

## Program 10

**Develop an aggregation pipeline to illustrate Text search on Catalog data collection.**

Create a database **TextDB** and collection **catalog** in Mongo IDE.

The screenshot shows the MongoDB Compass interface. On the left, the 'Databases' list is visible with a '+' icon highlighted by a red box and the number 1. A modal dialog titled 'Create Database' is open in the center. It contains two input fields: 'Database Name' with the value 'TextDB' (highlighted by a red box and the number 2) and 'Collection Name' with the value 'catalog' (highlighted by a red box and the number 3). Below these fields is an unchecked checkbox for 'Time-Series'. At the bottom of the dialog are 'Cancel' and 'Create Database' buttons. The 'Create Database' button is highlighted in green.

Add the following documents in the **catalog** collection in MongoDB Shell.

```
{
  "name": "Apple iPhone 14",
  "description": "Latest model of iPhone with advanced features",
  "category": "Electronics"
}
```

Insert Document

To collection TextDB.catalog

VIEW

```
1  /**
2  * Paste one or more documents here
3  */
4  {
5    "name": "Apple iPhone 14",
6    "description": "Latest model of iPhone with advanced featu
7    "category": "Electronics"
8  }
9
```

Cancel

Insert

```
{
  "name": "Samsung Galaxy S21",
  "description": "Newest Samsung smartphone with great camera",
  "category": "Electronics"
}
```

Insert Document

To collection TextDB.catalog

VIEW

```
1  /**
2  * Paste one or more documents here
3  */
4  {
5    "name": "Samsung Galaxy S21",
6    "description": "Newest Samsung smartphone with great
7    "category": "Electronics"
8  }
9
```

Cancel

Insert

```
{  
  
  "name": "Sony Headphones",  
  
  "description": "Noise-cancelling headphones for immersive sound",  
  
  "category": "Audio"  
  
}
```



### In MongoShell

```
>use TextDB
```

### Create a Text Index

First, create a text index on the 'name' and 'description' fields.

```
db.catalog.createIndex({ name: "text", description: "text" });
```

Output:

```
< name_text_description_text
```

## Define the Aggregation Pipeline

Now, create an aggregation pipeline to perform the text search and process the results. Below is an example pipeline:

```
db.catalog.aggregate([  
  { $match: {  
    $text: { $search: " Apple iPhone 14" } }  
  },  
  { $project: {  
    name: 1,  
    description: 1,  
    category: 1,  
    score: { $meta: "textScore" }  
  }  
  },  
  {  
    $sort: {  
      score: { $meta: "textScore" }  
    }  
  },  
  {  
    $limit: 10  
  }  
]);
```

**Output:**

```
< {  
  _id: ObjectId('6684f0ff8dcd2f606e5e6243'),  
  name: 'Apple iPhone 14',  
  description: 'Latest model of iPhone with advanced features',  
  category: 'Electronics',  
  score: 2.5999999999999996  
}
```

SN AIML ATME

SN AIML ATME

SN AIML ATME