**Read-heavy Workloads (SELECT queries):** If your application is primarily read-heavy, adding these indices will generally result in improved performance as long as you’re querying the indexed columns. Indices will provide faster data retrieval for queries that filter or join on these indexed columns.

balance between optimizing read performance and avoiding excessive overhead on write operations.

How tests are performed:

30 common queries are presented. All of these queries are tested on the database populated with the same data on indexed and not indexed versions. The results reveal that most of the queries are positively affected by the indices. However there were certain occurrences, where the created overhead was seen to be more costly, hence not every query’s runtime increased.

The python file “performance\_test.py”, runs all of the common queries one by one on the database, and records each query’s runtime in a file. This file is run 2 times, once with the database initialization script with no indices and once with the database script with indices. Each individual query result is presented in “query\_times\_no\_indices.txt” and “query\_times\_with\_indices.txt”.

Then another python script “performance\_test\_print.py”, calculates the time improvement between each query in the indiced and non- indiced format. These results are held in the file “time\_imporvement\_with\_indices.txt”. In this file, negative percentage increase’s correlate that the query was indeed ran faster. First 5 queries are also presented below in order to better showcase the logic: