**Practical No : 06**

**Aim :** Text search in MongoDB – Text fields in collection, create an index, prepare search items, text search.

* **Sample collection of the five highest mountains in the world.**

**Insert Code :**

db.peaks.insertMany([

{

"name": "Everest",

“description”: “Mount **Everest** is Earth's highest mountain above sea level”

"height": 8848,

"location": ["Nepal", "China"],

"ascents": {

"first": {

"year": 1953

},

"first\_winter": {

"year": 1980

},

"total": 5656

}

},

{

"name": "K2",

“description”: “K2 is the second highest mountain on Earth”,

"height": 8611,

"location": ["Pakistan", "China"],

"ascents": {

"first": {

"year": 1954

},

"first\_winter": {

"year": 1921

},

"total": 306

}

},

{

"name": "Kangchenjunga",

“description”: “Kangchenjunga is the third highest mountain in the world”,

"height": 8586,

"location": ["Nepal", "India"],

"ascents": {

"first": {

"year": 1955

},

"first\_winter": {

"year": 1986

},

"total": 283

}

},

{

"name": "Lhotse",

“description”: “**Lhotse** is the fourth highest mountain in the world”,

"height": 8516,

"location": ["Nepal", "China"],

"ascents": {

"first": {

"year": 1956

},

"first\_winter": {

"year": 1988

},

"total": 461

}

},

{

"name": "Makalu",

“description”: “**Makalu** is the fifth highest mountain in the world”,

"height": 8485,

"location": ["China", "Nepal"],

"ascents": {

"first": {

"year": 1955

},

"first\_winter": {

"year": 2009

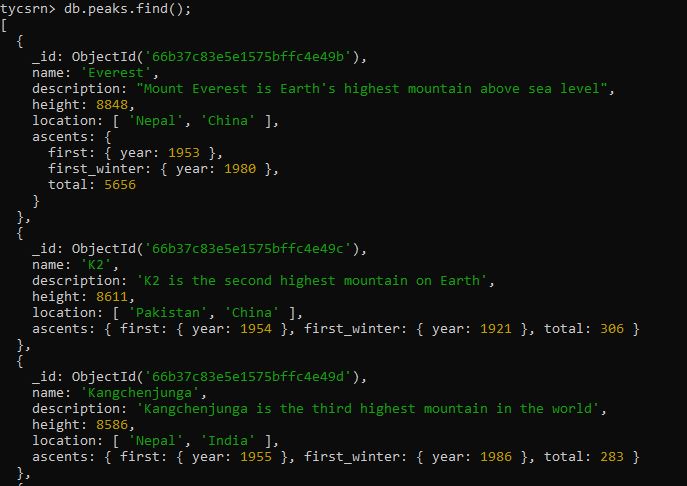
},

"total": 361

])

**1) Verify that the documents were properly inserted by running the find() method.**

> db.peaks.find()



**2) Retrieve documents that describe a mountain peak with a height value greater than 8700. Check the returned and total keys examined.**

db.peaks.find(

{ "height": { $gt: 8700 } }

).explain("executionStats")



**3) Create an index on the height field in the peaks collection.**

db.peaks.createIndex( { "height": 1 } )



**4) Run the same query from Q. 2 and check the returned and total keys examined.**

db.peaks.find(

{ "height": { $gt: 8700 } }

).explain("executionStats")

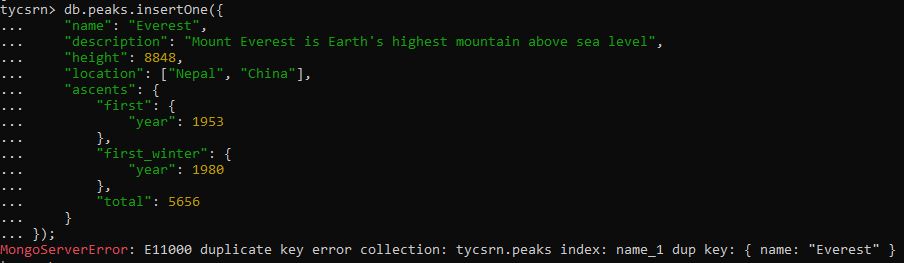


**5) Create an indexe to ensure the values of a given field “name” will be unique for every document in a collection.**

db.peaks.createIndex( { "name": 1 }, { "unique": true } )



**6) Check whether you’re able to add a second document representing Mt. Everest to the collection using insertOne().**

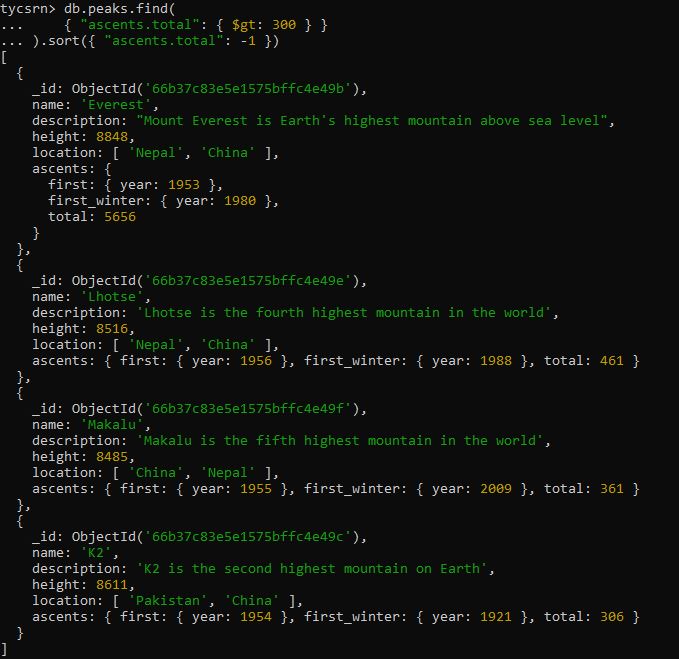


**7) Return documents whose total — a field nested within the ascents document found in each document in the peaks collection is greater than 300 and sorts the results in descending order.**

db.peaks.find(

{ "ascents.total": { $gt: 300 } }

).sort({ "ascents.total": -1 })



**8) Run the same query, but include the explain("executionStats") method used previously.**

db.peaks.find(

{ "ascents.total": { $gt: 300 } }

).sort({ "ascents.total": -1 }).explain("executionStats")



**9) Create an index on the total field within the ascents document. (Embedded Indexing).**

db.peaks.createIndex( { "ascents.total": 1 } )



**10) Run the previous query once again to check whether the index helped MongoDB avoid performing a full collection scan.**

db.peaks.find(

{ "ascents.total": { $gt: 300 } }

).sort({ "ascents.total": -1 }).explain("executionStats")



**11) Find mountains with a height of less than 8600 meters whose first winter ascent occurred after the year 1990.**

db.peaks.find(

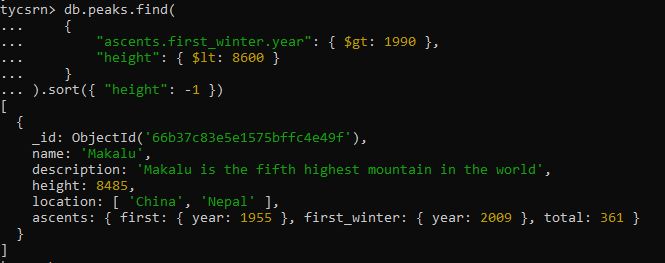
{

"ascents.first\_winter.year": { $gt: 1990 },

"height": { $lt: 8600 }

}

).sort({ "height": -1 })



**12) Find how MongoDB performed this same query by adding explain() method**.

db.peaks.find(

{

"ascents.first\_winter.year": { $gt: 1990 },

"height": { $lt: 8600 }

}

).sort({ "height": -1 }).explain("executionStats")



**13) Create an ascending index for peaks’ first winter ascents and descending index for their heights. (Compound Field Index)**

db.peaks.createIndex(

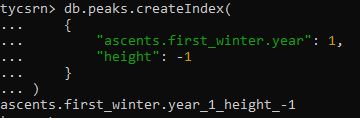
{

"ascents.first\_winter.year": 1,

"height": -1

}

)



**14) Run the previous query once again to test whether there was any change in how the query was performed.**

db.peaks.find(

{

"ascents.first\_winter.year": { $gt: 1990 },

"height": { $lt: 8600 }

}

).sort({ "height": -1 }).explain("executionStats")



**15) Finding all the mountains in the collection that are located in Nepal.**

db.peaks.find(

{ "location": "Nepal" }

)



**16) Create a new index for the location field. (Multi-key Index)**

db.peaks.createIndex( { "location": 1 } )



**17) Run the previous query once again to test whether there was any change in how the query was performed for Multi-key Index.**

db.peaks.find(

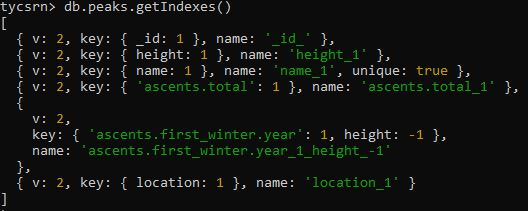
{ "location": "Nepal" }

).explain("executionStats")



**18) List all the indexes you’ve defined on the peaks collection.**

db.peaks.getIndexes()

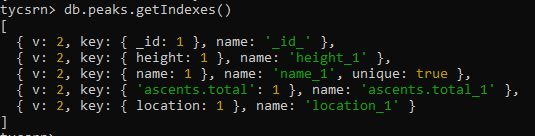


**19) To remove the index created on the first winter ascent and height.**

db.peaks.dropIndex("ascents.first\_winter.year\_1\_height\_-1")

db.peaks.getIndexes()



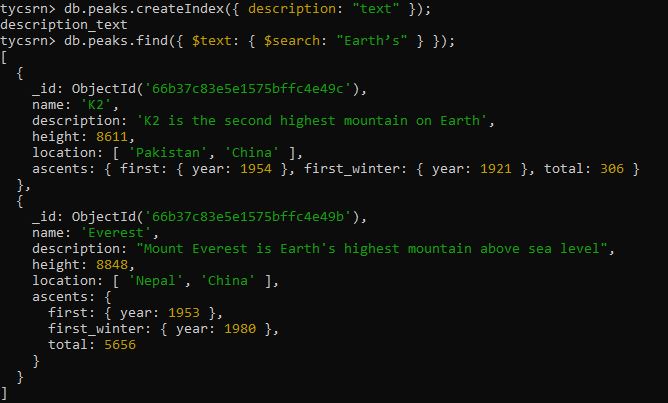


**20) Retrieve documents matching with the keyword “Earth’s” in description field using an index.**

**Index creation:** db.content.createIndex({key:"text")

**For Phrase search:**

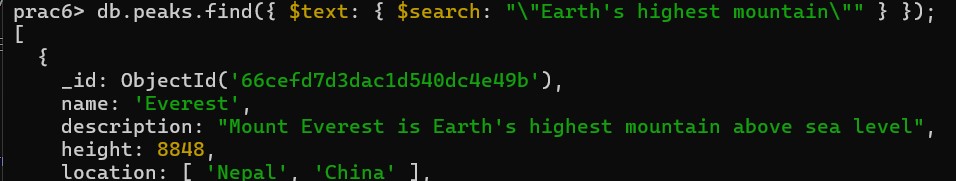
*db.collectionName.find({$text:{$search:”Phrase”}})*



**21)** **Retrieve documents matching with the keyword “Earth’s highest mountain” in description field using an index.**

**For exact phrase search:**

*db.collectionName.find({$text:{$search:”\”Phrase”\”}})*



**22) Retrieve documents containing the keywords “highest mountain” but not the keyword “fourth” in description field using an index.**

**Exclude using - (minus sign).**

db.collectionName.find({$text:{$search:"include\_key –exclude\_key"}})

