For query optimization we needed to research on advanced techniques of indexing.

Here is how we plan to improve the performance of queries.

**We will be creating 2 indexes**:

1. The first index will be an ascending single field index on the field “type”. This is a very efficient optimization as we analyzed the queries and found that the number of documents scanned for the query reduced significantly after creating the index on the “type” field. Attached below is a screenshot.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

Here is how the query performs without indexing

Graphical user interface, application, Teams

Description automatically generated

The whole table had to be scanned just to return the results of type “INFJ”.

1. The second index will be a text-based index(Compund). A text-based index helps perform efficient queries on fields containing strings. In our case we will be optimizing search queries on posts using text-based indexing on the field.

Attached is a screenshot

Graphical user interface, text, application, email

Description automatically generated

“mbti\_posts\_index” is an index on the field “posts”. Let’s analyze the efficiency of the index by executing the following query

db.mbti\_example.find( { $text: { $search:"youtube.com"}}).explain("executionStats");

So, this query explains the execution stats of running a search query with posts containing the phrase “youtube.com”.

Here is a screenshot of the execution stats:

A picture containing table

Description automatically generated

Total docs examined is way lower than the number of documents in the collection. The collection has 8675 documents. So, applying indexing helped us a lot to perform optimized queries on the database.