**10.Aggregation Pipeline to illustrate Text search on Catalog data collection**

db.createCollection("products")

db.products.insertMany([

{

product\_id: 1,

name: "Laptop",

category: "Electronics",

price: 1200,

reviews: [

{ user: "Alice", rating: 5, comment: "Excellent!" },

{ user: "Bob", rating: 4, comment: "Very good" },

{ user: "Charlie", rating: 3, comment: "Average" }

]

},

{

product\_id: 2,

name: "Smartphone",

category: "Electronics",

price: 800,

reviews: [

{ user: "Dave", rating: 4, comment: "Good phone" },

{ user: "Eve", rating: 2, comment: "Not satisfied" },

{ user: "Frank", rating: 5, comment: "Amazing!" }

]

},

{

product\_id: 3,

name: "Laptop",

category: "Electronics",

price: 800,

reviews: [

{ user: "Damu", rating: 4, comment: "Good phone" },

{ user: "Esh", rating: 2, comment: "Not satisfied" },

{ user: "Fauda", rating: 5, comment: "Amazing!" }

]

},

{

product\_id: 4,

name: "Headphones",

category: "Accessories",

price: 150,

reviews: [

{ user: "Grace", rating: 5, comment: "Great sound" },

{ user: "Heidi", rating: 3, comment: "Okay" }

]

}

])

**1. Create a Text Index**

> db.products.createIndex({

name: "text",

category: "text",

"reviews.comment": "text"

});

**2. Construct the Aggregation Pipeline**

>db.products.aggregate([

{

$match: {

$text: { $search: "Excellent" }

}

},

{

$project: {

product\_id: 1,

name: 1,

category: 1,

price: 1,

reviews: 1,

score: { $meta: "textScore" }

}

},

{

$sort: {

score: -1

}

}

])

### Restrictions

When using $text in the aggregation pipeline, be aware of the following restrictions:

* The $match stage containing $text must be the first stage in the pipeline.
* Only one $text expression is allowed in the $match stage.
* The $text operator cannot be used within $or or $not expressions.
* By default, the text search does not return documents in order of relevance score; you must explicitly sort by the textScore.