**TEMPLATE FOR ASSIGNMENT #4 REPORT**

**Student Name and CCID:**

1. Sidharth Srinivasan\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Name) sr11\_\_\_\_\_\_ (CCID)
2. James Schaefer-Pham\_\_\_\_\_\_\_\_\_\_\_\_\_(Name) jschaefe \_\_\_(CCID)
3. Adit Rada\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Name) rada\_\_\_\_\_\_\_ (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 | 0.262ms (0.0002628850936889648) |
| 1000 | 0.360ms (0.00035992860794067385) |
| 10,000 | 0.356ms (0.00033558368682861327) |
| 100,000 | 0.429ms (0.0004287815093994141) |
| 1,000,000 | 0.556ms (0.0005563831329345703) |

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index free Q2 (ms)** |
| 100 | 0.368ms (0.00036847829818725586) |
| 1000 | 0.370ms (0.0003700733184814453) |
| 10,000 | 4.45ms (0.004479818344116211) |
| 100,000 | 39.1ms (0.039133446216583254) |
| 1,000,000 | 379.1ms (0.3791451859474182) |

**Task B (2):**

|  |
| --- |
| **Compare, contrast and explain the trends observable in both tables above (Task A)** |
| The average time increases as the size of the database increases for both queries, this is consistent with the fact that as the no. of records increases, the query needs to iterate over more tuples and hence longer time.  The average time for executing query 2 is higher that that for query 1 for the corresponding instance of the database. This is as since needsPart is not the unique (unlike partNumber which was the primary key), the query must process over the entire database to find all all the records with that needsPart no.; however, query 1 could stop iterating over the database once the record is found hence it would scan, on average, about n/2 records; while query 2 scans all n records. |

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q1 (ms)** |
| 100 | 0.110ms (0.00011001348495483399) |
| 1000 | 0.112ms (0.00011253118515014649) |
| 10,000 | 0.191ms (0.00019063711166381837) |
| 100,000 | 0.190ms (0.00018982887268066407) |
| 1,000,000 | 0.291ms (0.00029119968414306643) |

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q2 (ms)** |
| 100 | 0.0898ms (0.00008979082107543946) |
| 1000 | 0.134ms (0.0001339411735534668) |
| 10,000 | 0.172ms (0.00017249822616577148) |
| 100,000 | 0.180ms (0.00017951250076293944) |
| 1,000,000 | 0.242ms (0.00024185895919799803) |

**Task D (4):**

|  |
| --- |
| **Compare, contrast and explain the trends observable in both tables above (Task C)** |
| The average time increases as the size of the database increases for both queries.  The times for both queries on corresponding instances of the databases is about the same despite the fact that query 2 should actually take longer as it scans all tuples in the relation whereas query 1 scans about n/2 tuples. The fact that both have similar times suggests that the index helps query 2 more than it helps query 1. This makes sense as the index is on needsPart and we are doing an equality search on needsPart in query 2; however, we are doing equality search on partNumber in query 1, so index unlikely to help. |

**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.** |
| Space before and after:  A4v100: 8kb -> 12kb  A4v1k: 36kb -> 60kb  A4v10k: 248kb -> 428kb  A4v100k: 2464kb -> 4224kb  A4v1M: 25,484kb -> 43,056kb  The average times are less with the index for both queries, though the times are only slightly less for query 1 but are significantly less for query 2.  For e.g., for query 2 on Av1M the average time reduces from 379.1ms to 0.242ms(Almost 100% reduction in time). Though we must note that the space needed, on average, for all the database instances increases by about 1.7 times.  We can conclude, that the benefits of an index are not worth the space costs for query 1 but are more than worth it (due to the enormous time savings) for query 2. |

**PART 2**

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index-free Q3 (ms)** |
| 100 | 0.310ms (0.0003096318244934082) |
| 1000 | 1.510ms (0.0015097546577453613) |
| 10,000 | 9.825ms (0.009825007915496826) |
| 100,000 | 115.1ms (0.11507606744766236) |
| 1,000,000 | 1611.7ms (1.6116997051239013) |

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q3 (ms)** |
| 100 | 0.289ms (0.00028920888900756835) |
| 1000 | 1.418ms (0.0014175820350646974) |
| 10,000 | 9.462ms (0.009461996555328369) |
| 100,000 | 250.0ms (0.25021379232406615) |
| 1,000,000 | 7643.7ms (7.6436997051239013) |

**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 both tests, as the size of the database increases, the average time increases.  It seems like the index does not make any difference in terms of time. For e.g., for A4v100k the time increased from 115ms to 250ms with the index.  So, the index is actually making the performance worse (at least on my machine).  Thus, we can say that the index is probably not worth the increased space costs. One hypothesis for the time not changing is that the time taken to iterate over all rows when calculating the average dominates the total time and using an index does not speed up this process anyway. So, looking up in the index and then iterating over all the rows is more time consuming than iterating over all the rows. |

**PART 3**

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for no-index Q4 (ms)** |
| 100 | 0.07976531982421875 |
| 1000 | 0.1404428482055664 |
| 10,000 | 0.8568572998046875 |
| 100,000 | 8.481192588806152 |
| 1,000,000 | 77.7010440826416 |

**Task J (10):**

|  |
| --- |
| **Define an index that you believe will optimize Q4 and explain why you think so.** |
| We used the command **CREATE INDEX MaxCost On Parts (madeIn, partPrice)** to make or index. |

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q4 (ms)** |
| 100 | 0.03988742828369141 |
| 1000 | 0.040040016174316406 |
| 10,000 | 0.06110668182373047 |
| 100,000 | 0.10442733764648438 |
| 1,000,000 | 0.19931793212890625 |

**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.** |
|  |

**PART 4**

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index-free Q5 (ms)** |
| 100 | 0.0012758779525756836 |
| 1000 | 0.07399272918701172 |
| 10,000 | 7.752774424552918 |
| 100,000 | N.A. |
| 1,000,000 | N.A. |

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for index-free Q6 (ms)** |
| 100 | 0.0002506828308105469 |
| 1000 | 0.0008909940719604492 |
| 10,000 | 0.01144637107849121 |
| 100,000 | 0.1694697141647339 |
| 1,000,000 | 5.3444664478302 |

**Task O (15):**

|  |
| --- |
| **Compare, contrast and explain the trends observed in Task M to the trends observed in Task N** |
| We can clearly see from the details from the two tasks that Query 6 is much faster than Query 5. On trying to run Query 5 on 100,000 and 1,000,000 Cardinality, we will notice that the time runs on O(n2). Every single run takes about 10 minutes for 100k in case of query 5 and hence had to be avoided.  On the other hand, Query 6 is definitely much faster in implementation, with an average of just approximately 5.3 seconds for 1M cardinality, in contrast to the 7.75 for just 10k cardinality |

**Task P (16):**

|  |
| --- |
| **Define an index that you believe will optimize Q6 and explain why you think so** |
| We used the command **CREATE INDEX idxneedParts On Parts (needParts)** to make our index. |

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

|  |  |
| --- | --- |
| **Cardinality of Table Parts** | **Average Processing time for indexed Q6 (ms)** |
| 100 | 8.619785308837891e-05 |
| 1000 | 0.0014877700805664062 |
| 10,000 | 0.006729097366333008 |
| 100,000 | 0.05674569606781006 |
| 1,000,000 | 2.817062759399414 |

**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 are able to notice that the use of an index has drastically reduced performance time. The time taken for operations on 1M Cardinality now took 2.5s lesser which is a major improvement.  We can easily say that the cost-benefit is much higher than the space cost of the index. This can be attributed to the ease in iterating over the indices as compared to everything |