Homework2

Ankit Sati

Akshat Kiritkumar Jain

Q1

Python script to generate trade.csv:

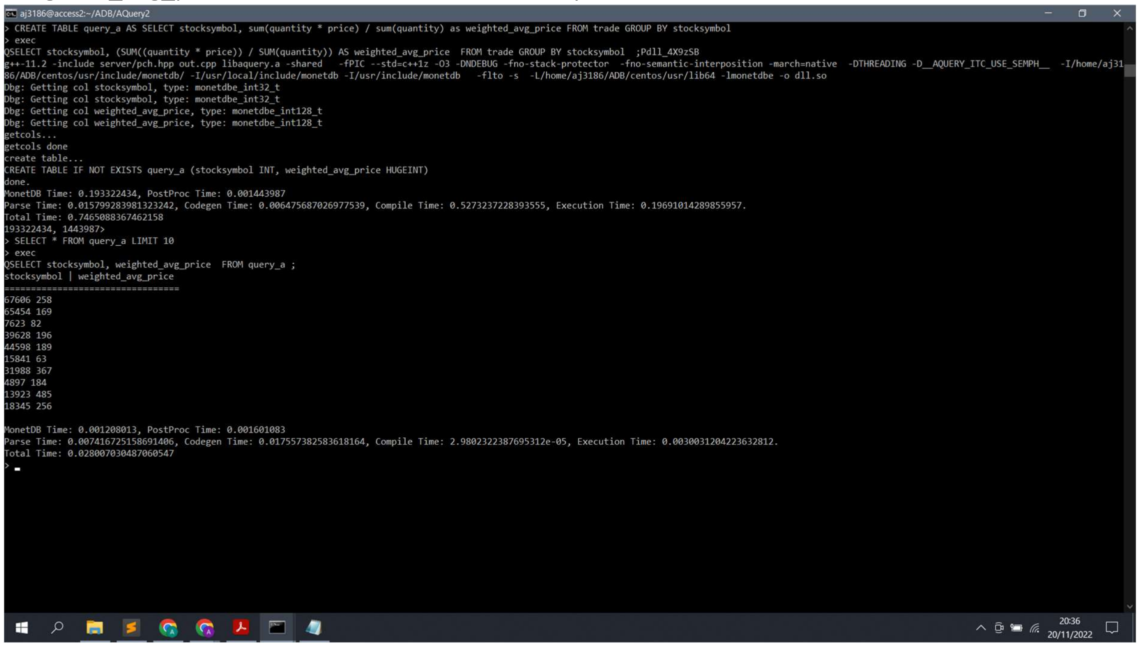
```
import numpy as np
import math
from random import randrange
import os

trade_file = 'trade.csv'

## Global variables
NUM_RECORDS = 10000000
NUM_STOCKSYMS = 100000
X = 70004

def gen(frac, N):
    p = np.random.permutation(N) + 1
    outvec = p
    while p.size > 1:
        p = p[:math.floor(frac * p.size)]
        outvec = np.concatenate([outvec, p])
    return np.random.permutation(outvec)

stock_sample = gen(0.3, X)

# checking if file already exists
if os.path.exists(trade_file):
    os.remove(trade_file)

f = open(trade_file, 'w')

f.write('stocksymbol,time,quantity,price' + '\n')

# generating data
old_prices = [-1 for i in range(NUM_STOCKSYMS)]

for i in range(NUM_RECORDS):
    stock = stock_sample[randrange(0, NUM_STOCKSYMS)]
    time = i
    quantity = randrange(100, 10001)
    price = randrange(50, 501)

    if old_prices[stock - 1] != -1:
        old_price = old_prices[stock - 1]
        while not (old_price - 5 <= price <= old_price - 1 or old_price + 1 <= price <= old_price + 5):
            price = randrange(50, 501)
    old_prices[stock - 1] = price

    f.write(str(stock) + ',' + str(time) + ',' + str(quantity) + ',' + str(price) + '\n')

f.close()
```

AQuery Commands:

CREATE TABLE trade (stocksymbol INT, time INT, quantity INT, price INT)

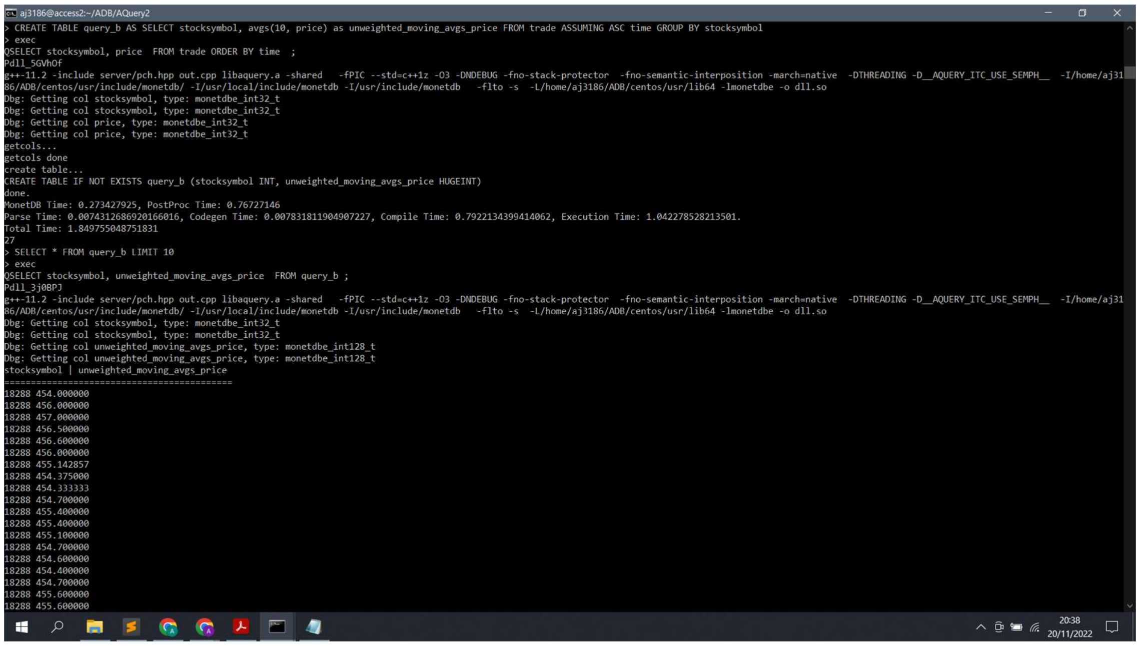
LOAD DATA INFILE "trade.csv" INTO TABLE trade FIELDS TERMINATED BY ","

a) CREATE TABLE query_a AS SELECT stocksymbol, sum(quantity * price) / sum(quantity) as weighted_avg_price FROM trade GROUP BY stocksymbol



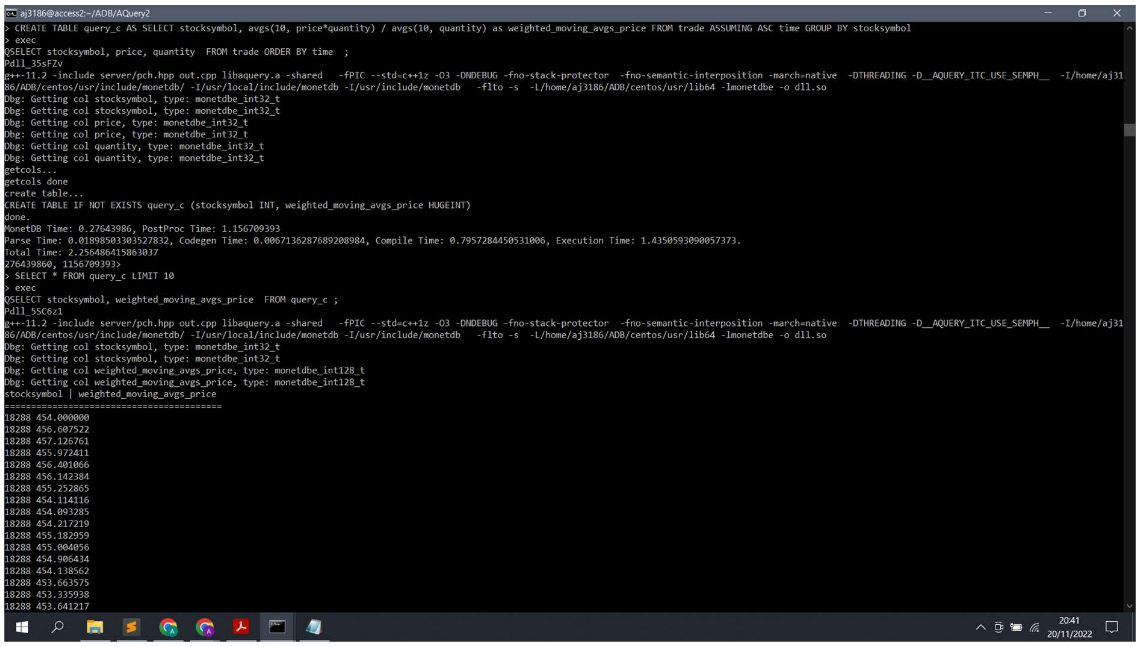
```
aj3186@access2:~/ADB/AQuery2
> CREATE TABLE query_a AS SELECT stocksymbol, sum(quantity * price) / sum(quantity) as weighted_avg_price FROM trade GROUP BY stocksymbol
> exec
QSELECT stocksymbol, (SUM((quantity * price)) / SUM(quantity)) AS weighted_avg_price  FROM trade GROUP BY stocksymbol  ;Pdll_4X9zSB
g++-11.2 -include server/pch.hpp out.cpp libaquery.a -shared   -fPIC --std=c++1z -O3 -DNDEBUG -fno-stack-protector  -fno-semantic-interposition -march=native  -DTHREADING -D__AQUERY_ITC_USE_SEMPH__  -I/home/aj31
86/ADB/centos/usr/include/monetdb/ -I/usr/local/include/monetdb -I/usr/include/monetdb   -flto -s  -L/home/aj3186/ADB/centos/usr/lib64 -lmonetdbe -o dll.so
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col weighted_avg_price, type: monetdbe_int128_t
Dbg: Getting col weighted_avg_price, type: monetdbe_int128_t
getcols...
getcols done
create table...
CREATE TABLE IF NOT EXISTS query_a (stocksymbol INT, weighted_avg_price HUGEINT)
done.
MonetDB Time: 0.193322434, PostProc Time: 0.001443987
Parse Time: 0.015799283981323242, Codegen Time: 0.006475687026977539, Compile Time: 0.5273237228393555, Execution Time: 0.19691014289855957.
Total Time: 0.7465088367462158
193322434, 1443987>
> SELECT * FROM query_a LIMIT 10
> exec
QSELECT stocksymbol, weighted_avg_price  FROM query_a ;
stocksymbol | weighted_avg_price
================================
67606 258
65454 169
7623 82
39628 196
44598 189
15841 63
31988 367
4897 184
13923 485
18345 256

MonetDB Time: 0.001208013, PostProc Time: 0.001601083
Parse Time: 0.007416725158691406, Codegen Time: 0.017557382583618164, Compile Time: 2.9802322387695312e-05, Execution Time: 0.0030031204223632812.
Total Time: 0.028007030487060547
> _
```

b) CREATE TABLE query_b AS SELECT stocksymbol, avgs(10, price) as unweighted_moving_avgs_price FROM trade ASSUMING ASC time GROUP BY stocksymbol



