

NAME :- Arya V. More .

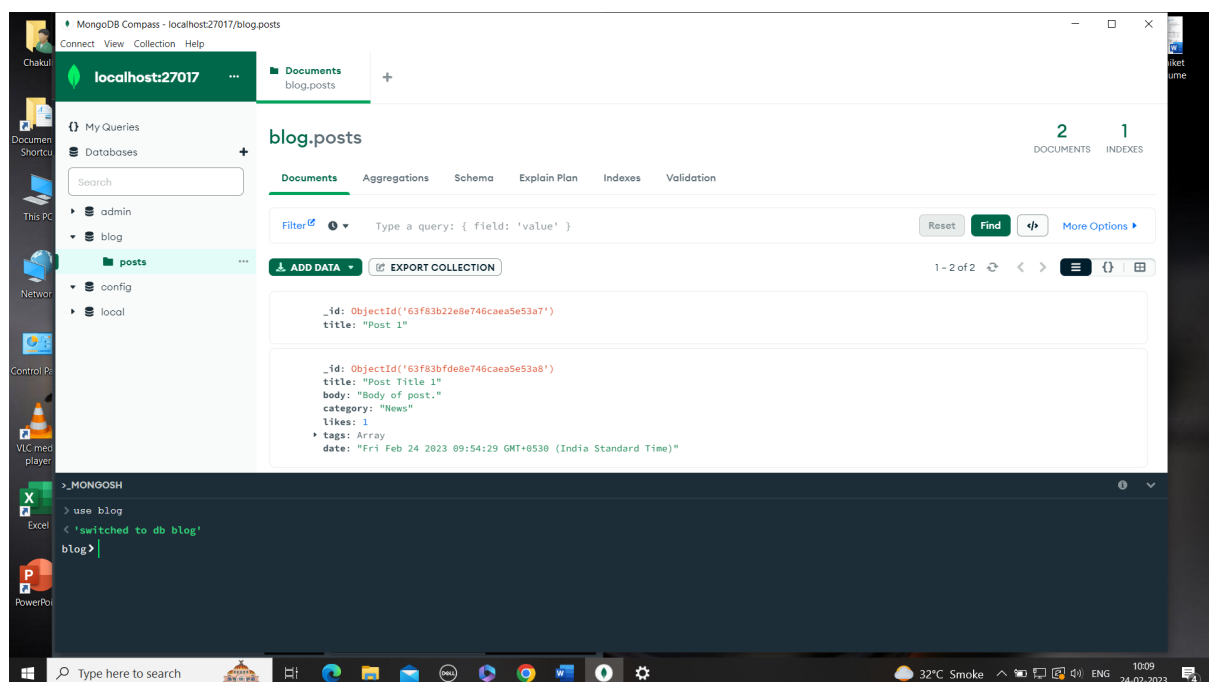
ROLL NO. :- 527

Mongodb-Performing queries

Mongodb:-

MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables.

STEP1 :- You can change or create a new database by typing **use** then the name of the database.



To insert a single document, use the `insertOne()` method.

The screenshot shows the MongoDB Compass interface for the `blog.posts` collection. The left sidebar shows the database structure with `posts` selected. The main panel displays the `blog.posts` collection with 2 documents and 1 index. The first document is visible:

```
{
  "_id": ObjectId("63f83b22e8e746cae5e53a7"),
  "title": "Post 1"
}
```

The second document is partially visible below it:

```
{
  "_id": ObjectId("63f83bfde8e746cae5e53a8"),
  "title": "Post Title 1",
  "body": "Body of post.",
  "category": "News",
  "likes": 1,
  "tags": Array,
  "date": "Fri Feb 24 2023 09:54:29 GMT+0530 (India Standard Time)"
}
```

Below the interface, a terminal window shows the command used to insert the document:

```
> use blog
> 'switched to db blog'
> db.posts.insertOne({title: "Post Title 1",
  body: "Body of post.",
  category: "News",
  likes: 1,
  tags: ["news", "events"],
  date: Date()})
```

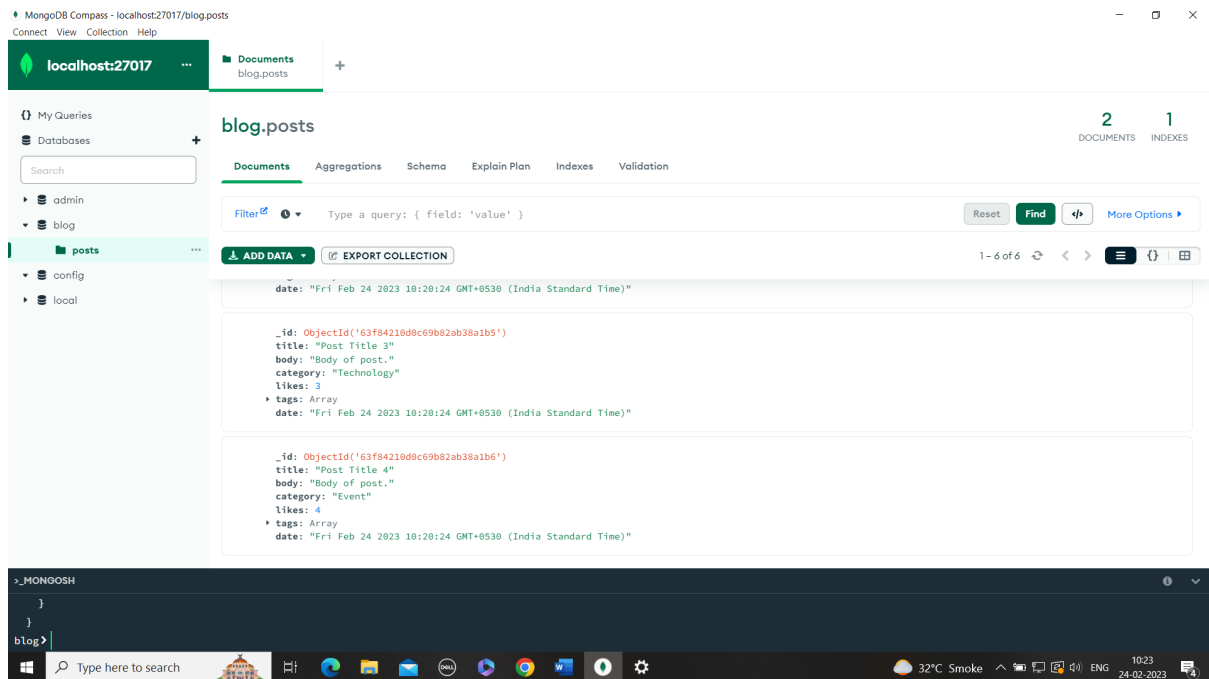
To insert multiple documents at once, use the `insertMany()` method.

The screenshot shows the MongoDB Compass interface for the `blog.posts` collection. The left sidebar shows the database structure with `posts` selected. The main panel displays the `blog.posts` collection with 2 documents and 1 index. The first document is visible:

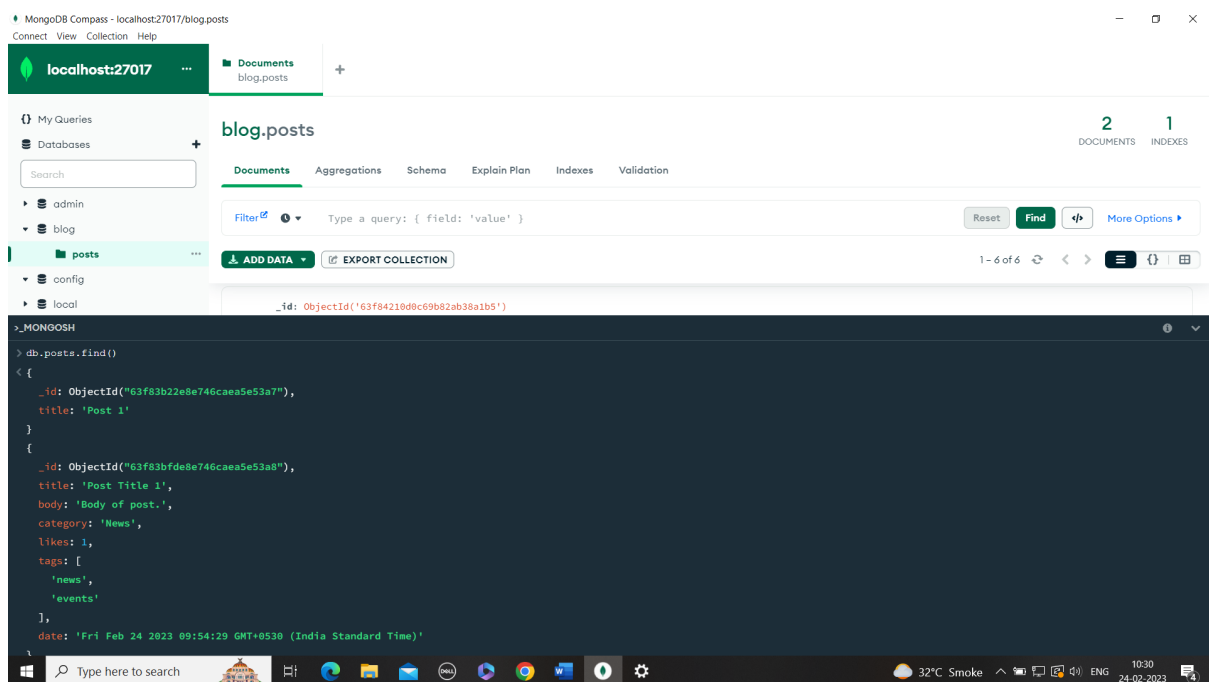
```
{
  "title": "Post Title 2",
  "body": "Body of post.",
  "category": "Event",
  "likes": 2,
  "tags": ["news", "events"],
  "date": Date()
},
{
  "title": "Post Title 3",
  "body": "Body of post.",
  "category": "Technology",
  "likes": 3,
  "tags": ["news", "events"],
  "date": Date()
},
{
  "title": "Post Title 4",
```

Below the interface, a terminal window shows the command used to insert multiple documents:

```
> db.posts.insertMany([
  {
    title: "Post Title 2",
    body: "Body of post.",
    category: "Event",
    likes: 2,
    tags: ["news", "events"],
    date: Date()
  },
  {
    title: "Post Title 3",
    body: "Body of post.",
    category: "Technology",
    likes: 3,
    tags: ["news", "events"],
    date: Date()
  },
  {
    title: "Post Title 4",
```



To select data from a collection in MongoDB, we can use the `find()` method.



To select only one document, we can use the `findOne()` method.

The screenshot shows the MongoDB Compass interface for the 'blog.posts' collection. The left sidebar shows the database structure with 'posts' selected. The main panel displays two documents in a list view. The first document has a title 'Post Title 3' and 3 likes. The second document has a title 'Post Title 4' and 4 likes. The bottom terminal window shows the command `db.posts.find()` being executed, returning an array of the two documents.

The screenshot shows the MongoDB Compass interface for the 'blog.posts' collection. The left sidebar shows the database structure with 'posts' selected. The main panel displays two documents in a list view. The second document, 'Post Title 4', is highlighted. The bottom terminal window shows the command `db.posts.findOne()` being executed, returning a single document object with the title 'Post 1'.

To query, or filter, data we can include a query in our `find()` or `findOne()` methods.

MongoDB Compass - localhost:27017/blog.posts

Connect View Collection Help

localhost:27017 Documents blog.posts

My Queries Databases Search

posts config local

blog.posts 2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 6 of 6

```
{
  "body": "Body of post.",
  "category": "Technology",
  "likes": 3,
  "tags": Array,
  "date": "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
}
```

```
{
  "_id": ObjectId('63f84210d8c69b82ab38a1b6'),
  "title": "Post Title 4",
  "body": "Body of post.",
  "category": "Event",
  "likes": 4,
  "tags": Array,
  "date": "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
}
```

>_MONGOSH

```
}
> db.posts.find( {category: "News"} )
< [
  {
    "_id": ObjectId("63f83bfde8e746caea5e53a8"),
    "title": "Post Title 1",
    "body": "Body of post.",
    "category": "News",
  }
]
```

Type here to search 32°C Smoke ENG 10:38 24-02-2023

This example will only display the **title** and **date** fields in the results.

MongoDB Compass - localhost:27017/blog.posts

Connect View Collection Help

localhost:27017 Documents blog.posts

My Queries Databases Search

posts config local

blog.posts 2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 6 of 6

```
{
  "body": "Body of post.",
  "category": "Technology",
  "likes": 3,
  "tags": Array,
  "date": "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
}
```

```
{
  "_id": ObjectId('63f84210d8c69b82ab38a1b6'),
  "title": "Post Title 4",
  "body": "Body of post.",
  "category": "Event",
  "likes": 4,
  "tags": Array,
  "date": "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
}
```

>_MONGOSH

```
}
> db.posts.find({}, {title: 1, date: 1})
< [
  {
    "_id": ObjectId("63f83b22e8e746caea5e53a7"),
    "title": "Post 1",
  }
]
```

Type here to search 32°C Smoke ENG 10:40 24-02-2023

We will get an error if we try to specify both 0 and 1 in the same object.

MongoDB Compass - localhost:27017/blog.posts

Connect View Collection Help

localhost:27017 Documents blog.posts

My Queries Databases

Search

admin blog posts config local

blog.posts 2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 6 of 6

```
body: "Body of post."
category: "Technology"
likes: 3
tags: Array
date: "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"

_id: ObjectId('63f84210d0c69b82ab38a1b6')
title: "Post Title 4"
body: "Body of post."
category: "Event"
likes: 4
tags: Array
date: "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
```

>_MONGOSH

```
}
> db.posts.find( { title: "Post Title 1" } )
< {
  _id: ObjectId("63f83bfde8e746caea5e53a8"),
  title: 'Post Title 1',
  body: 'Body of post.',
  category: 'News',
}
```

MongoDB Compass - localhost:27017/blog.posts

Connect View Collection Help

localhost:27017 Documents blog.posts

My Queries Databases

Search

admin blog posts config local

blog.posts 2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

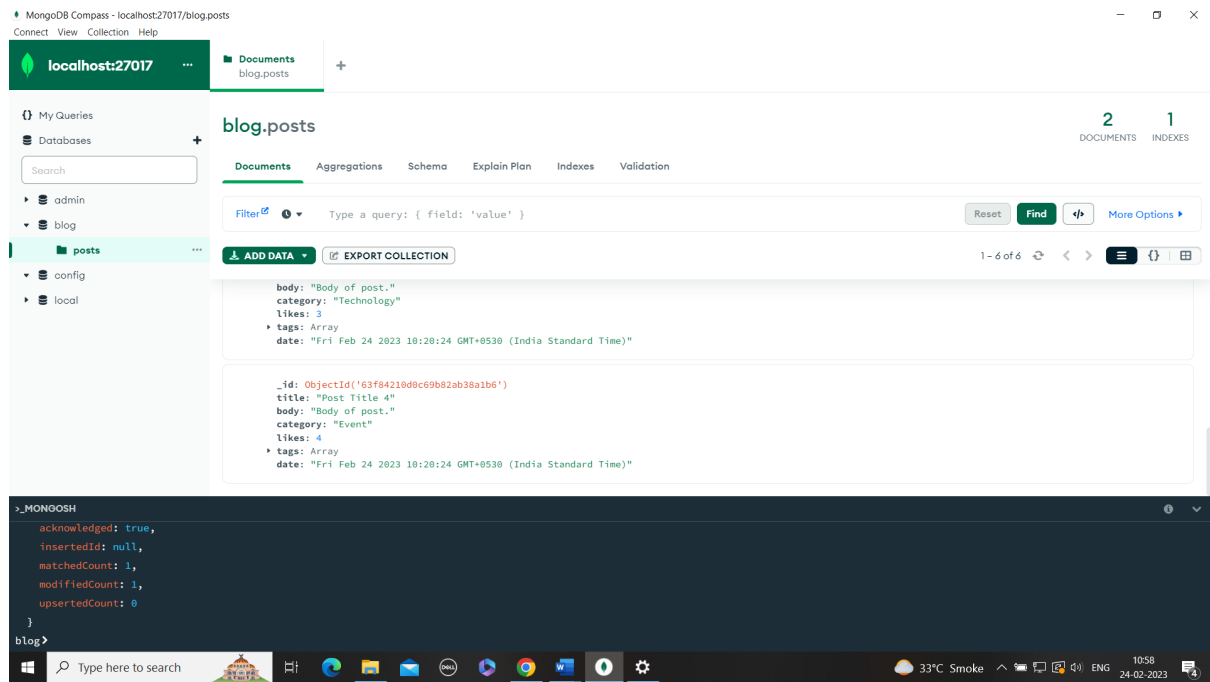
1 - 6 of 6

```
body: "Body of post."
category: "Technology"
likes: 3
tags: Array
date: "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"

_id: ObjectId('63f84210d0c69b82ab38a1b6')
title: "Post Title 4"
body: "Body of post."
category: "Event"
likes: 4
tags: Array
date: "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
```

>_MONGOSH

```
}
> db.posts.find( { title: "Post Title 1" } )
< {
  _id: ObjectId("63f83bfde8e746caea5e53a8"),
  title: 'Post Title 1',
  body: 'Body of post.',
  category: 'News',
}
```



If you would like to insert the document if it is not found, you can use the **upsert** option.

MongoDB Compass - localhost:27017/blog.posts

Connect View Collection Help

localhost:27017 Documents blog.posts

My Queries Databases Search

posts

blog.posts 2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 6 of 6

```
{
  category: "Event",
  likes: 4,
  tags: Array,
  date: "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
}
```

```
>_MONGOOSH
> db.posts.updateOne(
  { title: "Post Title 5" },
  {
    $set: {
      title: "Post Title 5",
      body: "Body of post.",
      category: "Event",
      likes: 5,
      tags: ["news", "events"],
      date: Date()
    }
  },
  { upsert: true }
)
```

Type here to search 33°C Smoke ENG 11:03 24-02-2023

MongoDB Compass - localhost:27017/blog.posts

Connect View Collection Help

localhost:27017 Documents blog.posts

My Queries Databases Search

posts

blog.posts 2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION

1 - 7 of 7

```
{
  tags: Array,
  date: "Fri Feb 24 2023 10:20:24 GMT+0530 (India Standard Time)"
}
```

```
{
  _id: ObjectId('63f84b8d3938ec14d902418'),
  title: "Post Title 5",
  body: "Body of post.",
  category: "Event",
  date: "Fri Feb 24 2023 11:00:41 GMT+0530 (India Standard Time)",
  likes: 5,
  tags: Array
}
```

```
>_MONGOOSH
{
  acknowledged: true,
  insertedId: ObjectId("63f84b8d3938ec14d902418"),
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 1
}
blog>
```

Type here to search 33°C Smoke ENG 11:07 24-02-2023

The `updateMany()` method will update all documents that match the provided query.

The screenshot shows the MongoDB Compass interface for the `localhost:27017/blog.posts` database. The left sidebar shows the database structure with `posts` selected. The main panel displays the `blog.posts` collection with 2 documents and 1 index. The `Documents` tab is active, showing a list of documents. The first document is expanded, showing its fields: `tags`, `date`, `_id`, `title`, `body`, `category`, `date`, `likes`, and `tags`. Below the document list, the `MONGOSH` terminal shows the command `db.posts.updateMany({}, { $inc: { likes: 1 } })` and its output, which indicates that 7 documents were matched and updated.

```
>_MONGOSH
> db.posts.updateMany({}, { $inc: { likes: 1 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 7,
  modifiedCount: 7,
  upsertedCount: 0
}
blog>
```

The `deleteOne()` method will delete the first document that matches the query provided.

The screenshot shows the MongoDB Compass interface for the `localhost:27017/blog.posts` database. The left sidebar shows the database structure with `posts` selected. The main panel displays the `blog.posts` collection with 2 documents and 1 index. The `Documents` tab is active, showing a list of documents. The first document is expanded, showing its fields: `tags`, `date`, `_id`, `title`, `body`, `category`, `likes`, and `tags`. Below the document list, the `MONGOSH` terminal shows the command `db.posts.deleteOne({ title: "Post Title 5" })` and its output, which indicates that 1 document was deleted.

```
>_MONGOSH
  modifiedCount: 7,
  upsertedCount: 0
}
> db.posts.deleteOne({ title: "Post Title 5" })
{
  acknowledged: true,
  deletedCount: 1
}
blog>
```

STEP :- The `deleteMany()` method will delete all documents that match the query provided.

The screenshot displays the MongoDB Compass interface for the 'blog.posts' collection on 'localhost:27017'. The left sidebar shows the database structure with 'posts' selected. The main panel shows the 'Documents' tab with a list of documents. The first document is expanded, showing its fields: '_id', 'tags', 'title', 'body', 'category', 'likes', and 'date'. Below the document list, a terminal window shows the execution of the `deleteMany()` command. The command is `db.posts.deleteMany({ category: "Technology" })`. The terminal output shows the command was acknowledged and the deleted count is 1.

```
>_MONGOSH
acknowledged: true,
deletedCount: 1
}
> db.posts.deleteMany({ category: "Technology" })
< {
  acknowledged: true,
  deletedCount: 0
}
blog>
```