT StockID, DateOfMeasurement, TotalReturn, AnnualizedReturn, RiskLevel
FROM Investment
WHERE InvestmentID = 9;
```

Output | Portfolio.Investment

1 row | Tx: Auto | DDL | CSV

	StockID	DateOfMeasurement	TotalReturn	AnnualizedReturn	RiskLevel
1	2	2023-04-11	<null>	<null>	0.012354

This query updates the performance metrics for a particular investment.

User Interface and Integration-

We have created a user interface through which various queries can be executed.

Tech Stack-

We have used ***SvelteKit***, a meta-framework built on top of the ***Svelte*** frontend framework, as the javascript framework for the frontend and the backend. ***Vite*** is used as a build tool for *SvelteKit.*

Readme Instructions-

How to start the project at your localhost?

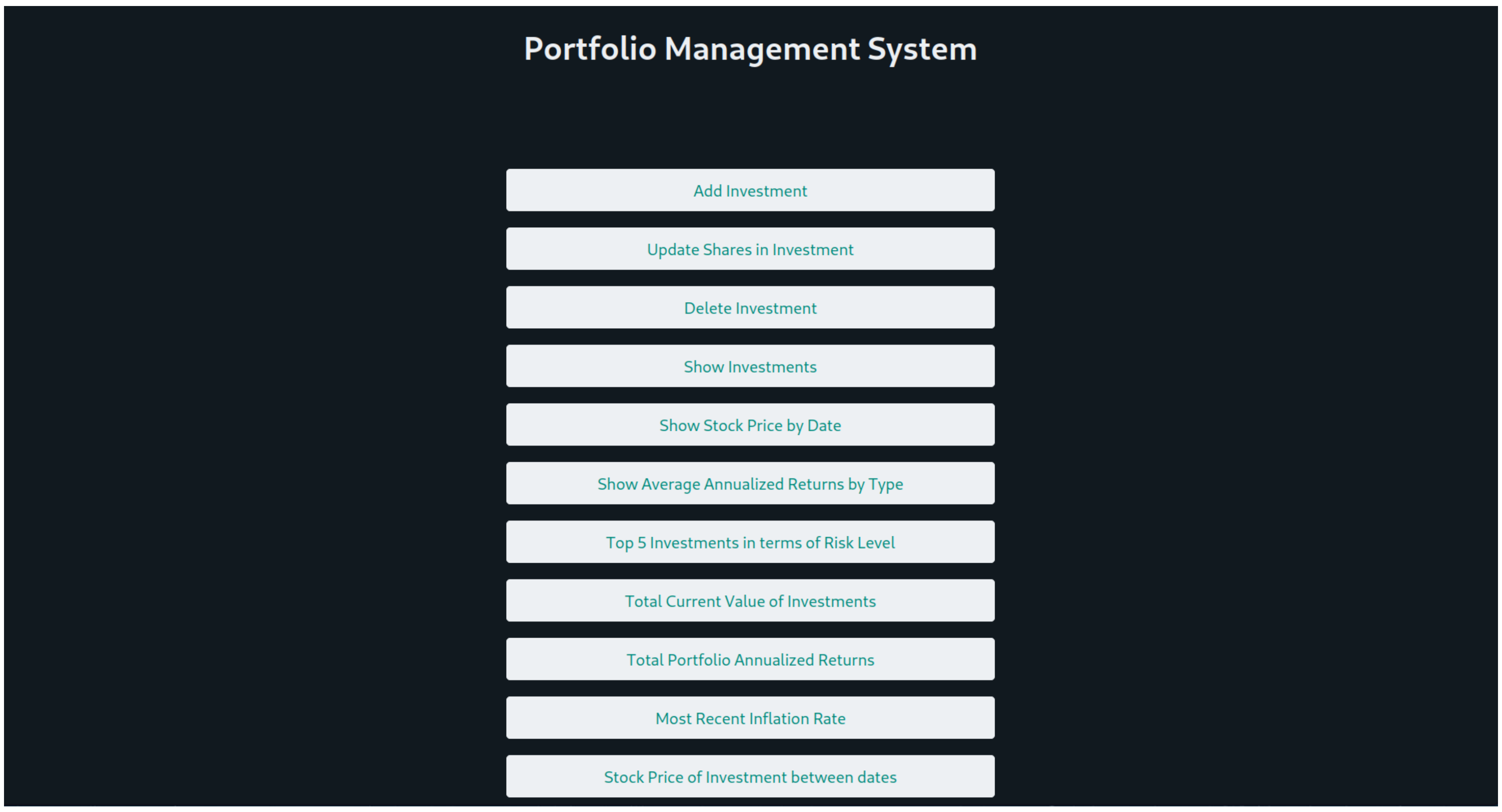
Requirements: *NodeJS*, *NPM* package manager and *MySQL*

In MySQL workbench, run the SQL file *sql/schema.sql*

In file *src/lib/database.ts*, enter the URL, user and password of your SQL server at line 3, 4 and 5 respectively.

Steps:
1. Install Node and npm. For the installation, the steps can be followed from here
2. Download the zip file and extract the zip folder.
3. Open the terminal inside the zip folder.
4. Open the *Portfolio Project* folder in the terminal.
5. Run the following commands:
   (a) npm i
   (b) npm run dev
6. Press the *o* key on the keyboard.

If all went well, the following screen should appear in the browser:


Portfolio Management System
Add Investment
Update Shares in Investment
Delete Investment
Show Investments
Show Stock Price by Date
Show Average Annualized Returns by Type
Top 5 Investments in terms of Risk Level
Total Current Value of Investments
Total Portfolio Annualized Returns
Most Recent Inflation Rate
Stock Price of Investment between dates

In case, there is any difficulty in creating the setup for frontend, kindly call me to help out.

Video Recording Link
GitHub Repository Link