Data Science Independent Project

WATCHING THE STOCK MARKET

By: Ghecel Joy De Chavez [GJBDC]

I. Overview

A. Background

This is an independent project posted in the Codecademy forum in relation to an off-platform challenge as a Data Scientist/Analyst. It is part of the Data Science and Data Analysis career path and something I found under the Learn SQL course.

Data Science Independent Project #1 – Watching the Stock Market - Projects

B. Suggested Technologies

As suggested in the forum, I utilized the DB Browser for SQLite tool. The above link for the project and below as well, provides brief background and functionalities of the tool.

DB Browser for SQLite (sqlitebrowser.org)

II. Project Tasks

A. Objective

As per Codecademy forum:

"You are asked by a company to help them make more informed decisions on investments. To start, you will be watching the stock market, collecting data, and identifying trends!"

B. Data

Manipulation: Collect data on your pick of 5 stocks.

I used Google Finance to track the stock prices of 5 companies, 3 times throughout the day for 1 week. The date range is from 9/1/2023, 9/5/2023-9/8/2023, no data for the weekends due to market closure. Tracked stock prices in the morning (09:30 AM), midday (12:00 PM), and closing (4:00 PM). 15 records per stocks and a total of 75 records were collected for this project.

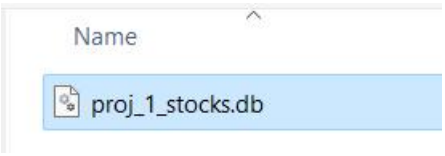
The 5 stocks are: Apple Inc (AAPL), Tesla Inc (TSLA), Intel Corporation (INTC), Amazon.com, Inc. (AMZN), and Uber Technologies Inc (UBER).

C. Methods

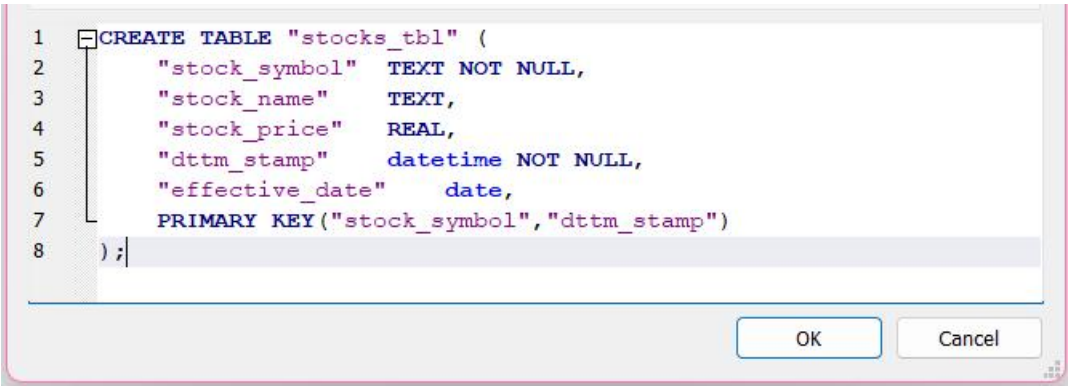
For this project, I built the database in SQLite. In addition, I have utilized the functionality of SQLite to import data via CSV file, but still included INSERT clause to insert some of the data we have, to practice.

For building bigger and future databases, I will most likely utilize the Import Table/Data via CSV file functionality of SQLite to quickly insert records and for a more streamlined approach.

Create the database [`proj_1_stocks.db`]:



To create the table [`stocks_tbl`]:



Column Name	Description
stock_symbol	Corresponds to the name or symbol of the stock, with constraint as NOT NULL as this will be used as primary key.
stock_name	Corresponds to the Business Name
stock_price	Corresponds to the stock price. The data type is real as they contain decimal values.
dttm_stamp	The date and time we captured the stock price. Tracked stock prices in the morning (09:30 AM), midday (12:00 PM), and closing (4:00 PM).
effective_date	The date we captured the stock price. This is the same value as the dttm_stamp, but without the time stamp, with constraint as NOT NULL as this will be used as primary key.
PRIMARY KEY("stock_symbol","dttm_stamp") We added two columns so we can have unique stock_symbol per dttm_stamp. Meaning we can have the multiple and same values in the stock_symbol column, but they cannot have the same dttm_stamp value. I have added an example of a unique constraint error here in the document.	

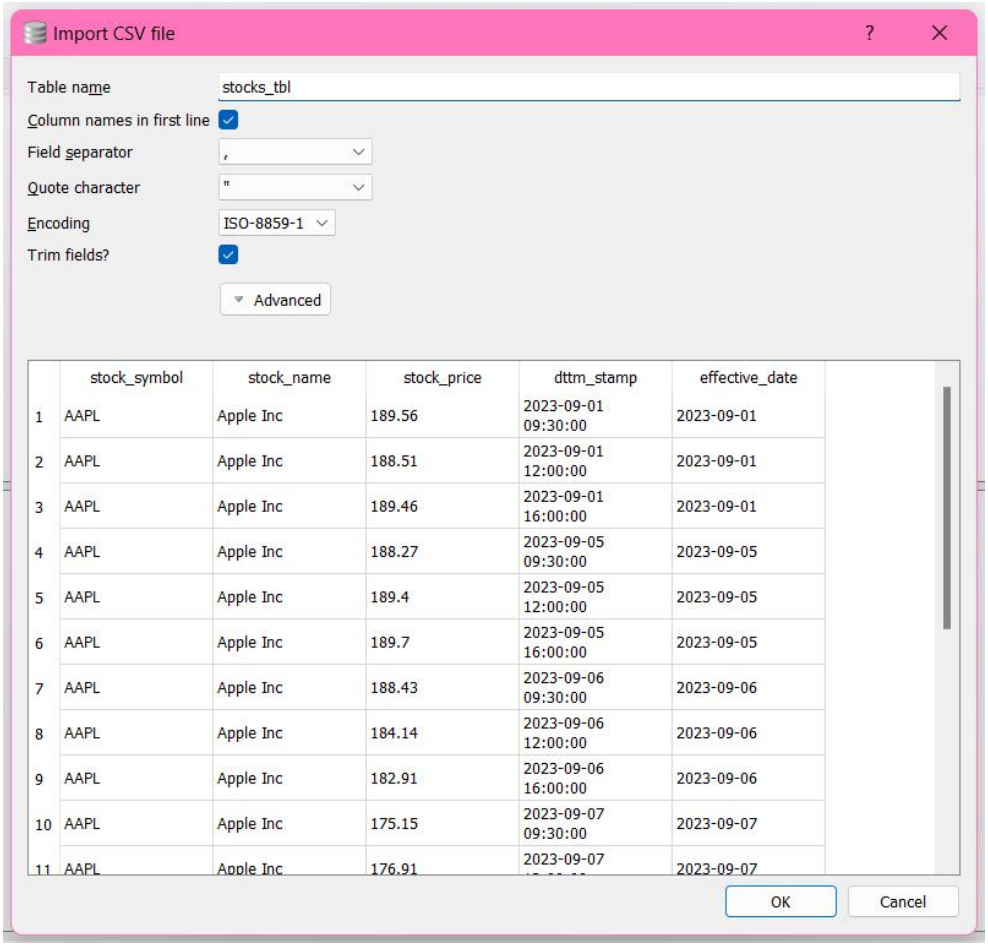
Data before importing/inserting any records:

```
1  SELECT count(*)
2  FROM stocks_tbl;
```

