**TEMPLATE FOR ASSIGNMENT #4 REPORT**

**Student Name and CCID:**

1. \_\_\_\_\_\_\_Ahmad Amin\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Name) \_\_\_\_\_aamin1\_\_\_\_ (CCID)
2. \_\_\_\_\_\_\_\_\_\_Chris Wen\_\_\_\_\_\_\_\_\_\_\_\_ (Name) \_\_\_\_\_chris2\_\_\_\_ (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 (1) (no index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index free Q1 (ms)** |
| 100 | 0.03997802734375 ms on average |
| 1000 | 0.039963722229003906 ms on average |
| 10,000 | 0.060062408447265625 ms on average |
| 100,000 | 0.09930133819580078 ms on average |
| 1,000,000 | 0.19540786743164062 ms on average |

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index free Q2 (ms)** |
| 100 | 0.08081436157226562 ms on average |
| 1000 | 0.0798797607421875 ms on average |
| 10,000 | 0.279998779296875 ms on average |
| 100,000 | 2.2298574447631836 ms on average |
| 1,000,000 | 15.000104904174805 ms on average |

**Task B (2):**

|  |
| --- |
| **Compare, contrast and explain the trends observable in both tables above (Task A)** |
| Upwards trend as table cardinality increases which is expected as there are more rows to check against.  Query 1 is the one of the simplest SQL queries possible, checking for rows that have a certain primary key. Query 2 has the potential to return multiple values, so it makes sense that it has a much longer execution time for larger table cardinalities than Query1. |

**Task C (3) (using index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q1 (ms)** |
| 100 | 0.061707496643066406 ms on average |
| 1000 | 0.070343017578125 ms on average |
| 10,000 | 0.12003898620605469 ms on average |
| 100,000 | 0.0995635986328125 ms on average |
| 1,000,000 | 0.20074844360351562 ms on average |

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q2 (ms)** |
| 100 | 0.06004810333251953 ms on average |
| 1000 | 0.07236003875732422 ms on average |
| 10,000 | 0.17885684967041016 ms on average |
| 100,000 | 0.10018348693847656 ms on average |
| 1,000,000 | 0.19998550415039062 ms on average |

**Task D (4):**

|  |
| --- |
| **Compare, contrast and explain the trends observable in both tables above (Task C)** |
| The values are quite similar in both tables above, with a general upward trend with some minor irregularities that can be explained through cache or OS-availability reasons. The increase in processing time however is very minimal as the size of the table increases. |

**Task E (5):**

|  |
| --- |
| **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.** |
| In Task B the increasing in processing time for Query 2 is exponential, whereas in Task D the increase in processing time for Query 2 is very minimal. This is due to the usage of an index. Although the index may take some space, the increase in query performance significantly outweighs that cost for Query 2. In Query 1, the query run without indexing is faster, and this is likely due to Query 1 being a very simple query which only searches for a primary key. In this case, the value of having an index is largely diminished. This shows that the more complex the query is, the more value indexing can have in terms of optimizing query performance vs index space cost. |

**PART 2**

**Task F (6) (no index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index-free Q3 (ms)** |
| 100 | 0.060830116271972656 ms on average |
| 1000 | 0.38018226623535156 ms on average |
| 10,000 | 3.853158950805664 ms on average |
| 100,000 | 52.478981018066406 ms on average |
| 1,000,000 | 468.9213752746582 ms on average |

**Task G (7) (using index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q3 (ms)** |
| 100 | 0.039887428283691406 ms on average |
| 1000 | 0.0599980354309082 ms on average |
| 10,000 | 0.09023904800415039 ms on average |
| 100,000 | 0.29296398162841797 ms on average |
| 1,000,000 | 15.404446125030518 ms on average |

**Task H (8):**

|  |
| --- |
| **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 this query, we can see that without the usage of index the processing time gets significantly exponentially larger. However, when an index is used, while the processing time does increase exponentially, the increase is not as significant. This is likely due to Query 3 being a very complex query. This shows that when using complex queries, indexing can increase query performance very significantly, and the query performance increase greatly outweighs the space cost of the index at large table sizes. |

**PART 3**

**Task I (9) (no index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for no-index Q4 (ms)** |
| 100 | 0.03998756408691406 ms on average |
| 1000 | 0.14000892639160156 ms on average |
| 10,000 | 0.6001615524291992 ms on average |
| 100,000 | 7.63401985168457 ms on average |
| 1,000,000 | 82.5016975402832 ms on average |

**Task J (10):**

|  |
| --- |
| **Define an index that you believe will optimize Q4 and explain why you think so.** |
| "CREATE INDEX idxQ4Optimize ON Parts (partPrice, madeIn)"  I believe this index will optimize the query since these two columns are the only columns that are accessed from the table. By indexing these two columns specifically, we can avoid having to access any unnecessary information, and by doing so optimize the processing time of Query 4. |

**Task K (11) (using index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q4 (ms)** |
| 100 | 0.056891441345214844 ms on average |
| 1000 | 0.09922504425048828 ms on average |
| 10,000 | 0.11074066162109375 ms on average |
| 100,000 | 0.10683536529541016 ms on average |
| 1,000,000 | 0.6117820739746094 ms on average |

**Task L (12):**

|  |
| --- |
| **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.** |
| As expected, the query performance significantly increases when an index is used. This is because the query is more complex and accesses many rows in the table. However, since we only need 2 specific columns from each row, we can further optimize the query by indexing those 2 specific columns. In this case I believe the increase in query performance greatly outweighs the index space cost. However, it is important to note that the larger the table size the greater the benefit as well. |

**PART 4**

**Task M (13) (no index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index-free Q5 (ms)** |
| 100 | 0.0600433349609375 ms on average |
| 1000 | 0.13998031616210938 ms on average |
| 10,000 | 0.7808732986450195 ms on average |
| 100,000 | NA |
| 1,000,000 | NA |

**Task N (14) (no index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index-free Q6 (ms)** |
| 100 | 0.2009725570678711 ms on average |
| 1000 | 1.8228483200073242 ms on average |
| 10,000 | 17.976484298706055 ms on average |
| 100,000 | 171.92165851593018 ms on average |
| 1,000,000 | 1614.9428844451904 ms on average |

**Task O (15):**

|  |
| --- |
| **Compare, contrast and explain the trends observed in Task M to the trends observed in Task N** |
| In both queries, the processing times increase exponentially, but Query 6 increases at a significantly higher rate. This implies that the usage of NOT EXISTS can be faster than the usage of NOT IN. |

**Task P (16):**

|  |
| --- |
| **Define an index that you believe will optimize Q6 and explain why you think so** |
| "CREATE INDEX idxQ6Optimize ON Parts (needsPart, partNumber);"  I believe that this index will optimize Query 6 since the only columns used in this query are needsPart and partNumber. By indexing these two columns specifically, we can avoid accessing any unnecessary information from the other columns. |

**Task Q (17) (with index):**

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q6 (ms)** |
| 100 | 0.24481296539306638 ms on average |
| 1000 | 0.6799793243408203 ms on average |
| 10,000 | 3.8686227798461914 ms on average |
| 100,000 | 41.22483730316162 ms on average |
| 1,000,000 | 399.3643283843994 ms on average |

**Task R (18):**

|  |
| --- |
| **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.** |
| We can see that the usage of an index for Query 6 greatly improves the processing time. This is likely because the query is not accessing any irrelevant data with each selection. In this case the increase in query performance greatly outweighs the cost of the index space, since the processing time is reduced by a factor of around 10 for the larger table sizes. |