CS315A
Assignment - I
Shreyash Raj, 170682

1. Queries in SQL and MongoDb :-

Q1: Find all A with A1 ≤ 50

query1_sql = SELECT * FROM A_100 WHERE A1<=50

query1_mongo = db.collection.find({"A1":{"$lte":50}})

Q2: Find all B in sorted order of B3.

query2_sql = SELECT * FROM B_100_3_1 ORDER BY B3

query2_mongo = db.collection.find().sort({“B3”:1})

Q3: Find average number of values per A1 by using only B table

query3_sql = SELECT AVG(COUNT) FROM (SELECT COUNT(*) as COUNT FROM B_100_3_1 GROUP BY B2)

query3_mongo = db.collection.aggregate([{"$group":{"_id":"$B2", "count":{"$sum":1}}}, {"$group":{"_id":"null", "average":{"$avg":"$count"}}}])

Q4: Find all A2 that corresponds to B by using B2 (output the fields of B and A2).

query4_sql = SELECT b.B1, b.B2, b.B3, a.A2 FROM A_100 AS a INNER JOIN B_100_3_1 AS b ON a.A1 = b.B2

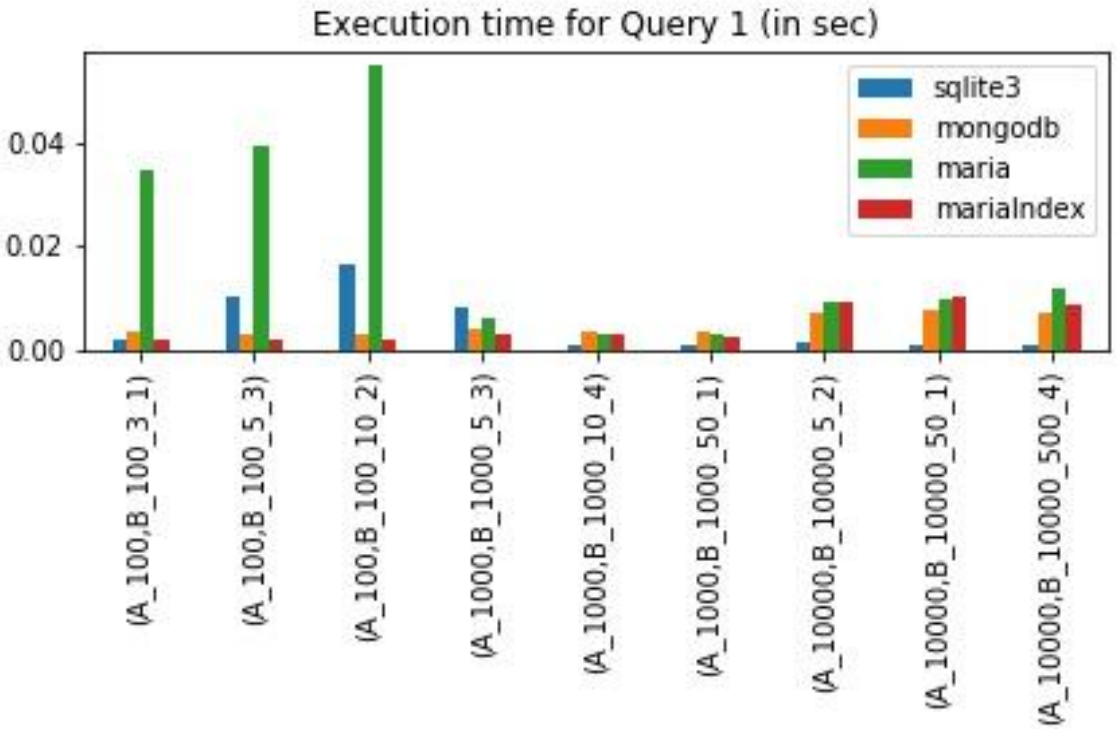
query4_mongo = db.collection.aggregate([{"$lookup":{"from":"A_100", "localField":"B2", "foreignField":"A1", "as":"A"}}, {"$project":{"B1":"$B1", "B2":"$B2", "B3":"$B3", "A2":"$A.A2"}}])

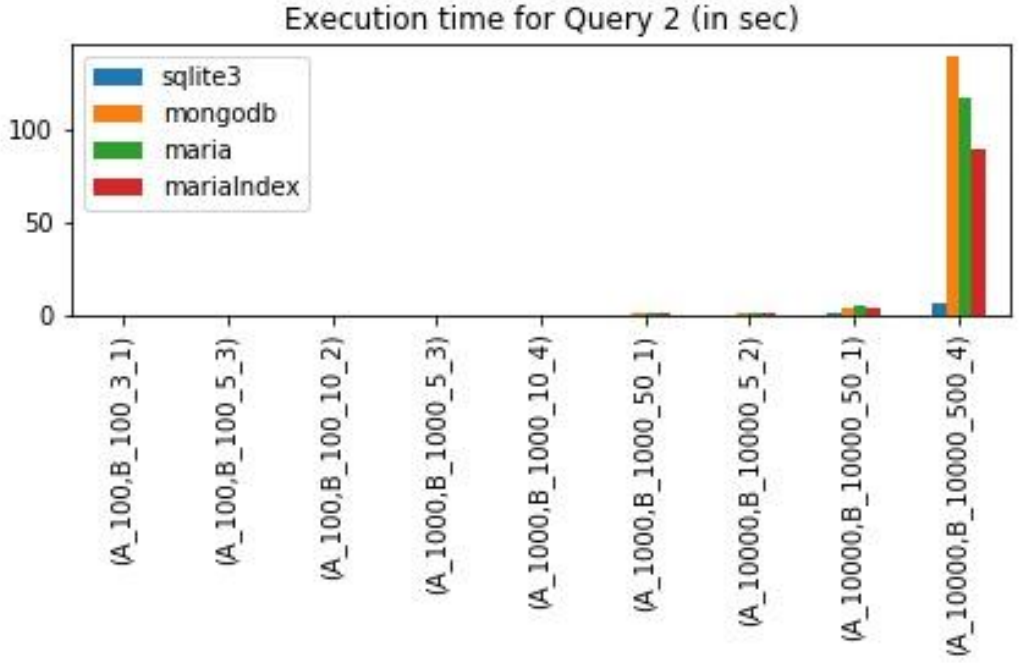
2.

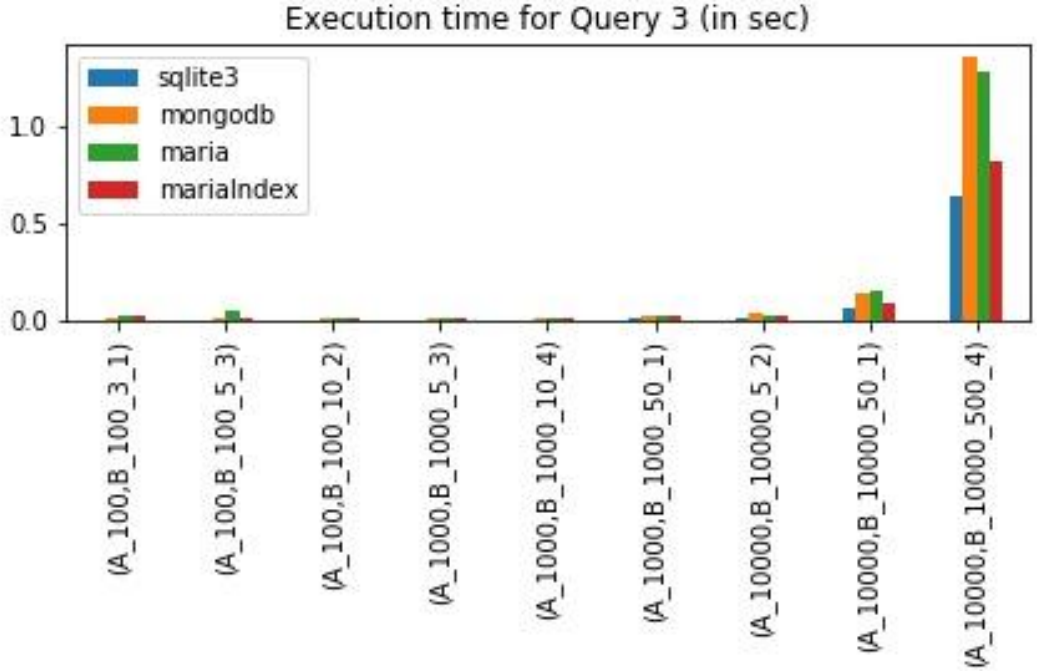
	(A_100,B_100_3_1)	**(A_100,B_100_5_3)**	**(A_100,B_100_10_2)**	**(A_1000,B_1000_5_3)**	**(A_1000,B_1000_10_4)**	**(A_1000,B_1000_50_1)**	**(A_10000,B_10000_5_2)**	**(A_10000,B_10000_50_1)**	**(A_10000,B_10000_500_4)**
sqlite_q1_time	0.001999	0.010313	0.0164	0.0081	0.00111	0.0007	0.001515	0.0007335	0.0007221
mongo_q1_time	0.0032793283	0.0030925 27	0.003197	0.0037951	0.0036425	0.003583	0.007114	0.0075432	0.0072370
maria_q1_time	0.034667849541	0.0394487381	0.054986	0.0058319	0.002810	0.002960	0.009276	0.00980878	0.011893
mariaIdx_q1_time	0.00212633609	0.0018295 05	0.001734 3	0.003137 9	0.00311	0.002685 5	0.009380	0.01014947	0.0084612
sqlite_q2_time	0.0008873939	0.0008382 79	0.001428 7	0.007151 4	0.010661	0.068022 8	0.092902	0.548771	6.192822
mongo_q2_time	0.00732636451	0.0071734 18	0.010112 04	0.035550 4	0.054901	0.2847	0.346506	3.1933619	139.2809
maria_q2_time	0.1062206029	0.036542	0.019236 5	0.074264	0.1506549	0.5052	0.652893	5.447992	117.44708
mariaIdx_q2_time	0.00704288	0.0074145 7	0.010794 0	0.056752	0.0953760	0.4224	0.559252	3.99998	89.09098
sqlite_q3_time	0.001969695	0.0011233 0	0.001113 65	0.001168	0.001947	0.0052	0.007712	0.055838	0.635612
mongo_q3_time	0.00276446	0.0028941 63	0.002768 6	0.004975	0.006463	0.0178	0.031387	0.1345394	1.351608
maria_q3_time	0.0245763	0.0532107 3	0.004230 26	0.009057 64	0.00631535	0.018695 8	0.025717 139	0.147968292 236328	1.2773852
mariaIdx_q3_time	0.02001667	0.0099405 05	0.003839 8	0.004443 8	0.005097 8	0.023170 82	0.020658 493	0.09018802	0.8181570
sqlite_q4_time	0.00111675	0.001577	0.0015	0.0080	0.0141	0.0633	0.085059	0.667809963	7.3163396

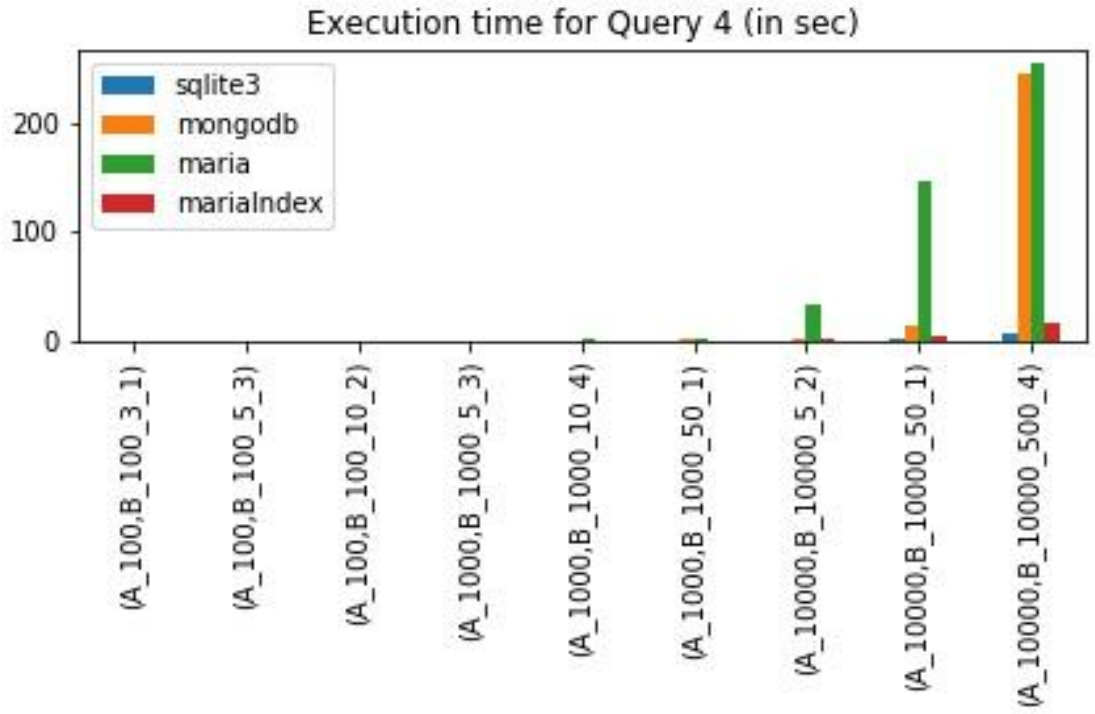
		258	095	1241	338	558			
mongo_q4_time	0.0173536 5	0.025632 5	0.0364 778	0.1862 1122	0.3101 09	1.4625 045	1.794323 563	13.6360275	245.68340253
maria_q4_time	0.0096441 50733948	0.009785 8905792 24	0.0160 447359 08508	0.3859 217166 90063	0.6590 034961 70044	2.8526 087999 3439	33.19258 91637802	148.6546367 54636	255.54364348
marialdx_q4_tim e	0.0114830 732	0.016226 05	0.0156 631	0.0721 674	0.1122 797	0.4489 272	0.576079 84	3.88727867	15.543544643

3.




Execution time for Query 2 (in sec)
sqlite3
mongodb
maria
mariaIndex
100
50
0
(A_100,B_100_3_1)
(A_100,B_100_5_3)
(A_100,B_100_10_2)
(A_1000,B_1000_5_3)
(A_1000,B_1000_10_4)
(A_1000,B_1000_50_1)
(A_10000,B_10000_5_2)
(A_10000,B_10000_50_1)
(A_10000,B_10000_500_4)


Execution time for Query 3 (in sec)
sqlite3
mongodb
maria
mariaIndex
1.0
0.5
0.0
(A_100,B_100_3_1)
(A_100,B_100_5_3)
(A_100,B_100_10_2)
(A_1000,B_1000_5_3)
(A_1000,B_1000_10_4)
(A_1000,B_1000_50_1)
(A_10000,B_10000_5_2)
(A_10000,B_10000_50_1)
(A_10000,B_10000_500_4)



4. The conclusions are:
   - As the database table size increases the time of execution of each query increases for each database system.
   - For query1 and query3 since the output file would be of fixed size and significantly small therefore both these queries took least amount of time to execute as compared to the other two queries.
   - After indexing query1 was much faster because of the constant time complexity of such queries after indexing, execution time for other queries improved as well after indexing.
   - query4 was the most expensive query followed by query2 then query3 and finally query1
   - Mongodb requires more space on RAM as compared to the other because sorting could barely happen on my pc with 8GB of ram for the largest file with mongodb.
   - Relational databases are faster than MongoDb (NoSql) for the above queries
   - Indexing decreases the execution time to a great extent.

- Sort and join operations are much more expensive than selection based on a boolean condition or count and average.
- There is a huge execution time difference between printing the output to a file vs database engine just executing the query.

My machine configuration :

- OS - Linux 64-bit (Ubuntu 20.04)
- RAM - 8GB
- HDD - 1TB
- CPU - Intel core i5, 8th gen, speed - 1.6 GHz

5.

About the script and code to run queries :-

There is one file named **“run_me.py”** which is supposed to be run in the same folder containing the databases. It will ask for a username and password to connect to mysql database for the mariadb database.

There are a couple of non conventional imports in the python file, which can be installed using the following command:

- pymongo - pip install pymongo
- mysql.connector - pip install mysql-connector-python-rf
- sqlalchemy - pip install sqlalchemy
- pymysql - pip install PyMySql

Give the permission for the file to be executable by using the command **chmod 754 run_me.py**

The script will create the required databases and tables in sqlite3, mongodb, mariadb without index and mariadb with index.
It will use the following .csv files based on my roll number i.e. 170682 :

A_100, A_1000, A_10000, B_100_3_1, B_100_5_3, B_100_10_2, B_1000_5_3, B_1000_10_4, B_1000_50_1, B_10000_5_2, B_10000_50_1, B_10000_500_4

Then it will process the queries with 7 iterations per query per database. It will then output 21 files, out of which 16 will be the query output, 4 will be the graphs depicting execution time for each query against the size of database tables and on different database systems, then the rest 1 file will be the table for execution times of each query wrt all database tables and all database system.