Indexes

1. Check the Query Execution

```
> db.mybookstore.find({store:"cine book office"}).explain()
```

```
{
      "cursor" : "BasicCursor",
      "isMultiKey" : false,
      "n" : 2,
      "nscannedObjects" : 12,
      "nscanned" : 12,
      "nscannedObjectsAllPlans" : 12,
      "nscannedAllPlans" : 12,
      "scanAndOrder" : false,
      "indexOnly" : false,
      "nYields" : 0,
      "nChunkSkips" : 0,
      "millis" : 0,
      "server" : "localhost.localdomain:27017",
      "filterSet" : false
}
```

2. Now, apply a single filed index

```
> db.mybookstore.ensureIndex({store:1})
```

```
{
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "ok" : 1
}
```

3. Now check the difference in the query execution

```
> db.mybookstore.find({store:"cine book office"}).explain()
```

```
{
      "cursor" : "BtreeCursor store_1",
      "isMultiKey" : false,
      "n" : 2,
      "nscannedObjects" : 2,
      "nscanned" : 2,
      "nscannedObjectsAllPlans" : 2,
      "nscannedAllPlans" : 2,
      "scanAndOrder" : false,
      "indexOnly" : false,
      "nYields" : 0,
      "nChunkSkips" : 0,
      "millis" : 2,
      "indexBounds" : {
            "store" : [
                  [
                        "cine book office",
                        "cine book office"
                  ]
            ]
      },
      "server" : "localhost.localdomain:27017",
      "filterSet" : false
}
```

4. Check the query execution using two fields

```
> db.mybookstore.find({store:"cine book office",author:"george
haligs"}).explain()
```

```
{
      "cursor" : "BtreeCursor store_1",
      "isMultiKey" : false,
      "n" : 1,
      "nscannedObjects" : 2,
      "nscanned" : 2,
      "nscannedObjectsAllPlans" : 2,
      "nscannedAllPlans" : 2,
      "scanAndOrder" : false,
      "indexOnly" : false,
      "nYields" : 0,
      "nChunkSkips" : 0,
      "millis" : 1,
      "indexBounds" : {
            "store" : [
                  [
                        "cine book office",
                        "cine book office"
                  ]
            ]
      },
      "server" : "localhost.localdomain:27017",
      "filterSet" : false
}
```

5. Now apply compound index

```
> db.mybookstore.ensureIndex({store:1,author:1})
```

```
{
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 2,
      "numIndexesAfter" : 3,
      "ok" : 1
}
```

6. Now check the difference in the query execution

```
> db.mybookstore.find({store:"cine book office",author:"george
haligs"}).explain()
```

```
{
      "cursor" : "BtreeCursor store_1_author_1",
      "isMultiKey" : false,
      "n" : 1,
      "nscannedObjects" : 1,
      "nscanned" : 1,
      "nscannedObjectsAllPlans" : 3,
      "nscannedAllPlans" : 3,
      "scanAndOrder" : false,
      "indexOnly" : false,
      "nYields" : 0,
      "nChunkSkips" : 0,
      "millis" : 1,
      "indexBounds" : {
            "store" : [
                  [
                        "cine book office",
                        "cine book office"
                  ]
            ],
            "author" : [
                  [
                        "george haligs",
                        "george haligs"
                  ]
            ]
      },
      "server" : "localhost.localdomain:27017",
      "filterSet" : false
}
```

7. Drop all indexes on a collection

```
> db.mybookstore.dropIndexes()
```

```
{
      "nIndexesWas" : 3,
      "msg" : "non-_id indexes dropped for collection",
      "ok" : 1
}
```

8. Create a unique index

```
> db.mybookstore.ensureIndex({title:1},{unique:true})
```

```
{
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "ok" : 1
}
```

9. Observe the problem if the unique index is not sparse

```
>db.mybookstore.insert({_id:15,category:["laughter","suspense"],publis
her:{name:"nesco",city:"pune"},author:"george"})
```

```
WriteResult({
      "nInserted" : 0,
      "writeError" : {
            "code" : 11000,
            "errmsg" : "insertDocument :: caused by :: 11000 E11000
duplicate key error index: bookstore.mybookstore.$title_1  dup key: {
: null }"
      }
})
```

10. Drop all Indexes

```
> db.mybookstore.dropIndexes()
```

```
{
      "nIndexesWas" : 2,
      "msg" : "non-_id indexes dropped for collection",
      "ok" : 1
}
```

11. Apply unique sparse index

```
> db.mybookstore.ensureIndex({title:1},{unique:true,sparse:true})
```

```
{
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "ok" : 1
}
```

12. Now try the previously failed query again

```
>db.mybookstore.insert({_id:15,category:["laughter","suspense"],publis
her:{name:"nesco",city:"pune"},author:"george"})
```

```
WriteResult({ "nInserted" : 1 })
```

13. Drop specific index

```
> db.mybookstore.dropIndex({"title":1})
```

```
{ "nIndexesWas" : 2, "ok" : 1 }
```

14. Restore all indexes

```
> db.mybookstore.reIndex()
```

```
{
	"nIndexesWas" : 1,
	"nIndexes" : 1,
	"indexes" : [
		{
			"key" : {
				"_id" : 1
			},
			"name" : "_id_",
			"ns" : "bookstore.mybookstore"
		}
	],
	"ok" : 1
}
```

15. Check all indexes on a collection

```
> db.mybookstore.getIndexes()
```

```
[
	{
		"v" : 1,
		"key" : {
			"_id" : 1
		},
		"name" : "_id_",
		"ns" : "bookstore.mybookstore"
	}
]
```

16. Apply a text index

```
> db.mybookstore.ensureIndex({title:"text"})
```

```
{
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "ok" : 1
}
```

17. Perform a text search using a text index

```
> db.mybookstore.find({$text:{$search:"spy"}}).count()
```

```
2
```

18. Drop text index

```
> db.mybookstore.dropIndex("title_text")
```

```
{ "nIndexesWas" : 2, "ok" : 1 }
```

19. Perform text search using Regex

```
> db.mybookstore.find({title:{$regex:"spy"}}).count()
```

```
2
```