Name: Shabbar Adamjee
Roll No.: PB57
PRN: 1032221508

BDT LAB ASSIGNMENT 2

1. Create database 'Restaurant'.

```
use Restaurant
```

2. Create collection hotel.

```
db.createCollection("hotel")
```

3. Insert 10 documents with above mentioned structure.

```
db.hotel.insertMany([
        { },
        { },
        { }.
        ...
])
```

4. Display all Hotel information.

```
db.hotel.find()
```

5. Display no of rooms in each hotel

```
db.hotel.aggregate(
 {$unwind: "$rooms"},
 {$group: {_id: "$name", noOfRooms: {$sum: 1}}}
)
```

6. Compute the top five hotels.

```
db.hotel.aggregate([
 {$group: {_id: '$name', totalLikes: {$sum: likes}}},
 {$sort: {totallikes: -1}},
 {$limit: 5},
 {$project: {name: 1, totalLikes:1}}
])
```

7. Return hotels having likes above 1000.

```
db.hotel.aggregate([
  {$group: {_id: '$name', totalLikes: {$sum: likes}}},
  {$match: {totalLikes: {$gt: 1000}}},
  {$project: {name: 1, totalLikes:1}}
])
```

8. Return the Five Most Common Cuisines.

```
db.hotel.aggregate([
  {$unwind: "$cuisines"},
  {$group: {_id: "$cuisines", count: {$sum: 1}}},
  {$sort: {count: -1}},
  {$limit: 5}
])
```

9. Return all prices of room in different hotel of type ‘Deluxe' .

```
db.hotel.aggregate([
  {$unwind: '$rooms'},
  {$match: {'rooms.type': 'Deluxe'}},
  {$project: {'name': 1, 'rooms.price': 1, _id: 0}}
])
```

10. Get the total count of hotels having ratings ‘5 star’

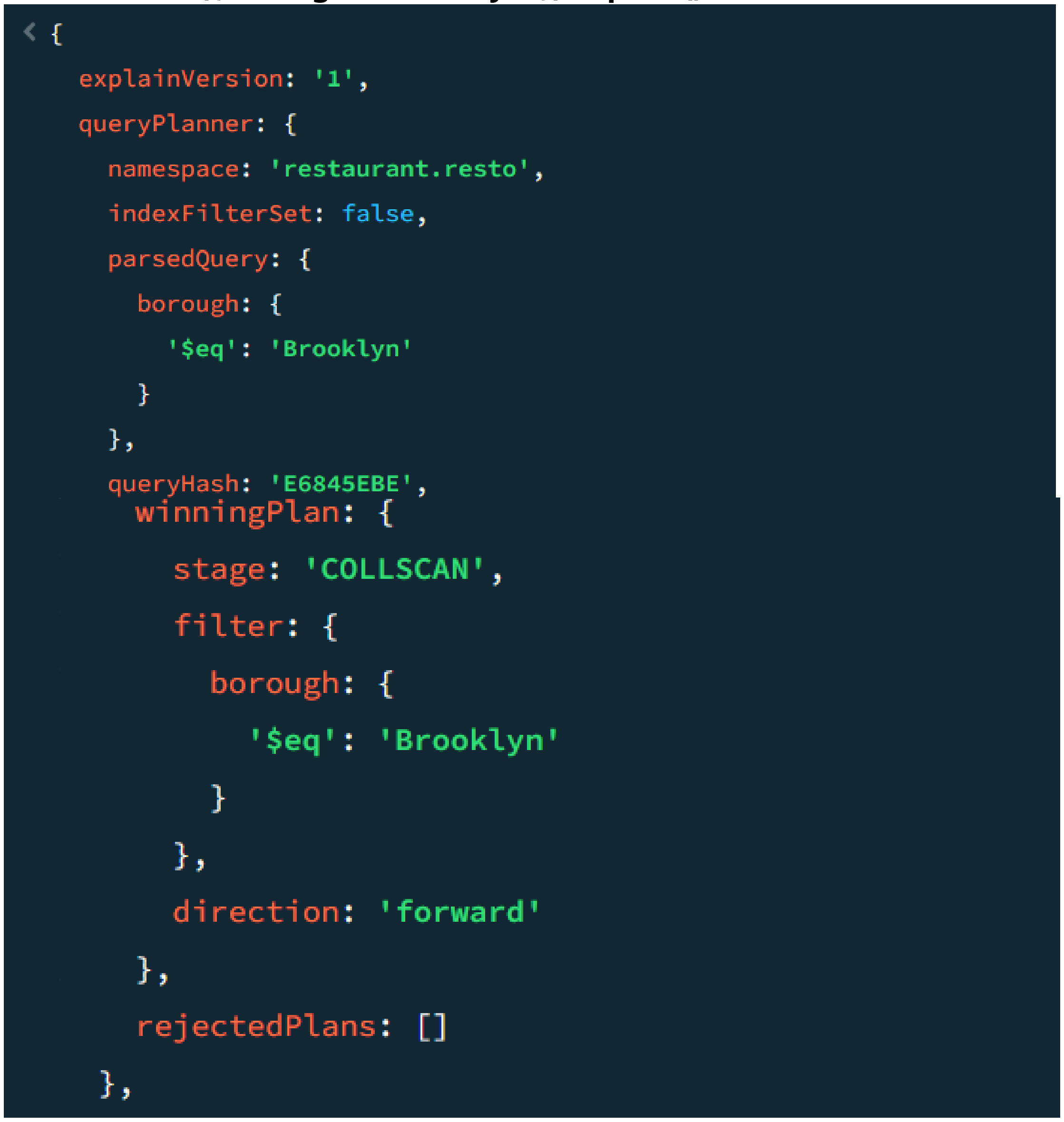
```
db.hotel.aggregate([
  {$match: {'rating': 5}},
  {$group: {_id: 'rating', count: {$sum: 1}}}
])
```

11. Display the count of hotels from ‘Pune’ city.

```
db.hotel.aggregate([
  {$match: {'address.city': 'Pune'}},
  {$count: 'name'}
])
```

1. **Get Indexes method : db.resto.getIndexes()**
   [{ v: 2, key: { _id: 1 }, name: '_id_' }] – default index generated by system

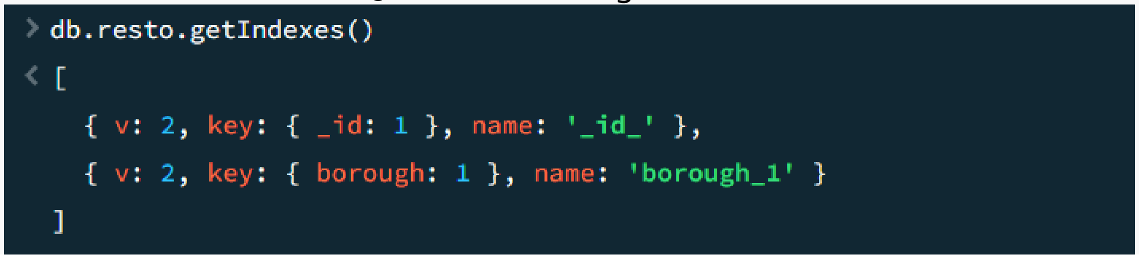
2. **db.resto.find({borough : "Brooklyn"}).explain()**



```
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'restaurant.resto',
      indexFilterSet: false,
      parsedQuery: {
        borough: {
          '$eq': 'Brooklyn'
        }
      },
      queryHash: 'E6845EBE',
      winningPlan: {
        stage: 'COLLSCAN',
        filter: {
          borough: {
            '$eq': 'Brooklyn'
          }
        },
        direction: 'forward'
      },
      rejectedPlans: []
    },
```

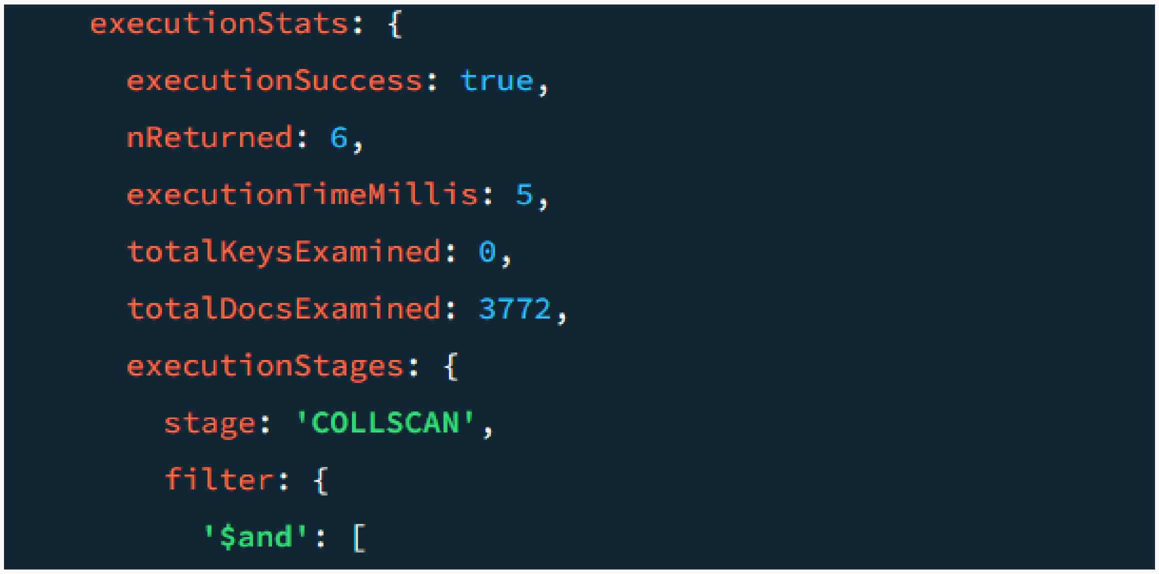
db.resto.find({borough : "Brooklyn"}).explain("executionStats")

3. db.resto.createIndex({borough : 1}) -> Creating index on a field

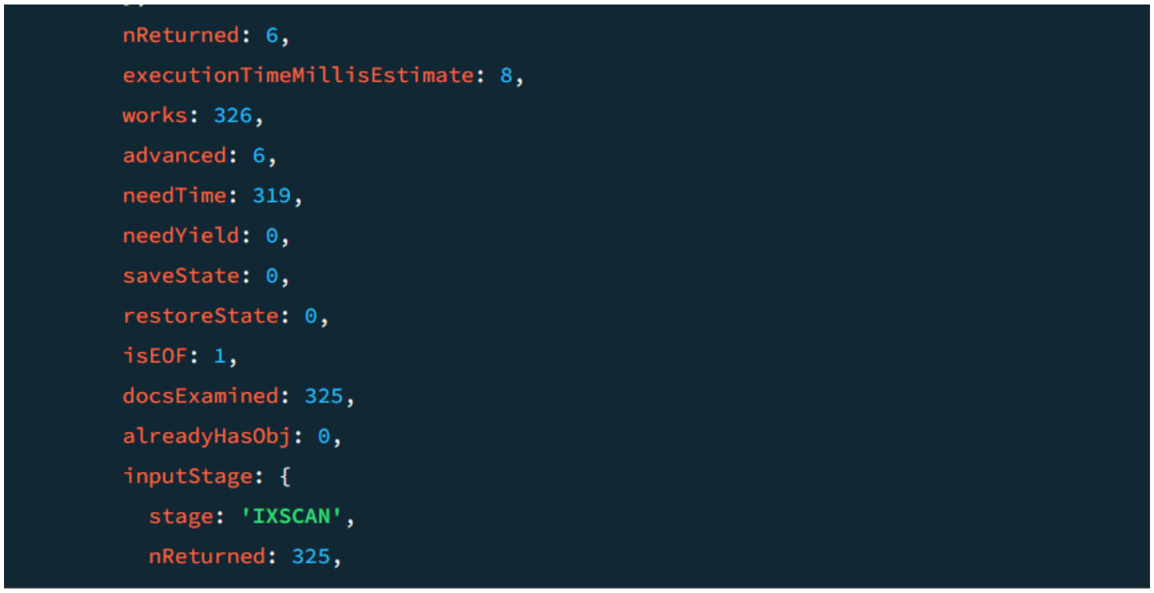


```
> db.resto.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { borough: 1 }, name: 'borough_1' }
  ]
```

4. db.resto.find({$and : [{"cuisine": {$eq : "Italian"}}, {"grades.score" : {$gt : 50}}]}).explain("executionStats")

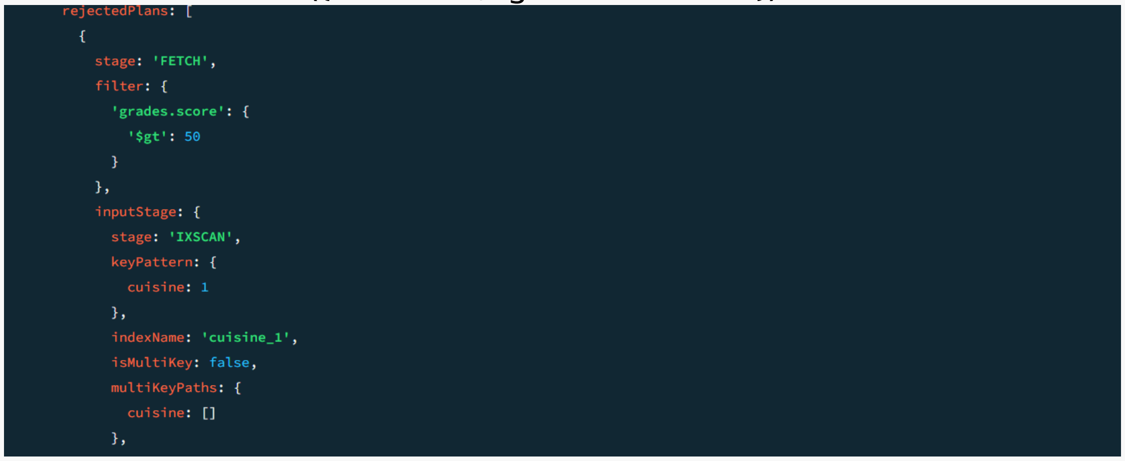


```
    executionStats: {
      executionSuccess: true,
      nReturned: 6,
      executionTimeMillis: 5,
      totalKeysExamined: 0,
      totalDocsExamined: 3772,
      executionStages: {
        stage: 'COLLSCAN',
        filter: {
          '$and': [
```



```
          nReturned: 6,
          executionTimeMillisEstimate: 8,
          works: 326,
          advanced: 6,
          needTime: 319,
          needYield: 0,
          saveState: 0,
          restoreState: 0,
          isEOF: 1,
          docsExamined: 325,
          alreadyHasObj: 0,
          inputStage: {
            stage: 'IXSCAN',
            nReturned: 325,
```

5. db.resto.createIndex({cuisine : 1, "grades.score" : 1})



```
      rejectedPlans: [
        {
          stage: 'FETCH',
          filter: {
            'grades.score': {
              '$gt': 50
            }
          },
          inputStage: {
            stage: 'IXSCAN',
            keyPattern: {
              cuisine: 1
            },
            indexName: 'cuisine_1',
            isMultiKey: false,
            multiKeyPaths: {
              cuisine: []
            },
```