PORTFOLIO MANAGEMENT SYSTEM

Prepared in fulfilment of

**DATABASE SYSTEMS (CS F214)**

**By**

VARUN VARMA 2020B4A70844P

GAURAV MISHRA 2020B3A70917P

Submitted to

**Dr. Amit Dua**



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI, PILANI CAMPUS**

**ACKNOWLEDGEMENT**

Firstly, we would like to express our gratitude to Dr. Amit Dua for guiding us throughout the course and equipping us with the necessary tools with which we were able to complete our project in time.

Secondly, we would like to thank the entire DBS team for always being available and helping us clarify the issues that we faced in our journey of completing the project and in the course,

Thirdly, We would like to thank the institute for providing us with the opportunity to work on a project of this magnitude.

**Introduction**

The Portfolio Management System is a well-designed database that can store and manage relevant data for the investors’ portfolios. The database stores information about investments, performance metrics, market data and other financial information. The database design has tables for investments, performance metrics, market data and other financial information. The design process of this project first involved the creation of an Entity Relationship (ER) Diagram. This was followed by conversion into a Relationship Schema where we define the different relations. This was followed by normalization into 3NF.

The database design also includes various measures to ensure data consistency through primary keys, foreign keys and check constraints. Primary keys will be used to uniquely identify each record in the tables, while foreign keys will be used to establish relationships between the tables. Check constraints will be used to ensure that only valid data is stored in the database.

**Entity Relationship Diagram**

The first step towards database design is to identify the entities and the relationships between said entities. These can be succinctly depicted using an Entity Relationship Diagram (ERD). To get started and to design the diagram we have taken the help of LucidChart, an online platform useful to draw ER diagrams with ease.

The major entities which we recognised along with their attributes are given below:

Investment

* investment\_ID (Primary key)
* investment\_name
* investment\_type
* shares\_held

Performance Metrics

* metric\_ID (Primary key)
* total\_return (Foreign key)
* annualized\_return
* risk\_level

Market Data

* market\_ID (Primary key)
* date
* stock\_price
* exchange\_rate
* commodity\_price

Other Financial Factors

* fin\_ID (Primary key)
* date
* interest\_rate
* inflation\_rate
* GDP\_growth\_rate

The Investment entity stores information pertaining to each investment made such the name and type of investment and the number of shares held. The performance metrics entity stores data relevant to how well the investment is performing in the market. The market data entity stores information about a particular stock or commodity such as the stock price and commodity. The other financial factors entity stores other financial information relevant while making and keeping track of investments in a portfolio.

The relationships between all these entities is described below:

* Investment -> Performance Metrics: One-to-Many relationship. This is chosen as each investment can have different performance metrics at different periods of time or different states of the market.
* Performance Metrics -> Market Data: One-to-Many relationship. This is because each Performance Metric can be drawn from different time periods of Market Data while each point in the Market Data will result in only one Performance Metric for the investment.
* Investment -> Other Financial Information: One-to-One relationship. Each investment corresponds to unique set of the information attributes specified in the other financial information entity.

The above information is hence presented as an ER Diagram given below. The cardinalities of each relationship is also specified in the diagram.

Diagram

Description automatically generated

*Figure 1: ER Diagram*

**Relationship Schema and Normalization**

Towards a Relationship Schema

The next stage in database design after the creation of the ER diagram is to convert it into a relationship schema which can be modelled in SQL. To perform this conversion we first convert all the entities into relationships. Based on the ER diagram given above we have the following relations:

* *Investment (investment\_ID (pk), investment\_name, investment\_type, shares\_held)*
* *Performance Metrics (metric\_ID (pk), investment\_ID (fk), total\_return, annualized\_return, risk\_level)*
* *Market Data (market\_ID (pk), metric\_ID (fk), date, stock\_price, exchange\_rate, commodity\_price)*
* *Other Financial Information (fin\_ID (pk), investment\_ID (fk), date, interest\_rate, inflation\_rate, GDP\_growth\_rate)*

These relationships are formed on the basis of the relationships in the ER diagram. To model the One-to-Many relationships two tables are used where the second entity will have a foreign key to the first one and for the One-to-One relationships one table is used for the remaining entity. The primary and foreign keys are indicated in the relationships itself. These relationships and the schema fully describes the ER diagram which was created earlier.

Identifying Functional Dependencies

Now that we have the relationship schema, we model all the functional dependencies present in each relationship. This is an important step as this will pave the way forward towards normalization, which is an important aspect where we aim to reduce redundancy in our database design.

