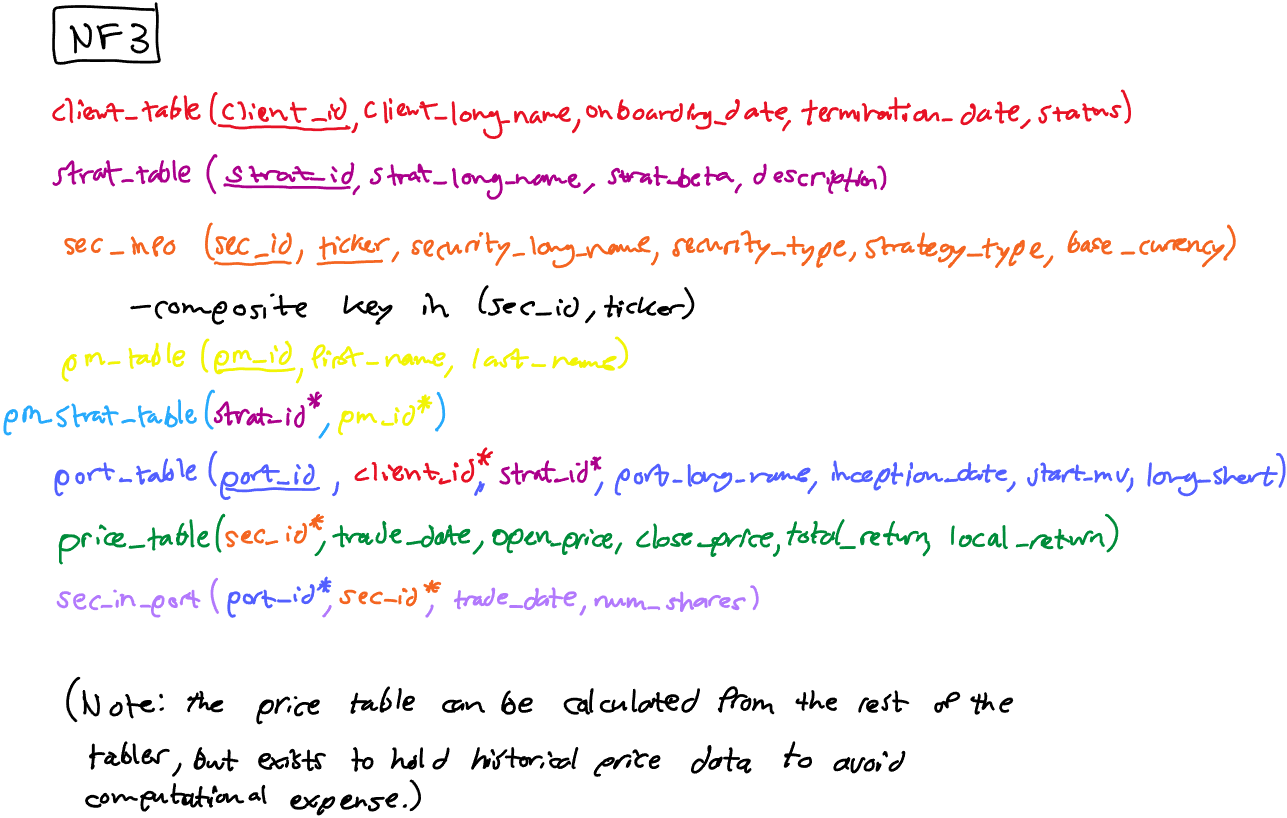
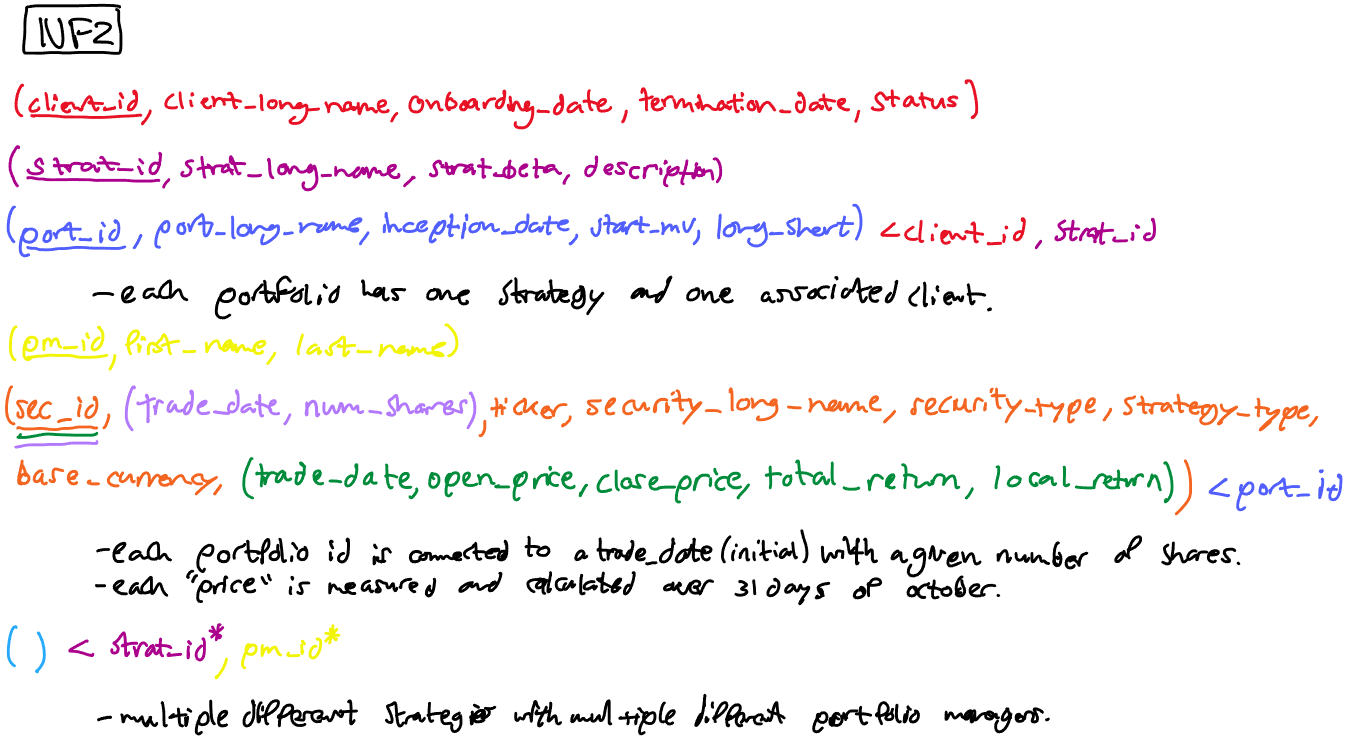
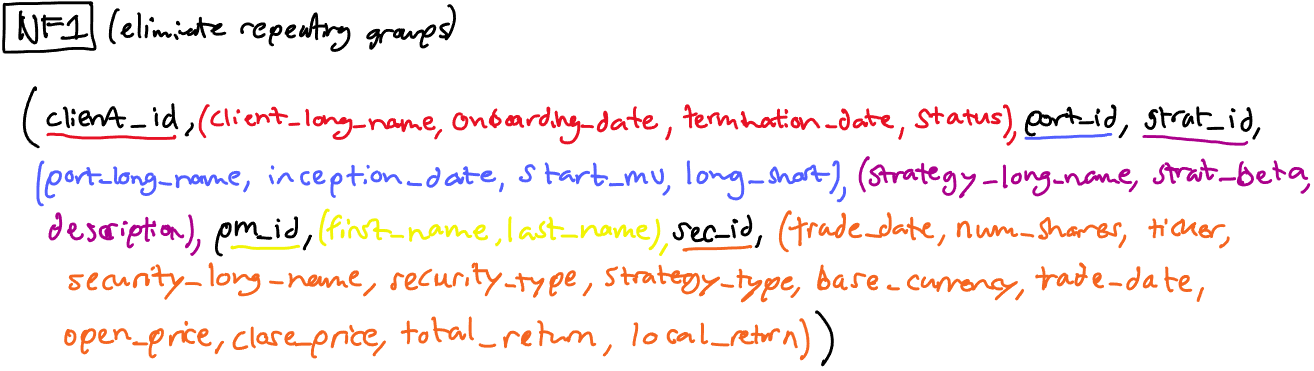
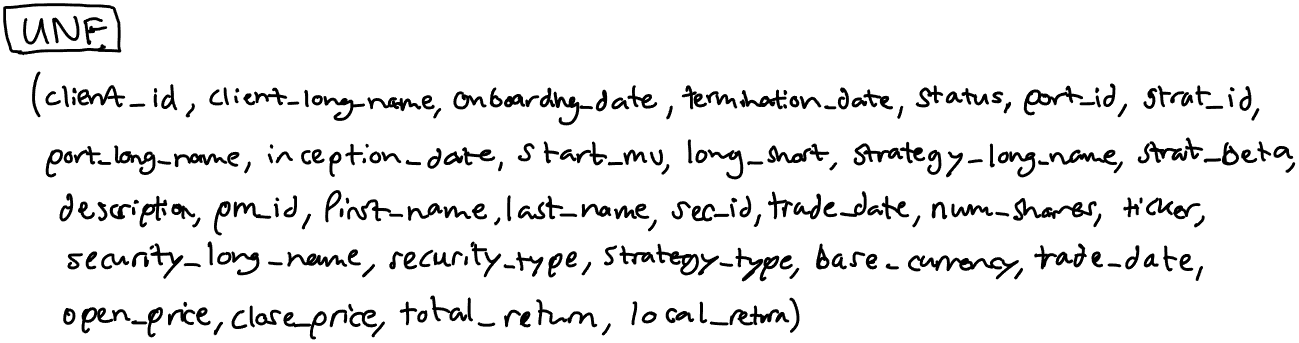
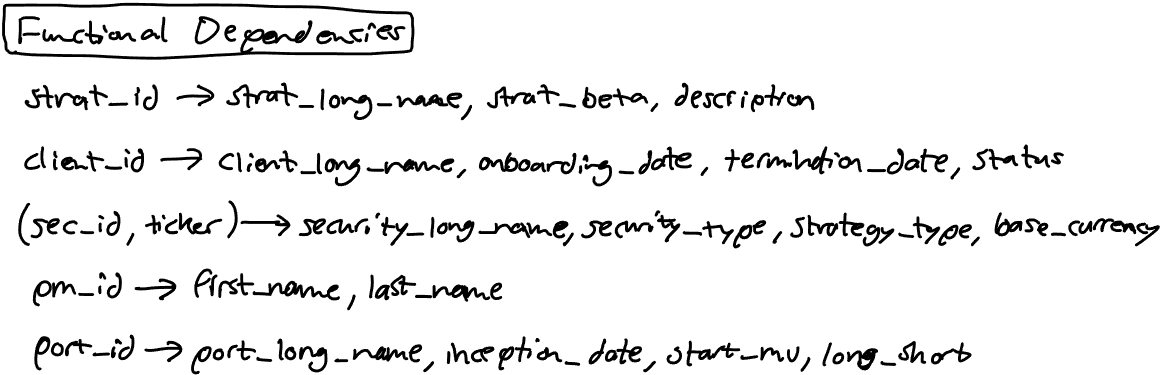
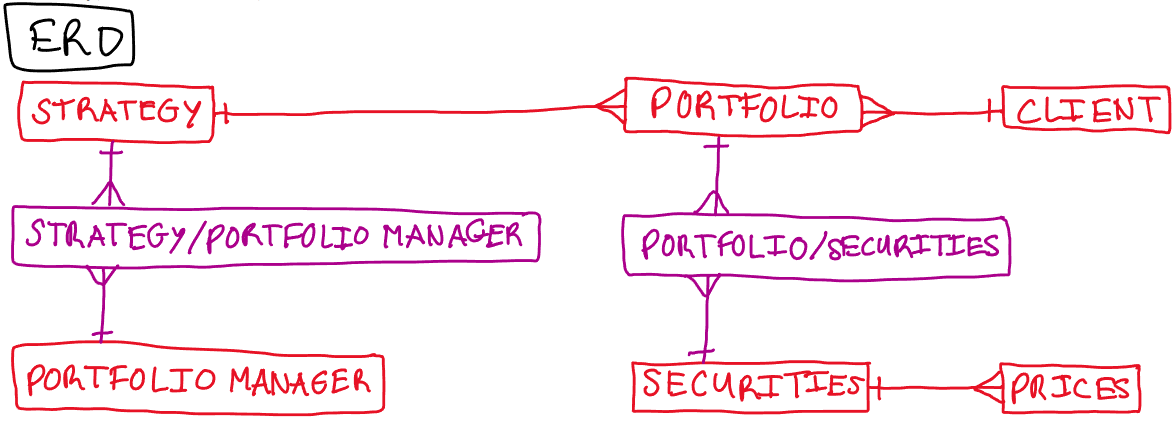
**Mini-World**

In our mini-world, we are the chief investment officer at a global asset management firm, and we want to take a look under the hood at the firms performance. The scope of this analysis will consist of looking at portfolios, securities, changes in the price of securities, investment strategies, clients, and portfolios. For simplicity’s sake, we will assume we have a small number of entities for each category to pick from. We will also be isolating any changes in prices among securities to a short time span (31 days) to reduce the combinatorial complexity of the project. Some things we might want to observe about the performance should include the best and worst performing portfolios by strategy, sub-strategy, and return percentage. To do this we must normalize a messy conglomerate of data that has been given to us by the data team (I guess this is a result of their department’s recent turnover). Once we normalize the data, we will aggregate, manipulate, and transform our tables into views that will accurately display the firm’s performance based on categories already defined.



Throughout the following discussion of the inner workings of my project, I will be referencing how I used relational algebra using **bold-text** words from the first list of terms and how I used other techniques using underlined words from the second list:

**Union**

**Intersection**

**Aggregation**

**Difference**

**Cartesian Product**

**Selection**

**Project**

**Join**

**Divide**

Referential Integrity

Multiple Tables

Multiple Joins

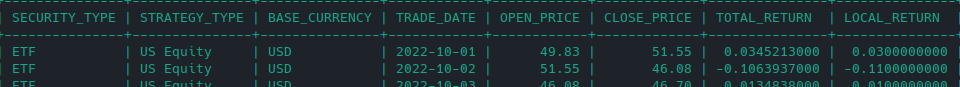
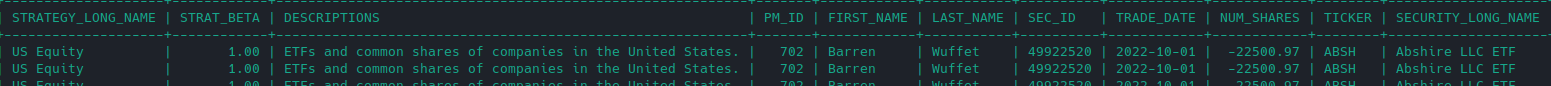
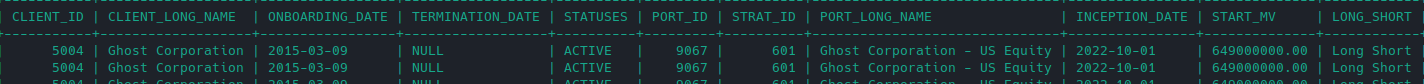
Nested Queries (Correlated And Uncorrelated)

Aggregated Queries (Or Nested Aggregation)

Functions

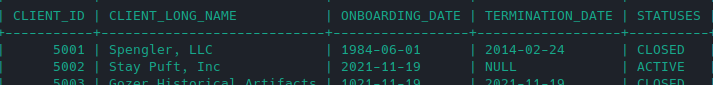
NOTE: In this project, it did not make sense to display the data as a **Cartesian Product**. Instead, it made the most sense to repeatedly use **intersections** (INNER JOIN) in order to aggregate portfolio and security attributes from different tables from the highest level to the most granular level.

1. My un-normalized form is essentially a **union** of all the final normalized tables.

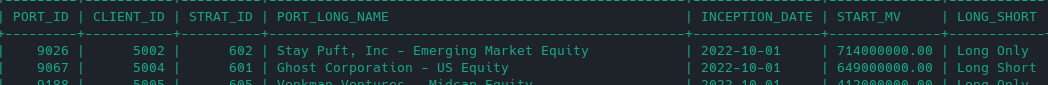


NOTE: The above 3 snippets are one single table.

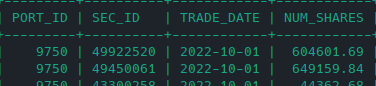
1. Normalizing the large table above results in the following 8 tables (multiple tables). The normalization process below clearly displays how referential integrity is maintained:
   1. CLIENT\_TABLE



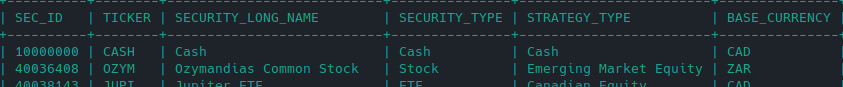
* 1. PORT\_TABLE



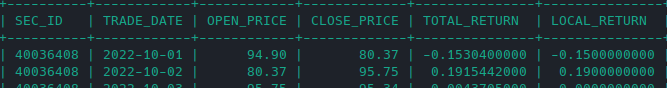
* 1. SEC\_IN\_PORT



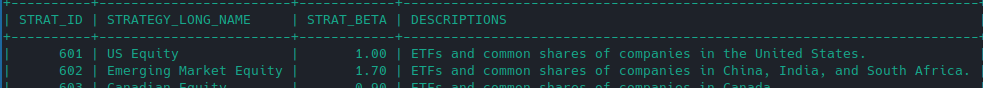
* 1. SEC\_INFO



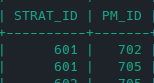
* 1. PRICE\_TABLE



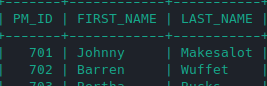
* 1. STRAT\_TABLE



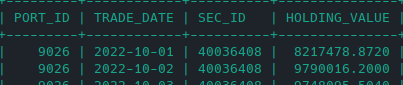
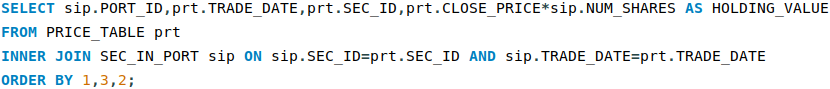
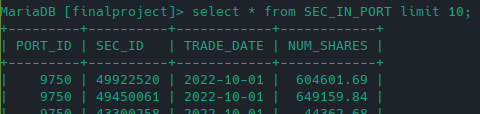
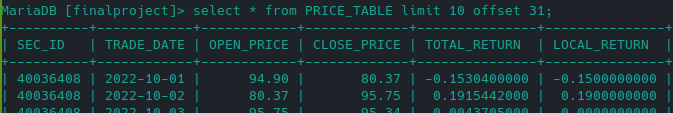
* 1. PM\_STRAT\_TABLE



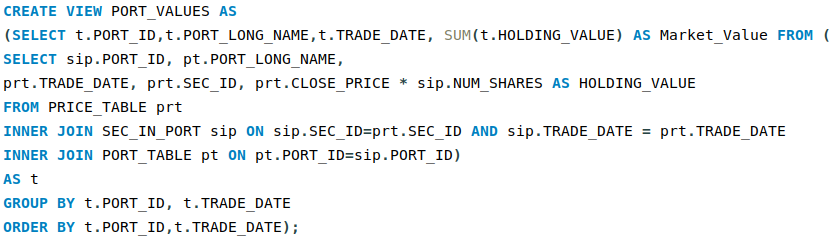
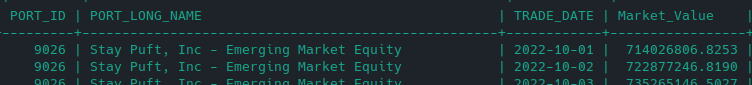
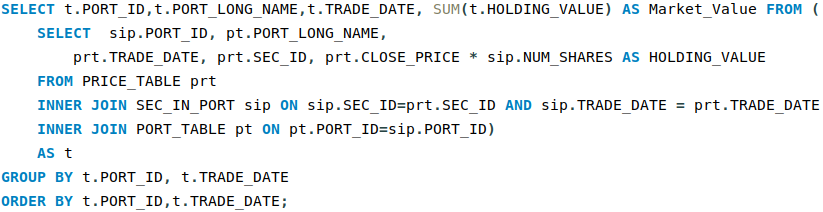
* 1. PM\_TABLE



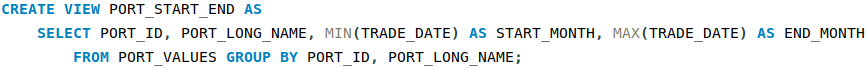
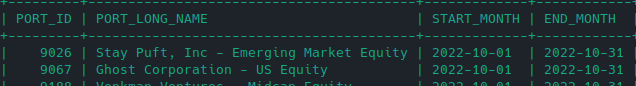
1. If we look at the price table, we can see that its displayed completely independently of each of the portfolios. In other words, no portfolio data exists in the price table. However, each portfolio is made up of securities which have prices that change every day, but the number of shares in each portfolio does not change (at least for this example). In order for us to display the value of the holdings in the portfolios, we must **intersect** the price table with the securities-in-port table.



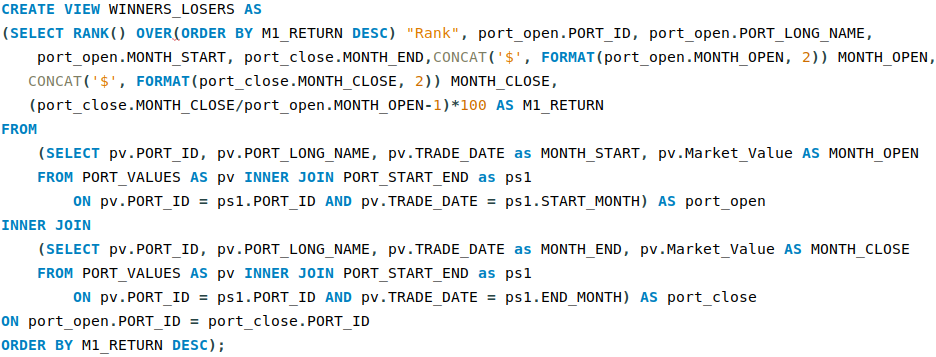
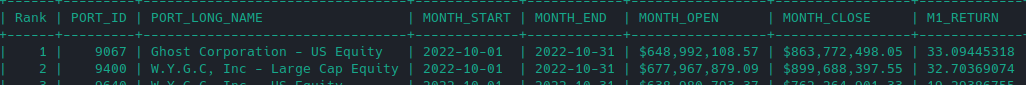
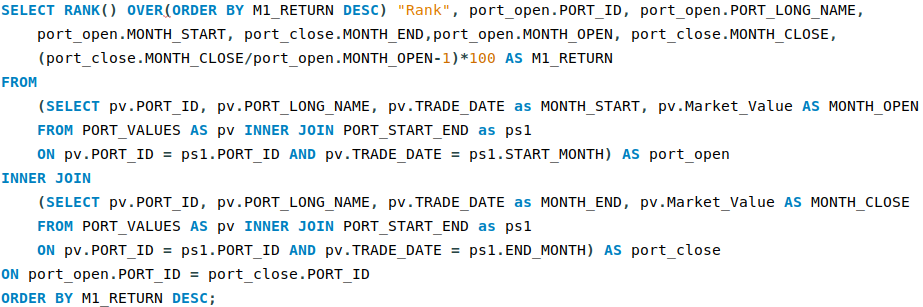
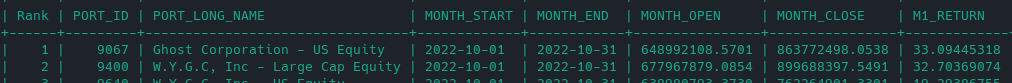
1. This query demonstrates the functional use of the SUM **aggregate function** to summarize market value information at the portfolio level using nested aggregation, multiple joins, and nested queries. Simply, this query displays portfolio level rather than security level (which is what is displayed in the above query). I then create a view for my query, to be utilized in more dynamic displays (in the following queries) related to the summarized values in each portfolio.



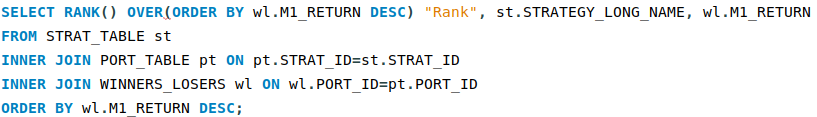
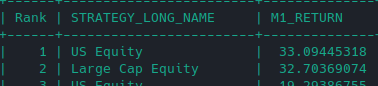
1. Since we want to measure the performance of the portfolios for October, I felt it pertinent to create a view that makes it simpler to **intersect** market values for both the beginning of the period and the end of the period for the query to follow.



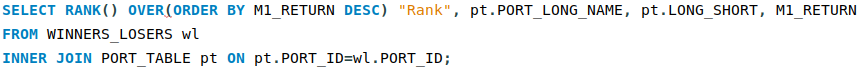
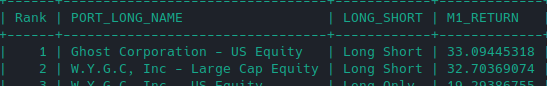
1. This selection query is a culmination of **intersections** of multiple tables, using multiple joins, and it gives a thorough illustration of the overall performance of each portfolio. In order for a neat display and analysis, I use the RANK function to show an ordered ranking of winning portfolios vs losing portfolios by monthly return. I then created a view containing the query, and modified the display of the open and close amounts via the CONCAT function to to display it in dollar format.



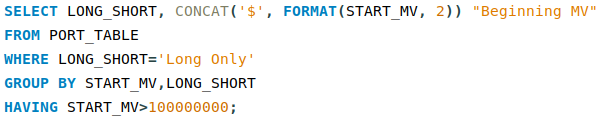
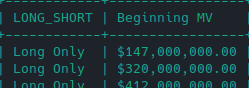
1. This query uses multiple joins to **intersect** (utilizing WINNERS\_LOSERS view) on my portfolio table and strategy table and displays the ranking based on the portfolio strategies, giving us relative strategy performance.



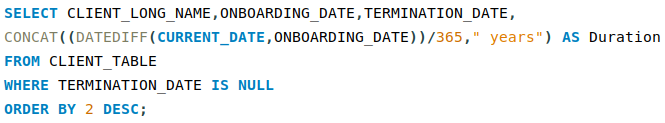
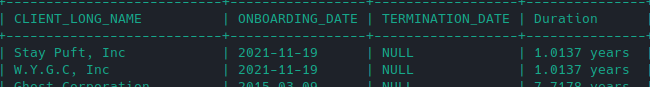
1. This query **intersects** (utilizing WINNERS\_LOSERS view) on our portfolio table and displays the ranking based on the long and long-short sub-strategies.



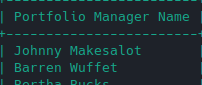
1. This query utilizes the GROUP BYaggregate function and filters beginning market values by any values larger than $100 million.



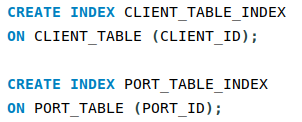
1. This query utilizes DATEDIFF function to determine the duration of time each client has been with my hypothetical investment firm based on the provided onboarding date and the current date. It also uses the CONCAT function to make the date readable.



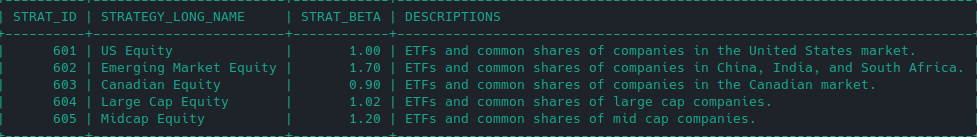
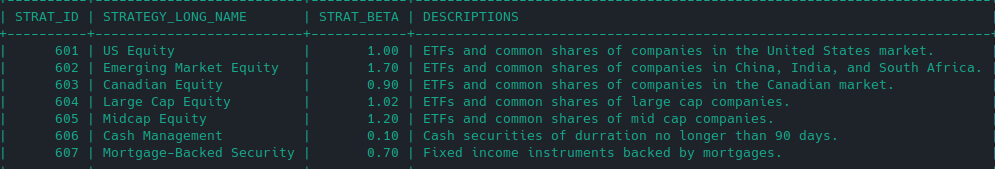
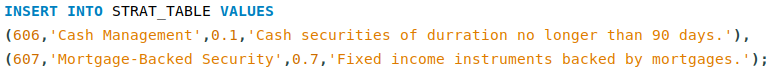
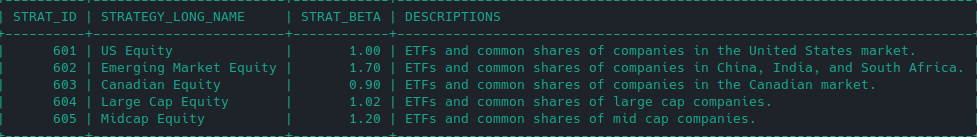
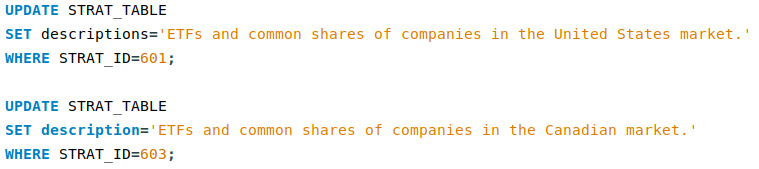
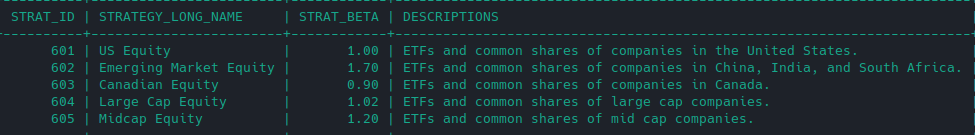
1. In the portfolio manager table, the names of the portfolio managers consist of a first and last name. This simple query uses the CONCAT function to display the full name.



1. Created two indices on the client table and portfolio table in order to more efficiently pull the data.



1. The following queries utilize UPDATE, DELETE, and ALTER functions to change some minor details within the strategy table to facilitate better descriptions for each strategy, and to demonstrate any necessary changes that might come with adding new strategies.



**APPENDIX**

**All Easter Eggs**

**CLIENT\_TABLE (Ghostbusters References)**

|  |  |
| --- | --- |
| Spengler, LLC | *Egon Spengler* |
| Stay Puft, Inc | *Stay Puft Marshmallow Man* |
| Gozer Historical Artifacts | *Sumerian Deity Gozer* |
| Ghost Corporation | *Ghost Corp.* |
| Venkman Ventures | *Peter Venkman* |
| Stanz Raymond Investments | *Ray Stantz* |
| Zeddemore Consulting | *Winston Zeddemore* |
| W.Y.G.C, Inc | *“Who You Gonna Call”* |
| 6/1/1984 | *Ghostbusters first movie release date* |
| 11/19/2021 | *Ghostbusters: Afterlife release date* |
| 3/9/2015 | *Ghost Corp Incorporation date* |

**PM\_TABLE (Famous Investors/Economists References)**

|  |  |
| --- | --- |
| Barren Wuffet | *Warren Buffet* |
| Teeter Peel | *Peter Thiel* |
| Beanjamin Slaham | *Benjamin Graham* |

**SEC\_INFO (References for Stock Names)**

|  |  |  |
| --- | --- | --- |
| ***One Piece References*** | ***Watchmen References*** | ***Mathematician References*** |
| *OnePiece ETF* | *Watchmen Common Stock* | *Einstein Common Stock* |
| *Luffy Common Stock* | *Jupiter ETF* | *Feynman ETF* |
| *Shanks Common Stock* | *Manhattan Common Stock* | *Turing Common Stock* |
| *Nami ETF* | *NiteOwl ETF* | *Archimedes Common Stock* |
| *Brooke ETF* | *Ozymandias Common Stock* | *Aryabhata Common Stock* |
| *Absalom Common Stock* | *Rorschach ETF* | *Euclid ETF* |
| *Abdullah ETF* |  | *Euler ETF* |
| *Linlin ETF* |  | *Fibonacci Common Stock* |
|  |  | *Godel ETF* |
|  |  | *Pythagoras ETF* |

***\*The rest are spoofs of existing companies***