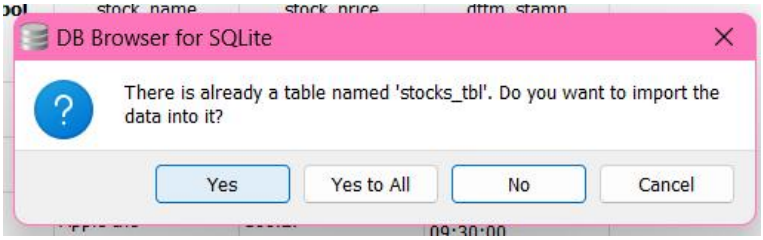
	count(*)
1	0

To import data via CSV file [*File > Import > Table from CSV file… > Open the CSV file*]:

- *Note: When importing records to an existing table in the database, make sure that the table name or filename (filename will be the default table name when using this functionality), column name(s), and number of columns are the same to the database table. Another thing to note as well are the constraints for each columns.*



	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	Apple Inc	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	Apple Inc	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	Apple Inc	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	Apple Inc	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	Apple Inc	176.91	2023-09-07	2023-09-07



```sql
SELECT count(*)
FROM stocks_tbl;
```

	count(*)
1	69

To insert records via INSERT clause:

```sql
-- For inserting records to the stocks_tbl table:
INSERT INTO stocks_tbl
VALUES
    ("UBER", "Uber Technologies Inc", 45.31, "2023-09-07 09:30:00", "2023-09-07"),
    ("UBER", "Uber Technologies Inc", 46.10, "2023-09-07 12:00:00", "2023-09-07"),
    ("UBER", "Uber Technologies Inc", 46.25, "2023-09-07 16:00:00", "2023-09-07"),
    ("UBER", "Uber Technologies Inc", 46.34, "2023-09-08 09:30:00", "2023-09-08"),
    ("UBER", "Uber Technologies Inc", 46.99, "2023-09-08 12:00:00", "2023-09-08"),
    ("UBER", "Uber Technologies Inc", 47.23, "2023-09-08 16:00:00", "2023-09-08")
;
```

```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 15:
-- For inserting records to the stocks_tbl table:
INSERT INTO stocks_tbl
VALUES
        ("UBER", "Uber Technologies Inc", 45.31, "2023-09-07 09:30:00", "2023-09-07"),
        ("UBER", "Uber Technologies Inc", 46.10, "2023-09-07 12:00:00", "2023-09-07"),
        ("UBER", "Uber Technologies Inc", 46.25, "2023-09-07 16:00:00", "2023-09-07"),
        ("UBER", "Uber Technologies Inc", 46.34, "2023-09-08 09:30:00", "2023-09-08"),
        ("UBER", "Uber Technologies Inc", 46.99, "2023-09-08 12:00:00", "2023-09-08"),
        ("UBER", "Uber Technologies Inc", 47.23, "2023-09-08 16:00:00", "2023-09-08")
;
```

```sql
SELECT count(*)
FROM stocks_tbl;
```

	count(*)
1	75

```
-- For the validation of data in stocks_tbl:
SELECT *
FROM stocks_tbl;
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	Apple Inc	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	Apple Inc	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	Apple Inc	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	Apple Inc	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	Apple Inc	176.91	2023-09-07 12:00:00	2023-09-07
12	AAPL	Apple Inc	177.56	2023-09-07 16:00:00	2023-09-07
13	AAPL	Apple Inc	178.34	2023-09-08 09:30:00	2023-09-08
14	AAPL	Apple Inc	179.57	2023-09-08 12:00:00	2023-09-08
15	AAPL	Apple Inc	178.18	2023-09-08 16:00:00	2023-09-08
16	TSLA	Tesla Inc	257.39	2023-09-01 09:30:00	2023-09-01
17	TSLA	Tesla Inc	246.32	2023-09-01 12:00:00	2023-09-01

```
Execution finished without errors.
Result: 75 rows returned in 10ms
At line 26:
-- For the validation of data in stocks_tbl:
SELECT *
FROM stocks_tbl;
```

Note: Executing the above INSERT clause again and with the same values will result to UNIQUE constraint error, just like below, as we setup the PRIMARY KEYs of the stocks_tbl with stock_symbol and dttm_stamp. We cannot have the same stock_symbol value with same dttm_stamp value.

```
Execution finished with errors.
Result: UNIQUE constraint failed: stocks_tbl.stock_symbol, stocks_tbl.dttm_stamp
At line 15:
-- For inserting records to the stocks_tbl table:
INSERT INTO stocks_tbl
VALUES
        ("UBER", "Uber Technologies Inc", 45.31, "2023-09-07 09:30:00", "2023-09-07"),
        ("UBER", "Uber Technologies Inc", 46.10, "2023-09-07 12:00:00", "2023-09-07"),
        ("UBER", "Uber Technologies Inc", 46.25, "2023-09-07 16:00:00", "2023-09-07"),
        ("UBER", "Uber Technologies Inc", 46.34, "2023-09-08 09:30:00", "2023-09-08"),
        ("UBER", "Uber Technologies Inc", 46.99, "2023-09-08 12:00:00", "2023-09-08"),
        ("UBER", "Uber Technologies Inc", 47.23, "2023-09-08 16:00:00", "2023-09-08")
;
```

III. Challenges

A. Basic Analysis

Queries: Perform basic analysis on the data and identify trends.

1. What are the distinct stocks in the table?

- These are the 5 **DISTINCT** stocks we have selected, which are those that I have chosen to track.

```
4   -- What are the distinct stocks in the table?
5   SELECT DISTINCT stock_symbol, stock_name
6   FROM stocks_tbl;
7
```

	stock_symbol	stock_name
1	AAPL	Apple Inc
2	AMZN	Amazon.com, Inc.
3	INTC	Intel Corporation
4	TSLA	Tesla Inc
5	UBER	Uber Technologies Inc

- *ADDITIONAL: I have included the **COUNT** for each stocks as well, to see how many records we have for each of them.*

```
8   SELECT stock_symbol, stock_name, count(*)
9   FROM stocks_tbl
10  GROUP BY stock_symbol, stock_name;
11
```

	stock_symbol	stock_name	count(*)
1	AAPL	Apple Inc	15
2	AMZN	Amazon.com, Inc.	15
3	INTC	Intel Corporation	15
4	TSLA	Tesla Inc	15
5	UBER	Uber Technologies Inc	15

2. Query all data for a single stock. Do you notice any overall trends?

- The stock price from 9/1-9/6 is consistently higher than $180 for AAPL. However, the stock price suddenly dropped and is lower than $180 from 9/7-9/8.

```
13  -- Query all data for a single stock. Do you notice any overall trends?
14  SELECT *
15  FROM stocks_tbl
16  WHERE stock_symbol = "AAPL";
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	Apple Inc	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	Apple Inc	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	Apple Inc	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	Apple Inc	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	Apple Inc	176.91	2023-09-07 12:00:00	2023-09-07
12	AAPL	Apple Inc	177.56	2023-09-07 16:00:00	2023-09-07
13	AAPL	Apple Inc	178.34	2023-09-08 09:30:00	2023-09-08
14	AAPL	Apple Inc	179.57	2023-09-08 12:00:00	2023-09-08
15	AAPL	Apple Inc	178.18	2023-09-08 16:00:00	2023-09-08

3. Which rows have a price above 100? Between 40 to 50, etc?

- The results showed that there are 45 records where stock price is above $100 and they are all data from AAPL, AMZN, and TSLA stocks (15 out of 15).

```
18  -- Which rows have a price above 100? between 40 to 50, etc?
19  SELECT *
20  FROM stocks_tbl
21  WHERE stock_price > 100;
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	Apple Inc	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	Apple Inc	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	Apple Inc	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	Apple Inc	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	Apple Inc	176.91	2023-09-07 12:00:00	2023-09-07
12	AAPL	Apple Inc	177.56	2023-09-07 16:00:00	2023-09-07
13	AAPL	Apple Inc	178.34	2023-09-08 09:30:00	2023-09-08
14	AAPL	Apple Inc	179.57	2023-09-08 12:00:00	2023-09-08

```
Execution finished without errors.
Result: 45 rows returned in 8ms
At line 18:
-- Which rows have a price above 100? between 40 to 50, etc?
SELECT *
FROM stocks_tbl
WHERE stock_price > 100;
```

- *ADDITIONAL: I have updated the **COUNT** query we have from Question #1 and added the **WHERE** clause to filter records that only have stock_price above $100.*

```
17  -- Which rows have a price above 100? between 40 to 50, etc?
18  SELECT *
19  FROM stocks_tbl
20  WHERE stock_price > 100;
21
22  SELECT stock_symbol, stock_name, count(*)
23  FROM stocks_tbl
24  WHERE stock_price > 100
25  GROUP BY stock_symbol, stock_name;
26
```

	stock_symbol	stock_name	count(*)
1	AAPL	Apple Inc	15
2	AMZN	Amazon.com, Inc.	15
3	TSLA	Tesla Inc	15

- For the **BETWEEN** query, I adjusted the range to $30-$50 to better fit the data I gathered. The results showed that there are 30 records where stock price is between $30-$50 and they are all data from INTC and UBER stocks (15 out of 15).

```
28  SELECT *
29  FROM stocks_tbl
30  WHERE stock_price BETWEEN 30 AND 50;
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	INTC	Intel Corporation	35.76	2023-09-01 09:30:00	2023-09-01
2	INTC	Intel Corporation	36.17	2023-09-01 12:00:00	2023-09-01
3	INTC	Intel Corporation	36.61	2023-09-01 16:00:00	2023-09-01
4	INTC	Intel Corporation	36.59	2023-09-05 09:30:00	2023-09-05
5	INTC	Intel Corporation	36.97	2023-09-05 12:00:00	2023-09-05
6	INTC	Intel Corporation	36.71	2023-09-05 16:00:00	2023-09-05
7	INTC	Intel Corporation	36.48	2023-09-06 09:30:00	2023-09-06
8	INTC	Intel Corporation	36.33	2023-09-06 12:00:00	2023-09-06
9	INTC	Intel Corporation	36.98	2023-09-06 16:00:00	2023-09-06
10	INTC	Intel Corporation	36.86	2023-09-07 09:30:00	2023-09-07
11	INTC	Intel Corporation	37.65	2023-09-07 12:00:00	2023-09-07
12	INTC	Intel Corporation	38.18	2023-09-07 16:00:00	2023-09-07
13	INTC	Intel Corporation	38.17	2023-09-08 09:30:00	2023-09-08
14	INTC	Intel Corporation	38.16	2023-09-08 12:00:00	2023-09-08
15	INTC	Intel Corporation	38.01	2023-09-08 16:00:00	2023-09-08
16	UBER	Uber Technologies Inc	47.5	2023-09-01 09:30:00	2023-09-01

```
Execution finished without errors.
Result: 30 rows returned in 4ms
At line 28:
SELECT *
FROM stocks_tbl
WHERE stock_price BETWEEN 30 AND 50;
```

- *ADDITIONAL: I have updated the **COUNT** query we have from Question #1 and added the **WHERE** clause to filter records that only have stock_price that is between $30-$50.*

```
27  SELECT *
28  FROM stocks_tbl
29  WHERE stock_price BETWEEN 30 AND 50;
30
31  SELECT stock_symbol, stock_name, count(*)
32  FROM stocks_tbl
33  WHERE stock_price BETWEEN 30 AND 50
34  GROUP BY stock_symbol, stock_name;
35
```

	stock_symbol	stock_name	count(*)
1	INTC	Intel Corporation	15
2	UBER	Uber Technologies Inc	15

4. Sort the table by price. What are the minimum and maximum prices?

- When we use **ORDER BY** on stock_price column (ascending order is the default), we can see that Intel Corporation (INTC) have the lowest stock price of $35.76 on the morning of September 1st, Friday. In the meantime, Tesla Inc (TSLA) have the highest stock price of $257.39 on the morning of September 1st, Friday.

```
37  -- Sort the table by price. What are the minimum and maximum prices?
38  SELECT *
39  FROM stocks_tbl
40  ORDER BY stock_price;
41
42
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	INTC	Intel Corporation	35.76	2023-09-01 09:30:00	2023-09-01
2	INTC	Intel Corporation	36.17	2023-09-01 12:00:00	2023-09-01
3	INTC	Intel Corporation	36.33	2023-09-06 12:00:00	2023-09-06
4	INTC	Intel Corporation	36.48	2023-09-06 09:30:00	2023-09-06
5	INTC	Intel Corporation	36.59	2023-09-05 09:30:00	2023-09-05
6	INTC	Intel Corporation	36.61	2023-09-01 16:00:00	2023-09-01
7	INTC	Intel Corporation	36.71	2023-09-05 16:00:00	2023-09-05
8	INTC	Intel Corporation	36.86	2023-09-07 09:30:00	2023-09-07
9	INTC	Intel Corporation	36.97	2023-09-05 12:00:00	2023-09-05
10	INTC	Intel Corporation	36.98	2023-09-06 16:00:00	2023-09-06
11	INTC	Intel Corporation	37.65	2023-09-07 12:00:00	2023-09-07
12	INTC	Intel Corporation	38.01	2023-09-08 16:00:00	2023-09-08
13	INTC	Intel Corporation	38.16	2023-09-08 12:00:00	2023-09-08
14	INTC	Intel Corporation	38.17	2023-09-08 09:30:00	2023-09-08
15	INTC	Intel Corporation	38.18	2023-09-07 16:00:00	2023-09-07
16	UBER	Uber Technologies Inc	45.31	2023-09-07 09:30:00	2023-09-07

```
Execution finished without errors.
Result: 75 rows returned in 10ms
At line 37:
-- Sort the table by price. What are the minimum and maximum prices?
SELECT *
FROM stocks_tbl
ORDER BY stock_price;
```

```sql
-- Sort the table by price. What are the minimum and maximum prices?
SELECT *
FROM stocks_tbl
ORDER BY stock_price;
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
61	TSLA	Tesla Inc	244.95	2023-09-05 09:30:00	2023-09-05
62	TSLA	Tesla Inc	244.97	2023-09-07 09:30:00	2023-09-07
63	TSLA	Tesla Inc	245.01	2023-09-01 16:00:00	2023-09-01
64	TSLA	Tesla Inc	246.32	2023-09-01 12:00:00	2023-09-01
65	TSLA	Tesla Inc	247.15	2023-09-07 12:00:00	2023-09-07
66	TSLA	Tesla Inc	248.5	2023-09-08 16:00:00	2023-09-08
67	TSLA	Tesla Inc	248.65	2023-09-06 12:00:00	2023-09-06
68	TSLA	Tesla Inc	251.35	2023-09-08 09:30:00	2023-09-08
69	TSLA	Tesla Inc	251.49	2023-09-07 16:00:00	2023-09-07
70	TSLA	Tesla Inc	251.92	2023-09-06 16:00:00	2023-09-06
71	TSLA	Tesla Inc	252.79	2023-09-05 12:00:00	2023-09-05
72	TSLA	Tesla Inc	253.1	2023-09-08 12:00:00	2023-09-08
73	TSLA	Tesla Inc	255.15	2023-09-06 09:30:00	2023-09-06
74	TSLA	Tesla Inc	256.49	2023-09-05 16:00:00	2023-09-05
75	TSLA	Tesla Inc	257.39	2023-09-01 09:30:00	2023-09-01

```
Execution finished without errors.
Result: 75 rows returned in 10ms
At line 37:
-- Sort the table by price. What are the minimum and maximum prices?
SELECT *
FROM stocks_tbl
ORDER BY stock_price;
```

B. Intermediate Challenge

1. Explore using aggregate functions to look at key statistics about the data (e.g., min, max, average).

- Below is the result set from our query using aggregate functions to look at the key statistics of the overall data. We can see that in using the **`min()`** function, we got the value of $35.76, and the value of $257.39 for **`max()`** function, which are the same as the values we got by using the **`ORDER BY`** query to manually get the minimum and maximum price (Basic Analysis Question #4). Instead of manually browsing and scrolling through the data to check the minimum and maximum value (stock_price) using the **`ORDER BY`** clause, whether in ascending or descending order, we can use aggregate functions such as **`min()`** and **`max()`** to do so automatically.
- *ADDITIONAL: Notice that to get the average price, we also use another function called **`round()`**, to round off the values in the column (stock_price) to the number of decimal places specified by the integer.*

```
43  -- B.Intermediate Challenge
44  -- 1.Explore using aggregate functions to look at key statistics about the data (e.g., min, max, average).
45  SELECT min(stock_price) AS [Lowest Price], round(avg(stock_price),2) AS [Average Price], max(stock_price) AS [Highest Price]
46  FROM stocks_tbl;
47
```

	Lowest Price	Average Price	Highest Price
1	35.76	130.92	257.39

- *In the above **`SELECT`** statement, we combined all our aggregate functions, **`min()`**, **`max()`**, **`avg()`**, into one statement, but we cannot see for which stock symbol the minimum and maximum stock price came from. To address this, we have created separate **`SELECT`** statements for each aggregate functions to show more details about them.*

```
47  SELECT stock_symbol, stock_name, min(stock_price) AS [Lowest Price], dttm_stamp
48  FROM stocks_tbl;
```

	stock_symbol	stock_name	Lowest Price	dttm_stamp
1	INTC	Intel Corporation	35.76	2023-09-01 09:30:00

```
50  SELECT stock_symbol, stock_name, max(stock_price) AS [Highest Price], dttm_stamp
51  FROM stocks_tbl;
```

	stock_symbol	stock_name	Highest Price	dttm_stamp
1	TSLA	Tesla Inc	257.39	2023-09-01 09:30:00

```
54  SELECT round(avg(stock_price),2) AS [Average Price]
55  FROM stocks_tbl;
56
```

	Average Price
1	130.92

2. Group the data by stock and repeat. How do the stocks compare to each other?

- In below result set, we can see the minimum, average, and the maximum price of each of our stocks. This way we can also see that Intel Corporation (INTC) have the lowest stock price of $35.76 and Tesla Inc have the highest stock price of $257.39, in comparison to all stocks we have. In addition, we can also perceive that the lowest price of AAPL, AMZN, and TSLA is higher than the overall average price ($130.92) of all the stocks that we have.
- Comparing the stocks to each other, we can focus on their range, which is the difference between the low and highest prices over a specific period of time. The relative difference between the high and the low, defines the historical volatility of the prices. Investors prefer lower volatility, so prices becoming significantly more volatile are said to indicate turmoil of some kind in the market.

  We can see that AAPL and TSLA have the highest range in prices, which might indicate turmoil of some kind in the market within the week. If we are going to check all the records we have for AAPL, we can see that a sudden drop of its stock price happened starting 4:00 PM of 9/6/2023. The stock price starting at that time is even lower than the average price for this specific stock. For TSLA, the record shows an up and down trend on the price from time to time and looks less stable. The price goes up and down at +/-1.6% on average.

  As for INTC, AMZN, and UBER, their stock prices moved a little over the week. Looking at these numbers, a Stock/Financial Analyst could say that they are stable with smaller price fluctuations. A conservative investor may prefer to invest in a more stable sectors.

```
62  SELECT stock_symbol, stock_name, min(stock_price) AS [Lowest Price], max(stock_price) AS [Highest Price],
63      round(avg(stock_price),2) AS [Average Price], round((max(stock_price) - min(stock_price)),2) AS [Range]
64  FROM stocks_tbl
65  GROUP BY stock_symbol, stock_name;
```

	stock_symbol	stock_name	Lowest Price	Highest Price	Average Price	Range
1	AAPL	Apple Inc	175.15	189.7	183.74	14.55
2	AMZN	Amazon.com, Inc.	133.95	139.41	136.96	5.46
3	INTC	Intel Corporation	35.76	38.18	37.04	2.42
4	TSLA	Tesla Inc	244.95	257.39	250.35	12.44
5	UBER	Uber Technologies Inc	45.31	47.5	46.52	2.19

AAPL records:

```
42  SELECT *
43  FROM stocks_tbl
44  WHERE stock_symbol = "AAPL";
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	Apple Inc	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	Apple Inc	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	Apple Inc	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	Apple Inc	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	Apple Inc	176.91	2023-09-07 12:00:00	2023-09-07
12	AAPL	Apple Inc	177.56	2023-09-07 16:00:00	2023-09-07
13	AAPL	Apple Inc	178.34	2023-09-08 09:30:00	2023-09-08
14	AAPL	Apple Inc	179.57	2023-09-08 12:00:00	2023-09-08
15	AAPL	Apple Inc	178.18	2023-09-08 16:00:00	2023-09-08

```
Execution finished without errors.
Result: 15 rows returned in 15ms
At line 42:
SELECT *
FROM stocks_tbl
WHERE stock_symbol = "AAPL";
```

TSLA records:

```
46  SELECT *
47  FROM stocks_tbl
48  WHERE stock_symbol = "TSLA";
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	TSLA	Tesla Inc	257.39	2023-09-01 09:30:00	2023-09-01
2	TSLA	Tesla Inc	246.32	2023-09-01 12:00:00	2023-09-01
3	TSLA	Tesla Inc	245.01	2023-09-01 16:00:00	2023-09-01
4	TSLA	Tesla Inc	244.95	2023-09-05 09:30:00	2023-09-05
5	TSLA	Tesla Inc	252.79	2023-09-05 12:00:00	2023-09-05
6	TSLA	Tesla Inc	256.49	2023-09-05 16:00:00	2023-09-05
7	TSLA	Tesla Inc	255.15	2023-09-06 09:30:00	2023-09-06
8	TSLA	Tesla Inc	248.65	2023-09-06 12:00:00	2023-09-06
9	TSLA	Tesla Inc	251.92	2023-09-06 16:00:00	2023-09-06
10	TSLA	Tesla Inc	244.97	2023-09-07 09:30:00	2023-09-07
11	TSLA	Tesla Inc	247.15	2023-09-07 12:00:00	2023-09-07
12	TSLA	Tesla Inc	251.49	2023-09-07 16:00:00	2023-09-07
13	TSLA	Tesla Inc	251.35	2023-09-08 09:30:00	2023-09-08
14	TSLA	Tesla Inc	253.1	2023-09-08 12:00:00	2023-09-08
15	TSLA	Tesla Inc	248.5	2023-09-08 16:00:00	2023-09-08

```
Execution finished without errors.
Result: 15 rows returned in 11ms
At line 46:
SELECT *
FROM stocks_tbl
WHERE stock_symbol = "TSLA";
```

3. Group the data by day or hour of day. Does day of week or time of day impact prices?

- I have grouped our records by date (effective_date) and get the opening balance (09:30 AM) and closing balance (04:00 PM) for each dates and stocks. We also wanted to get the Change and Change% of the price. If the sign is negative (-), it means that the price decreased. If it's positive, the price increased over time.
- For instance, we wanted to check if there is a win or loss for each stocks for 9/1/2023 date by getting the difference of closing balance and opening balance.
  - AAPL: We have an opening balance of $189.56 and closing balance of 189.46 - a loss of (-)0.1 (Change %: -0.05%).
  - AMZN: We have an opening balance of $139.41 and closing balance of 138.12 - a loss of (-)1.29 (Change %: -0.93%)
  - INTC: We have an opening balance of $35.76 and closing balance of 36.61 - a win of 0.85 (Change %: 2.38%)
  - TSLA: We have an opening balance of $257.39 and closing balance of 245.01 - a loss of (-)12.38 (Change %: -4.81%)
  - UBER: We have an opening balance of $47.5 and closing balance of 47.02 - a loss of (-)0.48 (Change %: -1.01%)

```sql
WITH change_comp_query AS (
                            SELECT DISTINCT stock_symbol, effective_date,
                                    (SELECT stock_price FROM stocks_tbl AS [TMP_1]
                                        WHERE stocks_tbl.stock_symbol = TMP_1.stock_symbol
                                        AND stocks_tbl.effective_date = TMP_1.effective_date
                                        AND time(dttm_stamp) = "09:30:00") AS opening_balance,
                                    (SELECT stock_price FROM stocks_tbl AS [TMP_1]
                                        WHERE stocks_tbl.stock_symbol = TMP_1.stock_symbol
                                        AND stocks_tbl.effective_date = TMP_1.effective_date
                                        AND time(dttm_stamp) = "16:00:00") AS closing_balance
                            FROM stocks_tbl
                        )
SELECT *,
        round((closing_balance - opening_balance), 2) AS [Change],
        round((100 * (closing_balance - opening_balance) / opening_balance), 2) AS [Change %]
FROM change_comp_query;
```

	stock_symbol	effective_date	opening_balance	closing_balance	Change	Change %
1	AAPL	2023-09-01	189.56	189.46	-0.1	-0.05
2	AAPL	2023-09-05	188.27	189.7	1.43	0.76
3	AAPL	2023-09-06	188.43	182.91	-5.52	-2.93
4	AAPL	2023-09-07	175.15	177.56	2.41	1.38
5	AAPL	2023-09-08	178.34	178.18	-0.16	-0.09
6	AMZN	2023-09-01	139.41	138.12	-1.29	-0.93
7	AMZN	2023-09-05	137.64	137.27	-0.37	-0.27
8	AMZN	2023-09-06	136.18	135.36	-0.82	-0.6
9	AMZN	2023-09-07	133.95	137.85	3.9	2.91
10	AMZN	2023-09-08	136.97	138.23	1.26	0.92
11	INTC	2023-09-01	35.76	36.61	0.85	2.38
12	INTC	2023-09-05	36.59	36.71	0.12	0.33
13	INTC	2023-09-06	36.48	36.98	0.5	1.37
14	INTC	2023-09-07	36.86	38.18	1.32	3.58
15	INTC	2023-09-08	38.17	38.01	-0.16	-0.42
16	TSLA	2023-09-01	257.39	245.01	-12.38	-4.81
17	TSLA	2023-09-05	244.95	256.49	11.54	4.71
18	TSLA	2023-09-06	255.15	251.92	-3.23	-1.27
19	TSLA	2023-09-07	244.97	251.49	6.52	2.66
20	TSLA	2023-09-08	251.35	248.5	-2.85	-1.13
21	UBER	2023-09-01	47.5	47.02	-0.48	-1.01
22	UBER	2023-09-05	46.87	46.54	-0.33	-0.7
23	UBER	2023-09-06	46.34	45.91	-0.43	-0.93
24	UBER	2023-09-07	45.31	46.25	0.94	2.07
25	UBER	2023-09-08	46.34	47.23	0.89	1.92

```
Execution finished without errors.
Result: 25 rows returned in 13ms
At line 88:
WITH change_comp_query AS (
                                                    SELECT DISTINCT stock_symbol, effective_date,
                                                                    (SELECT stock_price FROM stocks_tbl AS [TMP_1]
                                                                            WHERE stocks_tbl.stock_symbol = TMP_1.stock_symbol
                                                                            AND stocks_tbl.effective_date = TMP_1.effective_date
                                                                            AND time(dttm_stamp) = "09:30:00") AS opening_balance,
                                                                    (SELECT stock_price FROM stocks_tbl AS [TMP_1]
                                                                            WHERE stocks_tbl.stock_symbol = TMP_1.stock_symbol
                                                                            AND stocks_tbl.effective_date = TMP_1.effective_date
                                                                            AND time(dttm_stamp) = "16:00:00") AS closing_balance
                                                    FROM stocks_tbl
                                            )
SELECT *,
                round((closing_balance - opening_balance), 2) AS [Change],
                round((100 * (closing_balance - opening_balance) / opening_balance), 2) AS [Change %]
FROM change_comp_query;
```

4. Which of the rows have a price greater than the average of all prices in the data-set?

- A total of 45 out of 75 of our records have a higher price that the overall average. They all come from AAPL, AMZN, and TSLA stocks.

```
105  -- 4.Which of the rows have a price greater than the average of all prices in the dataset?
106  SELECT *
107  FROM stocks_tbl
108  WHERE stock_price > (SELECT avg(stock_price)
109                       FROM stocks_tbl)
110  ;
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	Apple Inc	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	Apple Inc	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	Apple Inc	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	Apple Inc	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	Apple Inc	176.91	2023-09-07 12:00:00	2023-09-07
12	AAPL	Apple Inc	177.56	2023-09-07 16:00:00	2023-09-07
13	AAPL	Apple Inc	178.34	2023-09-08 09:30:00	2023-09-08
14	AAPL	Apple Inc	179.57	2023-09-08 12:00:00	2023-09-08
15	AAPL	Apple Inc	178.18	2023-09-08 16:00:00	2023-09-08

```
Execution finished without errors.
Result: 45 rows returned in 12ms
At line 105:
-- 4.Which of the rows have a price greater than the average of all prices in the dataset?
SELECT *
FROM stocks_tbl
WHERE stock_price > (SELECT avg(stock_price)
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
16	TSLA	Tesla Inc	257.39	2023-09-01 09:30:00	2023-09-01
17	TSLA	Tesla Inc	246.32	2023-09-01 12:00:00	2023-09-01
18	TSLA	Tesla Inc	245.01	2023-09-01 16:00:00	2023-09-01
19	TSLA	Tesla Inc	244.95	2023-09-05 09:30:00	2023-09-05
20	TSLA	Tesla Inc	252.79	2023-09-05 12:00:00	2023-09-05
21	TSLA	Tesla Inc	256.49	2023-09-05 16:00:00	2023-09-05
22	TSLA	Tesla Inc	255.15	2023-09-06 09:30:00	2023-09-06
23	TSLA	Tesla Inc	248.65	2023-09-06 12:00:00	2023-09-06
24	TSLA	Tesla Inc	251.92	2023-09-06 16:00:00	2023-09-06
25	TSLA	Tesla Inc	244.97	2023-09-07 09:30:00	2023-09-07
26	TSLA	Tesla Inc	247.15	2023-09-07 12:00:00	2023-09-07
27	TSLA	Tesla Inc	251.49	2023-09-07 16:00:00	2023-09-07
28	TSLA	Tesla Inc	251.35	2023-09-08 09:30:00	2023-09-08
29	TSLA	Tesla Inc	253.1	2023-09-08 12:00:00	2023-09-08
30	TSLA	Tesla Inc	248.5	2023-09-08 16:00:00	2023-09-08

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
31	AMZN	Amazon.com, Inc.	139.41	2023-09-01 09:30:00	2023-09-01
32	AMZN	Amazon.com, Inc.	136.96	2023-09-01 12:00:00	2023-09-01
33	AMZN	Amazon.com, Inc.	138.12	2023-09-01 16:00:00	2023-09-01
34	AMZN	Amazon.com, Inc.	137.64	2023-09-05 09:30:00	2023-09-05
35	AMZN	Amazon.com, Inc.	136.39	2023-09-05 12:00:00	2023-09-05
36	AMZN	Amazon.com, Inc.	137.27	2023-09-05 16:00:00	2023-09-05
37	AMZN	Amazon.com, Inc.	136.18	2023-09-06 09:30:00	2023-09-06
38	AMZN	Amazon.com, Inc.	135.9	2023-09-06 12:00:00	2023-09-06
39	AMZN	Amazon.com, Inc.	135.36	2023-09-06 16:00:00	2023-09-06
40	AMZN	Amazon.com, Inc.	133.95	2023-09-07 09:30:00	2023-09-07
41	AMZN	Amazon.com, Inc.	135.76	2023-09-07 12:00:00	2023-09-07
42	AMZN	Amazon.com, Inc.	137.85	2023-09-07 16:00:00	2023-09-07
43	AMZN	Amazon.com, Inc.	136.97	2023-09-08 09:30:00	2023-09-08
44	AMZN	Amazon.com, Inc.	138.34	2023-09-08 12:00:00	2023-09-08
45	AMZN	Amazon.com, Inc.	138.23	2023-09-08 16:00:00	2023-09-08

```
112
113    SELECT stock_symbol, stock_name, count(*)
114    FROM stocks_tbl
115    WHERE stock_price > (SELECT avg(stock_price)
116                         FROM stocks_tbl)
117    GROUP BY stock_symbol, stock_name;
```

	stock_symbol	stock_name	count(*)
1	AAPL	Apple Inc	15
2	AMZN	Amazon.com, Inc.	15
3	TSLA	Tesla Inc	15

C. Advanced Challenge

1. In addition to the built-in aggregate functions, explore ways to calculate other key statistics about the data, such as the median or variance.

- Median is the element in the middle of an ordered list.

```
3   SELECT stock_symbol, stock_price
4   FROM stocks_tbl
5   ORDER BY stock_price
6   LIMIT 1
7   OFFSET (SELECT COUNT(*)
8           FROM stocks_tbl) / 2
9   ;
```

	stock_symbol	stock_price
1	AMZN	136.97

2. Let's refactor the data into 2 tables - stock_info to store general info about the stock itself (ie. symbol, name) and stock_prices to store the collected data on price (ie. symbol, datetime, price).

- Creation of new **`stock_info`** table that stores the general info about the stock, like the symbol and the name, with stock_symbol as the primary key. Records under this new table will be inserted from the original **`stocks_tbl`** table.

```
11  -- 2.Let's refactor the data into 2 tables - stock_info to
12  CREATE TABLE "stock_info" (
13      "stock_symbol"  TEXT NOT NULL,
14      "stock_name"    TEXT,
15      PRIMARY KEY("stock_symbol")
```

```
Execution finished without errors.
Result: query executed successfully. Took 99ms
At line 11:
-- 2.Let's refactor the data into 2 tables - stock_info to store
CREATE TABLE "stock_info" (
        "stock_symbol"    TEXT NOT NULL,
        "stock_name"      TEXT,
        PRIMARY KEY("stock_symbol")
);
```

```
18  SELECT count(*)
19  FROM stock_info;
```

	count(*)
1	0

- Insert records to **stock_info** table from the original **stocks_tbl** table, but we are only inserting unique records. With that said, there should only be 5 records to be inserted to **stock_info**.

```
22  -- Insert records to stock_info table from the original stocks_tbl table, bu
23  INSERT INTO stock_info (stock_symbol, stock_name)
24  SELECT DISTINCT stock_symbol, stock_name FROM stocks_tbl;
25
```

```
Execution finished without errors.
Result: query executed successfully. Took 2ms
At line 22:
-- Insert records to stock_info table from the original stocks_tbl table, but we
INSERT INTO stock_info (stock_symbol, stock_name)
SELECT DISTINCT stock_symbol, stock_name FROM stocks_tbl;
```

```
26  SELECT *
27  FROM stock_info;
28
```

	stock_symbol	stock_name
1	AAPL	Apple Inc
2	AMZN	Amazon.com, Inc.
3	INTC	Intel Corporation
4	TSLA	Tesla Inc
5	UBER	Uber Technologies Inc

- Creation of new **stock_prices** table that stores the collected data on price about the stock. Columns will include the stock_symbol, stock_price, dttm_stamp, and effective_date, with stock_symbol as the primary key. Records under this new table will be coming from the original **stocks_tbl** table and will be inserted upon the creation of the new table.

```
23  SELECT * FROM stocks_tbl;
```

	stock_symbol	stock_name	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	189.7	2023-09-05 16:00:00	2023-09-05

```sql
CREATE TABLE stock_prices AS
SELECT stock_symbol, stock_price, dttm_stamp, effective_date
FROM stocks_tbl;
```

```
Execution finished without errors.
Result: query executed successfully. Took 97ms
At line 25:
CREATE TABLE stock_prices AS
SELECT stock_symbol, stock_price, dttm_stamp, effective_date
FROM stocks_tbl;
```

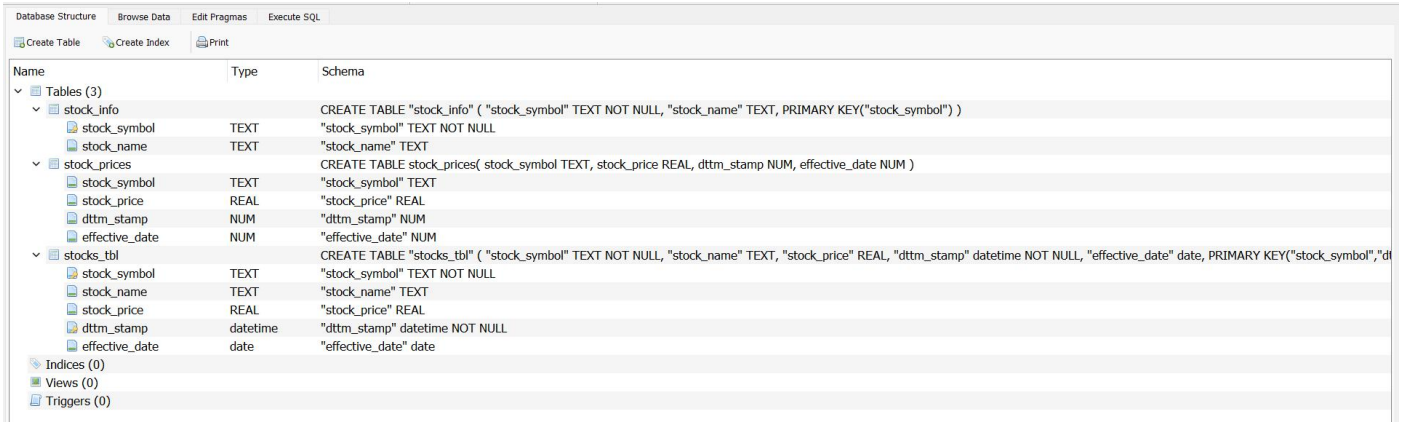
```sql
SELECT count(*)
FROM stock_prices;
```

	count(*)
1	75

```sql
SELECT *
FROM stock_prices;
```

	stock_symbol	stock_price	dttm_stamp	effective_date
1	AAPL	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	176.91	2023-09-07 12:00:00	2023-09-07
12	AAPL	177.56	2023-09-07 16:00:00	2023-09-07
13	AAPL	178.34	2023-09-08 09:30:00	2023-09-08
14	AAPL	179.57	2023-09-08 12:00:00	2023-09-08
15	AAPL	178.18	2023-09-08 16:00:00	2023-09-08
16	TSLA	257.39	2023-09-01 09:30:00	2023-09-01
17	TSLA	246.32	2023-09-01 12:00:00	2023-09-01
18	TSLA	245.01	2023-09-01 16:00:00	2023-09-01

```
Execution finished without errors.
Result: 75 rows returned in 11ms
At line 32:
SELECT *
FROM stock_prices;
```



3. Now, we do not need to repeat both symbol and name for each row of price data. Instead, join the 2 tables in order to view more information on the stock with each row of price.

```
42  -- 3.Now, we do not need to repeat both symbol and name for each row of price data. Instead, join t
43  SELECT *
44  FROM stock_info
45  JOIN stock_prices
46      ON stock_info.stock_symbol = stock_prices.stock_symbol;
```

	stock_symbol	stock_name	stock_symbol	stock_price	dttm_stamp	effective_date
1	AAPL	Apple Inc	AAPL	189.56	2023-09-01 09:30:00	2023-09-01
2	AAPL	Apple Inc	AAPL	188.51	2023-09-01 12:00:00	2023-09-01
3	AAPL	Apple Inc	AAPL	189.46	2023-09-01 16:00:00	2023-09-01
4	AAPL	Apple Inc	AAPL	188.27	2023-09-05 09:30:00	2023-09-05
5	AAPL	Apple Inc	AAPL	189.4	2023-09-05 12:00:00	2023-09-05
6	AAPL	Apple Inc	AAPL	189.7	2023-09-05 16:00:00	2023-09-05
7	AAPL	Apple Inc	AAPL	188.43	2023-09-06 09:30:00	2023-09-06
8	AAPL	Apple Inc	AAPL	184.14	2023-09-06 12:00:00	2023-09-06
9	AAPL	Apple Inc	AAPL	182.91	2023-09-06 16:00:00	2023-09-06
10	AAPL	Apple Inc	AAPL	175.15	2023-09-07 09:30:00	2023-09-07
11	AAPL	Apple Inc	AAPL	176.91	2023-09-07 12:00:00	2023-09-07
12	AAPL	Apple Inc	AAPL	177.56	2023-09-07 16:00:00	2023-09-07
13	AAPL	Apple Inc	AAPL	178.34	2023-09-08 09:30:00	2023-09-08
14	AAPL	Apple Inc	AAPL	179.57	2023-09-08 12:00:00	2023-09-08

```
Execution finished without errors.
Result: 75 rows returned in 27ms
At line 42:
-- 3.Now, we do not need to repeat both symbol and name for each row of price data. Instead, join the 2
SELECT *
FROM stock_info
JOIN stock_prices
        ON stock_info.stock_symbol = stock_prices.stock_symbol;
```

```
48  SELECT stock_info.stock_symbol, stock_info.stock_name, count(*)
49  FROM stock_info
50  JOIN stock_prices
51      ON stock_info.stock_symbol = stock_prices.stock_symbol
52  GROUP BY stock_info.stock_symbol, stock_info.stock_name;
```

	stock_symbol	stock_name	count(*)
1	AAPL	Apple Inc	15
2	AMZN	Amazon.com, Inc.	15
3	INTC	Intel Corporation	15
4	TSLA	Tesla Inc	15
5	UBER	Uber Technologies Inc	15

4. Add more variables to the stock_info table and update the data (e.g., sector, industry, etc).

- We will be executing an **ALTER** clause to add some columns to an existing table **stock_info**, such as stock_exchange and stock_sector. We will also be including some information about the company.

```
67  ALTER TABLE stock_info
68  ADD "stock_exchange" TEXT;
69
70  ALTER TABLE stock_info
71  ADD "stock_sector" TEXT;
72
73  ALTER TABLE stock_info
74  ADD "company_ceo" TEXT;
75
76  ALTER TABLE stock_info
77  ADD "company_founded" date;
78
79  ALTER TABLE stock_info
80  ADD "company_headquarters" TEXT;
81
82  ALTER TABLE stock_info
83  ADD "company_website" TEXT;
84
85  ALTER TABLE stock_info
86  ADD "company_employes" INTEGER;
```

stock_exchange	TEXT	"stock_exchange" TEXT
stock_sector	TEXT	"stock_sector" TEXT
company_ceo	TEXT	"company_ceo" TEXT
company_founded	date	"company_founded" date
company_headquarters	TEXT	"company_headquarters" TEXT
company_website	TEXT	"company_website" TEXT
company_employes	INTEGER	"company_employes" INTEGER

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Create Index | Print

Name	Type	Schema
Tables (3)		
stock_info		CREATE TABLE "stock_info" ("stock_symbol" TEXT NOT NULL, "stock_name" TEXT, "stock_exchange" TEXT, "stock_sector" TEXT, "company_ceo" TEXT, "company_founded" date, "company_head
stock_symbol	TEXT	"stock_symbol" TEXT NOT NULL
stock_name	TEXT	"stock_name" TEXT
stock_exchange	TEXT	"stock_exchange" TEXT
stock_sector	TEXT	"stock_sector" TEXT
company_ceo	TEXT	"company_ceo" TEXT
company_founded	date	"company_founded" date
company_headquarters	TEXT	"company_headquarters" TEXT
company_website	TEXT	"company_website" TEXT
company_employes	INTEGER	"company_employes" INTEGER
stock_prices		CREATE TABLE stock_prices(stock_symbol TEXT, stock_price REAL, dttm_stamp NUM, effective_date NUM)
stocks_tbl		CREATE TABLE "stocks_tbl" ("stock_symbol" TEXT NOT NULL, "stock_name" TEXT, "stock_price" REAL, "dttm_stamp" datetime NOT NULL, "effective_date" date, PRIMARY KEY("stock_symbol","d
Indices (0)		
Views (0)		
Triggers (0)		

```
56  SELECT *
57  FROM stock_info;
58
59  ALTER TABLE stock_info
60  ADD "stock_exchange" TEXT;
```

	stock_symbol	stock_name	stock_exchange	stock_sector	company_ceo	company_founded	company_headquarters	company_website	company_employes
1	AAPL	Apple Inc	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	AMZN	Amazon.com, Inc.	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	INTC	Intel Corporation	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	TSLA	Tesla Inc	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	UBER	Uber Technologies Inc	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
Execution finished without errors.
Result: 5 rows returned in 12ms
At line 54:
-- 4.Add more variables to the stock_info table and update the data (e.g., sector, industry, etc).
-- We will be executing an ALTER clause to add some columns to an existing table stock_info, such as stock_exchange and stock_sector. We will also be including some
SELECT *
FROM stock_info;
```

- We will be executing an **UPDATE** clause to update the value of the newly added columns.

```
-- We will be executing an UPDATE clause to update the value of the newly added columns.
UPDATE stock_info
SET stock_sector = "Technology",
    company_ceo = "Tim Cook",
    company_founded = "1976-04-01",
    company_headquarters = "Cupertino, California, United States",
    company_website = "apple.com",
    company_employes = 164000
WHERE stock_symbol = "AAPL";
```

```
Execution finished without errors.
Result: query executed successfully. Took 2ms
At line 80:
-- We will be executing an UPDATE clause to update the value of the newly added columns.
UPDATE stock_info
SET stock_sector = "Technology",
         company_ceo = "Tim Cook",
         company_founded = "1976-04-01",
         company_headquarters = "Cupertino, California, United States",
         company_website = "apple.com",
         company_employes = 164000
WHERE stock_symbol = "AAPL";
```

```
UPDATE stock_info
SET stock_exchange =
    CASE
        WHEN stock_symbol = "UBER" THEN "NYSE"
        ELSE "NASDAQ"
    END;
```

```
Execution finished without errors.
Result: query executed successfully. Took 0ms, 5 rows affected
At line 92:
UPDATE stock_info
SET stock_exchange =
         CASE
                  WHEN stock_symbol = "UBER" THEN "NYSE"
                  ELSE "NASDAQ"
         END;
```

```
SELECT * FROM stock_info;
```

	stock_symbol	stock_name	stock_exchange	stock_sector	company_ceo	company_founded	company_headquarters	company_website	company_employes
1	AAPL	Apple Inc	NASDAQ	Technology	Tim Cook	1976-04-01	Cupertino, California, United States	apple.com	164000
2	AMZN	Amazon.com, Inc.	NASDAQ	Technology	Andy Jassy	1994-07-05	Seattle, Washington, United States	aboutamazon.com	1461000
3	INTC	Intel Corporation	NASDAQ	Technology	Patrick P. Gelsinger	1968-07-18	Santa Clara, California, United States	intel.com	131900
4	TSLA	Tesla Inc	NASDAQ	Automotive	Elon Musk	2003-07-01	*NULL*	tesla.com	127855
5	UBER	Uber Technologies Inc	NYSE	Transportation	Dara Khosrowshahi	2009-03-01	San Francisco, California, United States	uber.com	32200

```
Execution finished without errors.
Result: 5 rows returned in 15ms
At line 80:
SELECT * FROM stock_info;
```

D. NEXT CHALLENGE:

On a larger scale in relation to this project, I want to track all the stocks found on NASDAQ 100 (cnbc.com) on a daily basis or on a bigger range. I would also like to show some visualizations through excel.

I will update this document again or just create a new one when I worked on to this project.

On to the next level!!!

I would like to thank you for going through this whole document and reading my findings.

Your feedback will be highly appreciated!

See you on my next adventure!

- GD