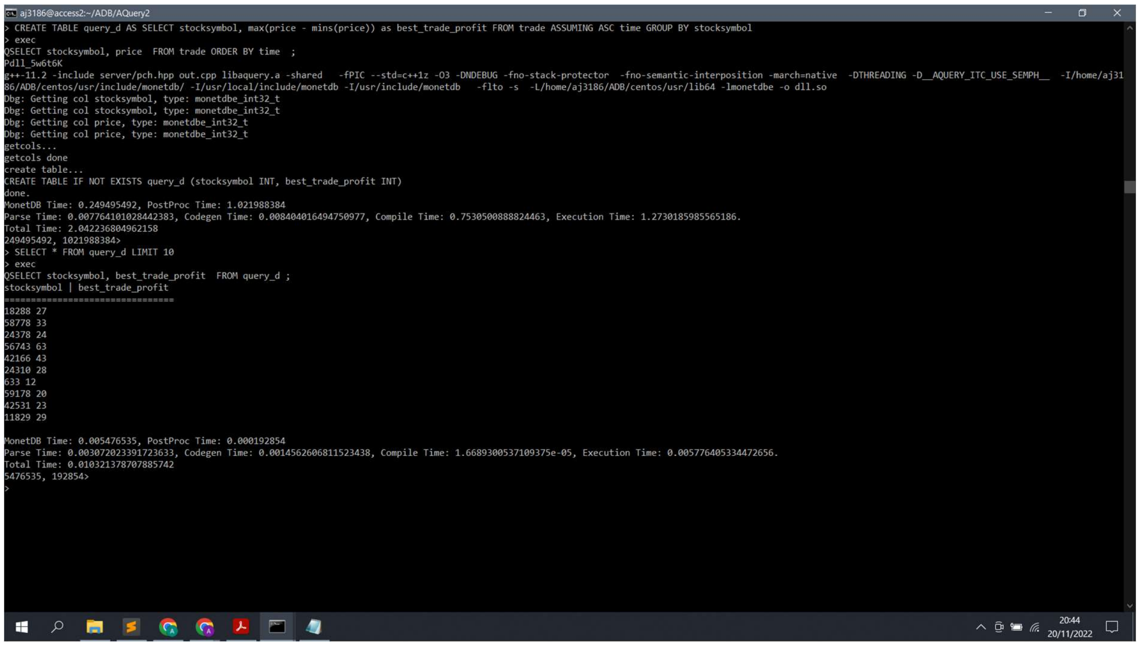
```
> CREATE TABLE query_b AS SELECT stocksymbol, avgs(10, price) as unweighted_moving_avgs_price FROM trade ASSUMING ASC time GROUP BY stocksymbol
> exec
QSELECT stocksymbol, price  FROM trade ORDER BY time  ;
Pdll_5GVhOf
g++-11.2 -include server/pch.hpp out.cpp libaquery.a -shared   -fPIC --std=c++1z -O3 -DNDEBUG -fno-stack-protector  -fno-semantic-interposition -march=native  -DTHREADING -D__AQUERY_ITC_USE_SEMPH__  -I/home/aj31
86/ADB/centos/usr/include/monetdb/ -I/usr/local/include/monetdb -I/usr/include/monetdb   -flto -s  -L/home/aj3186/ADB/centos/usr/lib64 -lmonetdbe -o dll.so
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col price, type: monetdbe_int32_t
Dbg: Getting col price, type: monetdbe_int32_t
getcols...
getcols done
create table...
CREATE TABLE IF NOT EXISTS query_b (stocksymbol INT, unweighted_moving_avgs_price HUGEINT)
done.
MonetDB Time: 0.273427925, PostProc Time: 0.76727146
Parse Time: 0.0074312686920166016, Codegen Time: 0.007831811904907227, Compile Time: 0.7922134399414062, Execution Time: 1.042278528213501.
Total Time: 1.849755048751831
27
> SELECT * FROM query_b LIMIT 10
> exec
QSELECT stocksymbol, unweighted_moving_avgs_price  FROM query_b ;
Pdll_3j0BPJ
g++-11.2 -include server/pch.hpp out.cpp libaquery.a -shared   -fPIC --std=c++1z -O3 -DNDEBUG -fno-stack-protector  -fno-semantic-interposition -march=native  -DTHREADING -D__AQUERY_ITC_USE_SEMPH__  -I/home/aj31
86/ADB/centos/usr/include/monetdb/ -I/usr/local/include/monetdb -I/usr/include/monetdb   -flto -s  -L/home/aj3186/ADB/centos/usr/lib64 -lmonetdbe -o dll.so
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col unweighted_moving_avgs_price, type: monetdbe_int128_t
Dbg: Getting col unweighted_moving_avgs_price, type: monetdbe_int128_t
stocksymbol | unweighted_moving_avgs_price
==========================================
18288 454.000000
18288 456.000000
18288 457.000000
18288 456.500000
18288 456.600000
18288 456.000000
18288 455.142857
18288 454.375000
18288 454.333333
18288 454.700000
18288 455.400000
18288 455.400000
18288 455.100000
18288 454.700000
18288 454.600000
18288 454.400000
18288 454.700000
18288 455.600000
18288 455.600000
```

c) CREATE TABLE query_c AS SELECT stocksymbol, avgs(10, price*quantity) / avgs(10, quantity) as weighted_moving_avgs_price FROM trade ASSUMING ASC time GROUP BY stocksymbol



```
> CREATE TABLE query_c AS SELECT stocksymbol, avgs(10, price*quantity) / avgs(10, quantity) as weighted_moving_avgs_price FROM trade ASSUMING ASC time GROUP BY stocksymbol
> exec
QSELECT stocksymbol, price, quantity  FROM trade ORDER BY time  ;
Pdll_35sFZv
g++-11.2 -include server/pch.hpp out.cpp libaquery.a -shared   -fPIC --std=c++1z -O3 -DNDEBUG -fno-stack-protector  -fno-semantic-interposition -march=native  -DTHREADING -D__AQUERY_ITC_USE_SEMPH__  -I/home/aj31
86/ADB/centos/usr/include/monetdb/ -I/usr/local/include/monetdb -I/usr/include/monetdb   -flto -s  -L/home/aj3186/ADB/centos/usr/lib64 -lmonetdbe -o dll.so
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col price, type: monetdbe_int32_t
Dbg: Getting col price, type: monetdbe_int32_t
Dbg: Getting col quantity, type: monetdbe_int32_t
Dbg: Getting col quantity, type: monetdbe_int32_t
getcols...
getcols done
create table...
CREATE TABLE IF NOT EXISTS query_c (stocksymbol INT, weighted_moving_avgs_price HUGEINT)
done.
MonetDB Time: 0.27643986, PostProc Time: 1.156709393
Parse Time: 0.01898503303527832, Codegen Time: 0.0067136287689208984, Compile Time: 0.7957284450531006, Execution Time: 1.4350593090057373.
Total Time: 2.256486415863037
276439860, 1156709393>
> SELECT * FROM query_c LIMIT 10
> exec
QSELECT stocksymbol, weighted_moving_avgs_price  FROM query_c ;
Pdll_5SC6z1
g++-11.2 -include server/pch.hpp out.cpp libaquery.a -shared   -fPIC --std=c++1z -O3 -DNDEBUG -fno-stack-protector  -fno-semantic-interposition -march=native  -DTHREADING -D__AQUERY_ITC_USE_SEMPH__  -I/home/aj31
86/ADB/centos/usr/include/monetdb/ -I/usr/local/include/monetdb -I/usr/include/monetdb   -flto -s  -L/home/aj3186/ADB/centos/usr/lib64 -lmonetdbe -o dll.so
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col weighted_moving_avgs_price, type: monetdbe_int128_t
Dbg: Getting col weighted_moving_avgs_price, type: monetdbe_int128_t
stocksymbol | weighted_moving_avgs_price
========================================
18288 454.000000
18288 456.607522
18288 457.126761
18288 455.972411
18288 456.401066
18288 456.142384
18288 455.252865
18288 454.114116
18288 454.093285
18288 454.217219
18288 455.182959
18288 455.004056
18288 454.906434
18288 454.138562
18288 453.663575
18288 453.335938
18288 453.641217
```

d) CREATE TABLE query_d AS SELECT stocksymbol, max(price - mins(price)) as best_trade_profit FROM trade ASSUMING ASC time GROUP BY stocksymbol



```
> CREATE TABLE query_d AS SELECT stocksymbol, max(price - mins(price)) as best_trade_profit FROM trade ASSUMING ASC time GROUP BY stocksymbol
> exec
QSELECT stocksymbol, price  FROM trade ORDER BY time  ;
Pdll_5w6t6K
g++-11.2 -include server/pch.hpp out.cpp libaquery.a -shared   -fPIC --std=c++1z -O3 -DNDEBUG -fno-stack-protector  -fno-semantic-interposition -march=native  -DTHREADING -D__AQUERY_ITC_USE_SEMPH__  -I/home/aj31
86/ADB/centos/usr/include/monetdb/ -I/usr/local/include/monetdb -I/usr/include/monetdb   -flto -s  -L/home/aj3186/ADB/centos/usr/lib64 -lmonetdbe -o dll.so
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col stocksymbol, type: monetdbe_int32_t
Dbg: Getting col price, type: monetdbe_int32_t
Dbg: Getting col price, type: monetdbe_int32_t
getcols...
getcols done
create table...
CREATE TABLE IF NOT EXISTS query_d (stocksymbol INT, best_trade_profit INT)
done.
MonetDB Time: 0.249495492, PostProc Time: 1.021988384
Parse Time: 0.007764101028442383, Codegen Time: 0.008404016494750977, Compile Time: 0.7530500888824463, Execution Time: 1.2730185985565186.
Total Time: 2.042236804962158
249495492, 1021988384>
> SELECT * FROM query_d LIMIT 10
> exec
QSELECT stocksymbol, best_trade_profit  FROM query_d ;
stocksymbol | best_trade_profit
================================
18288 27
58778 33
24378 24
56743 63
42166 43
24310 28
633 12
59178 20
42531 23
11829 29

MonetDB Time: 0.005476535, PostProc Time: 0.000192854
Parse Time: 0.003072023391723633, Codegen Time: 0.0014562606811523438, Compile Time: 1.6689300537109375e-05, Execution Time: 0.005776405334472656.
Total Time: 0.010321378707885742
5476535, 192854>
>
```

Q2

AQuery queries:

For fractal distribution:

CREATE TABLE trade_frac (stocksymbol INT, time INT, quantity INT, price INT)

LOAD DATA INFILE "trade.csv" INTO TABLE trade_frac FIELDS TERMINATED BY ","

Rule of thumbs used:

1. Remove irrelevant DISTINCT

   SELECT DISTINCT stocksymbol, time, quantity, price FROM trade_frac

   SELECT stocksymbol, time, quantity, price FROM trade_frac

2. With Covering indexes:

   SELECT stocksymbol FROM trade_frac WHERE quantity > 500

   CREATE INDEX quant_stocksymbol ON trade_frac (quantity, stocksymbol)

   SELECT stocksymbol FROM trade_frac WHERE quantity > 500

For Uniform distribution:

CREATE TABLE trade_uniform (stocksymbol INT, time INT, quantity INT, price INT)

LOAD DATA INFILE "trade_uniform.csv" INTO TABLE trade_uniform FIELDS TERMINATED BY ","

1. Remove irrelevant DISTINCT
   SELECT DISTINCT stocksymbol, time, quantity, price FROM trade_uniform
   SELECT stocksymbol, time, quantity, price FROM trade_uniform

2. With Covering indexes

   SELECT stocksymbol FROM trade_uniform WHERE quantity > 500

   CREATE INDEX quant_stocksymbol ON trade_uniform (quantity, stocksymbol)

   SELECT stocksymbol FROM trade_uniform WHERE quantity > 500

MySQL queries:

// Load both the tables.

CREATE TABLE trade_frac (stocksymbol INT, time INT, quantity INT, price INT);

LOAD DATA INFILE "trade.csv" INTO TABLE trade_frac FIELDS TERMINATED BY ",";

CREATE TABLE trade_uni (stocksymbol INT, time INT, quantity INT, price INT);

LOAD DATA INFILE "trade_uniform.csv" INTO TABLE trade_uni FIELDS TERMINATED BY ",";

1. Distinct:
   SELECT DISTINCT stocksymbol, time, quantity, price FROM trade_frac

   SELECT stocksymbol, time, quantity, price FROM trade_frac

   SELECT DISTINCT stocksymbol, time, quantity, price FROM trade_uni

   SELECT stocksymbol, time, quantity, price FROM trade_uni

2. Covering indexes:

   SELECT stocksymbol FROM trade_frac WHERE quantity > 500

   CREATE INDEX quant_stocksymbol ON trade_frac (quantity, stocksymbol)

   SELECT stocksymbol FROM trade_frac WHERE quantity > 500

   SELECT stocksymbol FROM trade_uni WHERE quantity > 500

   CREATE INDEX quant_stocksymbol1 ON trade_uni (quantity, stocksymbol)

```
SELECT stocksymbol FROM trade_uni WHERE quantity > 500
```

AQuery results:

	Fractal Data Distribution	Uniform Data Distribution
With DISTINT	253.89	277.66
Without DISTINCT	270.713	264.70
Without Indexing	214.77	200.35
With Indexing	209.00	205.27

We don't observe any increase in time without using Distinct for fractal distributions. In the case of Uniform distribution there is a slight decrease in time.

Indexing provides better results for fractal data which is not the case as observed in uniform data.

MySQL results:

	Fractal Data Distribution	Uniform Data Distribution
With DISTINT	2213.23	670.45
Without DISTINCT	21.3	17.8
With Indexing	15830.6	14754.7
Without Indexing	17348.3	16895.5

Without using distinct gives a huge boost in the results in both fractal and uniform distributions.

With covering indexes, we get noticeable results for both the distributions.

Q3

AQuery queries:

```
// Friends table

create table friends (person1 INT, person2 INT)

LOAD DATA INFILE "friends.txt" INTO TABLE friends FIELDS TERMINATED BY ","

// adding p2, p1 since they both are friends too and easier to perform joins

insert into friends select person2,person1 from friends


// Likes table

create table likes (person INT, artist INT)

LOAD DATA INFILE "like.txt" INTO TABLE likes FIELDS TERMINATED BY ","

// temp1 table with p1, p2, p2 who likes artist

create table temp1 as select friends.person1 as p1, friends.person2 as p2, likes.artist as artist from friends INNER JOIN likes on friends.person2 = likes.person


// final result where we remove those cases where p1 also likes the artist.

select * from temp1 except select * from temp1 INNER JOIN likes on temp1.p1 = likes.person and temp1.artist = likes.artist
```