DBMS Assignment 1

Piyush Rane
180511

The database files that I used according to my roll no are

```
A_100 A_1000 A_10000
B_100_3_0 B_100_5_0 B_100_10_0 B_1000_5_0 B_1000_10_1 B_1000_50_1 B_10000_5_0
B_10000_50_1 B_10000_500_1
```

Note:- I changed the names of files by replacing '-' with '_' for simplicity

1. The equivalent queries in SQL and MongoDB are given below

(a) SQlite3

```
(1).  SELECT * FROM A WHERE A1 <= 50;
(2).  SELECT * FROM B ORDER BY B3;
(3).  SELECT AVG(sum)
      FROM (
           SELECT COUNT(B1) AS sum
           FROM B
           GROUP BY B2
      );
(4).  SELECT B1,B2,B3,A2
      FROM B
      JOIN A
      ON A.A1 = B.B2;
```

(b) MongoDB

(1).

```
db.A.aggregate([
        {
            $match:
            {
            "A1":{
                "$lte" :  50
                }
            }
        }])
```

(2).

```
db.B.aggregate([
         {
              $sort : { "B3": 1 }
         }
]).pretty()
```

(3).

```
db.B.aggregate([
            {
                $group:
                {
                    _id:"$B2",
                    "count": { $sum:1 }
                }
            },
            {
                $group:
                {
                    _id:null,
                    "Avg per A1":{$avg:"$count"}
                }
            }
        ]).pretty()
```

(4).

```
db.B.aggregate([
            {
                $lookup:
                {
                    from:"A",
                    localField:"B2",
                    foreignField:"A1",
                    as:"new"
                }
            },
            { $unwind:"$new" },
            {
                $project: { "new.A2":1, B1:1, B2:1, B3:1 }
            }
        ]).pretty()
```

2.

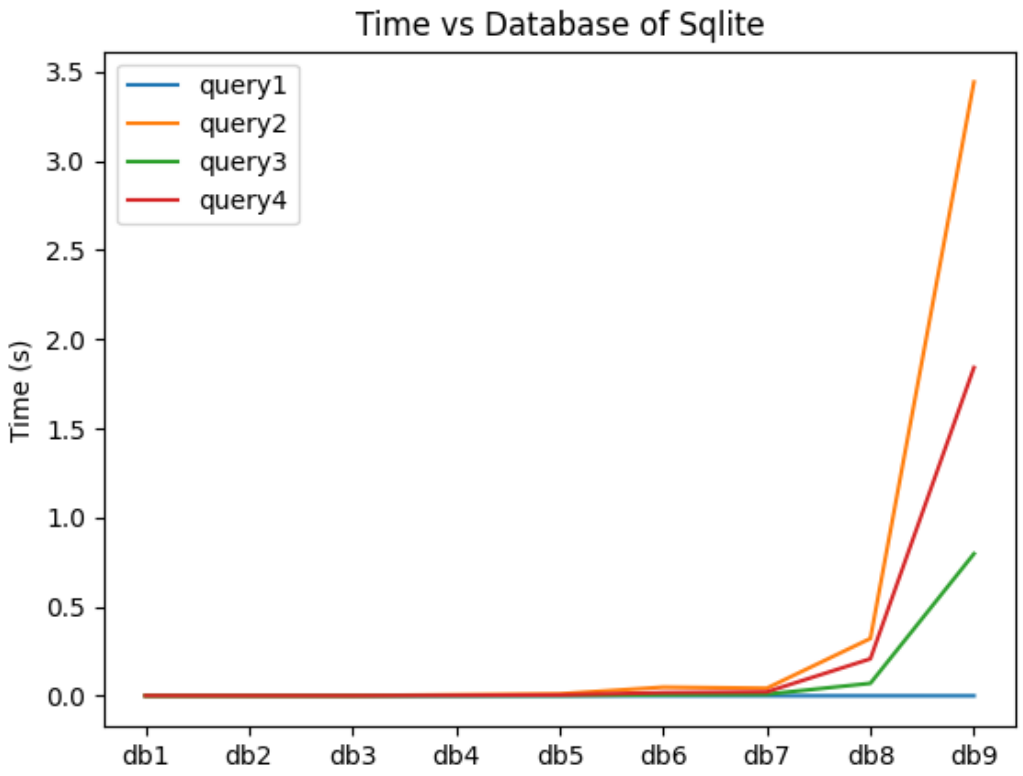
Table of all the Average values

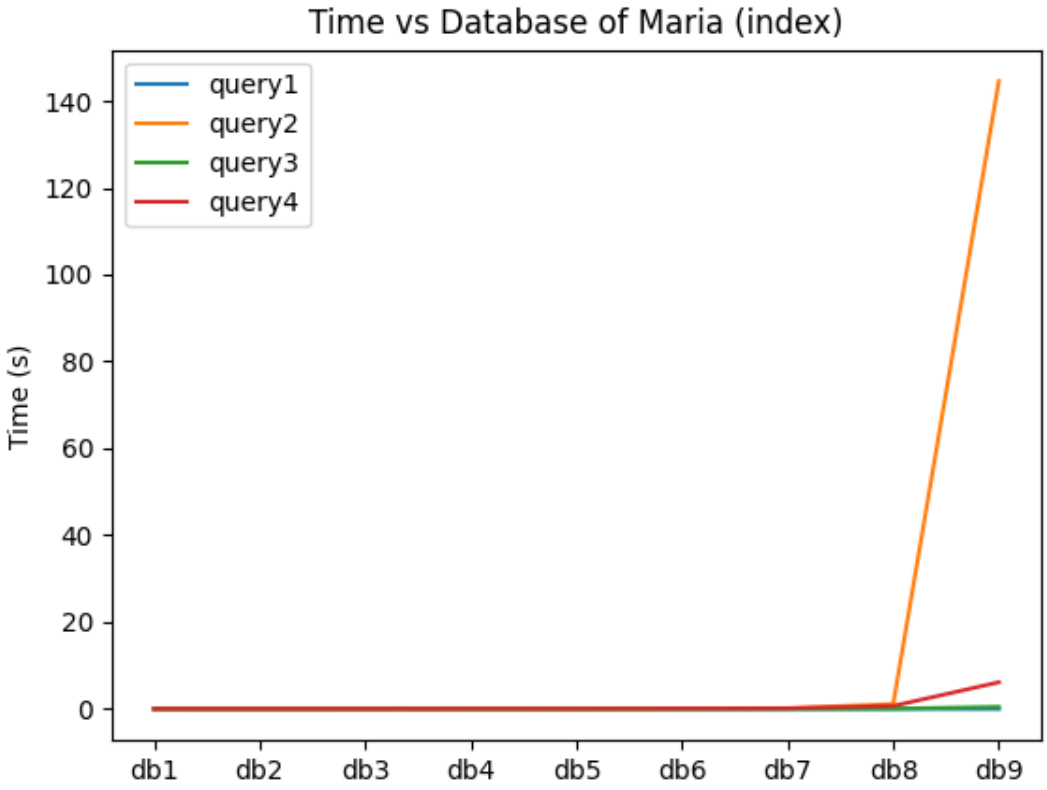
		db1(avg)	db2(avg)	db3(avg)	db4(avg)	db5(avg)	db6(avg)	db7(avg)	db8(avg)	db9(avg)
Sqlite	query1	0.000440	0.000407	0.000414	0.000435	0.000375	0.000426	0.000440	0.000459	0.000385
	query2	0.001084	0.001150	0.001678	0.006440	0.011126	0.048906	0.042965	0.321773	3.443755
	query3	0.000240	0.000398	0.000439	0.001045	0.001648	0.005899	0.009185	0.070337	0.797214
	query4	0.001272	0.001091	0.000950	0.003004	0.005240	0.018260	0.022384	0.207763	1.841938
Maria (index)	query1	0.000301	0.000247	0.000233	0.000251	0.000274	0.000251	0.000275	0.000239	0.000240
	query2	0.033447	0.013877	0.031766	0.090059	0.053257	0.125984	0.180923	1.123347	144.569065
	query3	0.000308	0.000322	0.000394	0.001104	0.001619	0.005250	0.009058	0.051623	0.483816
	query4	0.001188	0.001417	0.002062	0.012246	0.016458	0.068554	0.127128	0.687433	6.138065
Maria (no index)	query1	0.000291	0.000260	0.000276	0.000662	0.000649	0.000718	0.005234	0.004429	0.003897
	query2	0.002936	0.003877	0.003889	0.018328	0.024126	0.091414	0.113218	0.986303	104.356098
	query3	0.000594	0.000607	0.000899	0.003541	0.005508	0.019062	0.031459	0.171258	1.654273
	query4	0.002568	0.003244	0.005704	0.293350	0.502152	2.129112	27.954886	209.006185	2018.058060
MongoDB	query1	0.000200	0.000000	0.000000	0.001200	0.001400	0.001400	0.014200	0.014200	0.013400
	query2	0.001000	0.001000	0.002200	0.010600	0.019400	0.051800	0.062600	0.343200	4.353400
	query3	0.000800	0.000800	0.000800	0.011000	0.014600	0.045600	0.047200	0.233400	1.945400
	query4	0.016200	0.026800	0.018400	0.073600	0.079600	0.088600	0.515600	0.464000	0.475600

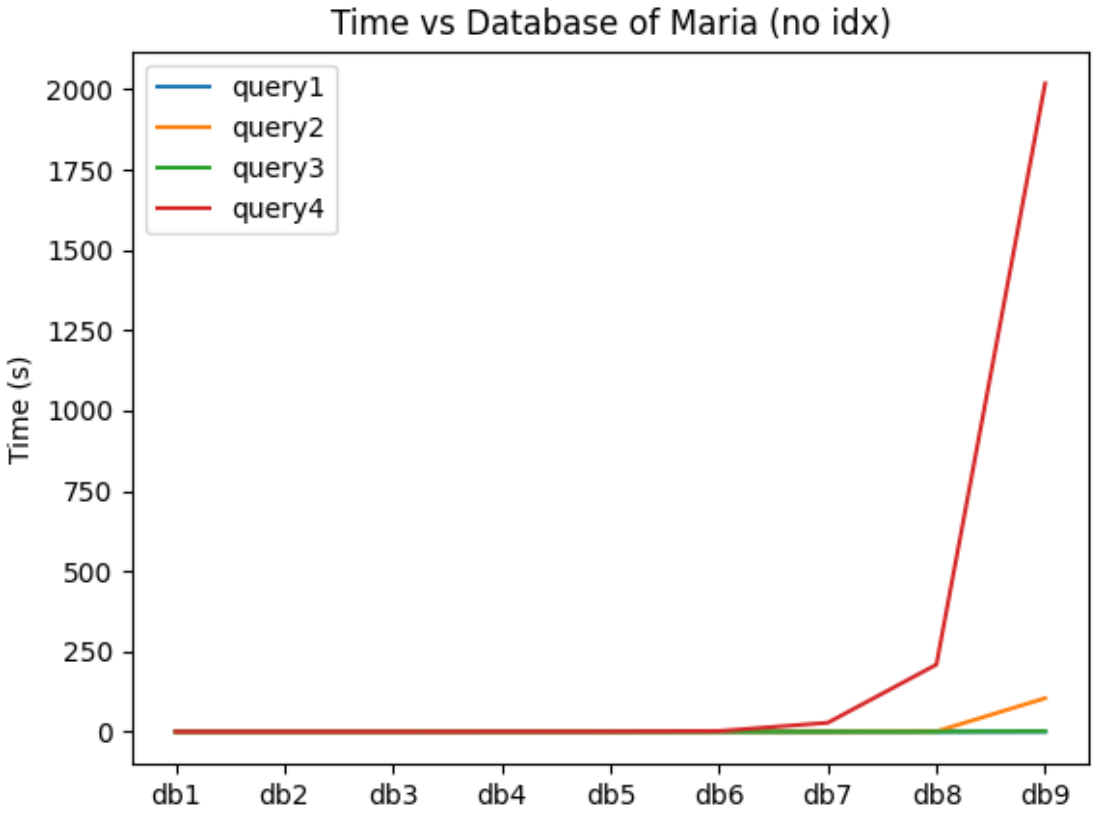
Table of all the standard deviation values

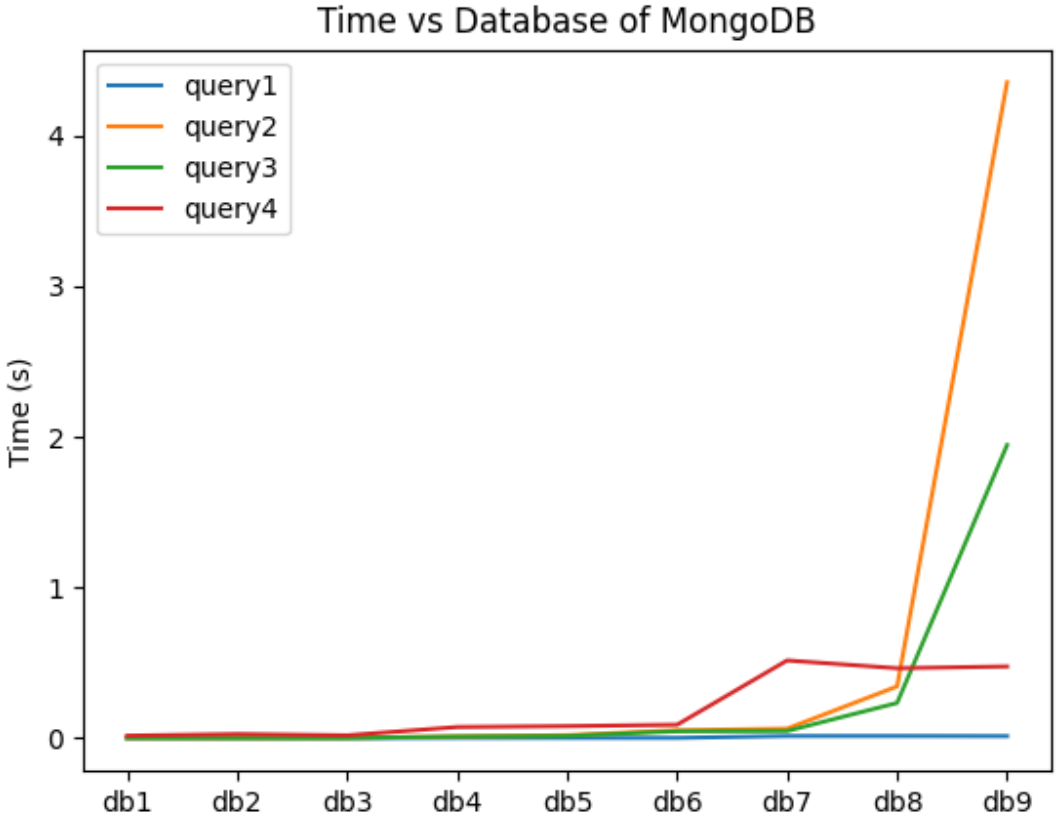
		db1(std)	db2(std)	db3(std)	db4(std)	db5(std)	db6(std)	db7(std)	db8(std)	db9(std)
Sqlite	query1	0.000017	0.000044	0.000070	0.000065	0.000075	0.000045	0.000012	0.000047	0.000063
	query2	0.000551	0.000534	0.000833	0.003033	0.005413	0.016018	0.004614	0.032844	0.131485
	query3	0.000045	0.000176	0.000164	0.000088	0.000111	0.000187	0.000388	0.002273	0.029194
	query4	0.000482	0.000516	0.000224	0.000395	0.000974	0.004598	0.000448	0.050518	0.111701
Maria (index)	query1	0.000067	0.000039	0.000023	0.000032	0.000044	0.000042	0.000043	0.000032	0.000032
	query2	0.017294	0.001909	0.023502	0.059083	0.011659	0.014079	0.026250	0.067592	35.510233
	query3	0.000007	0.000008	0.000050	0.000017	0.000080	0.000041	0.000136	0.001358	0.031818
	query4	0.000086	0.000189	0.000135	0.002613	0.001332	0.010757	0.008218	0.039528	0.304813
Maria (no index)	query1	0.000164	0.000093	0.000087	0.000156	0.000157	0.000196	0.001418	0.000781	0.000126
	query2	0.004761	0.006650	0.005338	0.010479	0.010066	0.021425	0.018572	0.083376	6.783621
	query3	0.000083	0.000008	0.000041	0.000159	0.000131	0.000391	0.000190	0.004355	0.028280
	query4	0.000014	0.000081	0.000082	0.032982	0.027207	0.016904	0.212758	4.565348	23.611835
MongoDB	query1	0.000447	0.000000	0.000000	0.000837	0.000548	0.000548	0.001643	0.001483	0.002408
	query2	0.000000	0.000000	0.000447	0.001817	0.001517	0.005541	0.001673	0.013535	0.151968
	query3	0.000447	0.000447	0.000447	0.002345	0.003715	0.004827	0.005357	0.028413	0.071717
	query4	0.009985	0.003834	0.005727	0.012582	0.009099	0.012876	0.081230	0.042831	0.052099

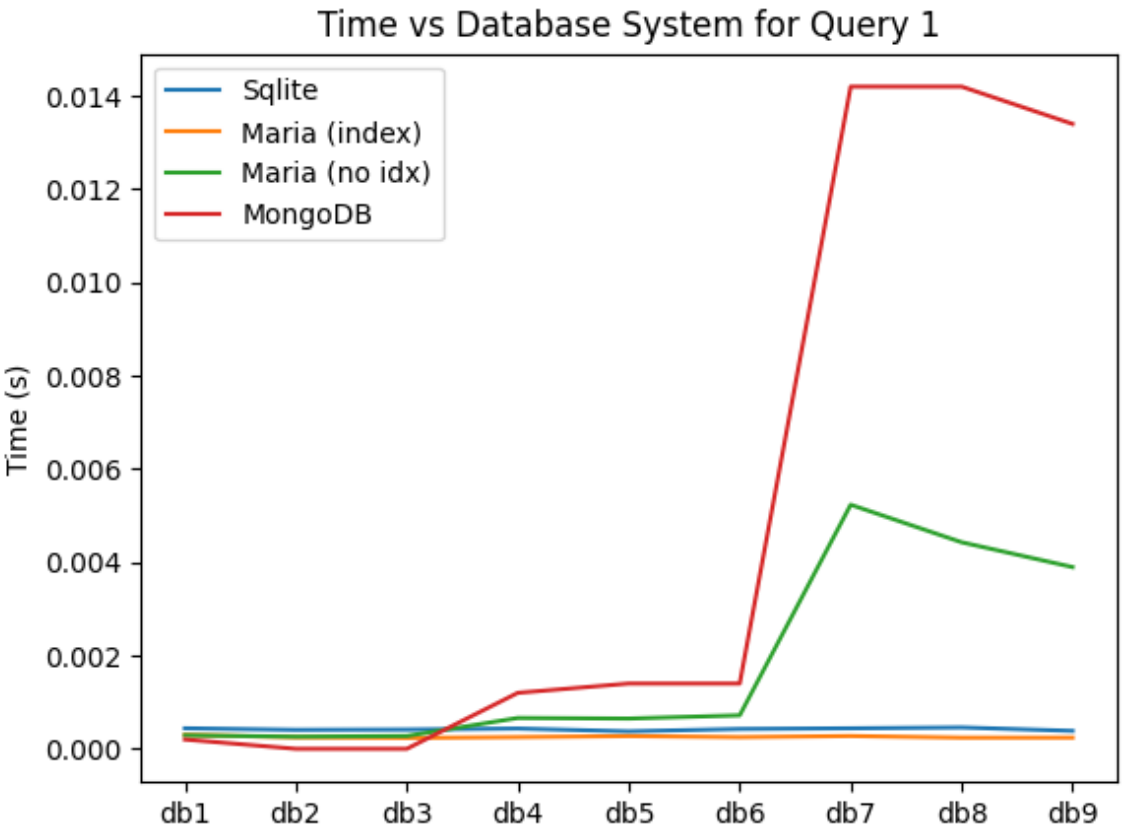
3.

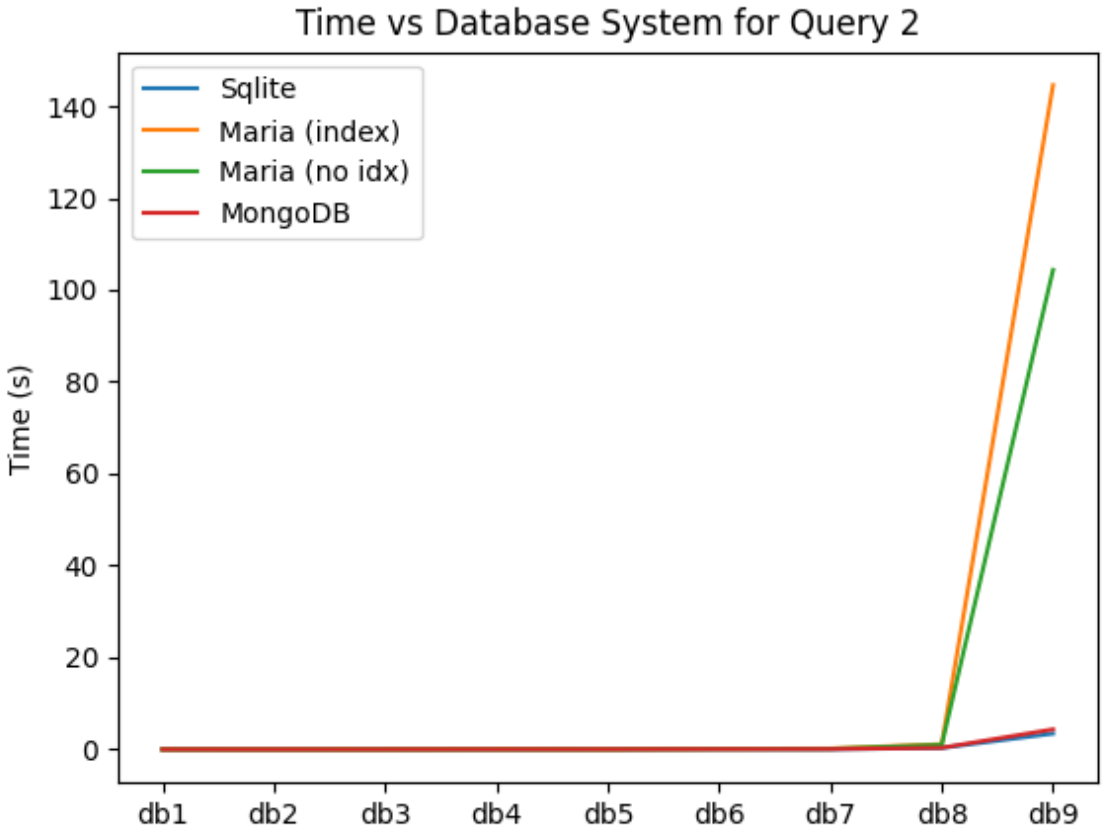

Time vs Database of Sqlite
query1
query2
query3
query4
Time (s)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
db1
db2
db3
db4
db5
db6
db7
db8
db9

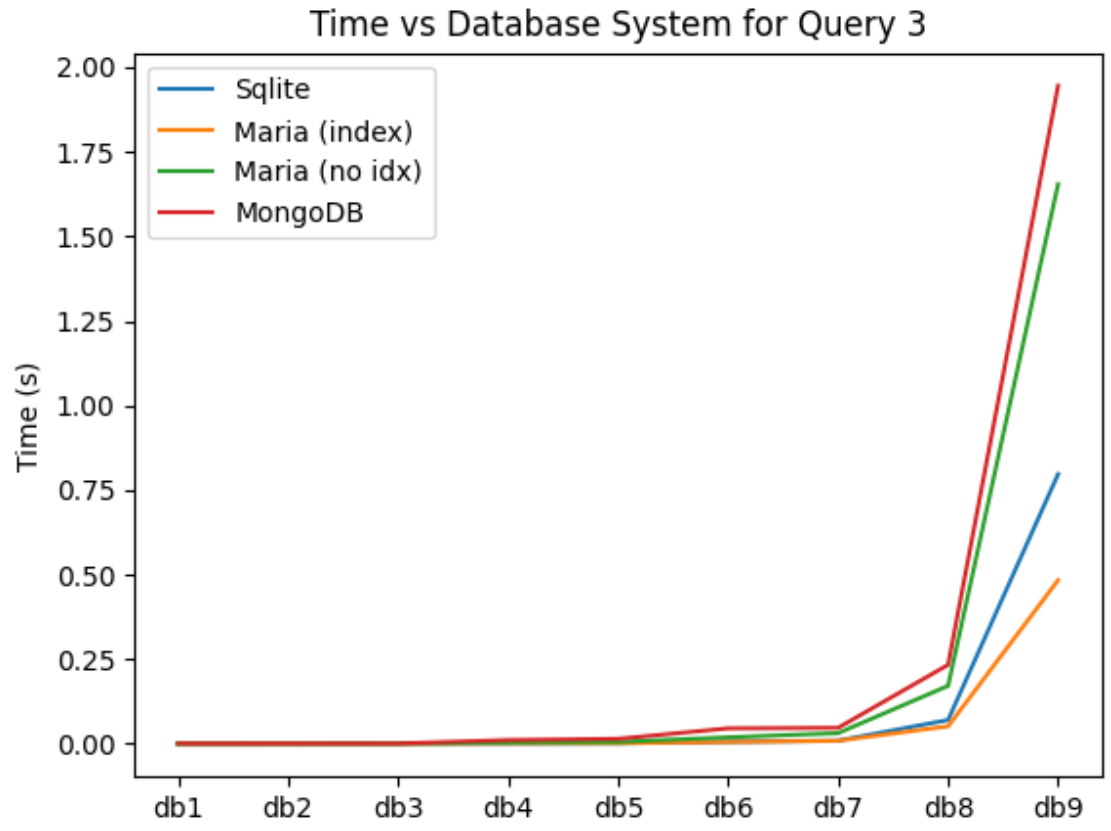

Time vs Database of Maria (index)
query1
query2
query3
query4
Time (s)
140
120
100
80
60
40
20
0
db1
db2
db3
db4
db5
db6
db7
db8
db9

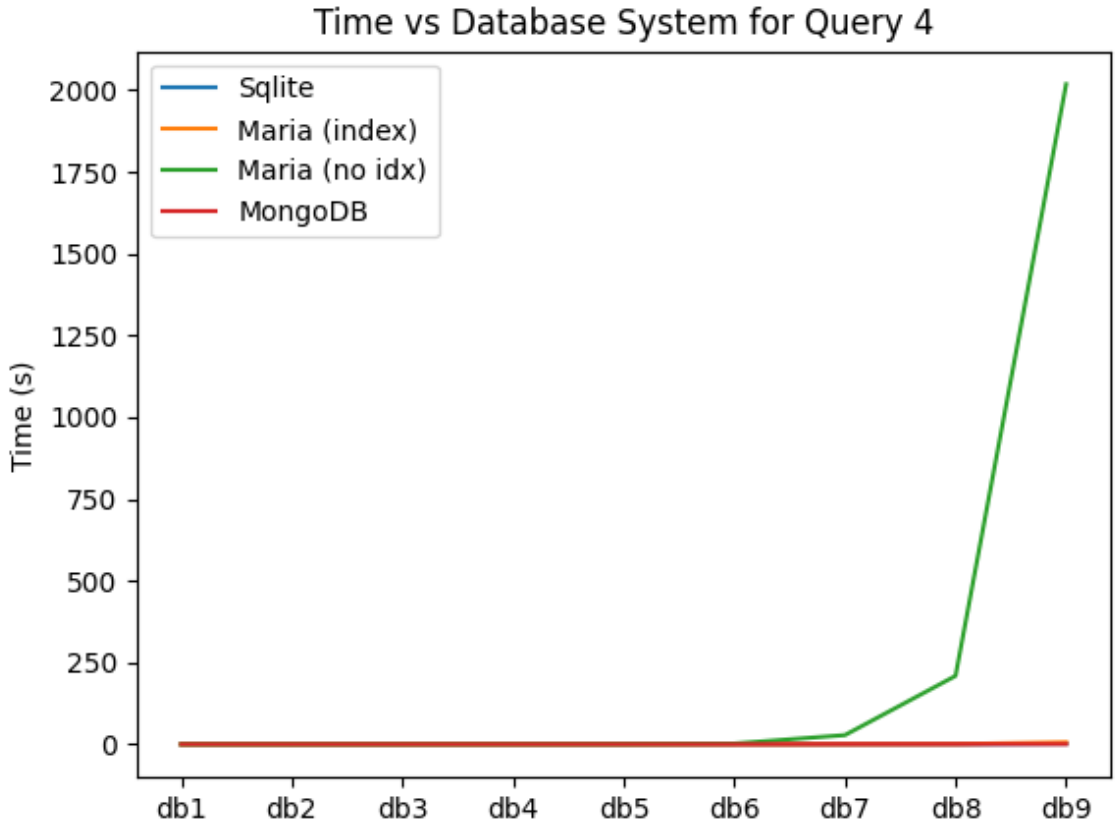

Time vs Database of Maria (no idx)
query1
query2
query3
query4
Time (s)
2000
1750
1500
1250
1000
750
500
250
0
db1
db2
db3
db4
db5
db6
db7
db8
db9


Time vs Database of MongoDB
query1
query2
query3
query4
Time (s)
4
3
2
1
0
db1
db2
db3
db4
db5
db6
db7
db8
db9


Time vs Database System for Query 1
Sqlite
Maria (index)
Maria (no idx)
MongoDB
Time (s)
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
db1
db2
db3
db4
db5
db6
db7
db8
db9


Time vs Database System for Query 2
Sqlite
Maria (index)
Maria (no idx)
MongoDB
Time (s)
140
120
100
80
60
40
20
0
db1
db2
db3
db4
db5
db6
db7
db8
db9


Time vs Database System for Query 3
Sqlite
Maria (index)
Maria (no idx)
MongoDB
Time (s)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
db1
db2
db3
db4
db5
db6
db7
db8
db9


Time vs Database System for Query 4
Sqlite
Maria (index)
Maria (no idx)
MongoDB
Time (s)
2000
1750
1500
1250
1000
750
500
250
0
db1
db2
db3
db4
db5
db6
db7
db8
db9

4.

The first thing that we observed is that in small databases it does not matter which system we are using for the query (e.g MongoDB or SQL), the time taken by the query will be similar. When we increase the size of the database these things changes and the execution time taken by the query will start to depend on various things like which system are we using (Maria or mongodb or SQLite), is there any indexing present or not, more the size of database more will be the time (in general), the number of other processes running on the computer will also affect the running time of the query.

Graph analysis by query:-
For query 1, we can see by the graph that for small dbs, MongoDB works fastest, and later on increasing the db size it behaves the worst, the 2nd worst is MariaDB (without index).

For query 2, we again saw the effect of size which increases the time for every database system.

For query 3, here we can see the role of indexing as maria without indexing took so much more time than maria with indexing, also MongoDB behaves worse here maybe due to the reason that it's not a relational database.

For query 4, here the indexing factor dominates and the time for all the other database is so small compared to MariaDB without indexing that they seem to look as they took an equal amount of time to run the query

Graph analysis by databases:-
In SQLite, the order of time taken by the query is q2>q4>q3>q1. q2 involves sorting so it takes longer and then comes q4 as join operation takes more time. q1 is simply a select statement with 1 condition and q2 involves subquery so it comes on 2nd.

In MariaDB with index, again ordering took more time as sort operation is time taking and it took time only after db8, before it approx all query took similar time.

In MariaDB without indexing, the time taken by the query is huge. It took approx 2000 seconds to run the 4th query while in MariaDB with indexing, the slowest query took about 140s, this shows that indexing plays a huge role in the speed of executing a query.

In MongoDB, the database is stored in collections and it is not a relational database so it behaves differently, but here also sorting takes the most of time, then grouping takes less time as compared to it.

Conclusion:- The execution time of a query can depend on the system on which it is running, the size of the database, the type of database (relational/non-relational, Sqlite, mongo, or maria), and the most important thing is that indexing plays a major role in execution time. One thing that we also observed while running the query is that the execution time of a query is not much, but the time it took to print the result on the terminal or in a file is so much more.

5.

The 180511.zip contains 4 folders corresponding to 4 databases. Each folder contains a separate “readme.txt” showing how to run the code and where the time outputs are getting stored. After getting all the time outputs in the “time” folder present in all 4 folders, make a directory “plots” and then run the command “python3 table.py”, it will create 8 plots that I have shown in this pdf above and also it will produce 2 csv files, one for Average time and one for standerd deviation and 2 html files again for the same. I have copied the table produced by the html file in this pdf above as answer to question number 2.

Note 1:- Put the data files in a separate folder and name that folder “data”, also place the folder in the same directory where the other 4 folders are present. Replace ‘-’ to ‘_’ as my code needs that (e.g A-100.csv -> A_100.csv).
Note 2:- Install "expect" in your system to run the expect script that I have used.

I am also writing the readme.txt of each folder here.

./sqlite/readme.txt :-
Run the command below and it will do everything needed to produce time per query per database:-

"bash run.sh"

It will take 1-2 mins to run

I have used expect script to import the data to sqlite database and to run the queries.
The “run.sh” script first imports the data using the "import.sh" file and then runs "time.sh" to get the time in file "time.txt" and the output of queries in folder "query".
After getting the time, run "clean.py" to get the time for each database in different file having 4 lines representing time of 4 queries and each line has 7 time data separated by ‘,’ stored in folder "time".

./maria_with_index/readme.txt :-
Run the command below and it will do everything needed to produce time per query per database:-

"bash run.sh"

It will take approx 30 minutes

I have used expect script to import the data to MariaDB database and bash script to run the queries. The “run.sh” script first imports the data using "expect import.sh" and then runs "bash time.sh" to get the time in 4 files naming "t1.txt" for query 1, "t2.txt" for query 2 and so on.
After getting the time, run "clean.py" to get the time for each database in a different file having 4 lines representing time of 4 queries and each line have 7 time data separated by ‘,’ stored in folder "time".

./maria_without_index/readme.txt :-
Run the command below and it will do everything needed to produce time per query per database:-

"bash run.sh"

It can take 3-4 hours to run

I have used expect script to import the data to MariaDB database and bash script to run the queries. The “run.sh” script first imports the data using "expect import.sh" and then runs "bash time.sh" to get the time in 4 files naming "t1.txt" for query 1, "t2.txt" for query 2 and so on. After getting the time, run "clean.py" to get the time for each database in a different file having 4 lines representing time of 4 queries and each line have 7 time data separated by ‘,’ stored in folder "time".

./mongo/readme.txt :-
Run the command below and it will do everything needed to produce time per query per database:-

"bash run.sh"

It can take 2-3 minutes to run

I have used a bash script to import the data to MongoDB database and java script to run the queries. The “run.sh” script first import the data using "bash import.sh" and then run "mongo s1.js | grep "millis" | tee t1.txt" and so on to get the time in 4 files naming "t1.txt" for query 1, "t2.txt" for query 2 and so on. After getting the time, run "clean.py" to get the time for each database in a different file having 4 lines representing time of 4 queries and each line have 7 time data separated by ‘,’ stored in folder "time".

My system configurations :-
OS -> Ubuntu 18.04.5 LTS
Memory -> 11.6 GiB
Processor -> Intel® Core™ i5-7200U CPU @ 2.50GHz × 4
Graphics -> Intel® HD Graphics 620 (KBL GT2)
GNOME -> 3.28.2
OS type -> 64-bit
Disk -> 141.0 GB