S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 8

Aim: To study and implement Indexing in MongoDB using

1. Single Field Index
2. Compound Index
3. Unique Index
4. Text Index

and analyze query performance using. explain(“executionStats”).

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: 7th / 4th

Academic Session: 2025-2026

Date of Performance: __________

Date of Submission: __________

AIM: To study and implement Indexing in MongoDB using

1. Single Field Index
2. Compound Index
3. Unique Index
4. Text Index

and analyze query performance using .explain("executionStats").

.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- A good indexing strategy is crucial to ensuring that your MongoDB database returns your results in the most efficient way possible
- An index measures the price performance of a basket of securities using a standardized metric and methodology.

Indexes are an essential feature of databases that can significantly improve query performance.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement: High Configuration computer

Software Requirement: MongoDB-7.0, Mongo Shell

THEORY:

Introduction: MongoDB indexes are crucial in optimizing database performance, as they improve the speed of data retrieval operations in a MongoDB database. They create a reference to the location of data within a collection based on the indexed field(s). When a query involves the indexed field, MongoDB can use the index to rapidly pinpoint the relevant documents, resulting in faster query execution times. By expediting data retrieval through efficient query execution, MongoDB indexes enhance the performance of your database.

MongoDB Indexes Importance: -

MongoDB indexes play a pivotal role in enhancing query performance and improving overall database efficiency. They enable MongoDB to retrieve data by providing a roadmap for the database to locate specific data quickly. When a query involves the indexed fields, MongoDB can use the index to navigate directly to the relevant documents, bypassing the need for a full collection scan. This process is akin to using an index in a book to find specific information instead of reading the entire book.

Especially helpful in collections with a large number of documents, indexes are instrumental in maintaining acceptable query performance. Without indexes, querying such collections could become prohibitively slow due to the need to scan through a massive amount of data. Indexes not only speed up queries but also minimize resource usage. With indexes, MongoDB performs fewer disk I/O operations and consumes less CPU, leading to improved server performance and reduced response times. By creating organized data structures that enable rapid data retrieval and efficient query execution, indexes empower MongoDB to locate and retrieve data that matches specific query conditions quickly. This capability is essential for achieving responsive applications, optimizing user experiences, and supporting applications dealing with significant data loads.

Use Indexes in MongoDB:-

Accelerating Queries: Indexes dramatically enhance query execution for frequently accessed fields, leading to quicker data retrieval and overall application responsiveness.

Sorting and Aggregation: Indexes expedite sorting and aggregation tasks, enabling efficient data manipulation for reports, analytics, and visualization.

Range-based Queries: Indexes bolster range-based queries, essential for retrieving data within specific value intervals, such as time ranges or numeric sequences.

Join Operations: In cases of data linking through multiple collections, indexes improve join efficiency, consolidating data from various sources promptly.

Text Searches: For applications requiring full-text search capabilities, text indexes accelerate searching within text fields, facilitating efficient content searches.

Geospatial Data: Indexes are vital for geospatial applications, where rapid location-based queries and analyses depend on efficient spatial indexes.

Data Deduplication: Indexes expedite processes involving duplicate detection and data cleanup, enhancing data quality and consistency.

Types of MongoDB Indexes:-

MongoDB offers various indexing options, each designed to optimize specific data access patterns and query scenarios. Understanding the different types of indexes is essential for harnessing MongoDB’s full potential in terms of query performance, data retrieval, and overall database efficiency. From simple single-field indexes to compound, text, and geospatial indexes, each type plays a crucial role in enhancing the speed and accuracy of data access.

Single Field Indexes

Single field indexes in MongoDB are a fundamental indexing strategy that significantly impacts query performance by expediting data retrieval based on a specific field. These indexes efficiently handle queries involving the indexed field, notably minimizing the necessity for full collection scans. They are especially advantageous for tasks such as filtering, sorting, or searching data based on a single attribute. For example, when queries frequently focus on unique user identifications or timestamps, a single field index can profoundly enhance query execution, ensuring prompt and responsive data access.

Compound Indexes

Compound indexes in MongoDB play a crucial role in optimizing queries that involve multiple fields. These indexes encompass more than one field and allow for enhanced query performance by facilitating the retrieval of data based on various attributes simultaneously. The order of fields in a compound index holds immense importance as it directly affects its efficiency in addressing specific query patterns. This order influences how the index is used; queries that match the order of fields in the index benefit the most. Careful consideration of the order in which fields are placed within a compound index can significantly impact query efficiency and overall database performance.

Multikey Indexes

Multikey indexes in MongoDB are indexing tools that involve arrays and nested documents and are particularly useful when dealing with fields containing arrays or documents that hold multiple values. A multikey index generates separate index entries for each value within an array or nested document, enabling efficient querying and filtering based on these nested values. While multikey indexes offer benefits like enhanced query performance for array-based searches, it’s crucial to consider potential downsides, such as increased index size and write operations’ impact.

Text Indexes

In MongoDB, these enable full-text search functionality within the database, making it easier to search for and retrieve relevant information from large collections of text data. Text indexes are particularly beneficial when natural language processing and text-based analysis are essential. Uses include content management systems, social media platforms, and e-commerce sites, where users need to locate specific textual content within datasets.

Geospatial Indexes

Geospatial indexes in MongoDB are designed to facilitate the efficient retrieval of location-based information, like proximity searches, polygon intersections, and distance-based analyses. They are particularly valuable in applications that involve geographic data, such as mapping and navigation systems, location-based services, and real estate platforms, providing responsive and accurate information and enhancing user experiences.

Hashed Indexes

Hashed indexes in MongoDB are specialized indexing tools for distributing data uniformly across a collection. Unlike other index types, hashed indexes use a hashing algorithm to map field values to index keys, ensuring even distribution. This approach is particularly beneficial for scenarios where write-intensive workloads are prevalent. In such cases, hashed indexes can mitigate issues related to index contention and hotspots, distributing write operations across the index evenly. Hashed indexes excel when the primary concern is maintaining consistent performance in high-write environments, such as in applications with rapidly changing data.

SYNTAX –

The createIndex() Method

```
db.COLLECTION_NAME.createIndex({KEY:1})
```

The key determines the field on the basis of which you want to create an index and 1 (or -1) determines the order in which these indexes will be arranged(ascending or descending).

The dropIndex() method

In order to drop an index, MongoDB provides the dropIndex() method.

```
db.NAME_OF_COLLECTION.dropIndex({KEY:1})
```

The dropIndexes() method

```
>db.COLLECTION_NAME.dropIndexes()
```

The getIndexes() method

```
db.NAME_OF_COLLECTION.getIndexes()
```

It will retrieve all the description of the indexes created within the collection.

Limitations And Challenges Of Mongodb Indexes:-

While MongoDB indexes offer significant benefits, they also come with certain limitations and challenges:

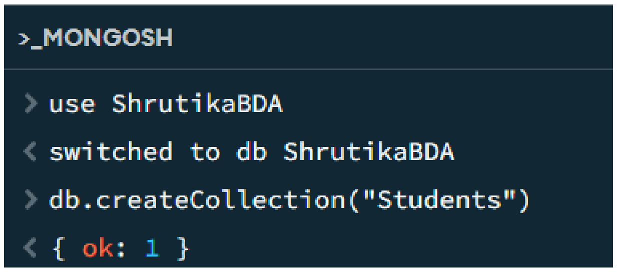
Index Size: Many indexes can lead to large storage requirements.

Write Operations: Additional indexes slow down write operations.

Query Planning: Poorly chosen indexes might lead to inefficient query execution plans.

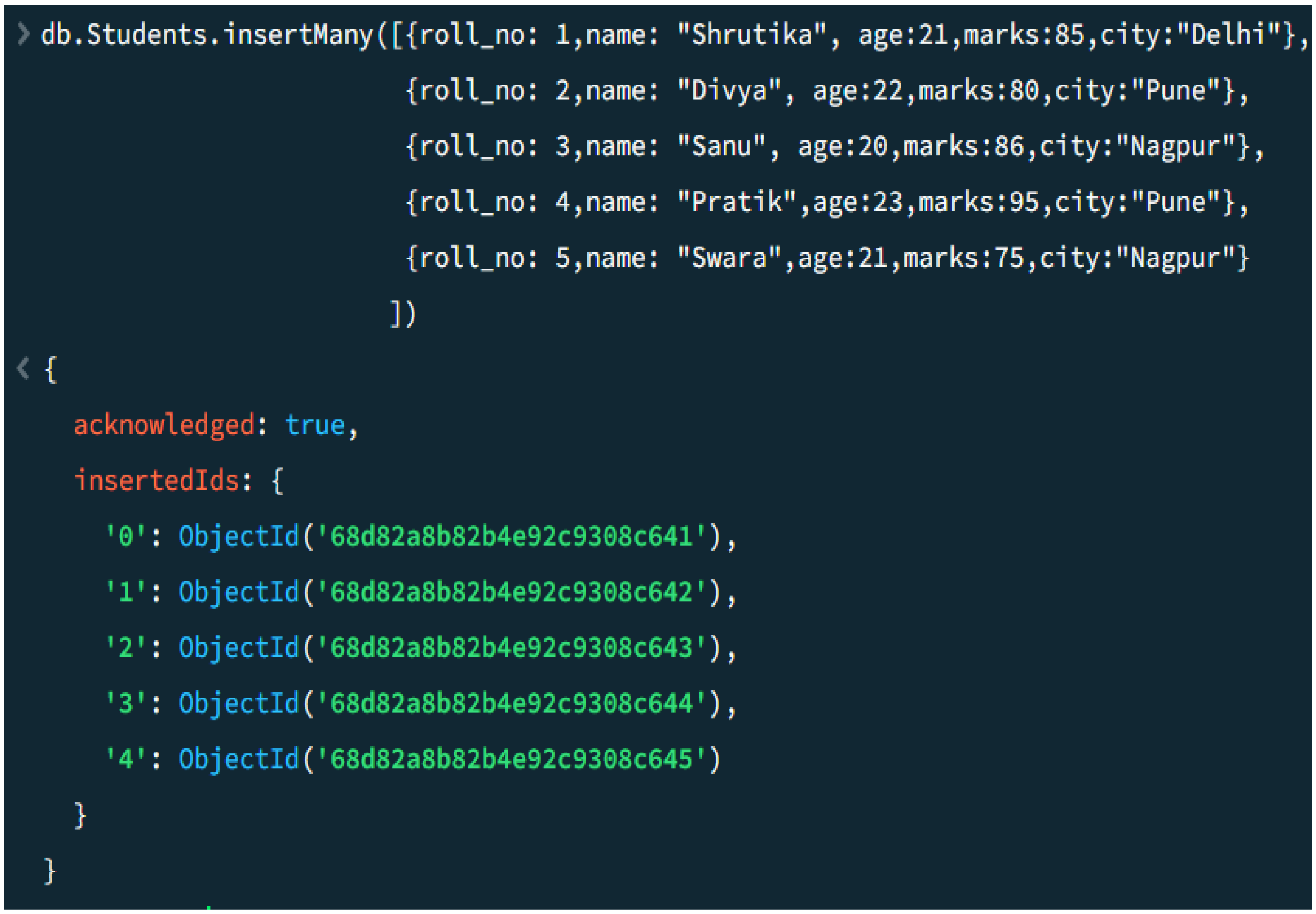
Index Updates: Frequent updates can lead to index fragmentation, impacting performance.

INPUT / OUTPUT (SCREENSHOTS):



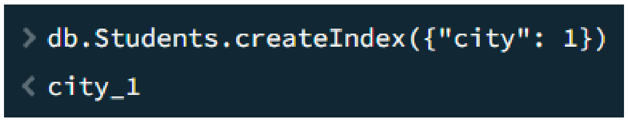
```
>_MONGOSH
> use ShrutikaBDA
< switched to db ShrutikaBDA
> db.createCollection("Students")
< { ok: 1 }
```

.

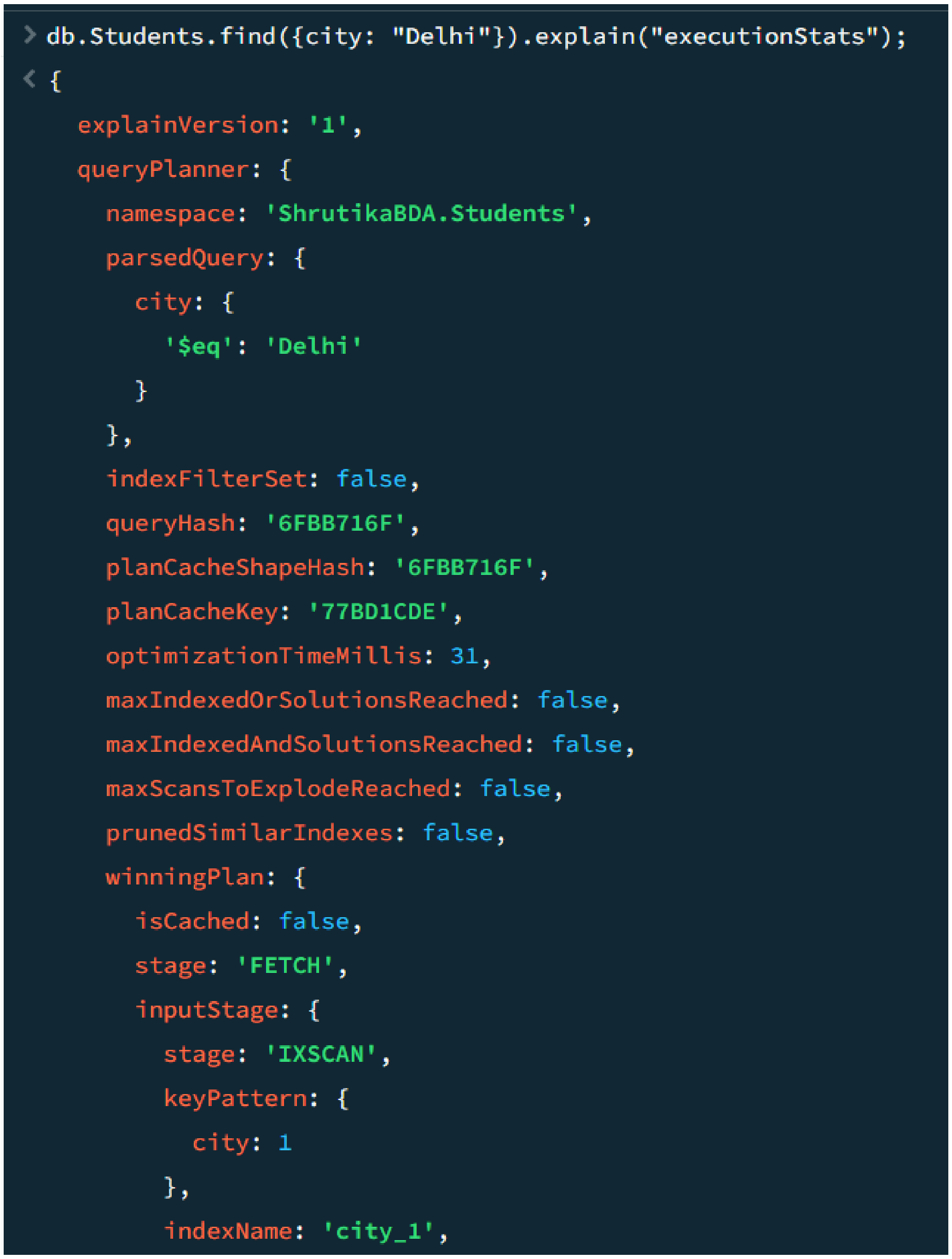


```
> db.Students.insertMany([{roll_no: 1,name: "Shrutika", age:21,marks:85,city:"Delhi"},
                         {roll_no: 2,name: "Divya", age:22,marks:80,city:"Pune"},
                         {roll_no: 3,name: "Sanu", age:20,marks:86,city:"Nagpur"},
                         {roll_no: 4,name: "Pratik",age:23,marks:95,city:"Pune"},
                         {roll_no: 5,name: "Swara",age:21,marks:75,city:"Nagpur"}
                        ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68d82a8b82b4e92c9308c641'),
      '1': ObjectId('68d82a8b82b4e92c9308c642'),
      '2': ObjectId('68d82a8b82b4e92c9308c643'),
      '3': ObjectId('68d82a8b82b4e92c9308c644'),
      '4': ObjectId('68d82a8b82b4e92c9308c645')
    }
  }
```

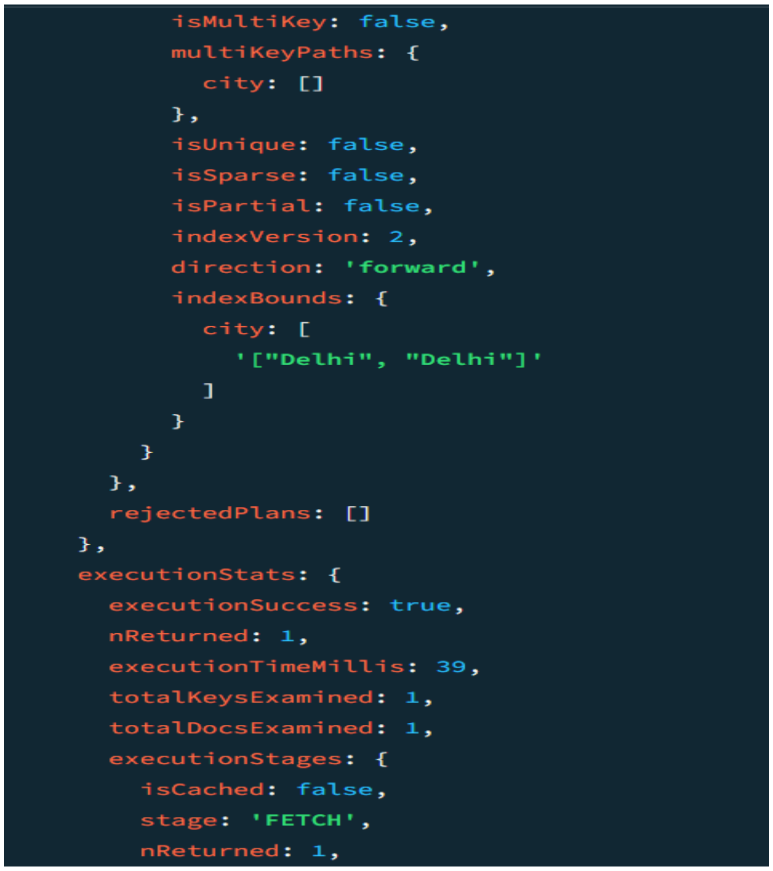
Single Field Index



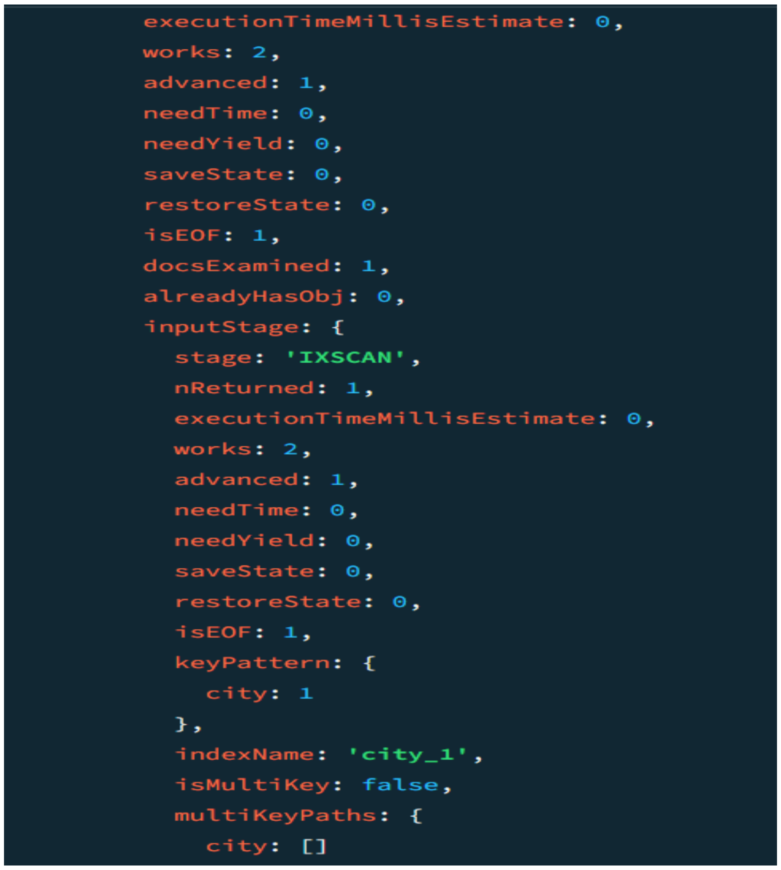
```
> db.Students.createIndex({"city": 1})
< city_1
```



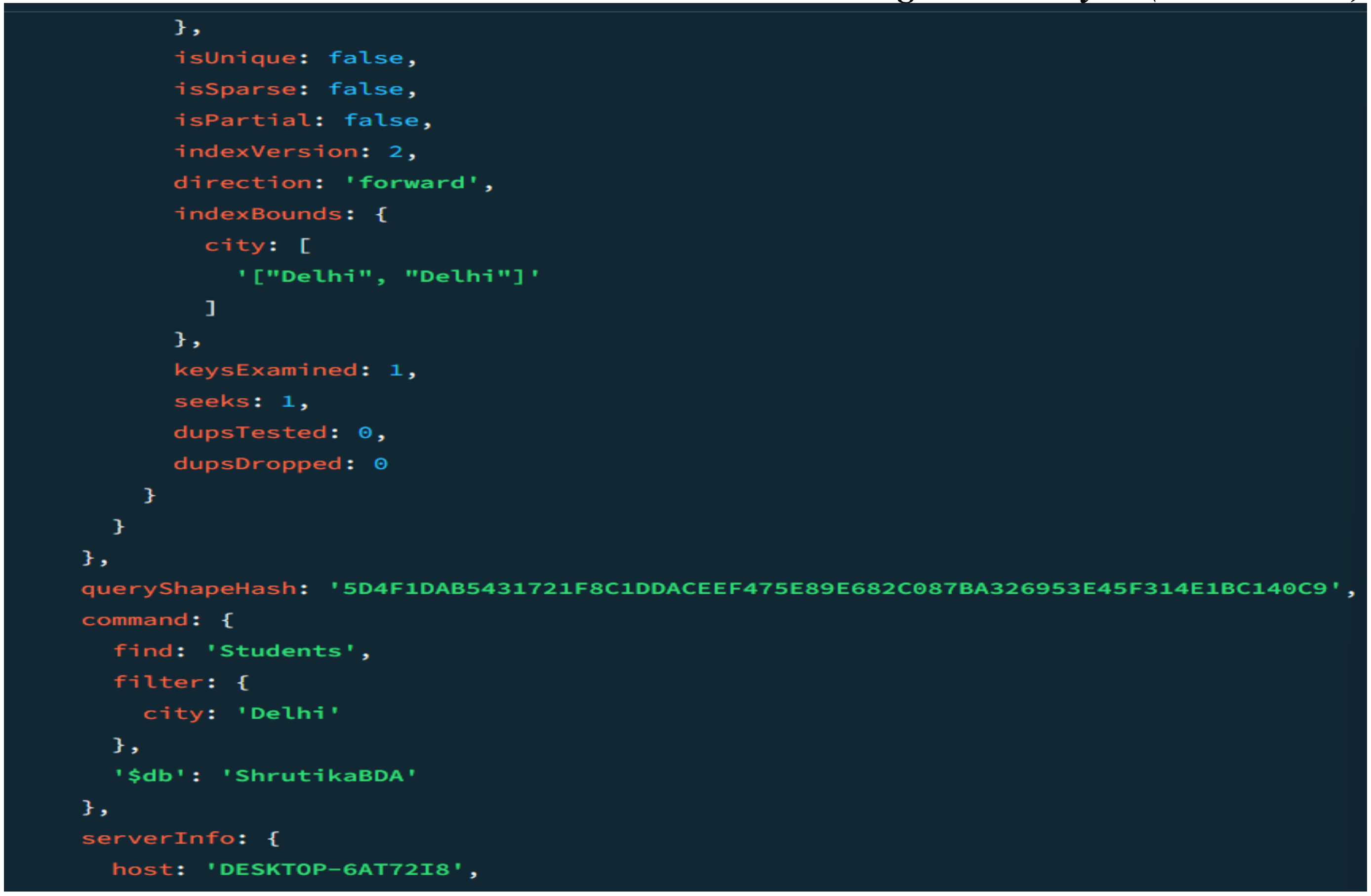
```
> db.Students.find({city: "Delhi"}).explain("executionStats");
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'ShrutikaBDA.Students',
      parsedQuery: {
        city: {
          '$eq': 'Delhi'
        }
      },
      indexFilterSet: false,
      queryHash: '6FBB716F',
      planCacheShapeHash: '6FBB716F',
      planCacheKey: '77BD1CDE',
      optimizationTimeMillis: 31,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: {
            city: 1
          },
          indexName: 'city_1',
```



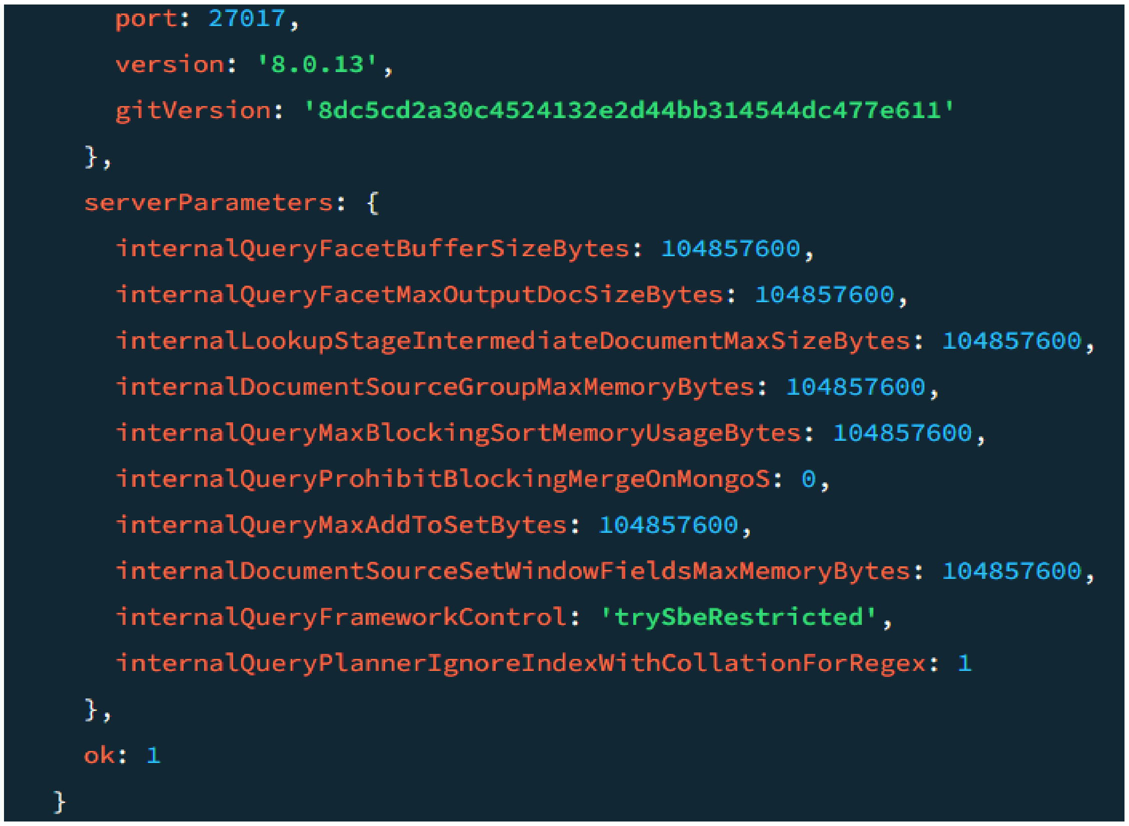
```
          isMultiKey: false,
          multiKeyPaths: {
            city: []
          },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: {
            city: [
              '["Delhi", "Delhi"]'
            ]
          }
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 1,
      executionTimeMillis: 39,
      totalKeysExamined: 1,
      totalDocsExamined: 1,
      executionStages: {
        isCached: false,
        stage: 'FETCH',
        nReturned: 1,
```



```
        executionTimeMillisEstimate: 0,
        works: 2,
        advanced: 1,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1,
        docsExamined: 1,
        alreadyHasObj: 0,
        inputStage: {
          stage: 'IXSCAN',
          nReturned: 1,
          executionTimeMillisEstimate: 0,
          works: 2,
          advanced: 1,
          needTime: 0,
          needYield: 0,
          saveState: 0,
          restoreState: 0,
          isEOF: 1,
          keyPattern: {
            city: 1
          },
          indexName: 'city_1',
          isMultiKey: false,
          multiKeyPaths: {
            city: []
```

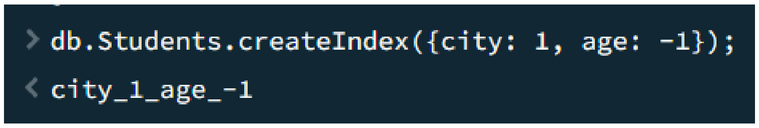


```
          },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: {
            city: [
              '["Delhi", "Delhi"]'
            ]
          },
          keysExamined: 1,
          seeks: 1,
          dupsTested: 0,
          dupsDropped: 0
        }
      }
    },
    queryShapeHash: '5D4F1DAB5431721F8C1DDACEEF475E89E682C087BA326953E45F314E1BC140C9',
    command: {
      find: 'Students',
      filter: {
        city: 'Delhi'
      },
      '$db': 'ShrutikaBDA'
    },
    serverInfo: {
      host: 'DESKTOP-6AT72I8',
```

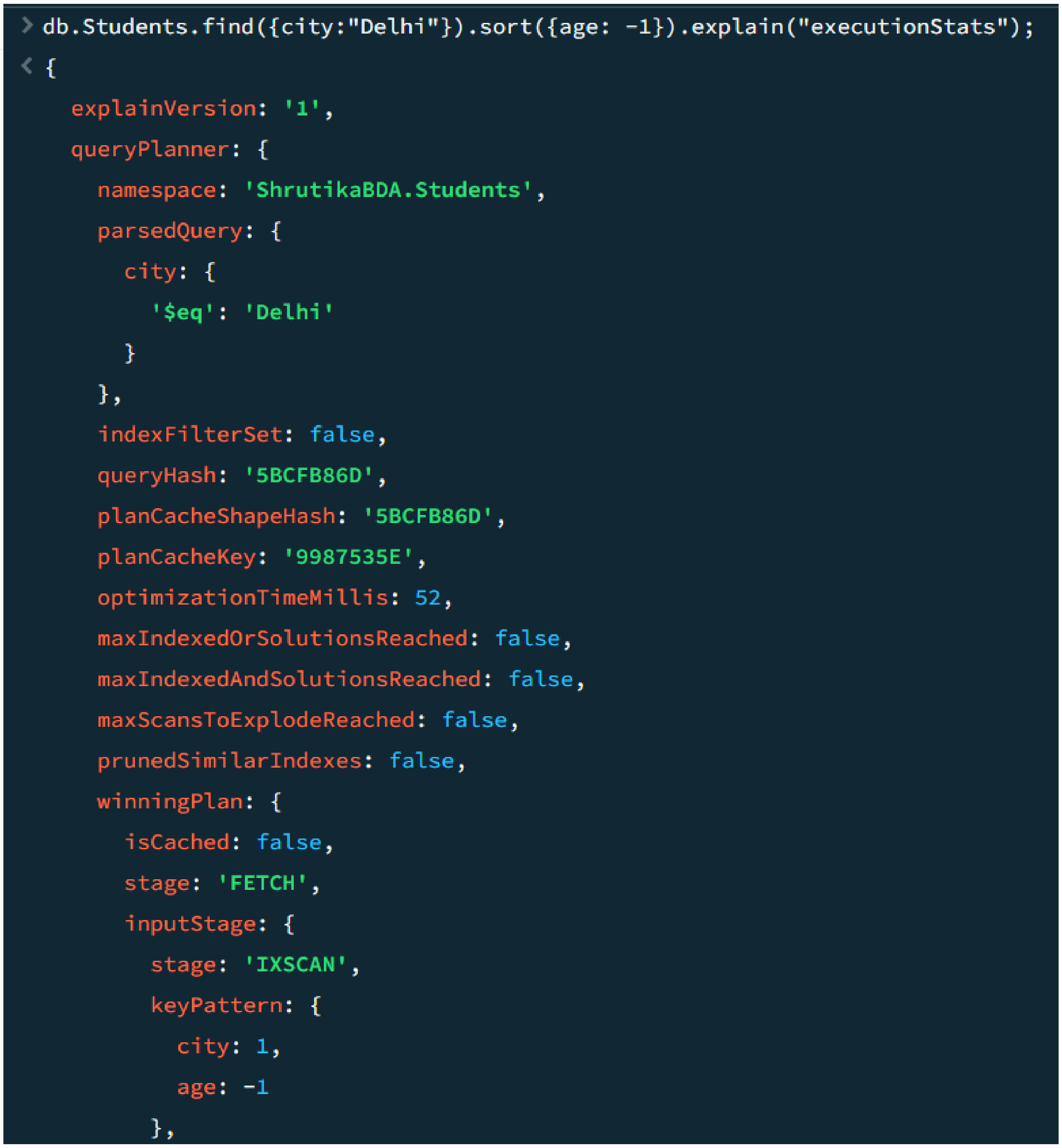


```
      port: 27017,
      version: '8.0.13',
      gitVersion: '8dc5cd2a30c4524132e2d44bb314544dc477e611'
    },
    serverParameters: {
      internalQueryFacetBufferSizeBytes: 104857600,
      internalQueryFacetMaxOutputDocSizeBytes: 104857600,
      internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
      internalDocumentSourceGroupMaxMemoryBytes: 104857600,
      internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
      internalQueryProhibitBlockingMergeOnMongoS: 0,
      internalQueryMaxAddToSetBytes: 104857600,
      internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
      internalQueryFrameworkControl: 'trySbeRestricted',
      internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
    },
    ok: 1
  }
```

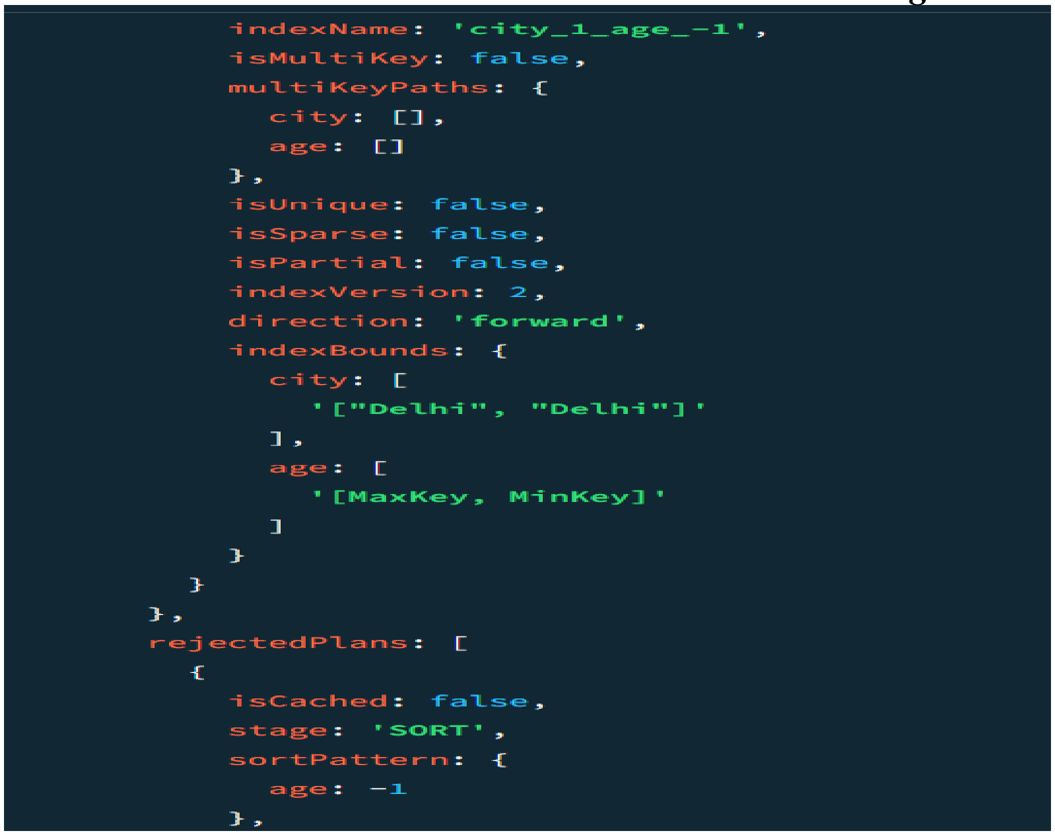
Compound Index



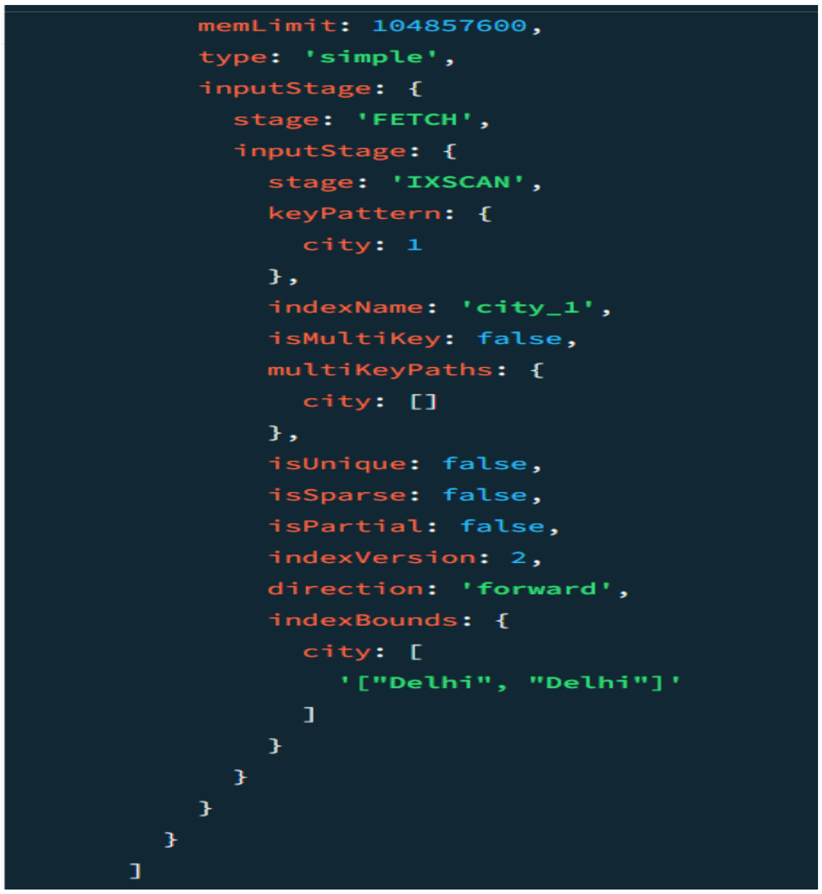
```
> db.Students.createIndex({city: 1, age: -1});
< city_1_age_-1
```



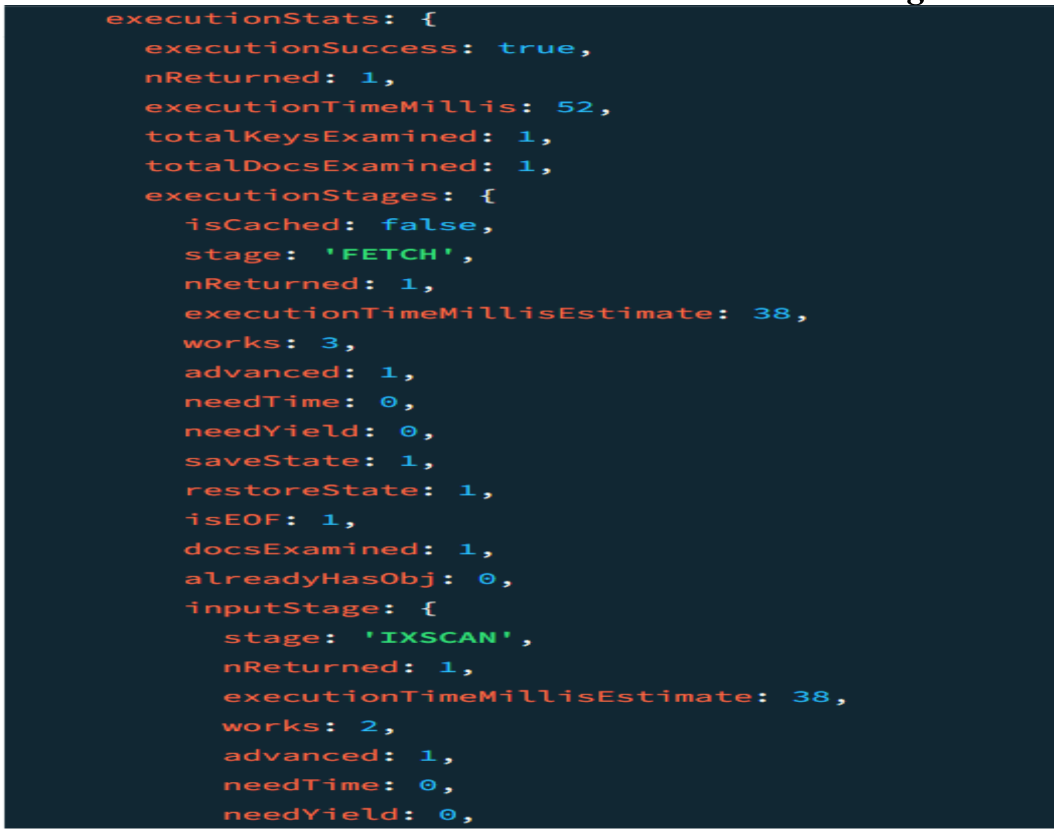
```
> db.Students.find({city:"Delhi"}).sort({age: -1}).explain("executionStats");
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'ShrutikaBDA.Students',
      parsedQuery: {
        city: {
          '$eq': 'Delhi'
        }
      },
      indexFilterSet: false,
      queryHash: '5BCFB86D',
      planCacheShapeHash: '5BCFB86D',
      planCacheKey: '9987535E',
      optimizationTimeMillis: 52,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: {
            city: 1,
            age: -1
          },
```



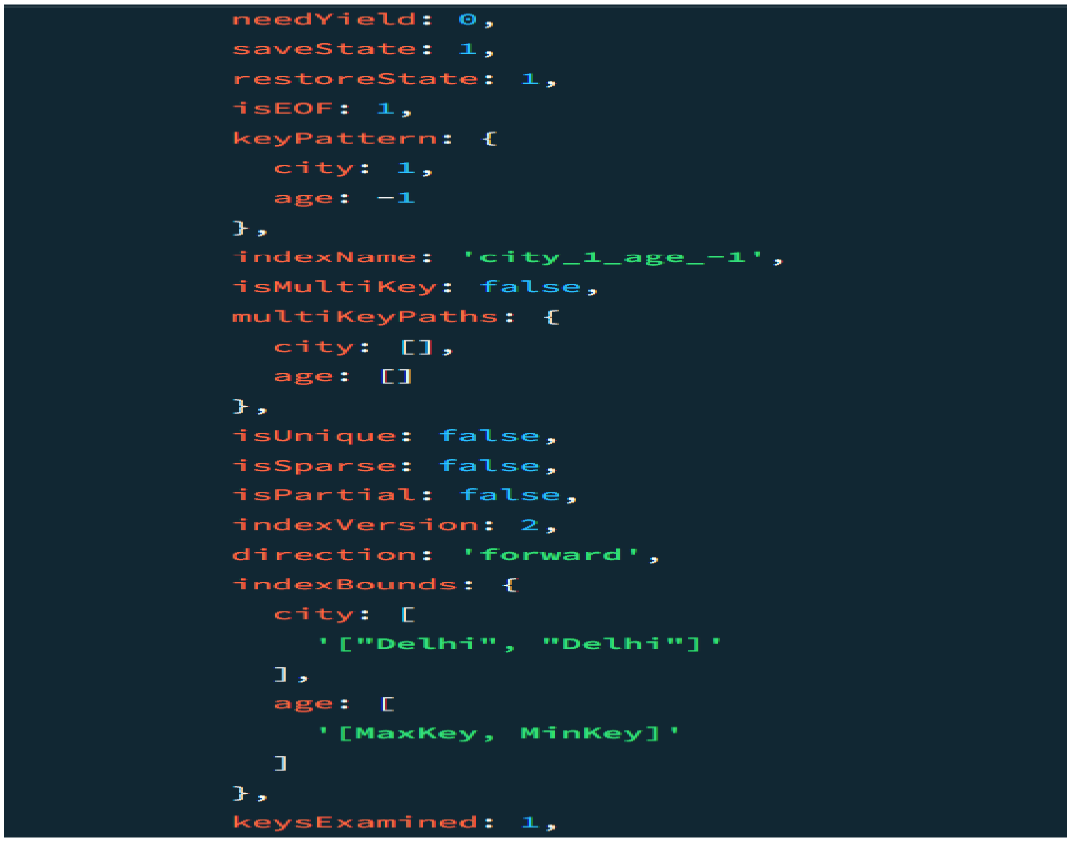
```
          indexName: 'city_1_age_-1',
          isMultiKey: false,
          multiKeyPaths: {
            city: [],
            age: []
          },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: {
            city: [
              '["Delhi", "Delhi"]'
            ],
            age: [
              '[MaxKey, MinKey]'
            ]
          }
        }
      },
      rejectedPlans: [
        {
          isCached: false,
          stage: 'SORT',
          sortPattern: {
            age: -1
          },
```



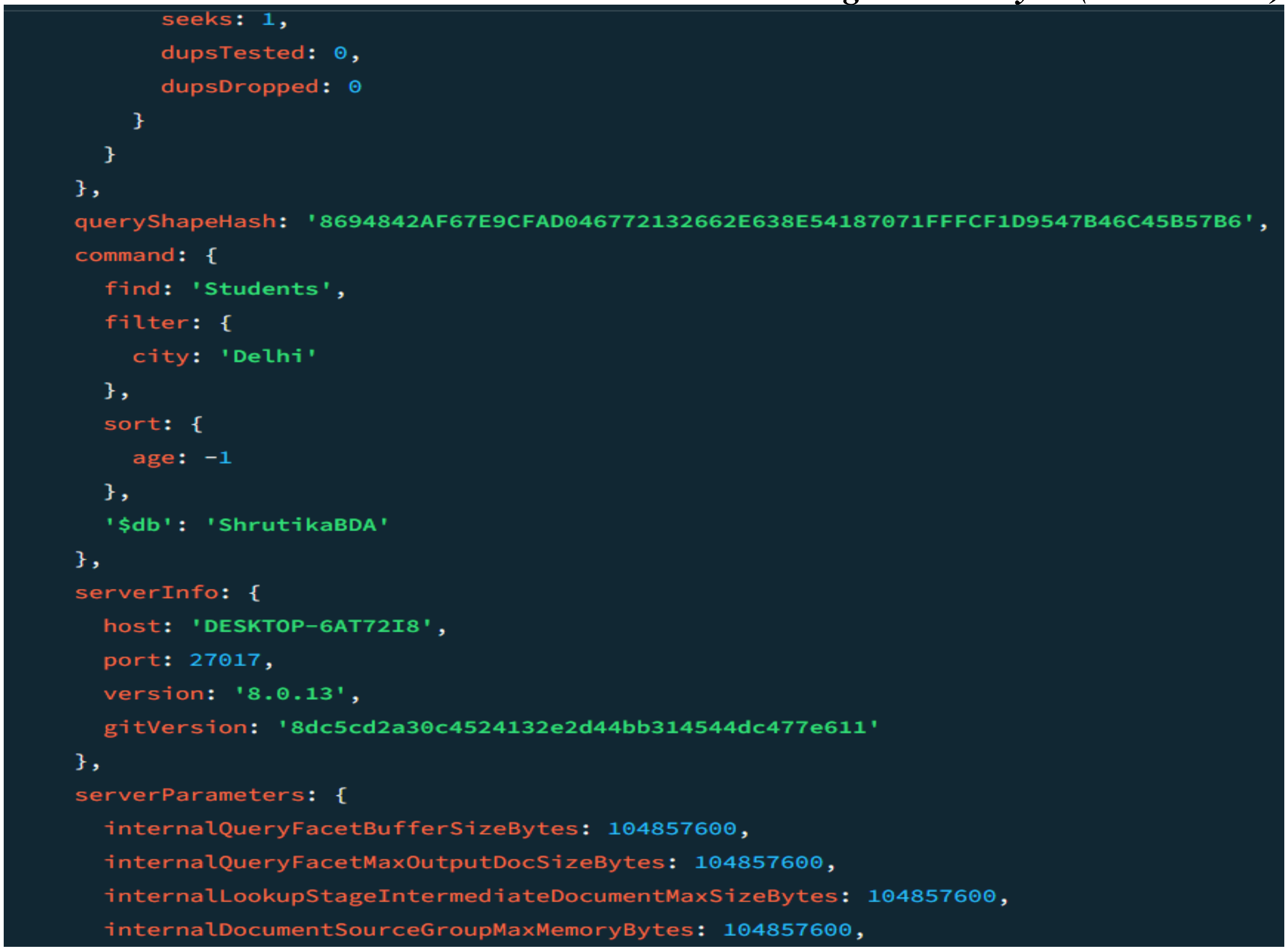
```
          memLimit: 104857600,
          type: 'simple',
          inputStage: {
            stage: 'FETCH',
            inputStage: {
              stage: 'IXSCAN',
              keyPattern: {
                city: 1
              },
              indexName: 'city_1',
              isMultiKey: false,
              multiKeyPaths: {
                city: []
              },
              isUnique: false,
              isSparse: false,
              isPartial: false,
              indexVersion: 2,
              direction: 'forward',
              indexBounds: {
                city: [
                  '["Delhi", "Delhi"]'
                ]
              }
            }
          }
        }
      ]
```



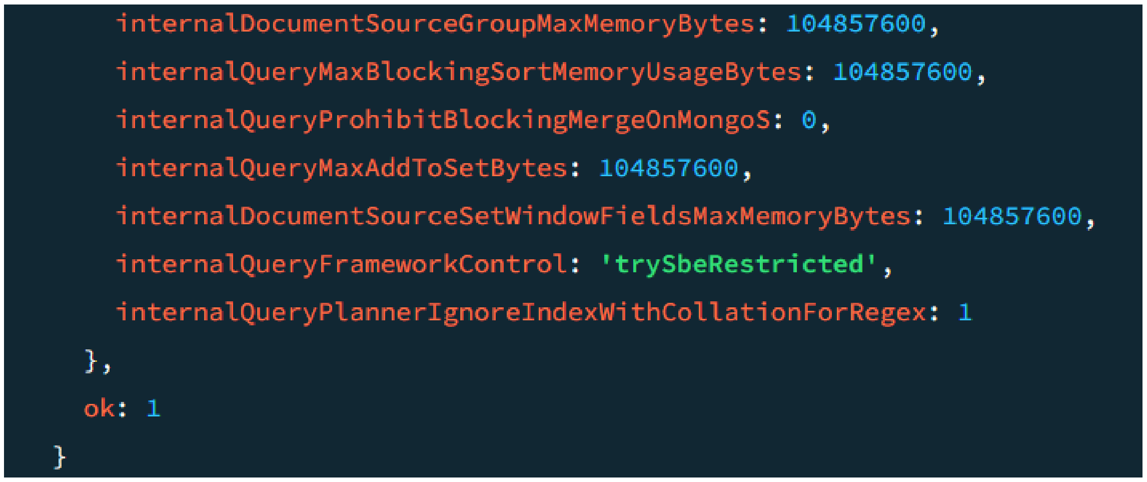
```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 52,
  totalKeysExamined: 1,
  totalDocsExamined: 1,
  executionStages: {
    isCached: false,
    stage: 'FETCH',
    nReturned: 1,
    executionTimeMillisEstimate: 38,
    works: 3,
    advanced: 1,
    needTime: 0,
    needYield: 0,
    saveState: 1,
    restoreState: 1,
    isEOF: 1,
    docsExamined: 1,
    alreadyHasObj: 0,
    inputStage: {
      stage: 'IXSCAN',
      nReturned: 1,
      executionTimeMillisEstimate: 38,
      works: 2,
      advanced: 1,
      needTime: 0,
      needYield: 0,
```



```
      needYield: 0,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      keyPattern: {
        city: 1,
        age: -1
      },
      indexName: 'city_1_age_-1',
      isMultiKey: false,
      multiKeyPaths: {
        city: [],
        age: []
      },
      isUnique: false,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: {
        city: [
          '["Delhi", "Delhi"]'
        ],
        age: [
          '[MaxKey, MinKey]'
        ]
      },
      keysExamined: 1,
```

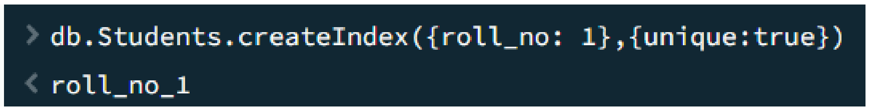


```
            seeks: 1,
            dupsTested: 0,
            dupsDropped: 0
          }
        }
      },
      queryShapeHash: '8694842AF67E9CFAD046772132662E638E54187071FFFCF1D9547B46C45B57B6',
      command: {
        find: 'Students',
        filter: {
          city: 'Delhi'
        },
        sort: {
          age: -1
        },
        '$db': 'ShrutikaBDA'
      },
      serverInfo: {
        host: 'DESKTOP-6AT72I8',
        port: 27017,
        version: '8.0.13',
        gitVersion: '8dc5cd2a30c4524132e2d44bb314544dc477e611'
      },
      serverParameters: {
        internalQueryFacetBufferSizeBytes: 104857600,
        internalQueryFacetMaxOutputDocSizeBytes: 104857600,
        internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
        internalDocumentSourceGroupMaxMemoryBytes: 104857600,
```

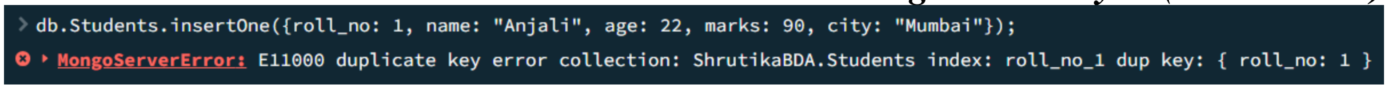


```
        internalDocumentSourceGroupMaxMemoryBytes: 104857600,
        internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
        internalQueryProhibitBlockingMergeOnMongoS: 0,
        internalQueryMaxAddToSetBytes: 104857600,
        internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
        internalQueryFrameworkControl: 'trySbeRestricted',
        internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
      },
      ok: 1
    }
```

Unique Index

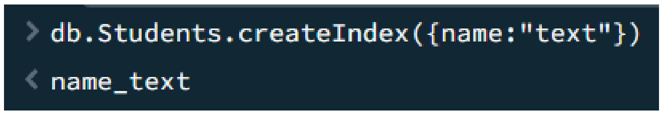


```
> db.Students.createIndex({roll_no: 1},{unique:true})
< roll_no_1
```

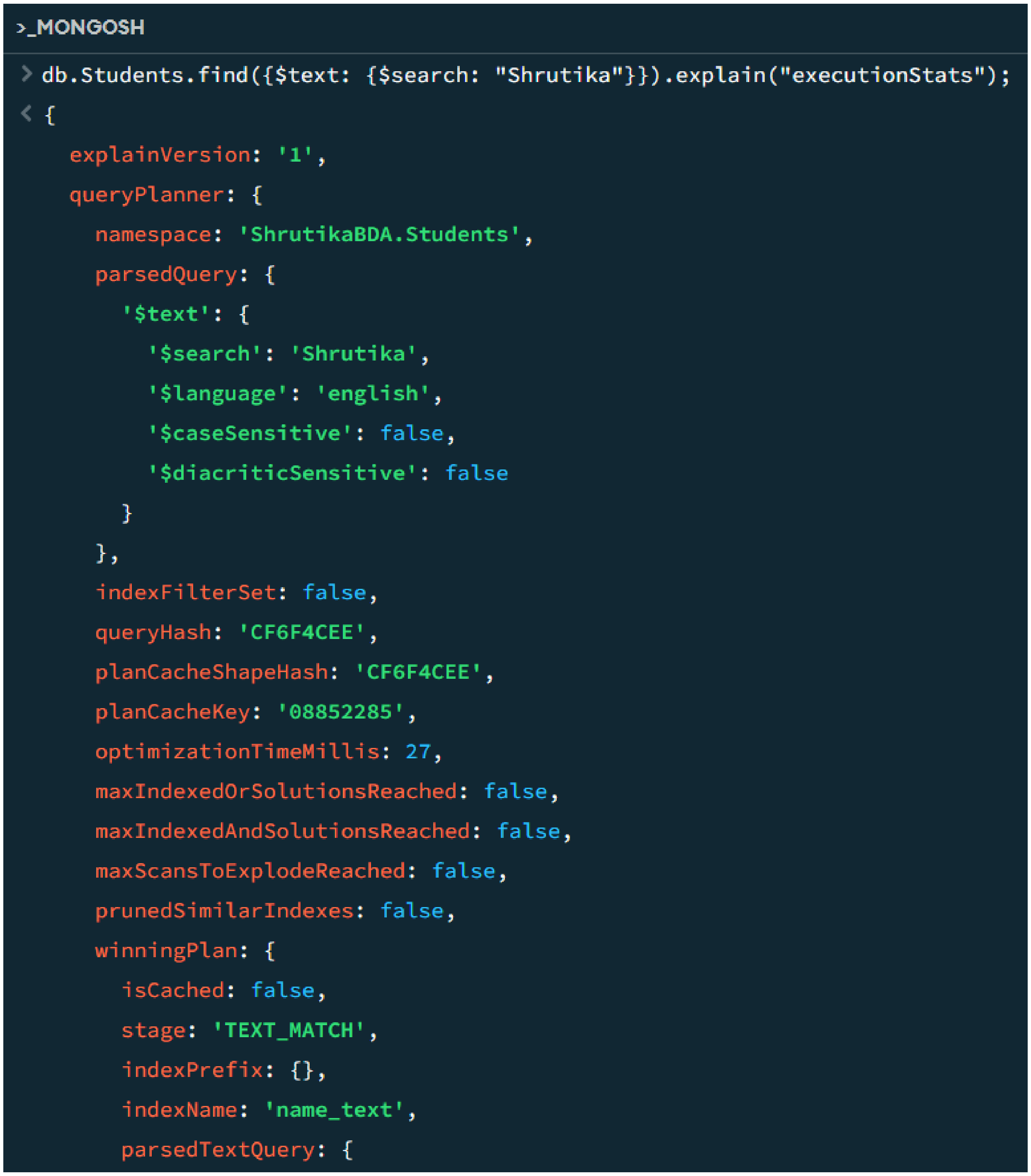


```
> db.Students.insertOne({roll_no: 1, name: "Anjali", age: 22, marks: 90, city: "Mumbai"});
✖ ▸ MongoServerError: E11000 duplicate key error collection: ShrutikaBDA.Students index: roll_no_1 dup key: { roll_no: 1 }
```

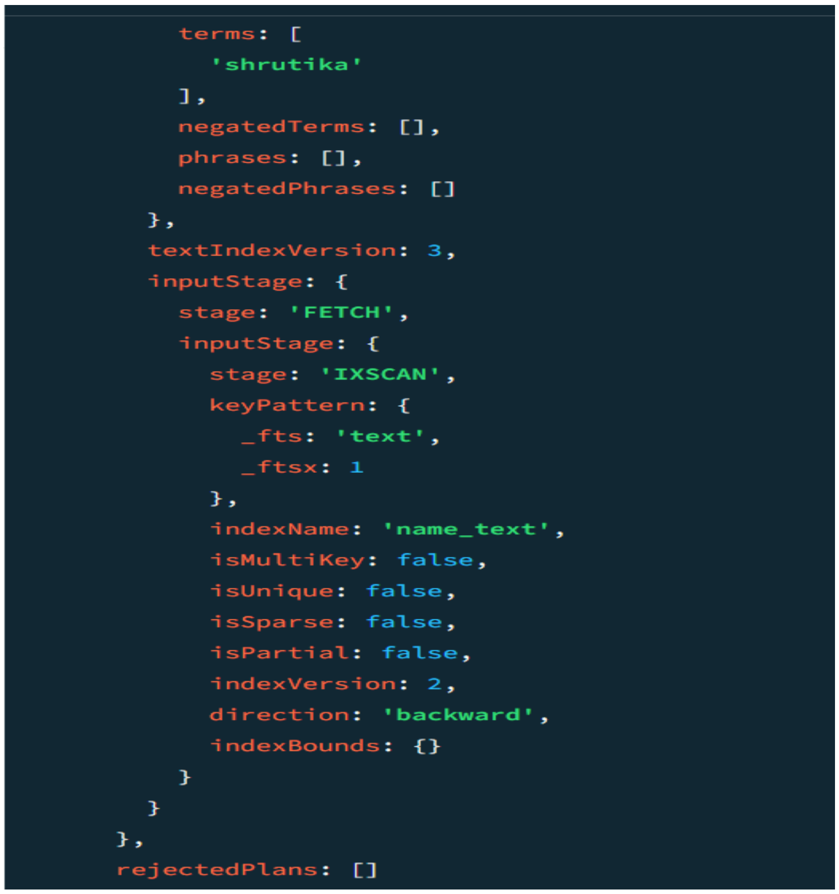
Text Index



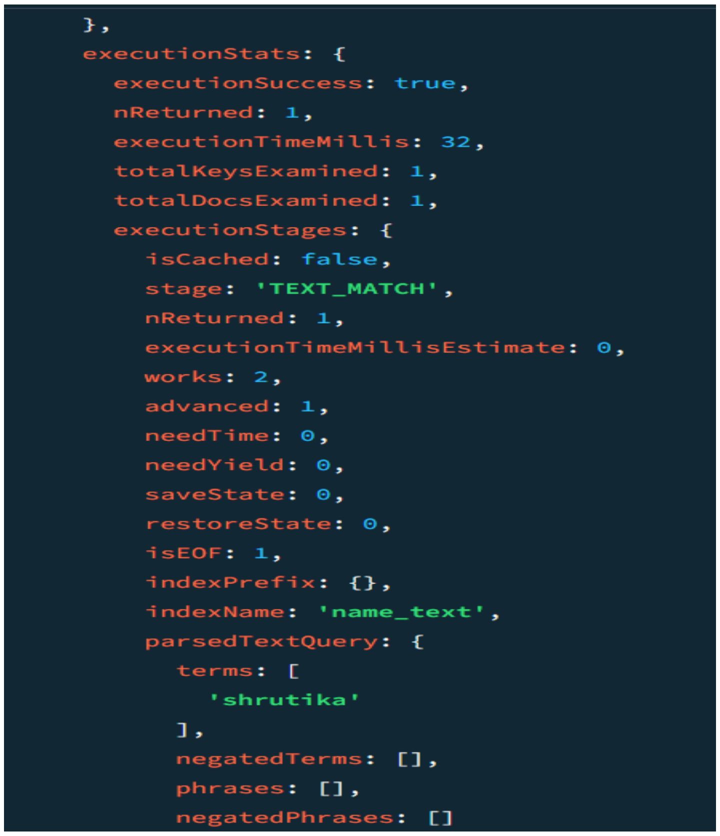
```
> db.Students.createIndex({name:"text"})
< name_text
```



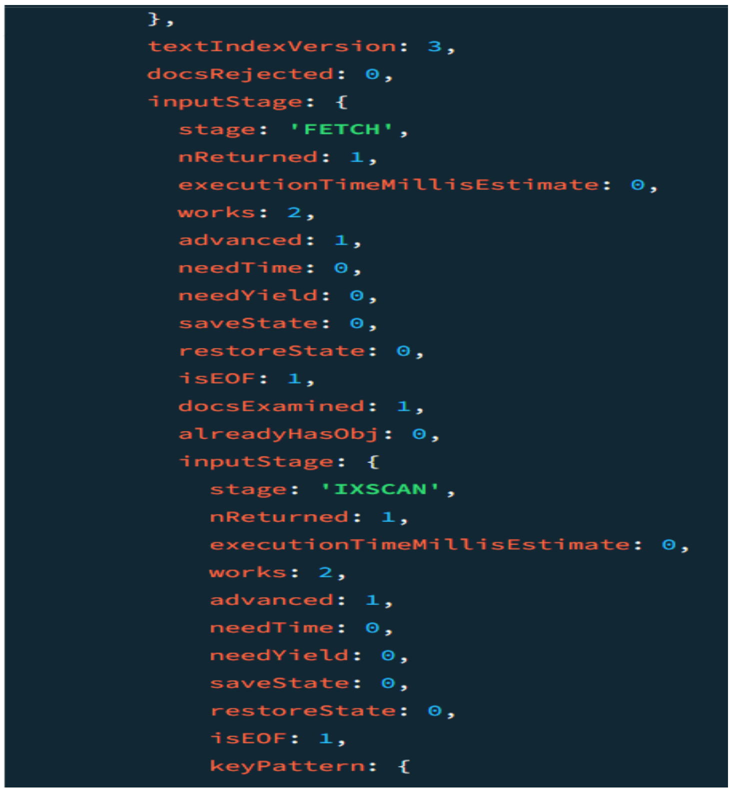
```
>_MONGOSH
> db.Students.find({$text: {$search: "Shrutika"}}).explain("executionStats");
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'ShrutikaBDA.Students',
      parsedQuery: {
        '$text': {
          '$search': 'Shrutika',
          '$language': 'english',
          '$caseSensitive': false,
          '$diacriticSensitive': false
        }
      },
      indexFilterSet: false,
      queryHash: 'CF6F4CEE',
      planCacheShapeHash: 'CF6F4CEE',
      planCacheKey: '08852285',
      optimizationTimeMillis: 27,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      prunedSimilarIndexes: false,
      winningPlan: {
        isCached: false,
        stage: 'TEXT_MATCH',
        indexPrefix: {},
        indexName: 'name_text',
        parsedTextQuery: {
```



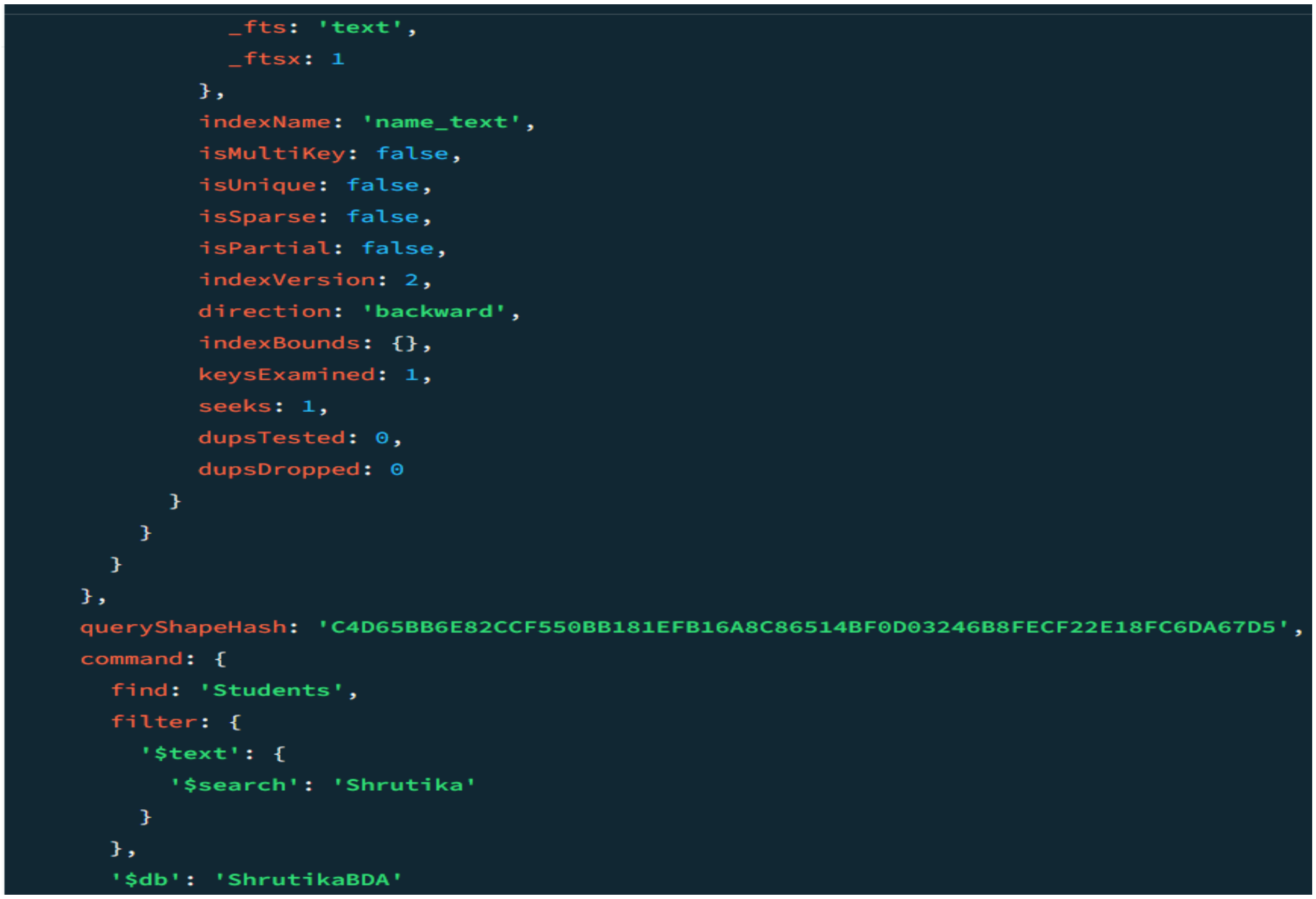
```
          terms: [
            'shrutika'
          ],
          negatedTerms: [],
          phrases: [],
          negatedPhrases: []
        },
        textIndexVersion: 3,
        inputStage: {
          stage: 'FETCH',
          inputStage: {
            stage: 'IXSCAN',
            keyPattern: {
              _fts: 'text',
              _ftsx: 1
            },
            indexName: 'name_text',
            isMultiKey: false,
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'backward',
            indexBounds: {}
          }
        }
      },
      rejectedPlans: []
```



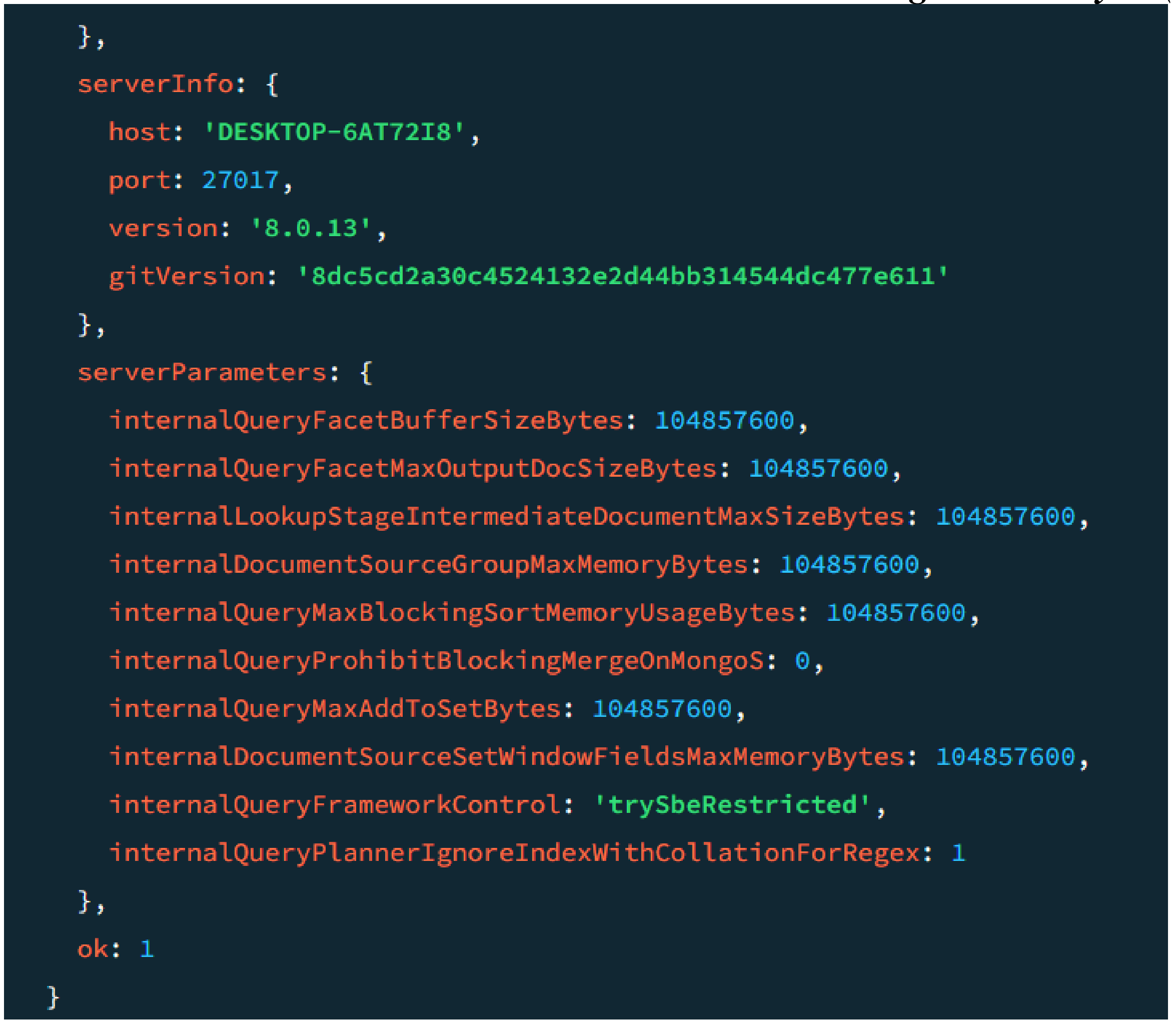
```
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 1,
      executionTimeMillis: 32,
      totalKeysExamined: 1,
      totalDocsExamined: 1,
      executionStages: {
        isCached: false,
        stage: 'TEXT_MATCH',
        nReturned: 1,
        executionTimeMillisEstimate: 0,
        works: 2,
        advanced: 1,
        needTime: 0,
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1,
        indexPrefix: {},
        indexName: 'name_text',
        parsedTextQuery: {
          terms: [
            'shrutika'
          ],
          negatedTerms: [],
          phrases: [],
          negatedPhrases: []
```



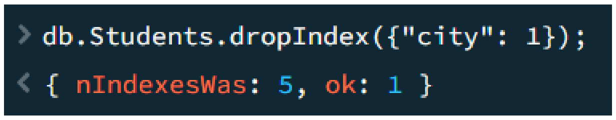
```
        },
        textIndexVersion: 3,
        docsRejected: 0,
        inputStage: {
          stage: 'FETCH',
          nReturned: 1,
          executionTimeMillisEstimate: 0,
          works: 2,
          advanced: 1,
          needTime: 0,
          needYield: 0,
          saveState: 0,
          restoreState: 0,
          isEOF: 1,
          docsExamined: 1,
          alreadyHasObj: 0,
          inputStage: {
            stage: 'IXSCAN',
            nReturned: 1,
            executionTimeMillisEstimate: 0,
            works: 2,
            advanced: 1,
            needTime: 0,
            needYield: 0,
            saveState: 0,
            restoreState: 0,
            isEOF: 1,
            keyPattern: {
```



```
              _fts: 'text',
              _ftsx: 1
            },
            indexName: 'name_text',
            isMultiKey: false,
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'backward',
            indexBounds: {},
            keysExamined: 1,
            seeks: 1,
            dupsTested: 0,
            dupsDropped: 0
          }
        }
      }
    },
    queryShapeHash: 'C4D65BB6E82CCF550BB181EFB16A8C86514BF0D03246B8FECF22E18FC6DA67D5',
    command: {
      find: 'Students',
      filter: {
        '$text': {
          '$search': 'Shrutika'
        }
      },
      '$db': 'ShrutikaBDA'
```



```
    },
    serverInfo: {
      host: 'DESKTOP-6AT72I8',
      port: 27017,
      version: '8.0.13',
      gitVersion: '8dc5cd2a30c4524132e2d44bb314544dc477e611'
    },
    serverParameters: {
      internalQueryFacetBufferSizeBytes: 104857600,
      internalQueryFacetMaxOutputDocSizeBytes: 104857600,
      internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
      internalDocumentSourceGroupMaxMemoryBytes: 104857600,
      internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
      internalQueryProhibitBlockingMergeOnMongoS: 0,
      internalQueryMaxAddToSetBytes: 104857600,
      internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
      internalQueryFrameworkControl: 'trySbeRestricted',
      internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
    },
    ok: 1
  }
```



```
> db.Students.dropIndex({"city": 1});
< { nIndexesWas: 5, ok: 1 }
```

CONCLUSION:

DISCUSSION AND VIVA VOCE:

- What are the Pros and Cons of MongoDB Indexing
- How does MongoDB choose which index to use?
- How to check indexing in MongoDB?
- What is the maximum index in MongoDB?
- How do you rename an index in MongoDB?

REFERENCE:

- https://www.tutorialspoint.com/mongodb/mongodb_indexing.htm
- https://www.simplilearn.com/tutorials/mongodb-tutorial/indexes-in-mongodb#:~:text=A%20MongoDB%20index%20supports%20the,that%20match%20the%20query%20statement.
- https://www.geeksforgeeks.org/indexing-in-mongodb

Observation book: (3)	Viva-Voce (3)	Quality of Submission and timely Evaluation (4)
Total:		Sign with date: