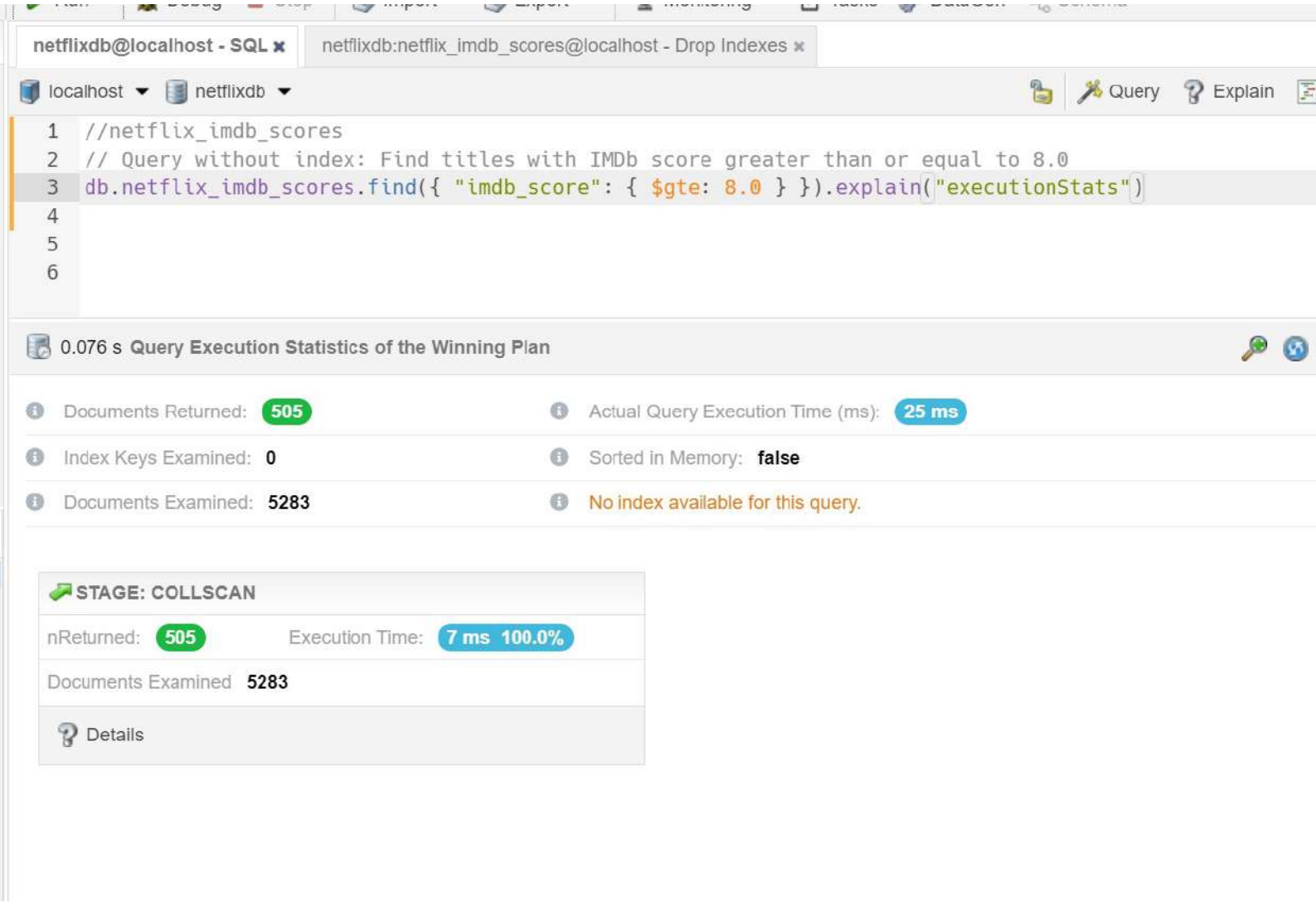
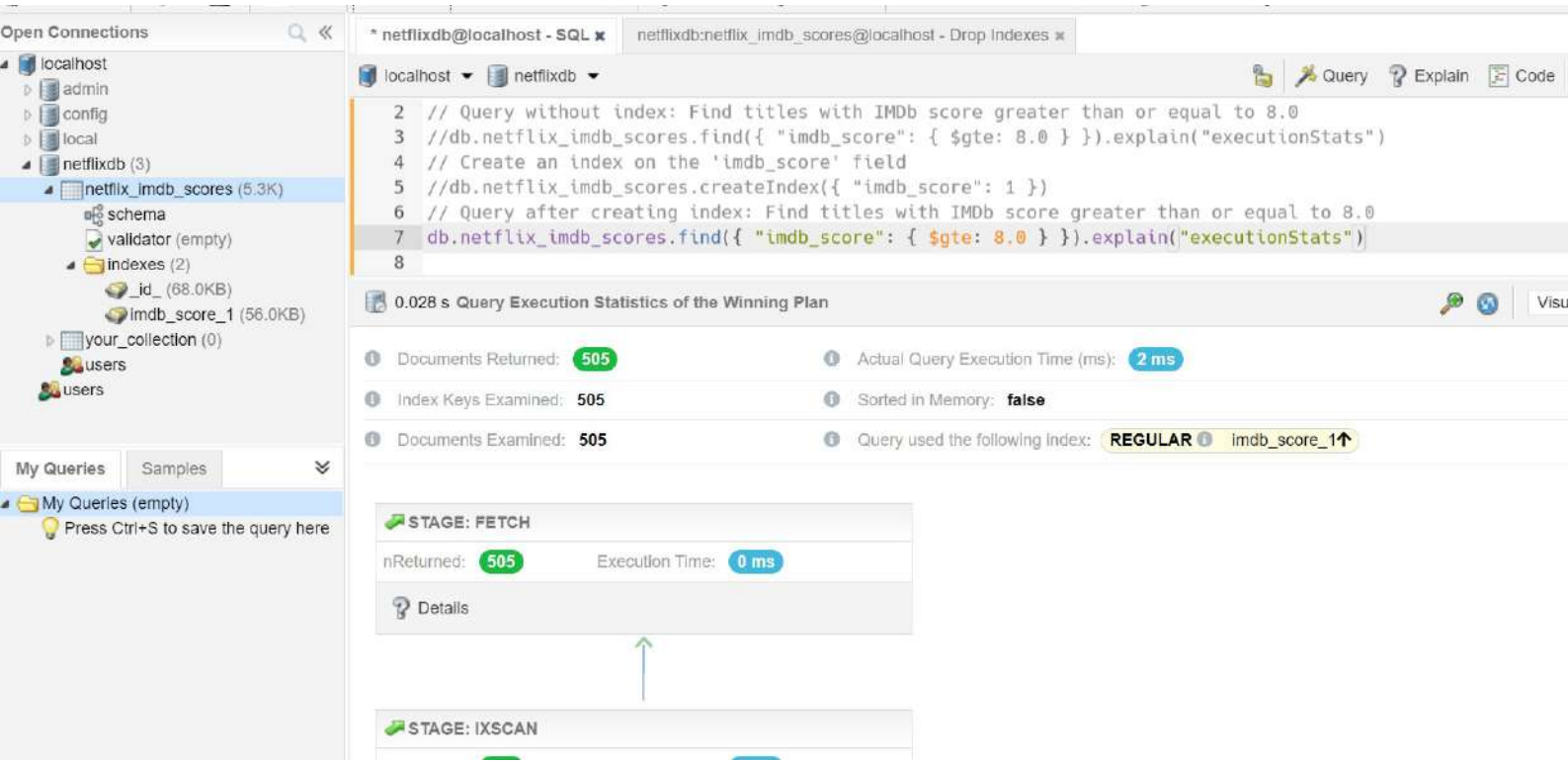
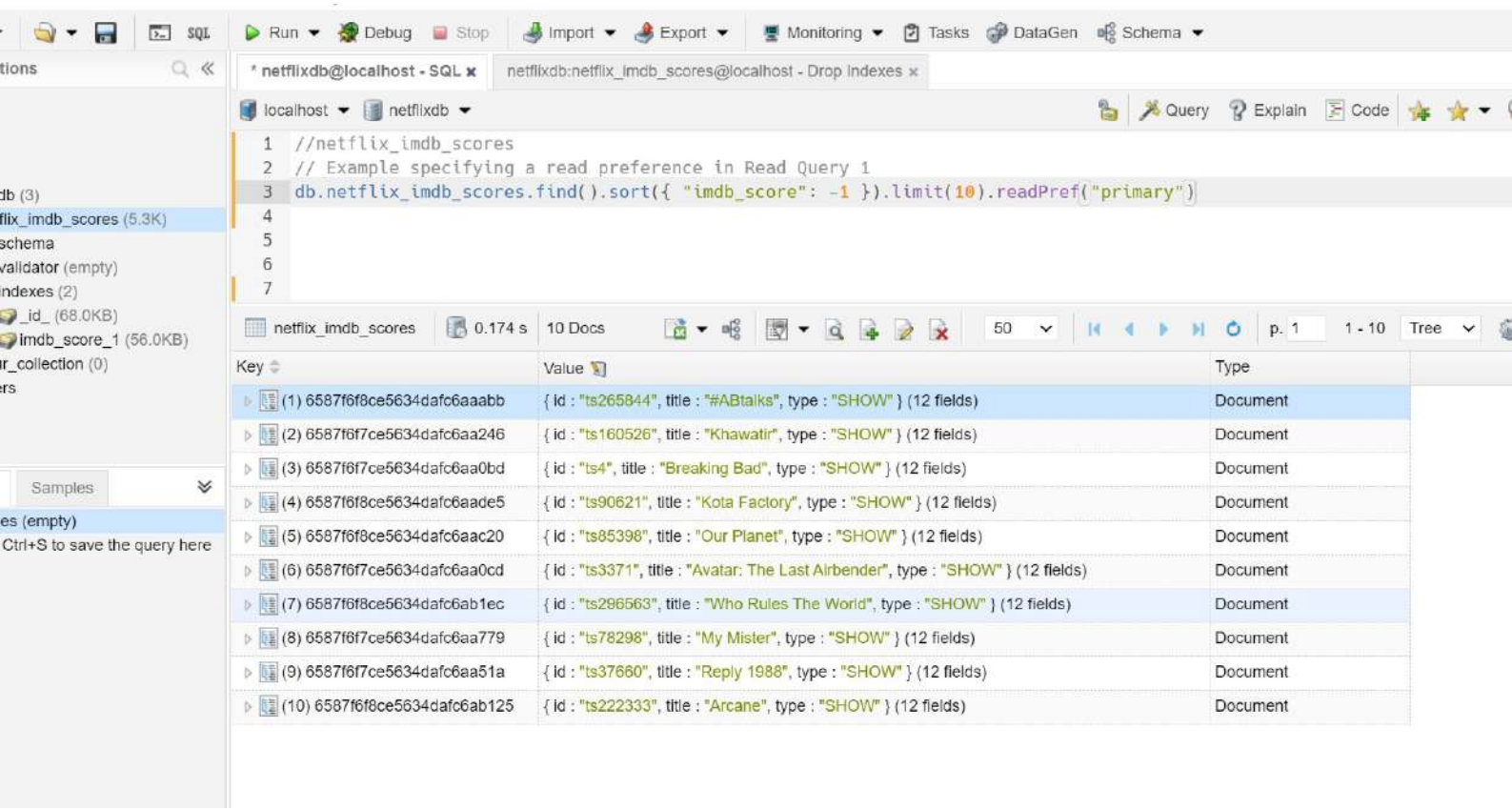
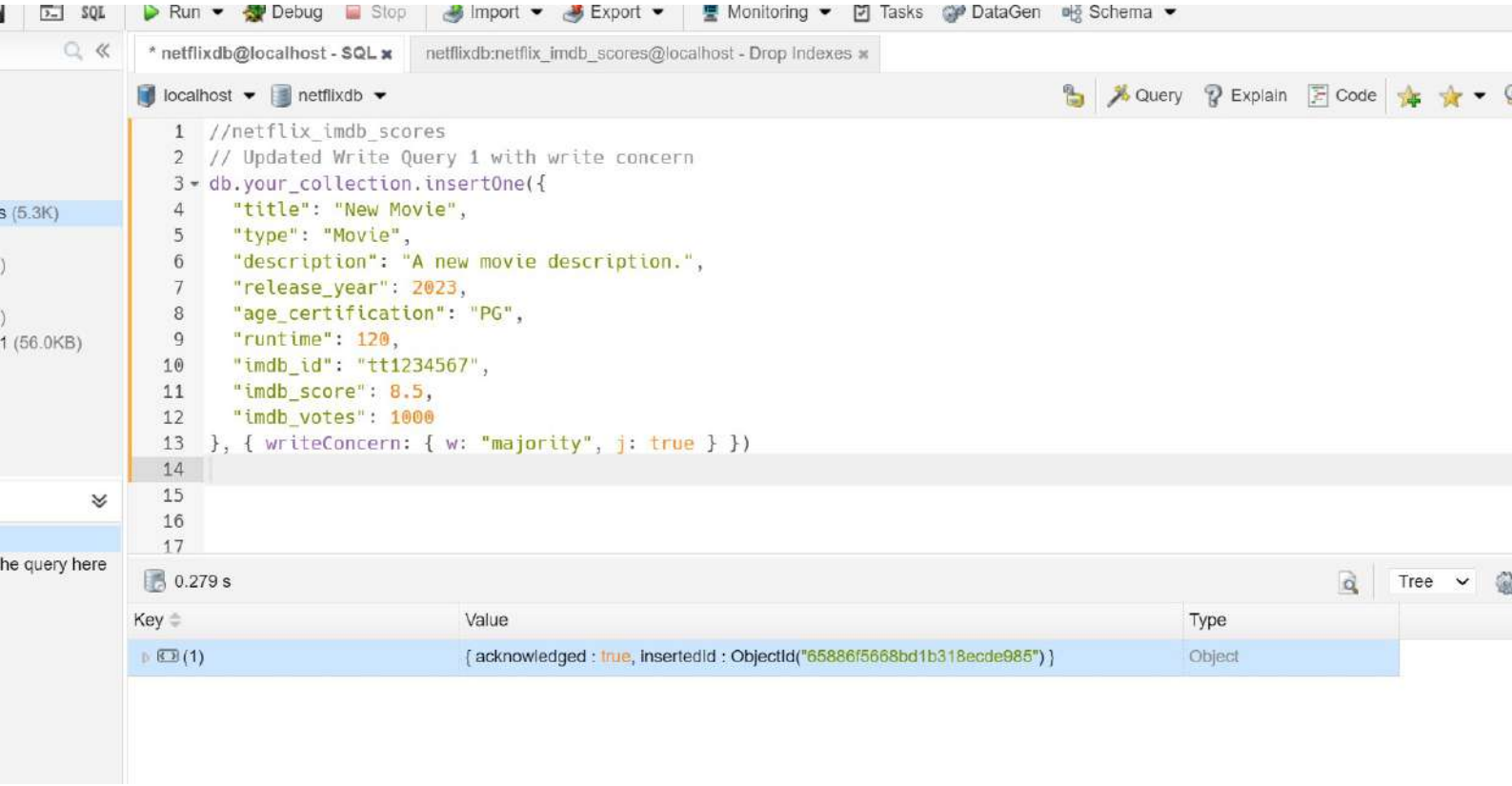

netflixdb@localhost - SQL
netflixdb:netflix_imdb_scores@localhost - Drop Indexes
localhost
netflixdb
Query
Explain
1 //netflix_imdb_scores
2 // Query without index: Find titles with IMDb score greater than or equal to 8.0
3 db.netflix_imdb_scores.find({ "imdb_score": { $gte: 8.0 } }).explain("executionStats")
4
5
6
0.076 s Query Execution Statistics of the Winning Plan
Documents Returned: 505
Actual Query Execution Time (ms): 25 ms
Index Keys Examined: 0
Sorted in Memory: false
Documents Examined: 5283
No index available for this query.
STAGE: COLLSCAN
nReturned: 505
Execution Time: 7 ms 100.0%
Documents Examined 5283
Details


Open Connections
localhost
admin
config
local
netflixdb (3)
netflix_imdb_scores (5.3K)
schema
validator (empty)
indexes (2)
_id_ (68.0KB)
imdb_score_1 (56.0KB)
your_collection (0)
users
users
My Queries
Samples
My Queries (empty)
Press Ctrl+S to save the query here
* netflixdb@localhost - SQL
netflixdb:netflix_imdb_scores@localhost - Drop Indexes
localhost
netflixdb
Query
Explain
Code
2 // Query without index: Find titles with IMDb score greater than or equal to 8.0
3 //db.netflix_imdb_scores.find({ "imdb_score": { $gte: 8.0 } }).explain("executionStats")
4 // Create an index on the 'imdb_score' field
5 //db.netflix_imdb_scores.createIndex({ "imdb_score": 1 })
6 // Query after creating index: Find titles with IMDb score greater than or equal to 8.0
7 db.netflix_imdb_scores.find({ "imdb_score": { $gte: 8.0 } }).explain("executionStats")
8
0.028 s Query Execution Statistics of the Winning Plan
Documents Returned: 505
Actual Query Execution Time (ms): 2 ms
Index Keys Examined: 505
Sorted in Memory: false
Documents Examined: 505
Query used the following index: REGULAR imdb_score_1
STAGE: FETCH
nReturned: 505
Execution Time: 0 ms
Details
STAGE: IXSCAN


Run
Debug
Stop
Import
Export
Monitoring
Tasks
DataGen
Schema
* netflixdb@localhost - SQL
netflixdb:netflix_imdb_scores@localhost - Drop Indexes
localhost
netflixdb
Query
Explain
Code
1 //netflix_imdb_scores
2 // Example specifying a read preference in Read Query 1
3 db.netflix_imdb_scores.find().sort({ "imdb_score": -1 }).limit(10).readPref("primary")
4
5
6
7
netflix_imdb_scores
0.174 s
10 Docs
50
p. 1
1 - 10
Tree
Key
Value
Type
(1) 6587f6f8ce5634dafc6aaabb { id : "ts265844", title : "#ABtalks", type : "SHOW" } (12 fields) Document
(2) 6587f6f7ce5634dafc6aa246 { id : "ts160526", title : "Khawatir", type : "SHOW" } (12 fields) Document
(3) 6587f6f7ce5634dafc6aa0bd { id : "ts4", title : "Breaking Bad", type : "SHOW" } (12 fields) Document
(4) 6587f6f8ce5634dafc6aade5 { id : "ts90621", title : "Kota Factory", type : "SHOW" } (12 fields) Document
(5) 6587f6f8ce5634dafc6aac20 { id : "ts85398", title : "Our Planet", type : "SHOW" } (12 fields) Document
(6) 6587f6f7ce5634dafc6aa0cd { id : "ts3371", title : "Avatar: The Last Airbender", type : "SHOW" } (12 fields) Document
(7) 6587f6f8ce5634dafc6ab1ec { id : "ts296563", title : "Who Rules The World", type : "SHOW" } (12 fields) Document
(8) 6587f6f7ce5634dafc6aa779 { id : "ts78298", title : "My Mister", type : "SHOW" } (12 fields) Document
(9) 6587f6f7ce5634dafc6aa51a { id : "ts37660", title : "Reply 1988", type : "SHOW" } (12 fields) Document
(10) 6587f6f8ce5634dafc6ab125 { id : "ts222333", title : "Arcane", type : "SHOW" } (12 fields) Document
Samples
Ctrl+S to save the query here


SQL
Run
Debug
Stop
Import
Export
Monitoring
Tasks
DataGen
Schema
* netflixdb@localhost - SQL
netflixdb:netflix_imdb_scores@localhost - Drop Indexes
localhost
netflixdb
Query
Explain
Code
//netflix_imdb_scores
// Updated Write Query 1 with write concern
db.your_collection.insertOne({
"title": "New Movie",
"type": "Movie",
"description": "A new movie description.",
"release_year": 2023,
"age_certification": "PG",
"runtime": 120,
"imdb_id": "tt1234567",
"imdb_score": 8.5,
"imdb_votes": 1000
}, { writeConcern: { w: "majority", j: true } })
0.279 s
Tree
Key
Value
Type
(1)
{ acknowledged : true, insertedId : ObjectId("65886f5668bd1b318ecde985") }
Object