

Part 1

Below are the Relational Algebra queries:

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

$$\pi_{\text{ticker}, \text{close}} \left[\sigma_{\substack{\text{date} > 12/31/2016 \\ \text{AND} \\ \text{date} < 01/01/2018}} (\text{Price}) \right]$$

2. 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'.

$$\pi_{\text{ticker}} \left[\sigma_{\substack{\text{date} = 9/20/2018 \\ \text{AND} \\ \text{ticker2} = \text{'IBM'} \\ \text{AND} \\ \text{close1} > \text{close}}} \left[(\text{Price}) \times \rho_{\substack{\text{close} \rightarrow \text{close1} \\ \text{AND} \\ \text{ticker} \rightarrow \text{ticker2}}} (\text{Price}) \right] \right]$$

$$\cap \pi_{\text{ticker}} \left[\sigma_{\substack{\text{date} = 9/20/2018 \\ \text{AND} \\ \text{ticker4} = \text{'GOOG'} \\ \text{AND} \\ \text{close3} < \text{close}}} \left[(\text{Price}) \times \rho_{\substack{\text{close} \rightarrow \text{close3} \\ \text{AND} \\ \text{ticker} \rightarrow \text{ticker4}}} (\text{Price}) \right] \right]$$

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

$$\pi_{\text{ticker}} \left[\sigma_{\substack{\text{date} = 09/20/2018}} (\text{Price}) \right]$$

$$- \pi_{\text{ticker}} \left[\sigma_{\substack{\text{date} = 9/20/2018 \\ \text{AND} \\ \text{close} < \text{close1}}} \left[(\text{Price}) \times \rho_{\substack{\text{ticker} \rightarrow \text{ticker1} \\ \text{AND} \\ \text{close} \rightarrow \text{close1}}} (\text{Price}) \right] \right]$$

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

$$\pi_{\text{ticker}} \left[\begin{array}{l} \text{ticker} = \text{ticker1} \\ \text{AND} \\ \text{exchange} = \text{'NYSE'} \\ \text{AND} \\ \text{date} = \text{'9/20/2018'} \\ \text{AND} \\ \text{close} < \$20 \text{ OR } \text{close} > \$100 \end{array} \left[(\text{Stock}) \times \rho (\text{Price}) \right] \right]_{\text{ticker} \rightarrow \text{ticker1}}$$

5. 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.

$$\pi_{\text{ticker}} \left[\begin{array}{l} \text{date} > 12/31/2017 \\ \text{AND} \\ \text{date} < 01/01/2019 \\ \text{AND} \\ \text{exchange} = \text{'NYSE'} \\ \text{AND} \\ \text{close} > 100 \end{array} \left[(\text{Stock}) \times \rho (\text{Price}) \right] \right]_{\text{ticker} \rightarrow \text{ticker1}}$$

N

$$\pi_{\text{ticker}} \left[\begin{array}{l} \pi_{\text{ticker, close}} (\text{Price}) - \pi_{\text{ticker, close}} \left[\begin{array}{l} \text{close} < \text{close2} - \text{close1} \\ \text{AND} \\ \text{exchange} = \text{'NYSE'} \\ \text{AND} \\ \text{date1} = \text{'9/20/2018'} \\ \text{AND} \\ \text{date2} = \text{'9/21/2018'} \end{array} \left[\begin{array}{l} \rho (\text{Price}) \times \rho (\text{Price}) \\ \text{close} \rightarrow \text{close1} \quad \text{close} \rightarrow \text{close2} \\ \text{AND} \\ \text{date} \rightarrow \text{date1} \quad \text{date} \rightarrow \text{date2} \end{array} \right] \end{array} \right] \right]_{\text{ticker} \rightarrow \text{ticker1}}$$

Part 2

2.1.1

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 a query language composed of a number of operators, each of which takes in relations as arguments and returns a single relation as result. However, there are a few operators which when applied (in primitive form) to relational algebra, goes beyond its scope. Relational Algebra does NOT support MAX, MIN, AVG operations.

To use these primitives MAX or MIN operations, we use SQL instead of relational algebra. This helps us make our queries seem less complex than relational algebra equations. The application of this can be seen in our second query where we use a nested subquery with MAX() function:

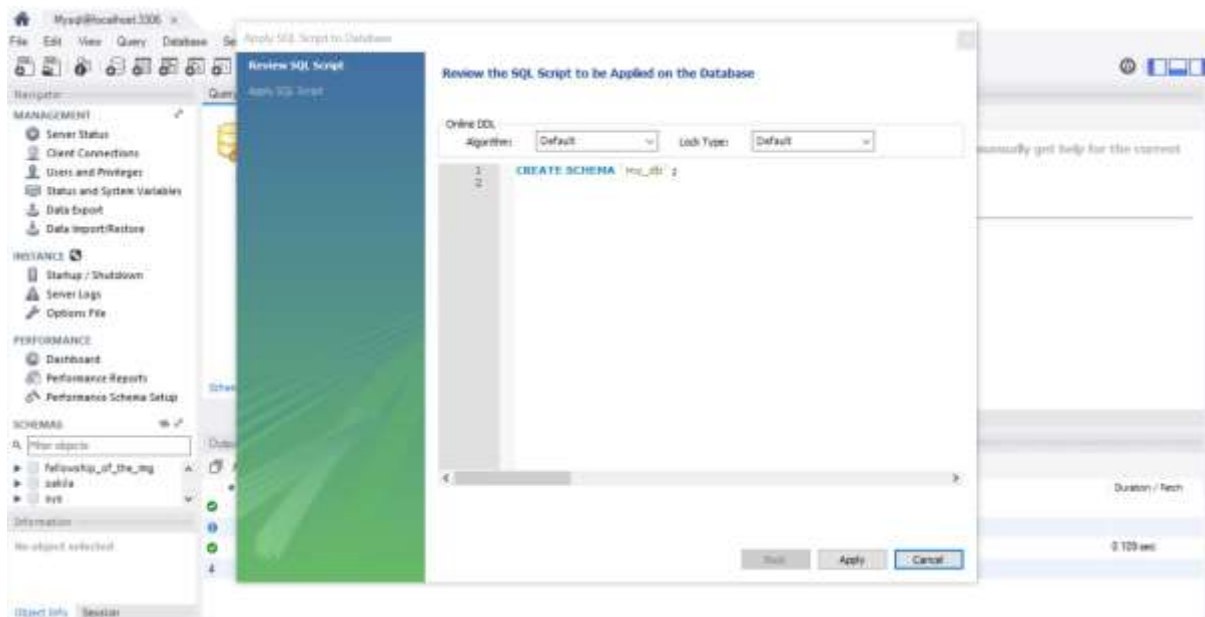
```
(SELECT DISTINCT MAX(p2.close) FROM my_db.price AS p2 WHERE p2.ticker='GOOG' && p2.date=20180920)
```

This helps us to get the desired output with ease.

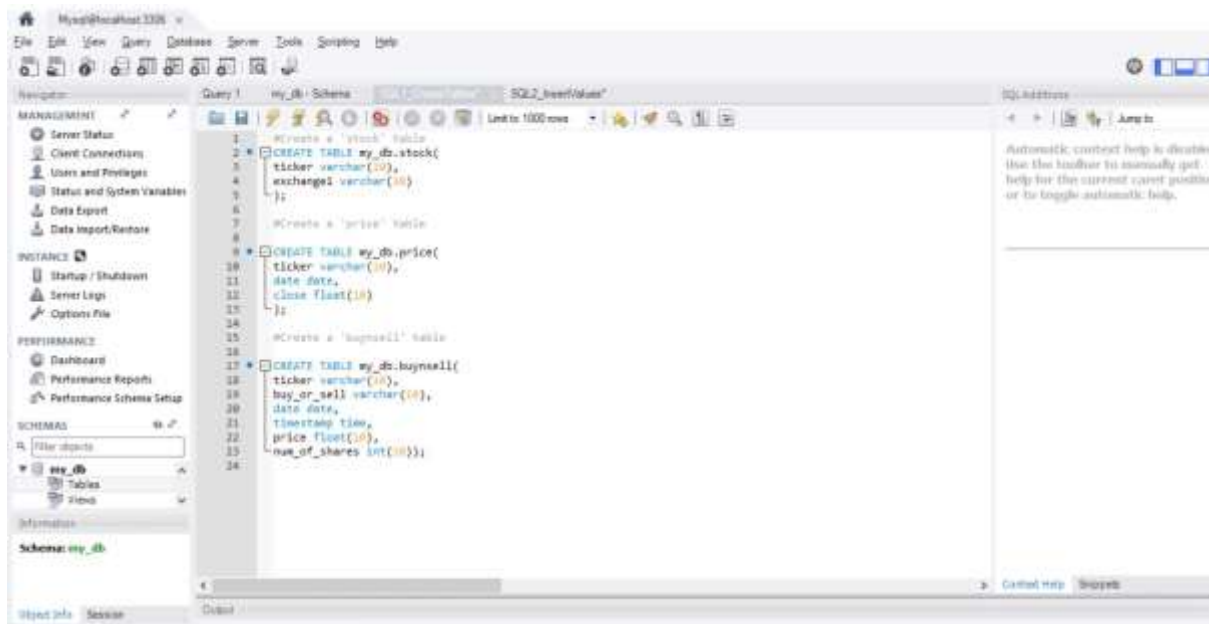
2.1.2

Following are the screenshots of MySQL queries along with their outputs:

#Create a database schema named **my_db**.

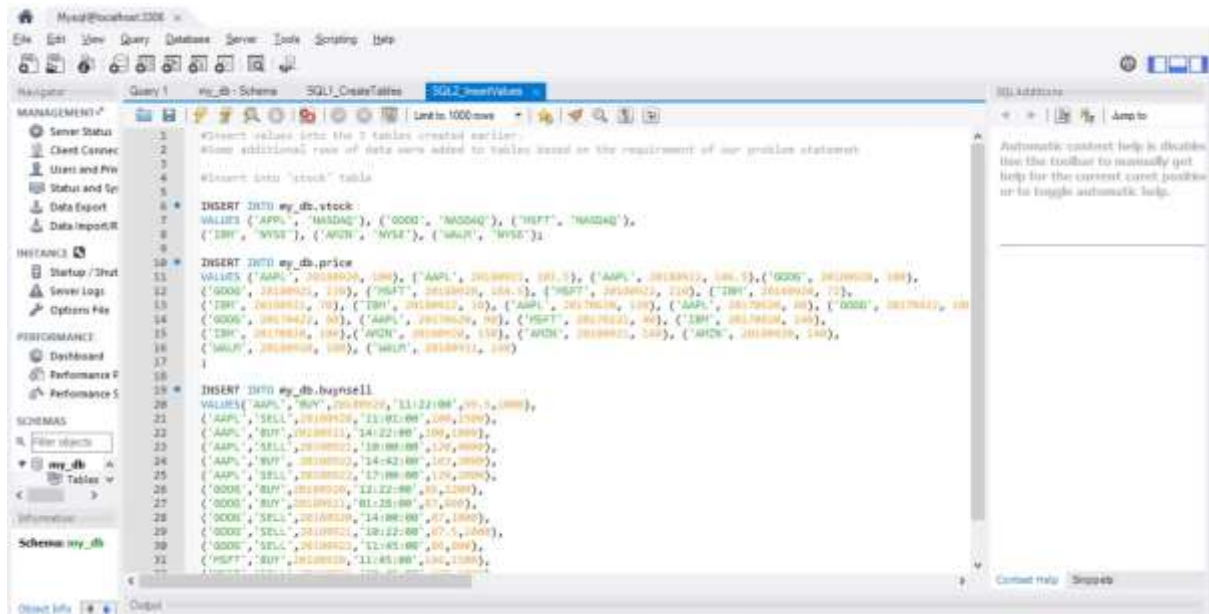


#Create tables **stock**, **price** and **buynsell** inside our schema



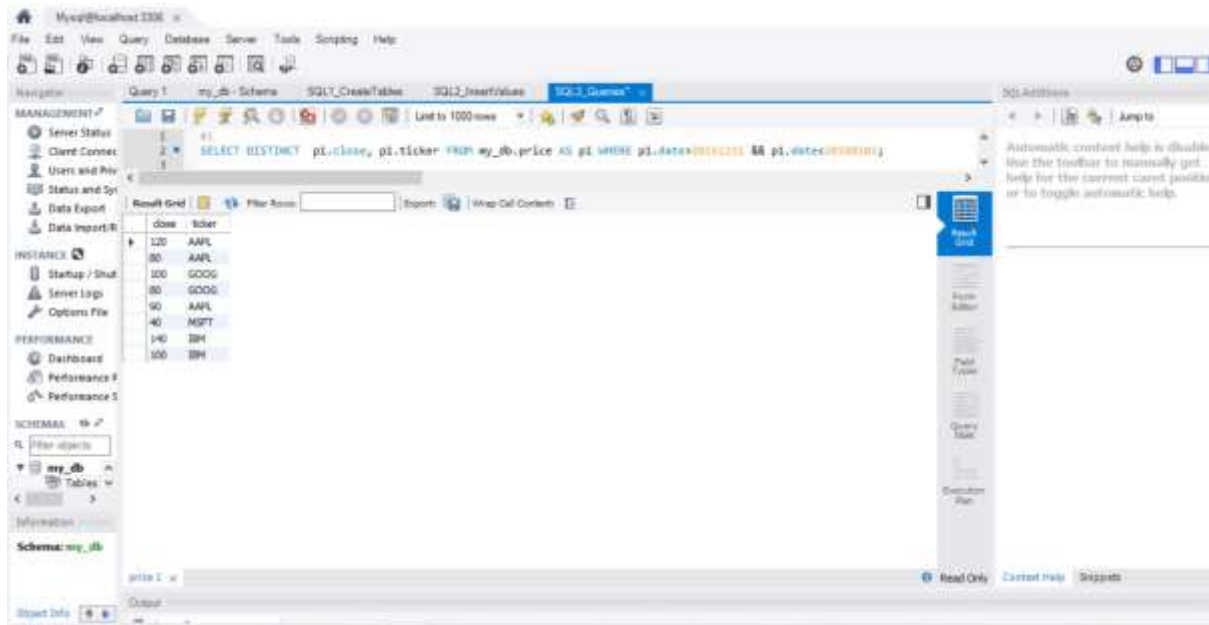
#Insert data into the tables created above

#Apart from the given rows of data, some additional rows of data were added to tables based on the requirement of our problem statement. E.g., In query #5 where we have to find all stocks in 'NYSE' whose closing price showed the highest increase between '9/20/2018' and '9/21/2018', We had to add new tickers to **stock** table and values of tickers in **price** table so that we can find the range with MAX increase.



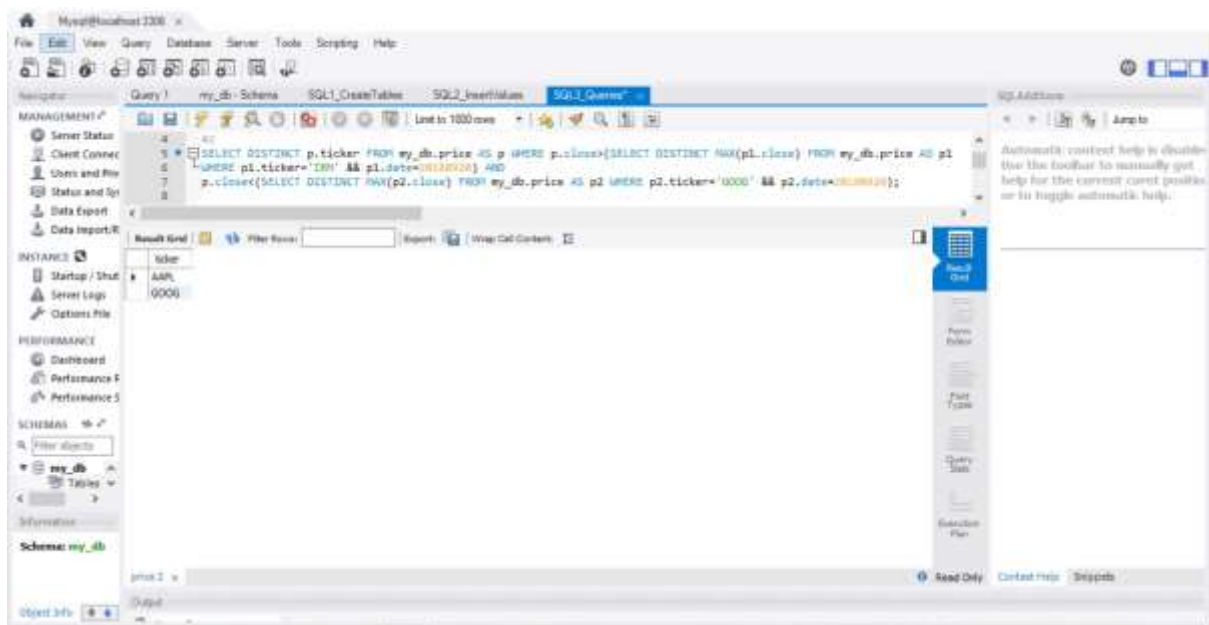
#SQL Query 1

#Find the tickers and all closing prices of all stocks exchanged in 2017



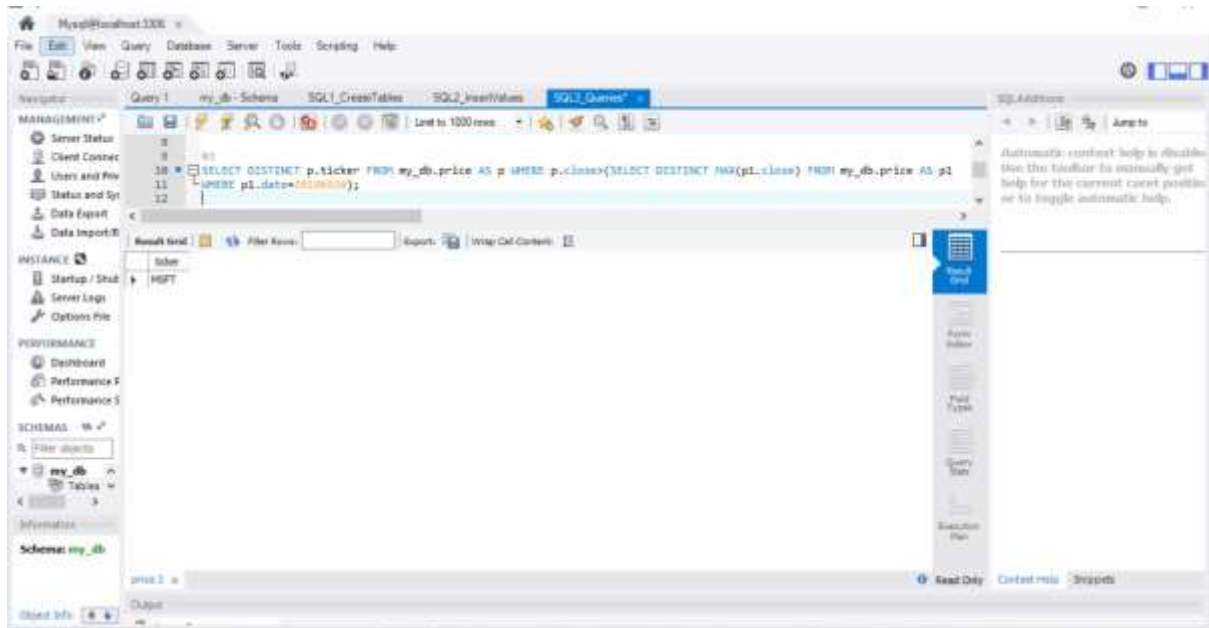
#SQL Query 2

#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'.



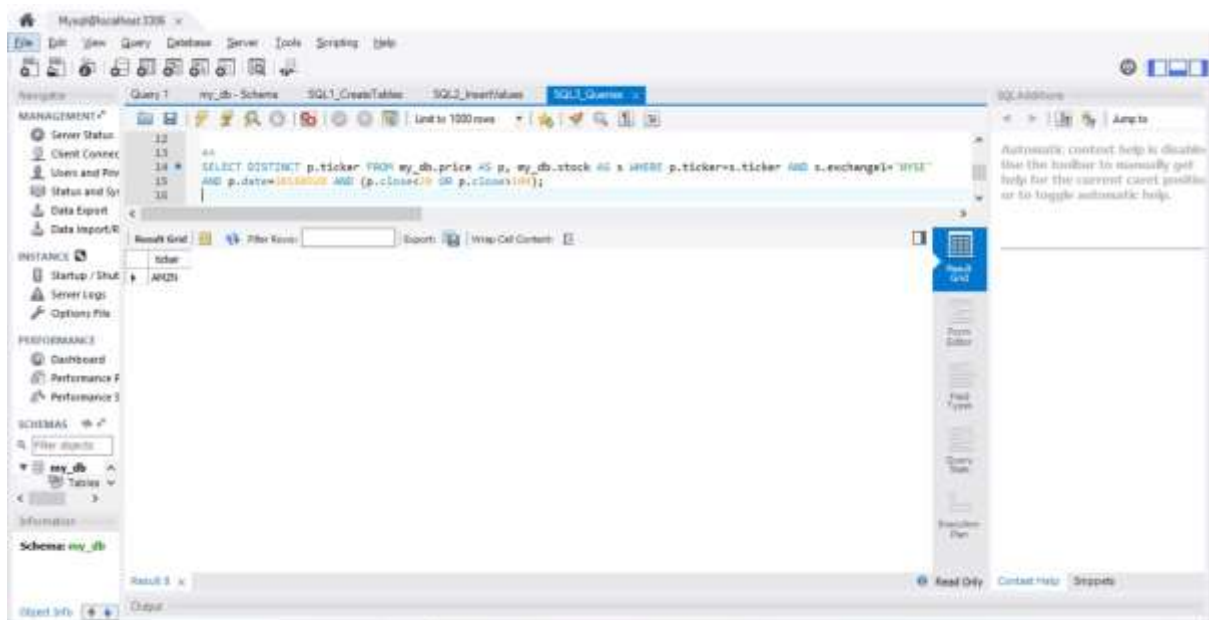
#SQL Query 3

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



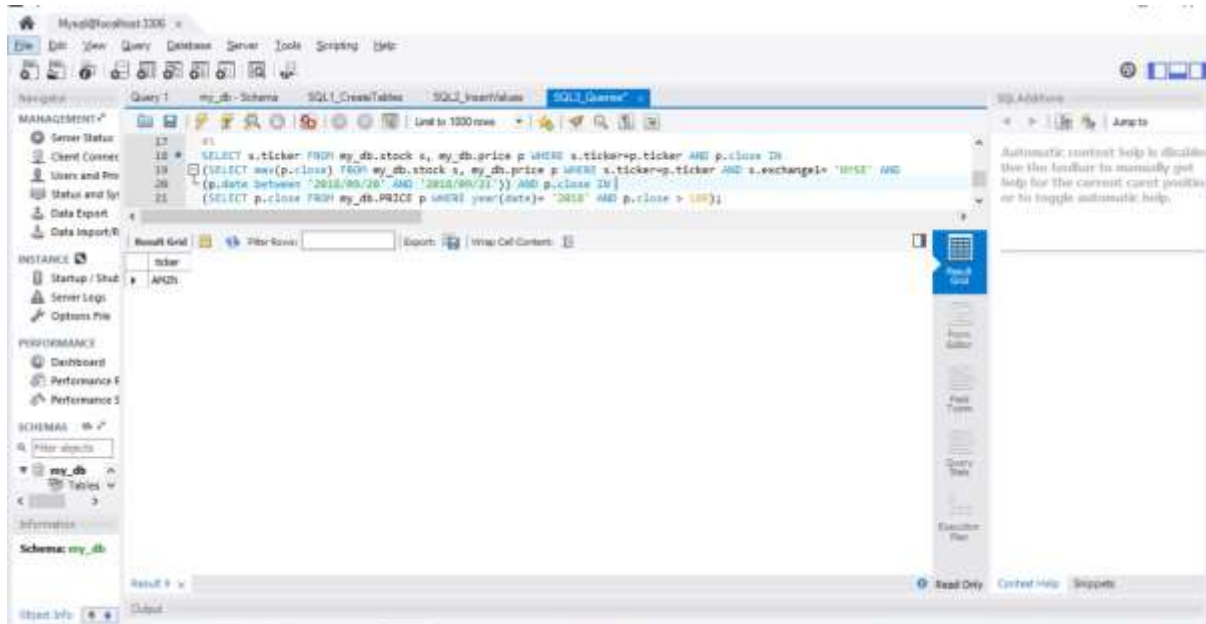
#SQL Query 3

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



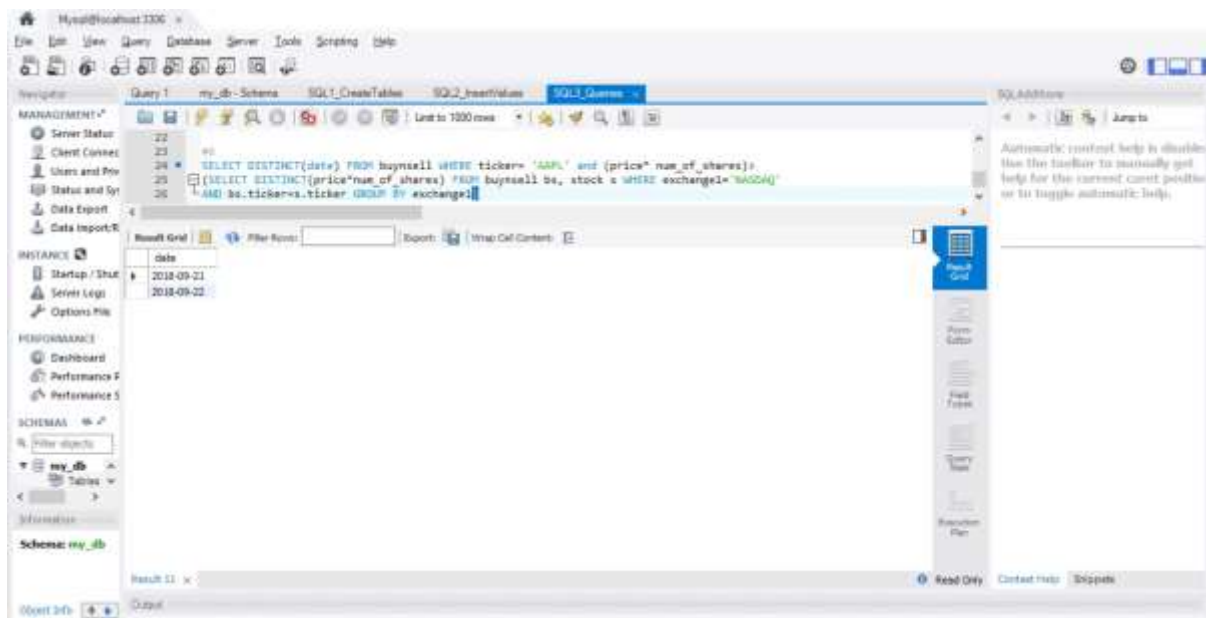
#SQL Query 5

#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.




#SQL Query 5

#Find the dates where the total price (i.e. price times num_of_shares) of 'AAPL' the firm (i.e. the trading firm which is using this database) sold was higher than what the firm bought in 'NASDAQ'.



2.2

Following python CGI was used in our program




```

1  #!/usr/bin/env python
2  print("Content-Type: text/html")
3  print()
4
5
6  import cgi, cgitb
7  cgitb.enable() #for debugging
8  form = cgi.FieldStorage()
9  name = form.getvalue('name')
10 print("Name of the user is:",name)
11
12
13 import psycopg2.cursor
14 # Open connection to the database
15 db = psycopg2.connect("host=localhost", "port=5432", "user=root")
16 print ("connect successful!!")
17 # Start a cursor object using cursor() method
18 cursor = db.cursor()
19
20 # Execute a SQL query using execute() method. This should return all the columns of the rows that use words.
21 cursor.execute(name)
22
23 # Fetch all the rows using fetchall() method.
24 data = cursor.fetchall()
25 print (data)
26
27 # Disconnect from server
28 db.close()

```

We used the following html command to read and print our message



```

1 <html>
2 <head>
3 <title>Demo HTML file</Title>
4 </head>
5 <body>
6 <form name="pyform" method="POST" action="http://localhost/cgi-bin/form.py">
7   <input type="text" name="fname" />
8   <input type="submit" name="submit" value="Submit" />
9 </form>
10 </body>
11 </html>

```


Following html page was generated

Part 3

3.1

What is a socket?

A socket may be considered a communication point within the same or different computers, to exchange data and to establish a line of communication. The sockets allow communication between the two different processes within the same or different machines. It is identified by a port number.

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

What is Socket Programing?

Socket programming is a way of connecting two nodes on a network to communicate with each other.

Following are the steps involved:

- The client machine knows the server machine's hostname (address) and the port number on which the server is listening. The client uses this to make a connection request.
- To identify itself to the server, the client binds to a local, system assigned port number.
- The server accepts the connection. The server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client.
- On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

This is called a Client-Server Architecture.

What are Servers?

A computer operation that provides application services is called servers. It can institute a linkage between applications through the use of socket programming. In other words, these are just computers or a program that provides functionality for other programs or devices (clients).

3.2

In this exercise, we create a webpage to query a MySQL database and display the result. Python CGI script is used to get results and generate webpage. Apache client is used to communicate between web client (HTML on chrome) and Database.

The whole query and response cycle begins when an input is submitted through html page.

Following are the steps involved:

- 1) Chrome asks Apache for a 'form.html'
- 2) Apache then reads the form.html file from disk.
- 3) Apache sends back the content of form.html to chrome
- 4) i. Chrome draws on the screen the webpage (form.html).
ii. The user then supply the SQL query in the textbox and then presses "Submit".
- 5) Chrome send *message* from the user to Apache.(Here, the *message* is the query that user has filled in the textbox. Now Apache knows that user has written query in chrome)
- 6) Apache now runs Python CGI(on server side)
- 7) Python CGI connects to the MySQL and starts to run the message.
- 8) MySQL goes to DB, evaluates the message and computes a response.
- 9) MySQL then sends the result to **our** python program.
- 10) Python CGI (our python program) receives results and compiles (on the fly) a webpage in a format understandable by HTML.
- 11) Apache sends the dynamically created message to Chrome.
- 12) Chrome draws on the screen, the output of file

The following diagram illustrates the same process:

