ASSIGNMENT #4 REPORT

Student Name and CCID:

A. Qasim Khawaja (Name) khawaja (CCID)
B. Muhammad Haris (Name) mharis2 (CCID)

By submitting this assignment, the students named above confirm that they have worked on it themselves without any help by other people. If any external resources were used, please state which ones and how they were used:

PART 1

Task A (no index):

Cardinality of Table Parts	**Average Processing time for index free Q1 (ms)**
100	0.78125 ms
1000	0.46875 ms
10,000	0.625 ms
100,000	0.625 ms
1,000,000	0.625 ms

Cardinality of Table Parts	**Average Processing time for index free Q2 (ms)**
100	0.625 ms
1000	0.78125 ms
10,000	1.71875 ms
100,000	12.03125 ms
1,000,000	97.65625 ms

Task B:

Compare, contrast and explain the trends observable in both tables above (Task A)

For Q1, the average processing time stayed relatively the same regardless of the size of entries; however, for Q2, as the entry size increased, the average processing time got slower. Note that for Q2, given the same size of entries, the average processing time was much longer, i.e. for 1 million entries, Q1 had an average processing speed of about 0.625 ms, whereas Q2 had an average processing time of 97.65625 ms (significantly longer)

Task C (using index):

Cardinality of Table Parts	**Average Processing time for indexed Q1 (ms)**
100	0.78125 ms
1000	0.78125 ms
10,000	0.625 ms
100,000	0.625 ms
1,000,000	0.625 ms

Cardinality of Table Parts	**Average Processing time for indexed Q2 (ms)**
100	0.78125 ms
1000	0.78125 ms
10,000	1.25 ms
100,000	1.5625 ms
1,000,000	8.90625 ms

Task D:

Compare, contrast and explain the trends observable in both tables above (Task C)
After creating the index, Q1's average processing time stayed relatively the same regardless of the size of entries; however, for Q2, the average process time increased at a slower pace for larger entries when compared to the average process time without the table. Note that on average, it is still true (even with the index) that Q2 has a longer average processing time than Q1. For example, 1 million entries Q1 had an average processing speed of about 0.625 ms, whereas Q2 had an average processing time of 8.90625 ms (still slower)

Task E:

Compare, contrast and explain the trends observed in Task D to the trends observed in Task B. Discuss the cost-benefit of the index space cost and query performance.
The query performance for Q2 was much faster with indexing, i.e. for 1 million entries, Task B had an average processing time of about 97.65625 ms. In contrast, for Task D, the average processing time for 1 million entries was about 8.90625 ms which was a significantly slower average processing time. Note that for Q1, the average processing time in both Task B and D remained relatively the same, meaning that the index had a negative cost with the increased database size, at least for Q1. The query performance was significantly increased as the average processing time was much faster for Q2 with the index; therefore, the space cost was justified for Q2.

PART 2

Task F (no index):

Cardinality of Table Parts	**Average Processing time for index-free Q3 (ms)**
100	0.3125 ms
1000	0.625 ms
10,000	2.5 ms
100,000	28.75 ms
1,000,000	271.71875 ms

Task G (using index):

Cardinality of Table Parts	**Average Processing time for indexed Q3 (ms)**
100	0.3125 ms
1000	0.3125 ms
10,000	0.46875 ms
100,000	2.03125 ms
1,000,000	14.21875 ms

Task H:

Compare, contrast and explain the trends observed in Task F to the trends observed in Task G. Discuss the cost-benefit of the index space cost and query performance.
For both Task F and G, as the entries' size increased, the average processing time became slower. However, after creating an index, note that the average processing time for Task G was much faster as the size of entries became large. For example, for 1 million entries, the average processing time for Task F was 271.71875 ms, whereas the average processing time for Task G was 14.21875 ms (significantly faster) This resulted in more efficient query performance. The cost of the space for the index was justified since the query performance increased significantly, especially for larger entries.

PART 3

Task I (no index):

Cardinality of Table Parts	**Average Processing time for no-index Q4 (ms)**
100	0.78114 ms
1000	1.15340 ms
10,000	2.97647 ms
100,000	16.50074 ms
1,000,000	136.11584 ms

Task J:

Define an index that you believe will optimize Q4 and explain why you think so.
CREATE INDEX idxPartPriceMadeIn ON Parts (madeIn, partPrice); This index optimizes Q4 since it is covering (includes all columns used in the query) and uses the correct sort key, madeIn should be primary since we need to first filter based on the country code and then filter on part price to find the max part price for the given country.

Task K (using index):

Cardinality of Table Parts	**Average Processing time for indexed Q4 (ms)**

100	0.90710 ms
1000	0.93213 ms
10,000	0.99707 ms
100,000	2.24802 ms
1,000,000	13.08205 ms

Task L:

Compare, contrast and explain the trends observed in Task K to the trends observed in Task I. Discuss the cost-benefit of the index space cost and query performance.
For both Task I and K, as the size of the entries increased, the average processing time became slower. However, after creating an index, the average processing time for Task K remained consistent up to 10K. It remained around ten times faster than Task I's execution times for entries of size 100k up to 1 million. The space for the index had minimal effect on the size of the database. Therefore the space cost was justified since the query performance increased speed tenfold for larger entries.

PART 4

Task M (no index):

Cardinality of Table Parts	**Average Processing time for index-free Q5 (ms)**
100	1.75373 ms
1000	70.67408 ms
10,000	6582.72218 ms
100,000	N/A
1,000,000	N/A

Task N (no index):

Cardinality of Table Parts	**Average Processing time for index-free Q6 (ms)**
100	0.78408 ms

1000	1.96144 ms
10,000	12.52025 ms
100,000	127.08096 ms
1,000,000	2944.90614 ms

Task O:

Compare, contrast and explain the trends observed in Task M to the trends observed in Task N
The average time to execute the query grows faster in Task M than in Task N. The reason for the difference in execution time is that Query 5 uses a correlated subquery, whereas Query 6 does not use a correlated subquery. Query 6 selects all the needed parts once and uses "IN" to check if the part number exists in that list, whereas Query 5 compares the part number for the row to all the parts that need that row, for all parts in Parts.

Task P:

Define an index that you believe will optimize Q6 and explain why you think so
CREATE INDEX idxPartNumberNeedsPart on Parts (needsPart, partNumber);   This index optimizes Q6 since it is covering (includes all columns used in the query) and uses the correct sort key, needsPart should be the first sort key since the "NOT IN" needs to compare the row partNumber to the needsPart list to see if it exists.

Task Q (with index):

Cardinality of Table Parts	**Average Processing time for indexed Q6 (ms)**
100	0.59061 ms
1000	1.40174 ms
10,000	5.20128 ms
100,000	56.00160 ms
1,000,000	2160.09378 ms

Task R:

Compare, contrast and explain the trends observed in Task N to the trends observed in Task Q. Discuss the cost-benefit of the index space cost and query performance.
Task Q's average execution times are on average faster than the times from Task N. However, the cost of the index space is significant compared to the advantage in query performance, especially for larger numbers of entries. Therefore cost and benefit of the index cancel out evenly as it would be up to the database admin to decide whether the increased query performance is worth the cost of the increase in the size of the database. Not to mention an index would also negatively affect read speeds as well.