

Part 1

Devise relational algebra queries for the following schema of a database storing information about daily stock prices and basic transactions made by a trading firm. You should define domains so as to make the schema complete.

- STOCK (ticker, exchange)

ticker: the stock's ticker symbol; e.g. GOOG, AAPL, GE

exchange: the exchange where the ticker is listed; e.g. NYSE, NASDAQ

- PRICE (ticker, date, close)

ticker: the stock's ticker symbol

date: the date of the price information

close: the closing price of the stock

- BUYnSELL (buy or sell, ticker, date, timestamp, value, num of shares)

buy or sell: 'BUY' or 'SELL'

ticker: the stock's ticker symbol

date: the date of the price information

timestamp: time of the transaction

value: the price of a single share

num_of_shares: number of shares (bought or sold)

- i. Find the tickers and all closing prices of all stocks exchanged in 2017.

①

$$\pi_{\text{ticker}, \text{close}} \left[\sigma_{\text{ticker} = \text{ticker}_2 \text{ AND } \text{date} = 2017} (\text{STOCK} \times \rho(\text{PRICE})) \right]$$

Handwritten notes:
 - $\text{ticker} \rightarrow \text{ticker}_2$
 - $\text{date} = 2017$
 - $\text{STOCK} \times \rho(\text{PRICE})$
 - $\pi_{\text{ticker}, \text{close}}$

- ii. Find all tickers (i.e. for all dates) whose closing price is both higher than 'IBM' on '9/20/2018' and no higher than 'GOOG' on '9/20/2018'.

②

Π_{tickers}	σ date = '2018-09-20' AND ticker1 = 'IBM' AND close1 > close	Price x P close \rightarrow close1 AND tickers \rightarrow ticker1 (Price)
------------------------	--	--

\cap

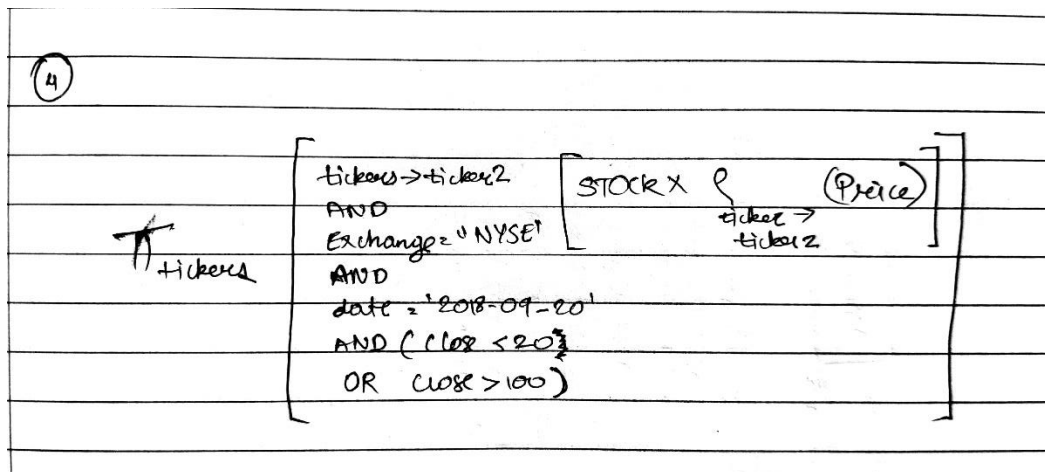
Π_{tickers}	σ date = '2018-09-20' AND ticker2 = 'GOOG' AND close2 < close	Price x P close \rightarrow close2 AND tickers \rightarrow ticker2 (Price)
------------------------	---	--

- iii. Find the tickers of all stocks that closed at the highest price on '9/20/2018'. (we are asking for "all stocks" since there may be more than one with the same "highest price").

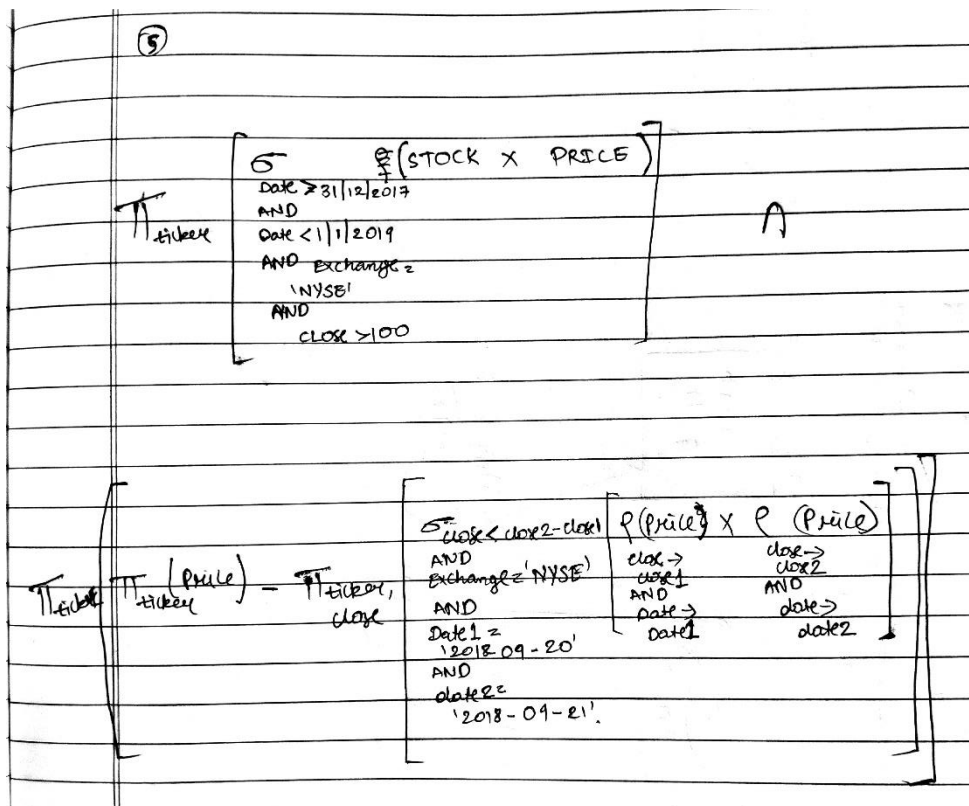
③

Π_{tickers}	σ date = '2018-09-20' (Price)	-	Π_{tickers}	σ date = '2018-09-20' AND close < close1	Price x P (Price) ticker \rightarrow ticker1 AND close \rightarrow close1
------------------------	--	---	------------------------	--	--

- iv. Find the tickers of all stocks in 'NYSE' whose closing price on '9/20/2018' was either strictly below \$20 or strictly above \$100



- v. Find all tickers in 'NYSE' of the stocks whose closing price showed the highest increase between '9/20/2018' and '9/21/2018' in 'NYSE' and whose closing price was (in 'NYSE') strictly above \$100 for the entire 2018.

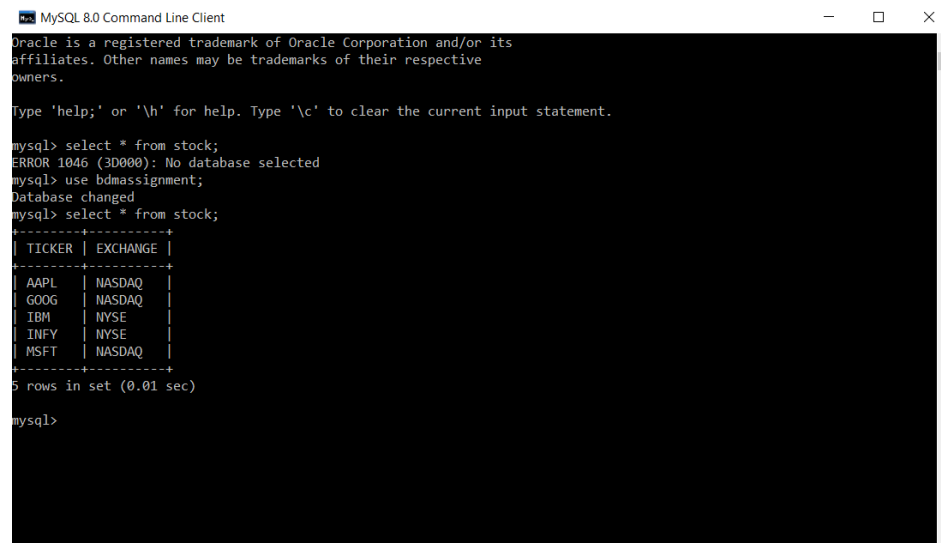


The Additional Database Instances that are added other than the ones mentioned in the end of lab exercise is separated in the different script file.

Table 1: Stock

Query : `mysql> INSERT INTO STOCK (TICKER, EXCHANGE) VALUES
-> ('INFY', 'NYSE');`

Output:



```
MySQL 8.0 Command Line Client
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from stock;
ERROR 1046 (3D000): No database selected
mysql> use bdmassignment;
Database changed
mysql> select * from stock;
+-----+-----+
| TICKER | EXCHANGE |
+-----+-----+
| AAPL   | NASDAQ   |
| GOOG   | NASDAQ   |
| IBM    | NYSE     |
| INFY   | NYSE     |
| MSFT   | NASDAQ   |
+-----+-----+
5 rows in set (0.01 sec)

mysql>
```

Table 2 : Price

Query :
`INSERT INTO PRICE (TICKER, DATE, CLOSE) VALUES
-> ('BSE', '2018-09-20', 80),
-> ('VZ', '2018-09-20', 75),
-> ('INFY', '2018-09-20', 100),
-> ('INFY', '2018-09-21', 290),
-> ('AAPL', '2017-09-20', 100),
-> ('MSFT', '2017-09-20', 160),
-> ('GOOG', '2017-09-22', 120),
-> ('GOOG', '2017-09-20', 110),
-> ('RUTG', '2017-09-21', 200),
-> ('VZ', '2017-09-23', 90),
-> ('AAPL', '2018-09-20', 101),
-> ('amzn', '2018-09-20', 86),
-> ('RUTG', '2018-09-20', 100);`

Output:

```

mysql> select * from Price;
+-----+-----+-----+
| TICKER | DATE       | CLOSE |
+-----+-----+-----+
| AAPL   | 2018-09-21 | 101.5 |
| AAPL   | 2018-09-22 | 106.5 |
| GOOG   | 2018-09-20 | 100   |
| GOOG   | 2018-09-21 | 130   |
| GOOG   | 2018-09-22 | 110   |
| MSFT   | 2018-09-20 | 50    |
| MSFT   | 2018-09-21 | 188.5 |
| MSFT   | 2018-09-22 | 210   |
| IBM    | 2018-09-20 | 80    |
| IBM    | 2018-09-21 | 270   |
| IBM    | 2018-09-22 | 10    |
| BSE    | 2018-09-20 | 80    |
| VZ     | 2018-09-20 | 75    |
| INFY   | 2018-09-20 | 100   |
| INFY   | 2018-09-21 | 290   |
| AAPL   | 2017-09-20 | 100   |
| MSFT   | 2017-09-20 | 160   |
| GOOG   | 2017-09-22 | 120   |
| GOOG   | 2017-09-20 | 110   |
| RUTG   | 2017-09-21 | 200   |
| VZ     | 2017-09-23 | 90    |
| AAPL   | 2018-09-20 | 90    |
| amzn   | 2018-09-20 | 86    |
| RUTG   | 2018-09-20 | 100   |
+-----+-----+-----+
24 rows in set (0.00 sec)

mysql>

```

Table 3 : BUYnSELL

Query :

```

INSERT INTO BUYnSELL (TICKER,BUY_OR_SELL,DATE,PRICE,NUM_OF_SHARES)
VALUES
-> ('AAPL', 'SELL', '2017-09-20', 99,900)
-> ('AAPL', 'SELL', '2017-09-22', 105,2000)
-> ('AAPL', 'SELL', '2017-09-21', 80,1000);

```

Output:

```

mysql>
mysql>
mysql>
mysql>
mysql> select * from buynsell;
+-----+-----+-----+-----+-----+-----+
| TICKER | BUY_OR_SELL | DATE       | TIMESTAMP                | PRICE | NUM_OF_SHARES |
+-----+-----+-----+-----+-----+-----+
| IBM    | BUY          | 2018-09-20 | 2018-11-14 14:28:35      | 273   | 1100           |
| IBM    | SELL         | 2018-09-22 | 2018-11-14 14:28:35      | 270.5 | 2500           |
| IBM    | BUY          | 2018-09-21 | 2018-11-14 14:28:35      | 271   | 2400           |
| GOOG   | BUY          | 2018-09-20 | 2018-11-14 14:28:35      | 86    | 2200           |
| GOOG   | SELL         | 2018-09-20 | 2018-11-14 14:28:35      | 87    | 1000           |
| GOOG   | SELL         | 2018-09-21 | 2018-11-14 14:28:35      | 87.5  | 1000           |
| GOOG   | BUY          | 2018-09-21 | 2018-11-14 14:28:35      | 87    | 800            |
| GOOG   | SELL         | 2018-09-22 | 2018-11-14 14:28:35      | 86    | 500            |
| AAPL   | BUY          | 2018-09-20 | 2018-11-14 14:28:35      | 99    | 1000           |
| AAPL   | BUY          | 2018-09-20 | 2018-11-14 14:28:35      | 99.5  | 1000           |
| AAPL   | BUY          | 2018-09-21 | 2018-11-14 14:28:35      | 100   | 1000           |
| AAPL   | SELL         | 2018-09-22 | 2018-11-14 14:28:35      | 103   | 3000           |
| MSFT   | BUY          | 2018-09-20 | 2018-11-14 14:28:35      | 186   | 1500           |
| MSFT   | SELL         | 2018-09-21 | 2018-11-14 14:28:35      | 188   | 1000           |
| MSFT   | BUY          | 2018-09-22 | 2018-11-14 14:28:35      | 187   | 5000           |
| AAPL   | SELL         | 2017-09-20 | 2018-11-14 14:28:35      | 99    | 900            |
| AAPL   | SELL         | 2017-09-22 | 2018-11-14 14:28:35      | 105   | 2000           |
| AAPL   | SELL         | 2017-09-21 | 2018-11-14 14:28:35      | 80    | 1000           |
| AAPL   | SELL         | 2018-09-21 | 2018-11-14 14:28:35      | 80    | 100            |
+-----+-----+-----+-----+-----+-----+
19 rows in set (0.00 sec)

mysql>

```

Part 2

Queries

- i. (40 points) Realize the database schema of Part 1 in MYSQL. Specify appropriate domains, keys, ICs, and all types of constraints. Then, implement as SQL queries all the relational algebra queries you devised in Part 1. Wherever you had to calculate a MAX or MIN value you are allowed in SQL to use the primitives MAX or MIN if you find it easier this way (but you should be careful because these queries might need additional more advanced SQL commands).

Q. Find the dates where the total price (i.e. price times num of shares) of 'AAPL' the firm.

Query :

```
SELECT DISTINCT B.DATE FROM BUYNSELL B WHERE B.TICKER =  
'AAPL' AND B.BUY_OR_SELL = 'SELL'  
-> AND (B.PRICE*B.NUM_OF_SHARES) > (SELECT SUM(DISTINCT  
BNS.PRICE*BNS.NUM_OF_SHARES)  
-> AS TOTAL_PRICE FROM BUYNSELL BNS WHERE BNS.TICKER IN  
(SELECT S.TICKER FROM STOCK S  
-> WHERE S.EXCHANGE = 'NASDAQ' AND S.TICKER = 'AAPL'))  
AND BNS.BUY_OR_SELL = 'BUY');
```

Output:

```
mysql> SELECT DISTINCT B.DATE FROM BUYNSELL B WHERE B.TICKER = 'AAPL' AND B.BUY_OR_SELL = 'SELL'  
-> AND (B.PRICE*B.NUM_OF_SHARES) > (SELECT SUM(DISTINCT BNS.PRICE*BNS.NUM_OF_SHARES)  
-> AS TOTAL_PRICE FROM BUYNSELL BNS WHERE BNS.TICKER IN (SELECT S.TICKER FROM STOCK S  
-> WHERE S.EXCHANGE = 'NASDAQ' AND S.TICKER = 'AAPL')) AND BNS.BUY_OR_SELL = 'BUY');  
+-----+  
| DATE |  
+-----+  
| 2018-09-22 |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

(1) What is the main reason for not asking to do the above query in Relational Algebra? Justify your answer. Use a detailed explanation and back it up with examples.

- Relational algebra is like normal algebra (as in $2+3*x-y$), except we use relations as values instead of numbers, and the operations and operators are different. In relational Algebra we cannot perform certain standard aggregate functions: sum, count, avg, min, max which is required in the above query, so we use SQL query instead of relational algebra.

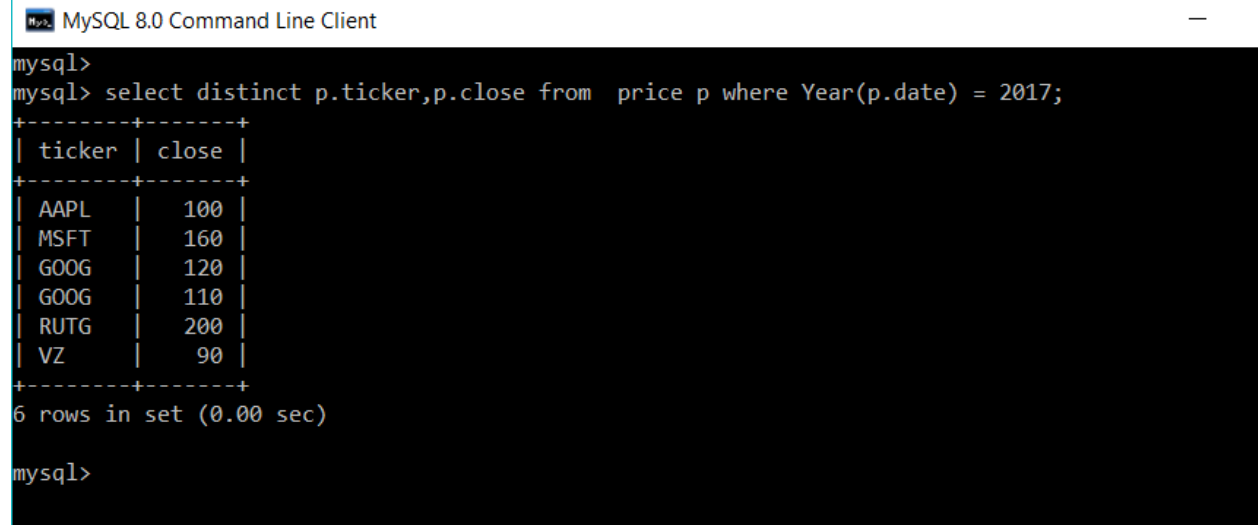
(2) Report the code that creates the database. For all the queries you wrote you should report the results of the queries on the example DB instance listed at the end of this lab exercise, but this example DB instance should be modified as follows. Whenever the answer to the query is empty (return nothing) you should add your own typical instances to the provide DB instance.

- i. Find the tickers and all closing prices of all stocks exchanged in 2017.

Query:

```
SELECT DISTINCT P.TICKER,P.CLOSE FROM PRICE P WHERE  
YEAR(P.DATE) = 2017;
```

Output:



```
MySQL 8.0 Command Line Client  
mysql>  
mysql> select distinct p.ticker,p.close from price p where Year(p.date) = 2017;  
+-----+-----+  
| ticker | close |  
+-----+-----+  
| AAPL  | 100   |  
| MSFT  | 160   |  
| GOOG  | 120   |  
| GOOG  | 110   |  
| RUTG  | 200   |  
| VZ    | 90    |  
+-----+-----+  
6 rows in set (0.00 sec)  
  
mysql>
```

Since we need to select all stocks that were exchanged in 2017 we will select price table for the operation.

- ii. Find all tickers (i.e. for all dates) whose closing price is both higher than 'IBM' on '9/20/2018' and no higher than 'GOOG' on '9/20/2018'.

Query :

```
SELECT DISTINCT P.TICKER FROM PRICE P WHERE P.CLOSE <  
(SELECT P.CLOSE FROM PRICE P WHERE P.TICKER = 'GOOG' AND  
P.DATE = '2018-09-20') AND P.CLOSE > (SELECT P.CLOSE FROM  
PRICE P WHERE P.TICKER = 'IBM' AND P.DATE = '2018-09-20');
```

Output:

```
mysql> SELECT DISTINCT P.TICKER FROM PRICE P WHERE P.CLOSE < (SELECT P.CLOSE FROM PRICE P WHERE P.TICKER = 'GOOG' AND P.DATE = '2018-09-20') AND P.CLOSE > (SELECT P.CLOSE FROM PRICE P WHERE P.TICKER = 'IBM' AND P.DATE = '2018-09-20');
+-----+
| TICKER |
+-----+
| AAPL   |
| GOOG   |
| BSE    |
| VZ     |
| amzn   |
| RUTG   |
+-----+
6 rows in set (0.02 sec)
```

Since we need to select all tickers for closing price below the closing price of GOOG to select this we need subquery that returns the closing prices less than GOOG and same with IBM we need closing price above the price of IBM on date 2018-09-20. This condition will return all the other tickers that satisfies the condition.

- iii. Find the tickers of all stocks that closed at the highest price on '9/20/2018'.

Query :

```
SELECT DISTINCT P.TICKER FROM PRICE P WHERE P.CLOSE =
(SELECT MAX(P.CLOSE ) FROM PRICE P WHERE P.DATE = '2018-09-
20');
```

Output:

```
mysql> SELECT DISTINCT P.TICKER FROM PRICE P WHERE P.CLOSE = (SELECT MAX(P.CLOSE ) FROM PRICE P WHERE P.
DATE = '2018-09-20');
+-----+
| TICKER |
+-----+
| GOOG   |
| INFY   |
+-----+
2 rows in set (0.00 sec)
```

By selecting max (closing price) will give the highest price and comparing it with the prices of the other tickers get all the tickers that have same highest prices.

- iv. Find the tickers of all stocks in 'NYSE' whose closing price on '9/20/2018' was either strictly below \$20 or strictly above \$100

Query :

```
SELECT DISTINCT P.TICKER FROM STOCK S , PRICE P WHERE
S.EXCHANGE = 'NYSE' AND P.DATE = '2018-09-20' AND S.TICKER
= P.TICKER AND (P.CLOSE < 20 OR P.CLOSE > 100);
```

Output:


```
mysql> SELECT DISTINCT P.TICKER FROM STOCK S , PRICE P WHERE S.EXCHANGE = 'NYSE' AND P.DATE = '2018-09-20' AND S.TICKER = P.TICKER AND (P.CLOSE < 20 OR P.CLOSE > 100);
+-----+
| TICKER |
+-----+
| INFY   |
+-----+
1 row in set (0.02 sec)
```

Explanation:

For getting tickers from NYSE we need to use STOCK table and price table for getting closing prices . s.ticker = p.ticker will give us distinct results. Rest conditions are straight forward with closing price less than 20 and greater than 100.

- v. Find all tickers in 'NYSE' of the stocks whose closing price showed the highest increase between '9/20/2018' and '9/21/2018' in 'NYSE' and whose closing price was (in 'NYSE') strictly above \$100 for the entire 2018

Query :

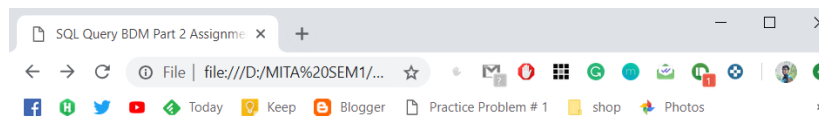
```
SELECT TICKER FROM (SELECT TICKER,MAX(DIFF) FROM (SELECT
TICKER,DIFF FROM (SELECT B.TICKER, ((B.MAX-B.MIN))
-> AS DIFF FROM (SELECT MAX(A.CLOSE) AS
MAX,MIN(A.CLOSE) AS MIN,A.TICKER FROM (SELECT
-> S.TICKER, S.EXCHANGE, P.CLOSE,P.DATE FROM STOCK
S,PRICE P WHERE P.TICKER=S.TICKER AND
-> S.EXCHANGE='NYSE' AND P.DATE BETWEEN '2018-09-20'
AND '2018-09-21') AS A GROUP BY A.TICKER) AS
-> B) AS C GROUP BY TICKER ORDER BY DIFF DESC) AS D) AS
MX WHERE TICKER IN (SELECT P.TICKER FROM
-> PRICE P WHERE P.DATE BETWEEN '2018-01-01' AND '201
8-12-31' AND P.CLOSE>100 GROUP BY TICKER);
```

Output:

```
mysql> SELECT TICKER FROM (SELECT TICKER,MAX(DIFF) FROM (SELECT TICKER,DIFF FROM (SELECT B.TICKER,((B.MA
X-B.MIN))
-> AS DIFF FROM (SELECT MAX(A.CLOSE) AS MAX,MIN(A.CLOSE) AS MIN,A.TICKER FROM (SELECT
-> S.TICKER, S.EXCHANGE, P.CLOSE,P.DATE FROM STOCK S,PRICE P WHERE P.TICKER=S.TICKER AND
-> S.EXCHANGE='NYSE' AND P.DATE BETWEEN '2018-09-20' AND '2018-09-21') AS A GROUP BY A.TICKER) AS
-> B) AS C GROUP BY TICKER ORDER BY DIFF DESC) AS D) AS MX WHERE TICKER IN (SELECT P.TICKER FROM
-> PRICE P WHERE P.DATE BETWEEN '2018-01-01' AND '2018-12-31' and p.close >100 group by ticker);
+-----+
| TICKER |
+-----+
| IBM    |
+-----+
1 row in set (0.00 sec)
```

2. (20 points) Make an HTML with a single textbox and a “submit button” where the user can enter a MYSQL query on the database you made in (i). Then, a CGI in Python (or in any other programming language you prefer) will submit this query to a MYSQL database and it will return an HTML file showing a simple table with the result of the query. The first row of the table should list the names of the attributes in the database and the remaining rows should list the tuples the query returned. Report the results of the HTML outputs on the example DB instance listed at the end of this document

Screenshot of the Html file



BDM Assignment

What's your Query?

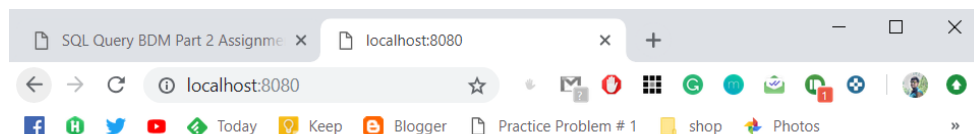
Select * from Price;

Submit!

Akhil Patil

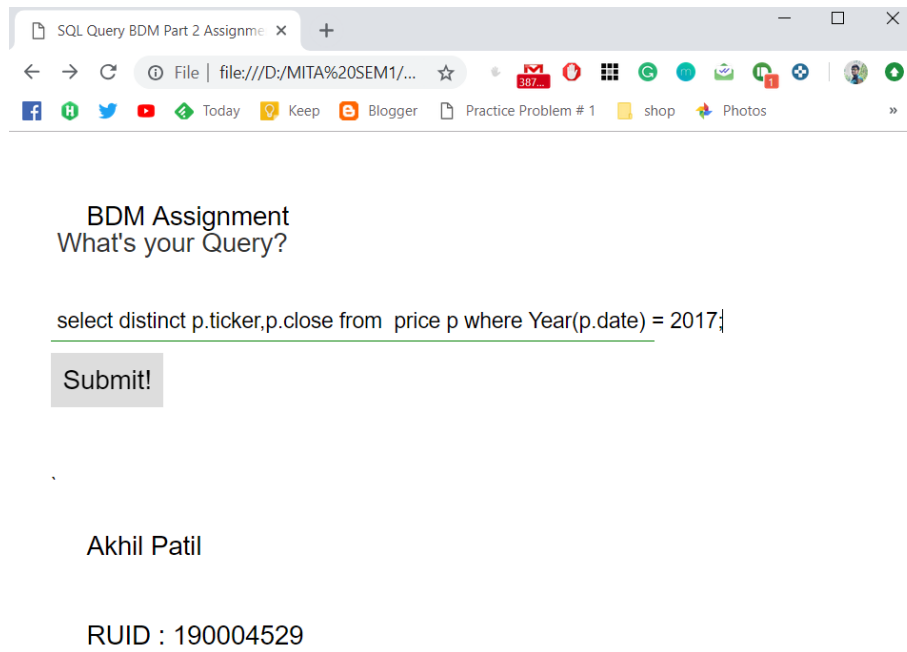
RUID : 190004529

Screenshot of Apache Localhost Connection working !



It works!

#Query1



The screenshot shows a web browser window with a single tab titled "SQL Query BDM Part 2 Assignment". The address bar shows a file path: "file:///D:/MITA%20SEM1/...". The browser's toolbar includes various icons for social media and applications. The main content area of the browser displays the following text:

BDM Assignment
What's your Query?

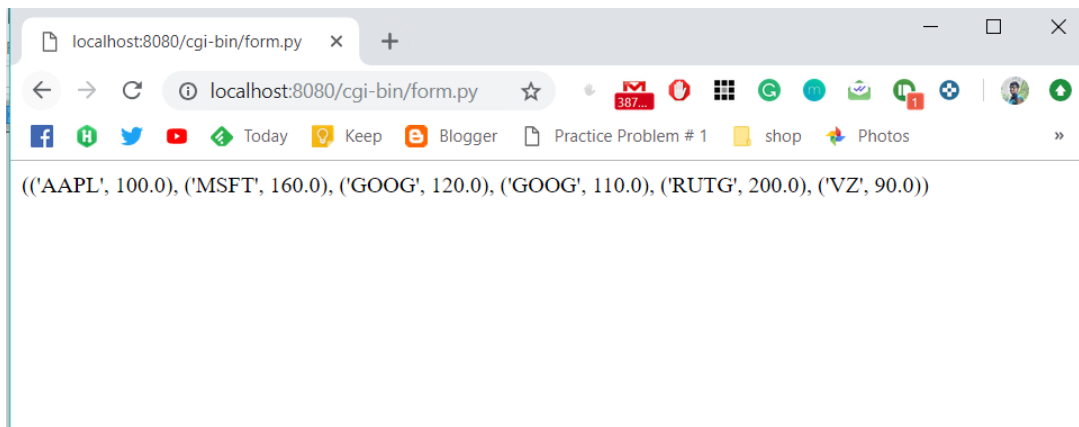
`select distinct p.ticker,p.close from price p where Year(p.date) = 2017;`

Submit!

Akhil Patil

RUID : 190004529

Figure SQL Query 1

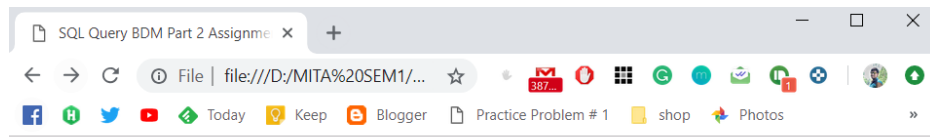


The screenshot shows a web browser window with a single tab titled "localhost:8080/cgi-bin/form.py". The address bar shows the URL "localhost:8080/cgi-bin/form.py". The browser's toolbar includes various icons for social media and applications. The main content area of the browser displays the following text:

((('AAPL', 100.0), ('MSFT', 160.0), ('GOOG', 120.0), ('GOOG', 110.0), ('RUTG', 200.0), ('VZ', 90.0)))

Figure Result of SQL query 1

#Query2



BDM Assignment
What's your Query?

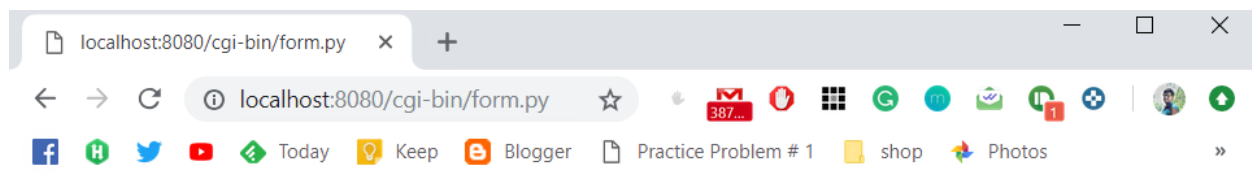
select p.ticker,p.close from price p where p.close < (select p.close from price p where p

Submit!

Akhil Patil

RUID : 190004529

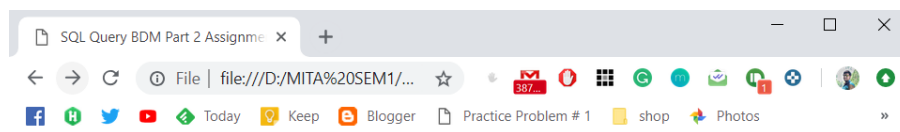
Figure SQL Query 2



((('AAPL'), ('GOOG'), ('BSE'), ('VZ'), ('amzn'), ('RUTG',))

Figure Result of SQL Query 2

#Query3



BDM Assignment
What's your Query?

SELECT DISTINCT P.TICKER FROM PRICE P WHERE P.CLOSE = (SELECT MAX(P.

Submit!

Figure SQL Query 3

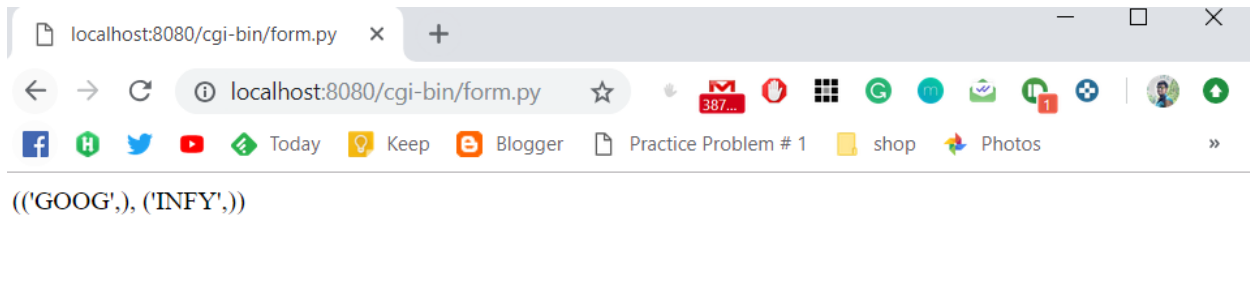


Figure Result of SQL Query 3

#Query4

A screenshot of a web browser window titled 'SQL Query BDM Part 2 Assignment'. The page asks 'What's your Query?' and shows a SQL query: `SELECT DISTINCT P.TICKER FROM PRICE P WHERE P.CLOSE = (SELECT MAX(P.`. Below the query is a 'Submit!' button.

Figure SQL Query 4

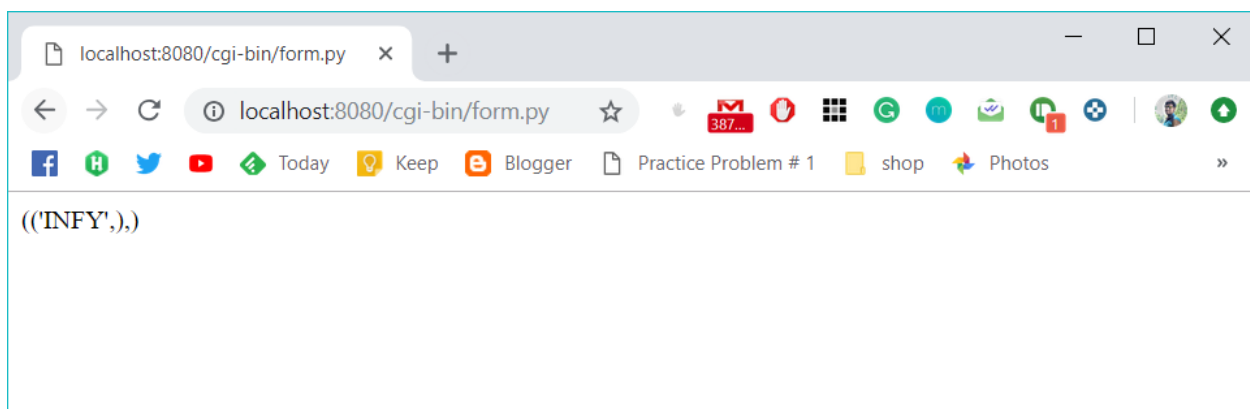
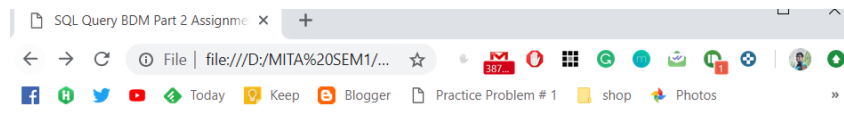


Figure Result of SQL Query 4

#Query5



BDM Assignment
What's your Query?

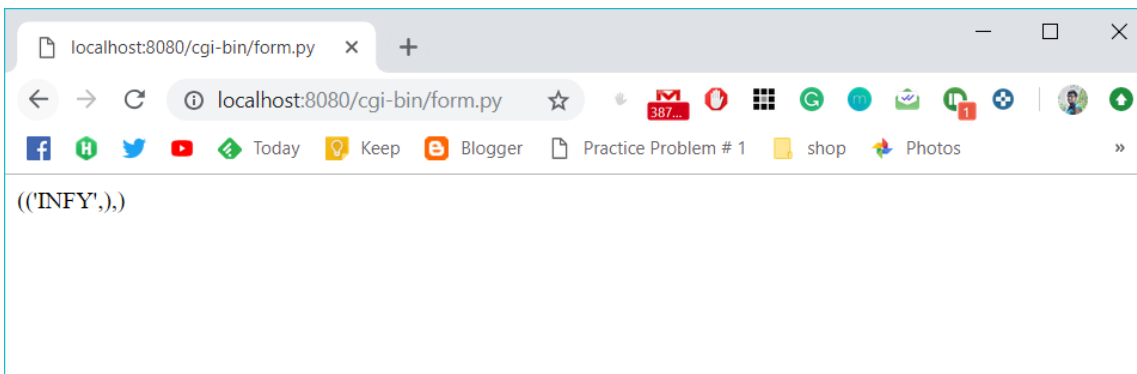
SELECT TICKER FROM (SELECT TICKER,MAX(DIFF) FROM (SELECT TICKER,DIF

Submit!

Akhil Patil

RUID : 190004529

Figure SQL Query 5



Result of SQL Query 5

Figure

Part 3

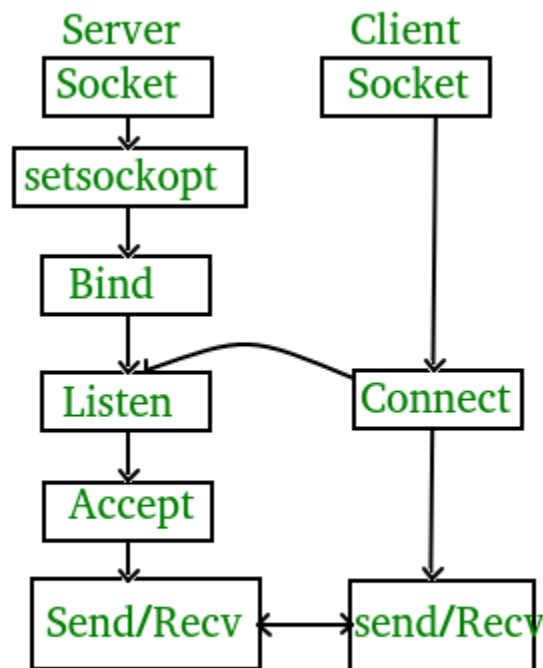
- a. Research the literature and write an essay explaining what the Sockets are, what is Socket Programming, and how these notions are related to the work that a “server” must do. Also, briefly explain what people mean when they use the term “server”. Give some examples if you think that they will help explaining what is going on. This should be an essay of at most 200 words.

➔ What are the Sockets?

Sockets are the endpoints of the two-way communications i.e. the interface between an application process and transport layer. Application process can send as well as receive data packets through the sockets. Sockets are Gateway for data packets.

➔ What is Socket Programming?

Socket Programming is the programming used to connect the client and server through the Sockets on client side and one on server side. One socket listen on a particular port at an IP address while the other reaches out to the server. There are certain stages to connect client and server via sockets. Below image depicts the stages of the connection.



Source: GeeksforGeeks: State Diagram for Client Server Model

Short Description on stages:

Server part:

1. Open the server Socket –Socket () create the socket
2. Bind() – Connects the port with socket to setup Connection.
3. Wait for client Request – Listen () - waits for the client to approach the server to make a connection.
4. Create streams for communicating to the client. -Accept ()
5. Perform communication with client. (Send/Receive)
6. Close Sockets. Close ()

Client Part:

1. Create a Socket. - Socket () create the socket
2. Create streams for communicating with the server.
3. Perform or communication with the server:
4. Close the socket when done.

➔ What is server?

Server is the messenger we can just request the data and it will provide the data. Think of this as person at reception who has knowledge about the office, we ask (REQUEST) receptionist for the information and we the required information (RESPONSE) . Similarly, Server has all the webpages data we request information by putting the web address on the browser which in turn connects to the server and we can retrieve the data from server.

References:

TCP/IP Illustrated, volumes 1-3 by W. Richard Stevens and Gary R. Wright

Beej's Guide to Network Programming: www.ecst.csuchico.edu/~beej/guide/net/

- b. You should also write a second essay of at most 300 words explaining the following (in addition to the essay you should provide a detailed diagram): which network services are related to your “Part 2” (above)? That is, explain how many and what kind of servers (and what do they do) are involved in the system, how your program gets involved in the process, and what kinds of messages the system is exchanging, and what do we mean by “message”.**

- ➔ Network Service** – a capability that facilitates a network operation. It is generally provided by a client server connection. Where in the server responds the requests of clients. Network Services used in part 2 are Apache web server and MySQL server. The components used in Part 2 are a browser – to act as interface between user and server,

Apache and MSQl Server, Database to store the records, Python CGI to communicate with MySQL Server.

- ➔ Message is the MySQL query that is passed from the user from Chrome browser to Apache Server to get the records from the database. System is exchanging messages in the form of queries to retrieve records form the database.
- ➔ The Communication process is at the backend of apache server. It starts with client when makes a request from Chrome for the file form.html which in turn fetches file from apache server running on the machine. Apache responds to the request of chrome for the webpage by returning the form.html file.

Chrome draws the webpage with the received form.html with all the elements. User fills in the textbox with Message as "Query" and once the user submits the message. Chrome sends these messages to apache server. Now, Apache server triggers the python CGI for executing the python script. Python script which our case is form.py which contains the code for connecting to the database and retrieving the Query. Python interpreter executes the code which requests the query to MySQL server where the Database Instance is stored.

MySQL runs the query on the database and sends the response for the request (RES) to Python CGI with proper format. Now, Python CGI takes RES and prints it to Html page which is dynamically created. Finally, the newly created Html file is sent to the Apache server. Apache sends the Html to chrome. Chrome Draws the Result on the browser window.

Diagram Flow chart for data flow in Web Application:

