Nicholas Frischkorn
BDM Assignment 2
April 30, 2020

1) The report should contain all SQL queries. You can add a brief explanation if you want.

Query 1 # find the tickers and all closing prices of all stocks exchanged in 2020

```
SELECT DISTINCT bs.ticker, p.close FROM buyNsell bs, price p WHERE bs.ticker = p.ticker
&& bs.date BETWEEN '2019-12-31' AND '2021-01-01';
```

Query 2 # Find all tickers (i.e. for all dates) whose closing price is both higher
# than ‘IBM’ on ‘3/25/2020’ and no higher than ‘GOOG’ on ‘3/25/2020’.

```
SELECT DISTINCT pb.ticker FROM  price p, price pa, price pb WHERE ((p.ticker = 'GOOG'
&& p.date = '2020-3-25') && (pa.ticker = 'IBM' && pa.date = '2020-3-25') && (pb.ticker != 'IBM'
&& pb.ticker != 'GOOG' && pb.date = '2020-3-25') && (pb.close <= p.close && pb.close >
pa.close));
```

Query 3 # Find the tickers of all stocks that closed at the highest price on ‘3/25/2020’. (we are asking for “all stocks” since there may be more than one with the same “highest price”)

```
SELECT DISTINCT p.ticker FROM price p WHERE p.ticker NOT IN (SELECT DISTINCT
pa.ticker FROM price p, price pa WHERE p.date = '2020-3-25' && pa.date = '2020-3-25' &&
pa.close < p.close);
```

Query 4 # Find the tickers of all stocks in ‘NYSE’ whose closing price on ‘3/25/2020’ was either strictly below $20 or strictly above $100

```
SELECT DISTINCT p.ticker FROM price p WHERE p.date = '2020-3-25' && (p.close<20 OR
p.close>100) && p.ticker IN (SELECT DISTINCT s.ticker FROM stock s WHERE (s.exchange =
'NYSE'));
```

Query 5 # Find all tickers in ‘NYSE’ of the stocks whose closing price showed the highest increase between ‘3/25/2020’ and ‘3/26/2020’ in ‘NYSE’ and whose closing price was (in ‘NYSE’) strictly above $100 for the entire 2020

```
SELECT DISTINCT s.ticker FROM stock s WHERE s.ticker NOT IN (SELECT DISTINCT
r.ticker FROM price p, price q, price r, price s WHERE p.date = '2020-3-25' && q.date = '2020-3-
26' && p.ticker = q.ticker && r.date = '2020-3-25' && s.date = '2020-3-26' &&  r.ticker = s.ticker
&&  p.ticker != r.ticker && q.close-p.close>s.close-r.close) && s.ticker NOT IN (SELECT
DISTINCT p.ticker FROM price p WHERE p.date BETWEEN '2019-12-31' AND '2021-01-01' &&
p.close<=100) && s.ticker IN (SELECT DISTINCT s.ticker FROM stock s WHERE s.exchange =
'NYSE');
```

Query 6 # In addition, to the above queries, also do in SQL the following one:
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’

SELECT DISTINCT buy.date FROM (SELECT b.date, SUM(b.price * b.num_of_shares) AS summed FROM buyNsell b, stock s WHERE s.ticker = b.ticker AND s.exchange = 'NASDAQ' AND b.buy_or_sell = 'BUY' GROUP BY b.date) AS buy, (SELECT b.date, SUM(b.price * b.num_of_shares) AS summed FROM buyNsell b WHERE b.buy_or_sell = 'SELL' AND b.ticker = 'AAPL' GROUP BY b.date) AS sell WHERE buy.date = sell.date AND sell.summed > buy.summed;

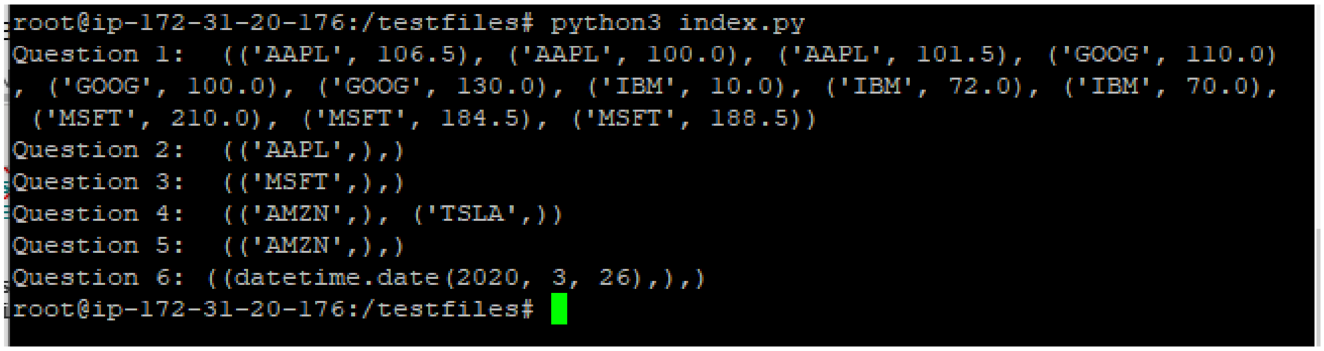
2) In the same report, you should list all the NEW entries you made to the database provided in the end of the Lab Exercise.

Additional entries made into stock table:
('TSLA','NYSE'),
('AMZN', 'NYSE');

Additional entries made into price table:
('TSLA', '2020-03-25', 138.50),
('TSLA', '2020-03-26', 138.00),
('TSLA', '2020-03-22', 137.50),
('AMZN', '2020-03-25', 103.00),
('AMZN', '2020-03-26', 195.00),
('AMZN', '2020-03-22', 200.00);

Additional entries made into buyNsell table:
('AAPL', 'SELL', '2020-03-26', '14:29:00', 120.00, 9000),
('MSFT', 'BUY', '2020-03-25', '11:48:00', 187.50, 3000),
('MSFT', 'SELL', '2020-03-26', '12:15:00', 189.00, 200);

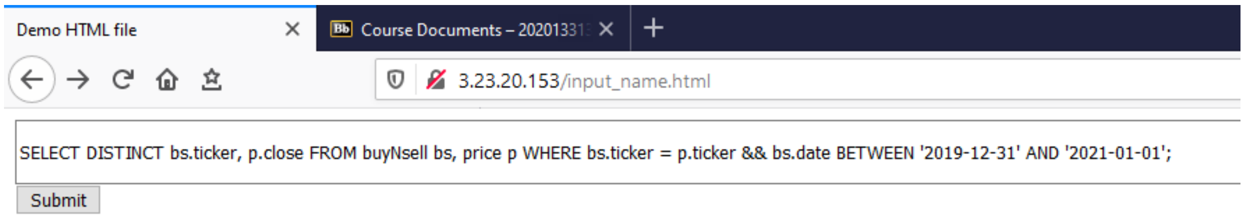
3) In the same report, you should show the result of each of your queries on the Database instance you used (i.e. the given one + the entries you added).



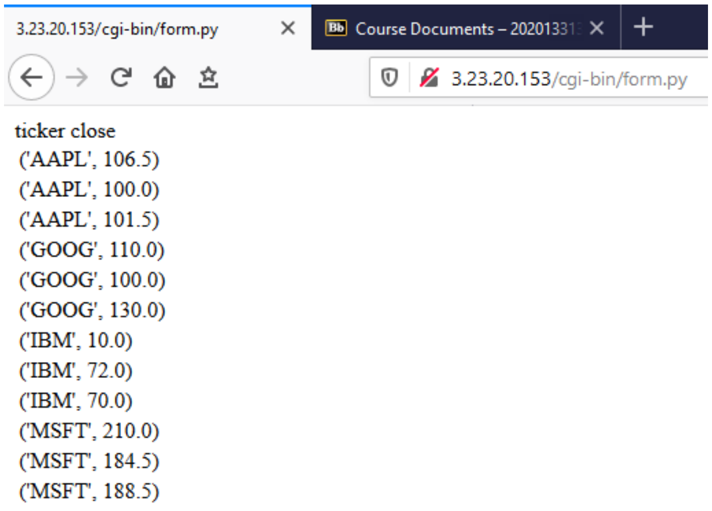
```
root@ip-172-31-20-176:/testfiles# python3 index.py
Question 1:  (('AAPL', 106.5), ('AAPL', 100.0), ('AAPL', 101.5), ('GOOG', 110.0)
, ('GOOG', 100.0), ('GOOG', 130.0), ('IBM', 10.0), ('IBM', 72.0), ('IBM', 70.0),
 ('MSFT', 210.0), ('MSFT', 184.5), ('MSFT', 188.5))
Question 2:  (('AAPL',),)
Question 3:  (('MSFT',),)
Question 4:  (('AMZN',), ('TSLA',))
Question 5:  (('AMZN',),)
Question 6: ((datetime.date(2020, 3, 26),),)
root@ip-172-31-20-176:/testfiles#
```

4) In the same report, you should provide screenshots for your web application and the tests you performed.

Question 1: Query and result

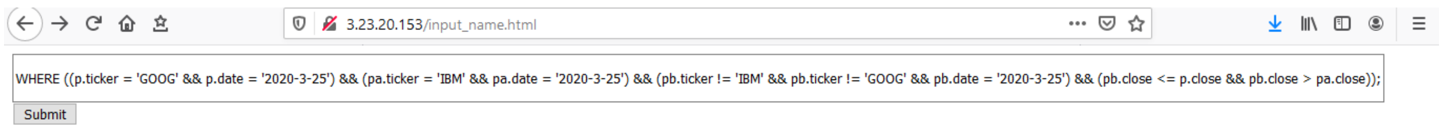


```
SELECT DISTINCT bs.ticker, p.close FROM buyNsell bs, price p WHERE bs.ticker = p.ticker && bs.date BETWEEN '2019-12-31' AND '2021-01-01';
```

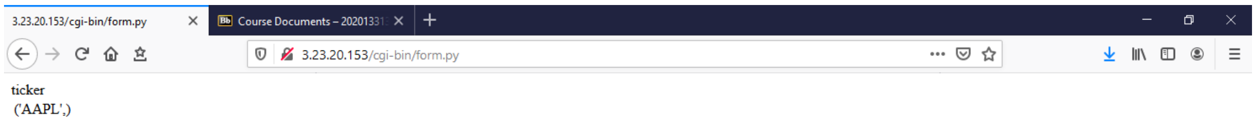

3.23.20.153/cgi-bin/form.py
Course Documents – 20201331
3.23.20.153/cgi-bin/form.py

```
ticker close
('AAPL', 106.5)
('AAPL', 100.0)
('AAPL', 101.5)
('GOOG', 110.0)
('GOOG', 100.0)
('GOOG', 130.0)
('IBM', 10.0)
('IBM', 72.0)
('IBM', 70.0)
('MSFT', 210.0)
('MSFT', 184.5)
('MSFT', 188.5)
```

Question 2: Query and result (The whole query cannot be seen in the textbox as it is to long)

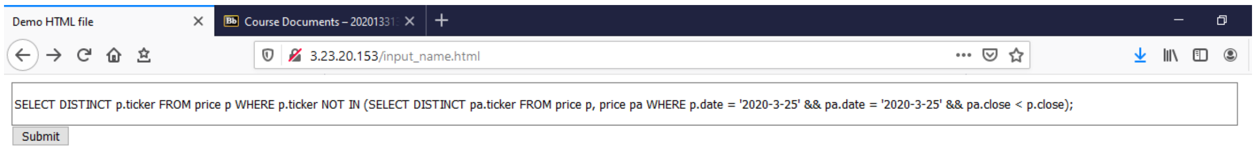


```
WHERE ((p.ticker = 'GOOG' && p.date = '2020-3-25') && (pa.ticker = 'IBM' && pa.date = '2020-3-25') && (pb.ticker != 'IBM' && pb.ticker != 'GOOG' && pb.date = '2020-3-25') && (pb.close <= p.close && pb.close > pa.close));
```

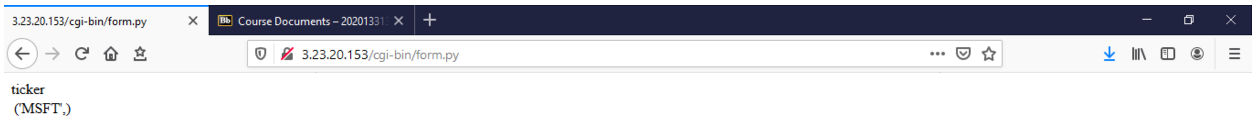

3.23.20.153/cgi-bin/form.py
Course Documents – 20201331
3.23.20.153/cgi-bin/form.py

```
ticker
('AAPL',)
```

Question 3: Query and result

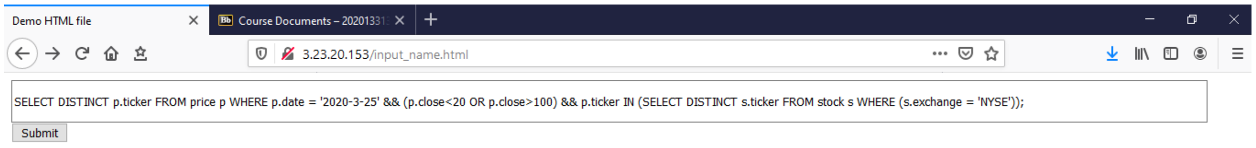


```
SELECT DISTINCT p.ticker FROM price p WHERE p.ticker NOT IN (SELECT DISTINCT pa.ticker FROM price p, price pa WHERE p.date = '2020-3-25' && pa.date = '2020-3-25' && pa.close < p.close);
```

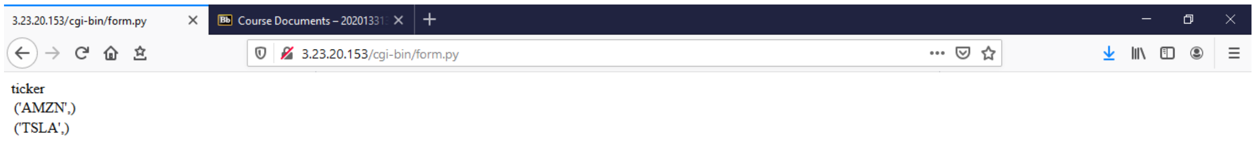

3.23.20.153/cgi-bin/form.py
Course Documents – 20201331
3.23.20.153/cgi-bin/form.py

```
ticker
('MSFT',)
```

Question 4: Query and result

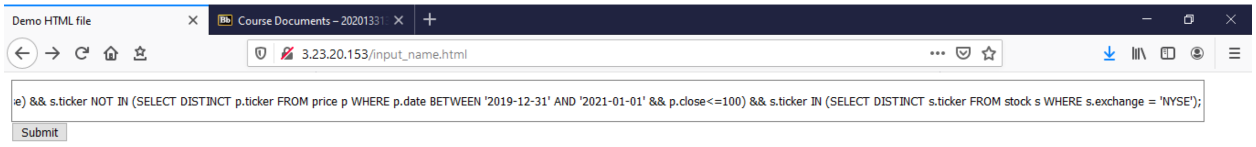


```
SELECT DISTINCT p.ticker FROM price p WHERE p.date = '2020-3-25' && (p.close<20 OR p.close>100) && p.ticker IN (SELECT DISTINCT s.ticker FROM stock s WHERE (s.exchange = 'NYSE'));
```

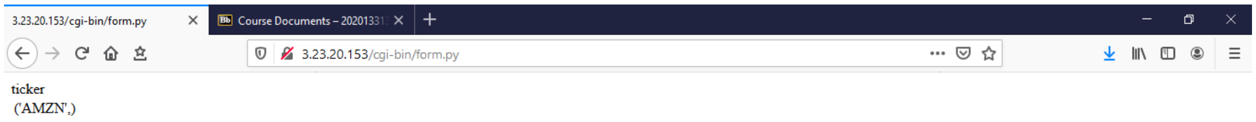

3.23.20.153/cgi-bin/form.py
Course Documents – 20201331
3.23.20.153/cgi-bin/form.py

```
ticker
('AMZN',)
('TSLA',)
```

Question 5: Query and result (The whole query cannot be seen in the textbox as it is to long)

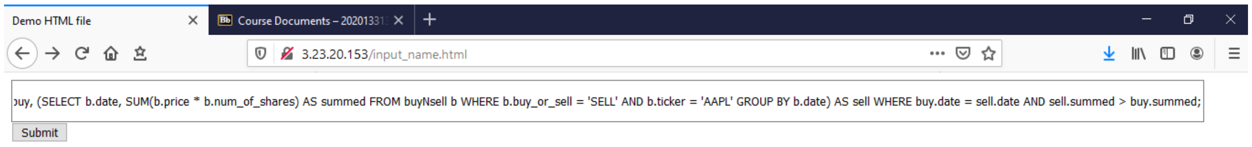


```
ːe) && s.ticker NOT IN (SELECT DISTINCT p.ticker FROM price p WHERE p.date BETWEEN '2019-12-31' AND '2021-01-01' && p.close<=100) && s.ticker IN (SELECT DISTINCT s.ticker FROM stock s WHERE s.exchange = 'NYSE');
```

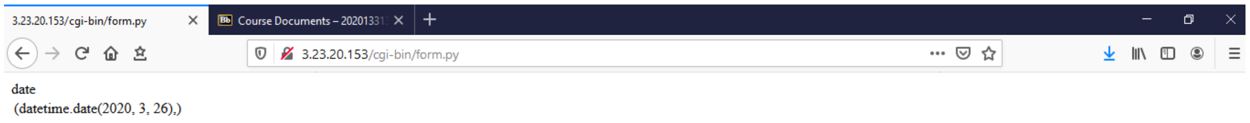

3.23.20.153/cgi-bin/form.py
Course Documents – 20201331
3.23.20.153/cgi-bin/form.py

```
ticker
('AMZN',)
```

Question 6: Query and result (The whole query cannot be seen in the textbox as it is to long)



```
ɔuy, (SELECT b.date, SUM(b.price * b.num_of_shares) AS summed FROM buyNsell b WHERE b.buy_or_sell = 'SELL' AND b.ticker = 'AAPL' GROUP BY b.date) AS sell WHERE buy.date = sell.date AND sell.summed > buy.summed;
```


3.23.20.153/cgi-bin/form.py
Course Documents – 20201331
3.23.20.153/cgi-bin/form.py

```
date
(datetime.date(2020, 3, 26),)
```

5) You should provide a script that generates the database that you used in your tests. The script should be in a form that the TA can run it on his MYSQL system. If you think it is necessary you should include a file with instructions of how to do this.

Please reference generateDB.py or the dump file spawnDB.txt

6) 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.

The primary reason for not performing the additional query in Relational Algebra is do to the fact it requires aggretation and the use of a sum or multiply function. This is necessary to compute the total price of AAPL shares sold by the firm. We are only allowed to act on tuples via the union, intersection, and difference functions, thus, there is no way for us to compute this.