Instructions for accessing the database is as follows

1. In DataGrip, select a MySQL data source.
2. Change the Name of the data source to: finalproject.cvfdrw5olghe.us-east-2.rds.amazonaws.com
3. Fill in the Host as: finalproject.cvfdrw5olghe.us-east-2.rds.amazonaws.com
4. Fill in the Port as: 3306
5. Fill in User as: austinw1995
6. Fill in Password as: twins111
7. Press Apply, Press Test Connection, Press Ok
8. Open a Query Console and type `USE FINAL_PROJECT;`
9. To access stock data, use table `Stocks_Cor`
10. To access index data, use table `Markets_Cor2`

SQL Queries

Simple Query 1:
Selecting the stock price over time of multiple companies over a specified period of time.

```
SELECT date, name, close
FROM Stocks_Cor
WHERE name IN ('AAPL', 'GOOGL', 'MSFT') AND date BETWEEN '2013-02-08' AND
'2018-02-07'
ORDER BY date ASC, name;
```

Simple Query 2:
Selecting the top 10 stocks by weighted relative market capitalization over a specified period of time, giving a sense of each company's contribution and significance in the overall market.

```
WITH TotalMarketCap AS (
   SELECT name, SUM(close * volume) AS numerator
   FROM Stocks_Cor
   WHERE date BETWEEN '2013-02-08' AND '2018-02-07'
   GROUP BY name
)
SELECT name, (numerator / (SELECT SUM(numerator) FROM TotalMarketCap)) AS
market_cap
FROM TotalMarketCap
ORDER BY market_cap DESC, name
LIMIT 10;
```

Simple Query 3:
Identify the top 10 stocks by positive percentage change in price in a selected period of time.

```sql
WITH BeginDate AS (
   SELECT name, close AS beg_price
   FROM Stocks
   WHERE date = '2013-02-08'
),
FinalDate AS (
   SELECT name, close AS end_price
   FROM Stocks
   WHERE date = '2018-02-07'
),
Combined AS (
SELECT BD.name, BD.beg_price, FD.end_price
FROM BeginDate BD JOIN FinalDate FD ON BD.name = FD.name
)
SELECT C.name, ((C.end_price - C.beg_price)/(C.beg_price)) * 100 AS pct_change
FROM Combined C
WHERE C.end_price >= C.beg_price
ORDER BY pct_change DESC, name
LIMIT 10;
```

Simple Query 4:
Identify the top 10 stocks by negative percentage change in price in a selected period of time.

```sql
WITH BeginDate AS (
   SELECT name, close AS beg_price
   FROM Stocks
   WHERE date = '2013-02-08'
),
FinalDate AS (
   SELECT name, close AS end_price
   FROM Stocks
   WHERE date = '2018-02-07'
),
Combined AS (
SELECT BD.name, BD.beg_price, FD.end_price
FROM BeginDate BD JOIN FinalDate FD ON BD.name = FD.name
)
SELECT C.name, ((C.end_price - C.beg_price)/(C.beg_price)) * 100 AS pct_change
FROM Combined C
WHERE C.end_price < C.beg_price
ORDER BY pct_change ASC, name
LIMIT 10;
```

Simple Query 5:

Identify the top 10 stocks by percentage change in price on a selected date.

```
SELECT name, ((close - open) / open) * 100 AS pct_change
FROM Stocks
WHERE date = '2013-02-08'
ORDER BY ABS(pct_change) DESC, name
LIMIT 10;
```

Simple Query 6:
Top 10 volatile stocks over a selected period of time.

```
WITH DailyReturns AS (
   SELECT
       sc.date,
       sc.name,
       (sc.close - LAG(sc.close) OVER (PARTITION BY sc.name ORDER BY sc.date))
/ LAG(sc.close) OVER (PARTITION BY sc.name ORDER BY sc.date) AS daily_return
   FROM
       Stocks_Cor sc
   WHERE
       sc.date BETWEEN '2017-02-08' AND '2018-02-07'
),
Volatility AS (
   SELECT
       name,
       SQRT(AVG(daily_return * daily_return)) AS volatility
   FROM
       DailyReturns
   GROUP BY
       name
)
SELECT
   v.name,
   v.volatility * 100 AS vol
FROM
   Volatility v
```

```
ORDER BY
   v.volatility DESC
Limit 10;
```

Simple Query 7:

View the prices of multiple selected indices over a specified period of time (time series of index prices).

```
SELECT date, marketIndex, closeUSD
FROM Markets_Cor2
WHERE marketIndex IN ('HSI', 'NYA', 'N100', 'NSEI') AND date BETWEEN
'1986-12-31' AND '2021-05-31'
ORDER BY date, marketIndex;
```

Simple Query 8:

Query to determine cumulative expected returns based on a trailing 1 year period for selected stocks.

```
WITH SelectedStocks AS (
   SELECT *
   FROM Stocks_Cor
   WHERE name IN ('AMD', 'NWL')
),
DailyReturns AS (
   SELECT
       sc.name,
       sc.date,
       (sc.close - lag(sc.close) OVER (PARTITION BY sc.name ORDER BY sc.date))
/ lag(sc.close) OVER (PARTITION BY sc.name ORDER BY sc.date) AS daily_return
   FROM SelectedStocks sc
),
ProjectedReturns AS (
   SELECT
       name,
       MAX(date) AS end_date,
       EXP(SUM(LOG(1 + daily_return)) OVER (PARTITION BY name ORDER BY date
ROWS BETWEEN 252 PRECEDING AND CURRENT ROW)) - 1 AS one_year_cumulative_return
   FROM DailyReturns
   GROUP BY name, date
)
SELECT name, one_year_cumulative_return
FROM ProjectedReturns
WHERE end_date = (SELECT MAX(date) FROM Stocks_Cor);
```

Complex Query 9 (Longer than 15s):

Query to get the beta of selected stocks with respect to a chosen index.

```
ITH SELECTED_STOCKS AS (
   SELECT name, date, close
   FROM Stocks_Cor
   WHERE name IN ('AAPL', 'GOOGL', 'MSFT') AND date BETWEEN '2014-02-07' AND
'2018-02-07'
),
STOCK_DAILY_RETURNS AS (
   SELECT
       S1.name,
       S1.date,
       (S1.close - S2.close) / S2.close AS daily_return
   FROM
       SELECTED_STOCKS S1
   JOIN
       SELECTED_STOCKS S2 ON S1.name = S2.name AND S2.date = (SELECT MAX(date)
FROM SELECTED_STOCKS WHERE date < S1.date)
   ORDER BY
       S1.date, S1.name
),
AVG_STOCK_RETURNS AS (
   SELECT
       name,
       AVG(daily_return) AS avg_stock_return
   FROM
       STOCK_DAILY_RETURNS
   GROUP BY
       name
),
SELECTED_MKT AS (
   SELECT date, closeUSD AS close
   FROM Markets_Cor2
   WHERE marketIndex = 'NYA' AND date BETWEEN '2014-02-07' AND '2018-02-07'
),
INDEX_DAILY_RETURNS AS (
   SELECT
       I1.date,
       (I1.close - I2.close) / I2.close AS daily_return
   FROM
       SELECTED_MKT I1
   JOIN
       SELECTED_MKT I2 ON I2.date = (SELECT MAX(date) FROM SELECTED_MKT WHERE
date < I1.date)
   ORDER BY
       I1.date
),
AVG_INX_RETURNS AS (
```

```
   SELECT AVG(daily_return) AS avg_inx_ret
   FROM INDEX_DAILY_RETURNS
)
SELECT
   SDR.name,
   ASR.avg_stock_return AS avg_stock_return,
   IDR.daily_return AS avg_index_return,
   (SUM((SDR.daily_return - ASR.avg_stock_return) * (IDR.daily_return - (SELECT
avg_inx_ret FROM AVG_INX_RETURNS))) / (COUNT(*) - 1)) /
   SUM(POW(IDR.daily_return - (SELECT avg_inx_ret FROM AVG_INX_RETURNS), 2)) /
(COUNT(*) - 1) AS beta
FROM
   STOCK_DAILY_RETURNS SDR
JOIN
   AVG_STOCK_RETURNS ASR ON SDR.name = ASR.name
JOIN
   INDEX_DAILY_RETURNS IDR ON SDR.date = IDR.date
GROUP BY
   SDR.name;
```

Complex Query 10:
Calculates correlation between the average price of multiple stocks (can also just be one stock) and a selected index.

```
WITH SELECTED_STOCKS AS (
   SELECT date, AVG(close) AS close
   FROM Stocks_Cor
   WHERE name IN ('AAPL', 'GOOGL', 'MSFT')
   GROUP BY date
   ORDER BY date ASC
),
StockAndIndex AS (
   SELECT S.date, S.close AS stock_price, I.closeUSD AS index_price
   FROM SELECTED_STOCKS S JOIN Markets_Cor2 I ON S.date = I.date
   WHERE I.marketIndex = 'NYA'
),
Averages AS (
   SELECT
       AVG(stock_price) AS avg_stock_price,
      AVG(index_price) AS avg_index_price
   FROM StockAndIndex
)
SELECT (SUM((S.stock_price - A.avg_stock_price) * (S.index_price -
A.avg_index_price)) / (COUNT(*) - 1)) /
(SQRT(SUM(POW(S.stock_price - A.avg_stock_price, 2)) / (COUNT(*) - 1)) *
SQRT(SUM(POW(S.index_price - A.avg_index_price, 2)) / (COUNT(*) - 1))) AS
correlation
FROM StockAndIndex S, Averages A;
```

Complex Query 11:
Compare and contrast the performance of selected S&P 500 stocks with selected indices.

```
WITH SELECTED_STOCKS AS (
   SELECT date, name, close
   FROM Stocks_Cor
   WHERE name IN ('AAPL', 'GOOGL', 'MSFT') AND date BETWEEN '2013-02-08' AND
'2018-02-07'
   ORDER BY date ASC, name
),
SELECTED_INX AS (
   SELECT date, marketIndex, closeUSD
   FROM Markets_Cor2
   WHERE marketIndex IN ('HSI', 'NYA', 'N100', 'NSEI') AND date BETWEEN
'2013-02-08' AND '2018-02-07'
   ORDER BY date, marketIndex
)
SELECT date, name AS ticker, close AS close
FROM SELECTED_STOCKS
UNION
SELECT date, marketIndex AS ticker, closeUSD AS close
FROM SELECTED_INX
ORDER BY date ASC, ticker;
```

Complex Query 12:
Query to take the mean of selected stocks and compare with selected indices to see how a batch of stocks have performed in comparison to the overall market performance of particular markets.

```
WITH SELECTED_STOCKS AS (
   SELECT date, 'SELECTED_STOCKS', AVG(close) AS close
FROM Stocks_Cor
WHERE name IN ('AAPL', 'GOOGL', 'MSFT')
GROUP BY date
ORDER BY date ASC
),
SELECTED_INX AS (
```

```
   SELECT date, marketIndex, closeUSD
   FROM Markets_Cor2
   WHERE marketIndex IN ('HSI', 'NYA', 'N100', 'NSEI') AND date BETWEEN
'2013-02-08' AND '2018-02-07'
   ORDER BY date, marketIndex
)
SELECT date, 'SELECTED_STOCKS' AS ticker, close AS close
FROM SELECTED_STOCKS
UNION
SELECT date, marketIndex AS ticker, closeUSD AS close
FROM SELECTED_INX
ORDER BY date ASC, ticker;
```

Complex Query 13:

Table that contains the S&P500 index and table that contains the weight of each stock in the S&P500. We don't have this index, so we will be artificially creating it with a price-weighted index of the component stocks.

```
CREATE TABLE SP500 AS
   SELECT date, SUM(close) AS pw_sp500_value
   FROM Stocks_Cor
   GROUP BY date;

CREATE TABLE Stock_Weightings AS
   SELECT SP.date, SC.name, (SC.close / SP.pw_sp500_value) AS stock_weight
   FROM SP500 SP JOIN Stocks_Cor SC ON SP.date = SC.date;
```

Complex Query 14 (Longer than 15s):

Calculates the relative strength index of selected stocks, and ranks them in order.

```
WITH SELECTED_STOCKS AS (
  SELECT name, date, close
  FROM Stocks_Cor
  WHERE name IN ('AAPL', 'GOOGL', 'MSFT') AND date BETWEEN '2014-02-07' AND
'2018-02-07'
),
```

```
STOCK_PRICE_CHANGES AS (
   SELECT
   S1.name,
   S1.date,
   (S1.close - S2.close) AS price_change
   FROM SELECTED_STOCKS S1 JOIN
   SELECTED_STOCKS S2 ON S1.name = S2.name AND S2.date = (SELECT MAX(date) FROM
SELECTED_STOCKS WHERE date < S1.date)
   ORDER BY S1.date, S1.name
),
AVG_GAINS_LOSSES AS (
   SELECT
       name,
       AVG(CASE WHEN price_change > 0 THEN price_change ELSE 0 END) AS
avg_gain,
       AVG(CASE WHEN price_change < 0 THEN ABS(price_change) ELSE 0 END) AS
avg_loss
   FROM STOCK_PRICE_CHANGES
   GROUP BY name
)
SELECT name, 100 - (100 / (1 + avg_gain / avg_loss)) AS rsi
FROM AVG_GAINS_LOSSES
WHERE avg_loss <> 0
ORDER BY rsi;
```