Based on the actual meaning and dependency of each attributes, the following functional dependencies arise for each relationship:

Investment

* investment\_ID —> investment\_name
* investment\_ID —> investment\_type
* investment\_ID —> shares\_held

Performance Metrics

* metric\_ID —> total\_return
* metric\_ID —> annualized\_return
* metric\_ID —> risk\_level
* risk\_level —> total\_return
* risk\_level —> annualized\_return

Market Data

* market\_ID —> stock\_price
* market\_ID —> exchange\_rate
* market\_ID —> commodity\_price
* market\_ID —> date

Other Financial Information

* fin\_ID —> date
* fin\_ID —> interest\_rate
* fin\_ID —> inflation\_rate
* fin\_ID —> GDP\_growth\_rate

Note that we can also model the functional dependencies of the Other Financial Information table where both the fin\_ID and date form a composite primary key, but we choose not to do so.

Normalization

Now that we have obtained all functional dependencies we are ready to normalize our relationship schema. We observe that every table is already in 1NF as there aren’t any composite attributes. We also observe that every table is also in 2NF as every functional dependency in every table is *fully functionally dependent*.

However we observe that not every table in the schema is in 3NF. This is because in the Performance Metrics table, we see that metric\_ID —> total\_return and metric\_ID —> annualized\_return are *transitive functional dependencies* as these attributes are also functionally dependent on risk\_level which is a *non-prime attribute* apart from metric\_ID which is a *prime attribute.*

Hence to perform 3NF normalization we split the Performance Metrics table into two tables as shown:

Performance Metrics

* metric\_ID —> total\_return
* metric\_ID —> annualized\_return
* metric\_ID —> risk\_level\_ID

Risk Level

* risk\_level\_ID —> risk\_level
* risk\_level\_ID —> total\_return
* risk\_level\_ID —> annualized\_return

By doing so both these tables are now in 3NF and hence the relationship schema is in 3NF as every other table is already in 3NF. Hence the final set of functional dependencies are:

Investment

* investment\_ID —> investment\_name
* investment\_ID —> investment\_type
* investment\_ID —> shares\_held

Performance Metrics

* metric\_ID —> total\_return
* metric\_ID —> annualized\_return
* metric\_ID —> risk\_level\_ID

Risk Level

* risk\_level\_ID —> risk\_level
* risk\_level\_ID —> total\_return
* risk\_level\_ID —> annualized\_return

Market Data

* market\_ID —> stock\_price
* market\_ID —> exchange\_rate
* market\_ID —> commodity\_price
* market\_ID —> date

Other Financial Information

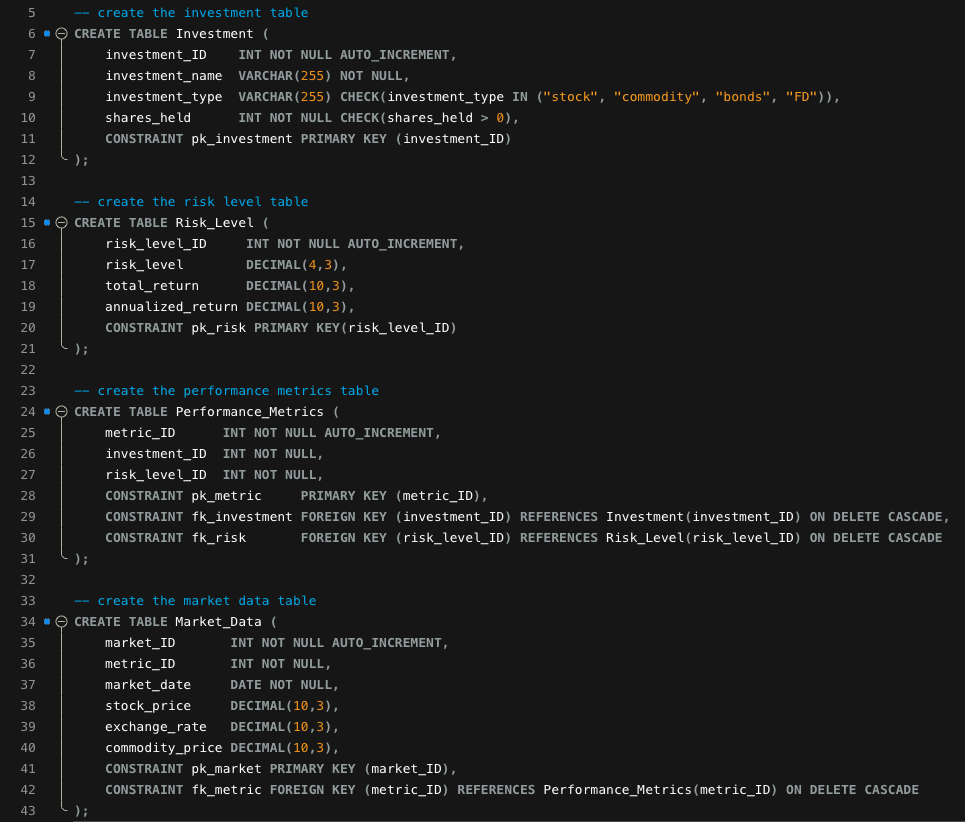
* fin\_ID —> date
* fin\_ID —> interest\_rate
* fin\_ID —> inflation\_rate
* fin\_ID —> GDP\_growth\_rate

We are now done with normalization and are ready to move into implementing the relationship schema in SQL and perform querying operations on it.

**Modelling the Schema in SQL and SQL Queries**

We now implement the relationship schema which was finalized earlier in SQL and perform queries on it. The code snippets of each part are given below:

Creation of tables specified in the schema with appropriate key attributes and check constraints.



We then defined procedures for adding and deleting an investment which can be seen in the SQL script file. The procedure consists of basic insert and delete operations with appropriate sub-queries whenever required.

Queries

Add, update and delete an investment by calling our stored procedures.

Graphical user interface, text

Description automatically generated

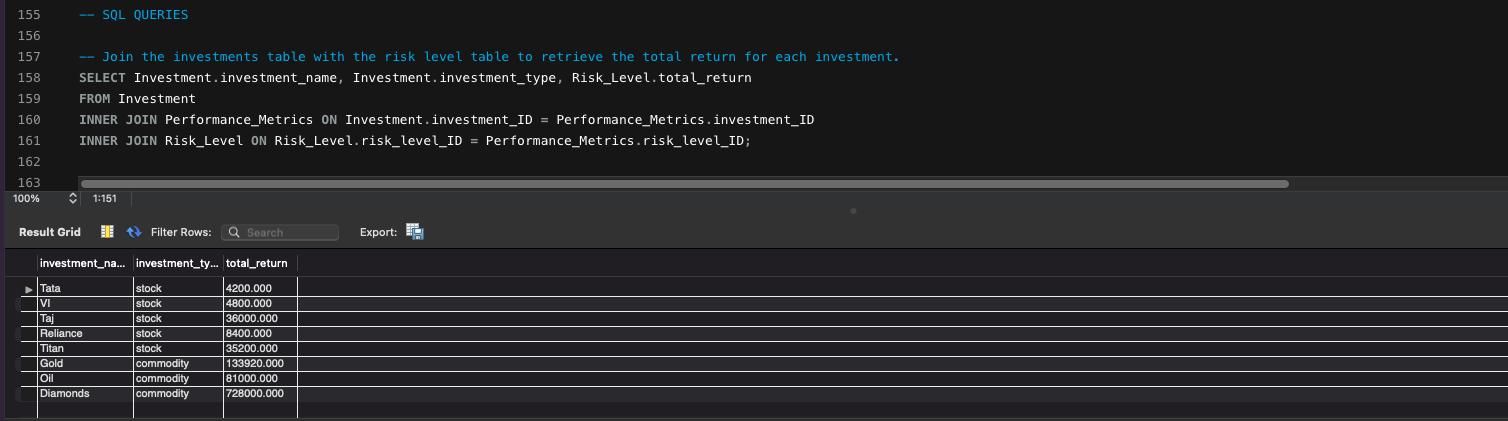
Text

Description automatically generated

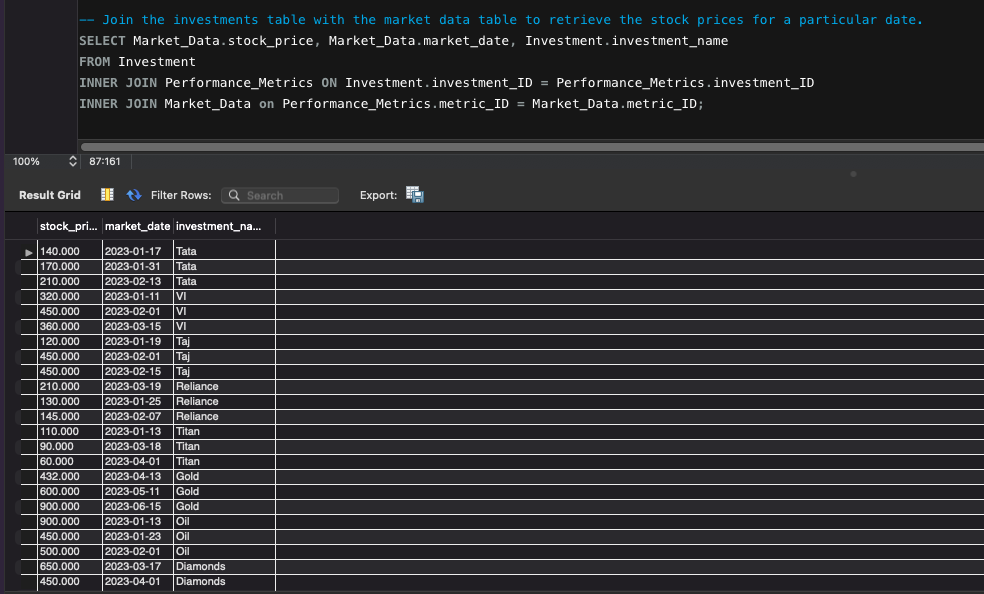
We then add more investments and market data to perform the query operations. The code for that is again present in the SQL script file.

Now we perform the remaining queries.

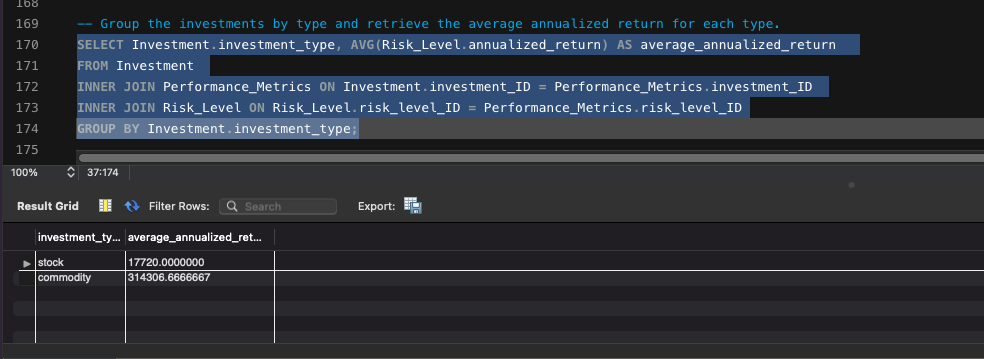
We use join here to join multiple tables and query info from all of them.



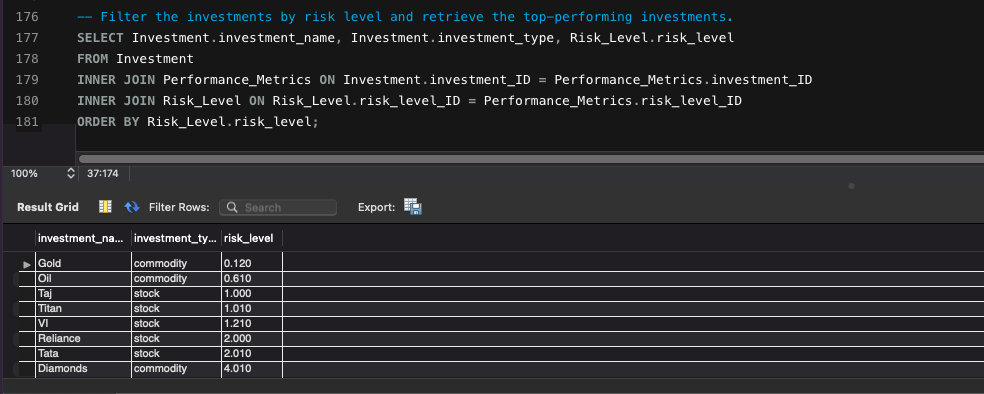
Similar usage of INNER JOIN.



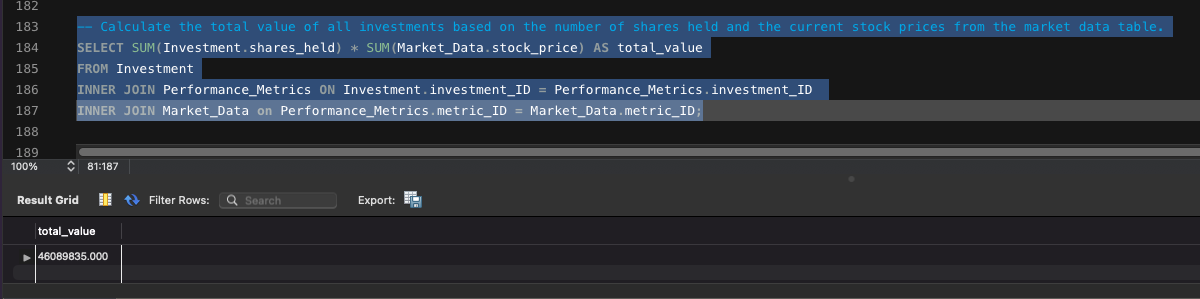
We use GROUP BY to group the investments by investment\_type.



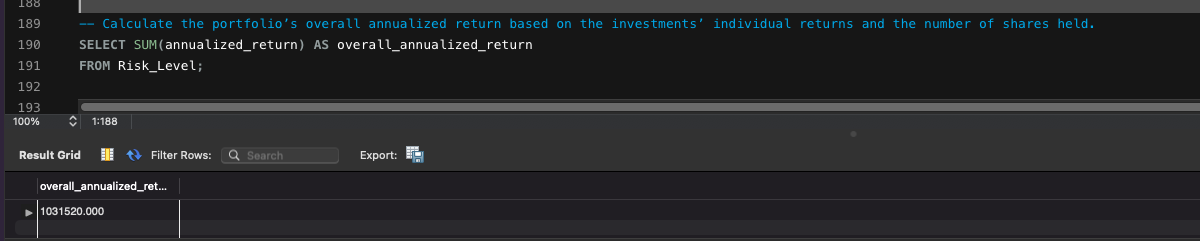
We order by risk level to get the top performing investments.



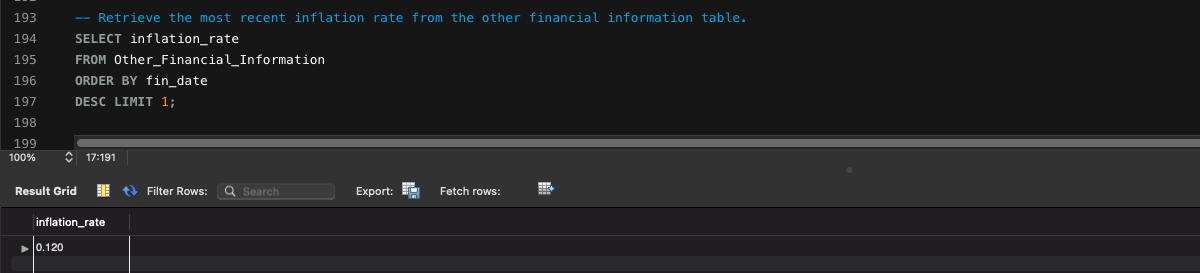
Multiply the shares held and stock\_price in total



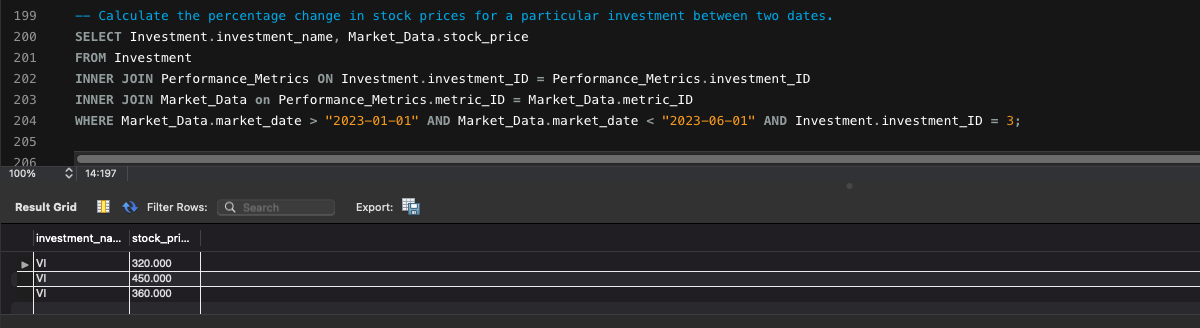
We sum over the annualized\_returns here. (Note values are dummy values need not be to scale)

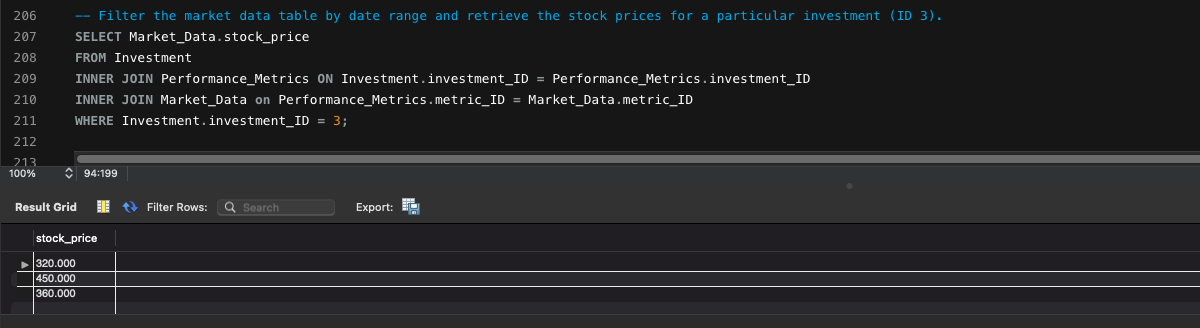


Order the selection by the date to get the most recent inflation rate and impose a limit of 1.

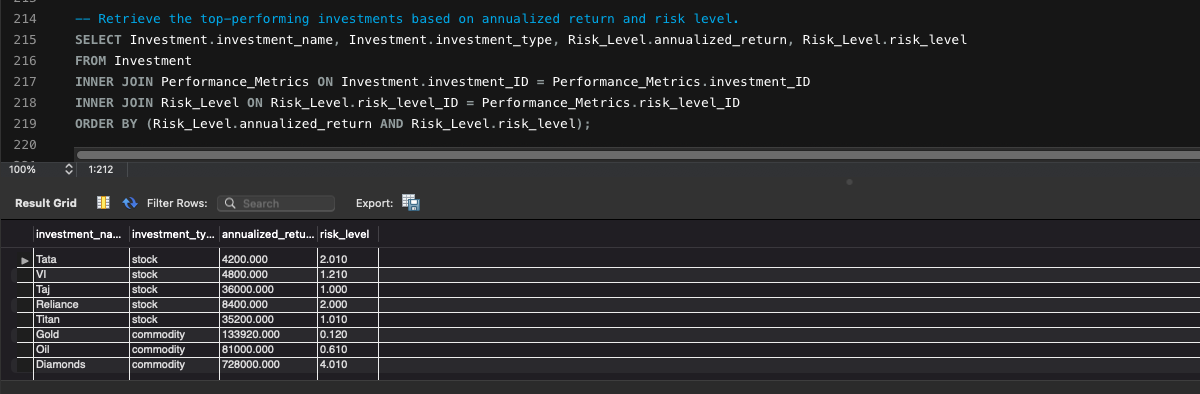


We use where clause to set the date range.

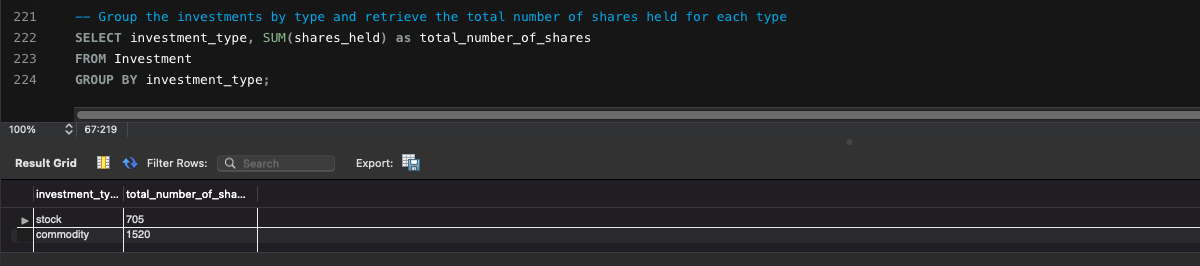




Multiple INNER JOIN and ORDER BY clause is used to check with attributes for different tables.



Finally a GROUP BY clause is used here directly.



**Conclusion and Scope for Improvement**

We have successfully implemented the relationship schema and performed query operations on the data stored. We have used multiple procedures, check constraints, and triggers to ensure data consistency.

The database can be further improved by setting a temporal stream of real-time market data and perform more fast and relevant information queries. More economic factors can be incorporated in the database to produce more robust and informative queries.

THANK YOU