MongoDB Insert Documents

1. How to Insert Documents?

You can insert documents into MongoDB collections using:

- Your programming language driver (e.g., Node.js, Python, Java)
- MongoDB Atlas UI (web interface)
- MongoDB Compass (GUI tool)

Method	Tool/Language	Example
Code (Python)	pymongo	insert_one({ name: "Alice" })
Code (Node.js)	mongodb package	insertOne({ name: "Alice" })
Web Interface	MongoDB Atlas	Click "Insert Document"
Desktop GUI	MongoDB Compass	Use visual editor to insert

2. Collection Creation

In **MongoDB**, a **collection** is like a folder that holds documents (like records in a database).

You **don’t need to create the collection manually**. If you try to add (insert) a document into a collection that doesn’t exist yet, **MongoDB will automatically create the collection for you**.

Example 1: Basic Insert

```
db.students.insertOne({ name: "Ali", age: 20 })
```

- Here, if students collection doesn't exist, MongoDB will create it.
- Then it will add the document: { name: "Ali", age: 20 }.

Example 2: Insert Many Documents

```
db.books.insertMany([
 { title: "Book A", author: "Author X" },
 { title: "Book B", author: "Author Y" }
])
```

- If the books collection doesn't exist, it will be created.
- Both documents will be inserted into the new books collection.

Example 3: Using a New Collection Name

```
db.orders.insertOne({ item: "Laptop", price: 1200 })
```

- Even if there was no orders collection before, this insert will **create it automatically**.

3. Insert Documents via MongoDB Atlas UI

Steps:

1. Go to **Clusters** page of your project.
2. Click **Browse Collections** on your cluster.
3. Select your **database** and then your **collection**.
4. Click **Insert Document**.
5. Paste a JSON array of documents.
6. Click **Insert** to add the documents.

Example JSON to insert multiple documents:

```
[
 { "prodId": 100, "price": 20, "quantity": 125 },
 { "prodId": 101, "price": 10, "quantity": 234 },
 { "prodId": 102, "price": 15, "quantity": 432 },
 { "prodId": 103, "price": 17, "quantity": 320 }
]
```

4. Insert a Single Document with Code

Use insertOne() method.

- insertOne() is a method used in **MongoDB** to **add one single document** (record) to a collection (like a table in SQL).
- If no _id is provided, MongoDB automatically generates a unique _id.
- insertOne() returns a promise that resolves with the result, including the new _id.

Example 1: Insert a Book

```
await db.collection('books').insertOne({
 title: 'The Hobbit',
 author: 'J.R.R. Tolkien',
 pages: 310,
 genres: ['fantasy', 'adventure']
});
```

- This adds a **book** to the books collection.

Example 2: Insert a User Profile

```
await db.collection('users').insertOne({
 name: 'Alice',
 email: 'alice@example.com',
 age: 27,
```

```
 interests: ['music', 'travel']
});
```

- Adds a **user** to the users collection.

Example 3: Insert a Product with _id

```
await db.collection('products').insertOne({
 _id: 'prod123',
 name: 'Laptop',
 price: 999.99,
 inStock: true
});
```

- This time, you **manually set _id** to 'prod123'.

5. Insert Multiple Documents with Code

Use insertMany() method.

- insertMany() is a method used in MongoDB to **insert multiple documents at once** into a collection.

Example 1: Insert Multiple Books

```
await db.collection('books').insertMany([
 {
   title: '1984',
   author: 'George Orwell',
   genre: 'dystopian'
 },
 {
   title: 'To Kill a Mockingbird',
   author: 'Harper Lee',
   genre: 'classic'
 },
 {
   title: 'Dune',
   author: 'Frank Herbert',
   genre: 'science fiction'
 }
]);
```

- Adds 3 books to the books collection.

Example 2: Insert Multiple Users

```
await db.collection('users').insertMany([
 { name: 'Bob', age: 30 },
 { name: 'Sarah', age: 24 },
 { name: 'Tom', age: 35 }
]);
```

- Adds 3 users to the users collection.

Example 3: Insert with Custom _id

```
await db.collection('products').insertMany([
 { _id: 'p1', name: 'Phone', price: 799 },
 { _id: 'p2', name: 'Tablet', price: 499 }
]);
```

- Here, you **manually give _ids** for each product.

InsertOne() vs insertMany():

Method	What it does
insertOne()	Inserts **1** document
insertMany()	Inserts **multiple** documents at once

6. Important Notes

- **Atomicity:** Each insert operation is atomic at the document level (each document is fully inserted or not at all).
- **_id Field:** MongoDB requires a unique _id field in each document. If omitted, MongoDB generates one automatically.
- **Write Acknowledgment:** You can control how MongoDB confirms your writes using "write concern" settings.

1. Atomicity

If MongoDB starts inserting a document, it will **either insert the whole document** or **insert nothing at all** if there's a problem.

- This keeps your data **safe and consistent**.

Example:

```
If you're inserting: await db.collection('orders').insertOne({
 orderId: 1,
 items: ['apple', 'banana'],
 total: 5.99
});
```

- If the insert works, the **entire document** is saved.
- If something goes wrong (e.g., invalid data), **nothing is saved**—no partial data.

2. _id Field

- _id is like a **unique ID or primary key** for each document.
- If you don't set it, MongoDB **automatically makes one**.
- If two documents have the **same _id**, the second one will **fail**.

Example 1: Let MongoDB handle it

```
await db.collection('products').insertOne({
        name: 'Book', price: 10
});
```

- MongoDB adds a unique _id like:
- { _id: ObjectId("64f2c123abc..."), name: 'Book', price: 10 }

Example 2: Set your own _id

```
await db.collection('products').insertOne({
 _id: 'prod001',
 name: 'Pen',
 price: 2
});
```

- Works fine if that _id is unique.

Example 3: Duplicate _id (Error)

```
// This will fail if 'prod001' already exists
await db.collection('products').insertOne({
 _id: 'prod001',
 name: 'Notebook',
 price: 5
});
```

- Error: Duplicate _id.

3. Write Acknowledgment (Write Concern)

- MongoDB can confirm your insert in different ways.
- You can choose to:
  - Wait until the data is **written to memory**
  - Or wait until it's written to **multiple servers** (in a replica set)
  - Or even **not wait at all** (less safe)

Example: Set write concern to wait for acknowledgment

```
await db.collection('logs').insertOne(
 { message: 'Server started' },
 { writeConcern: { w: 1 } } // Wait for confirmation from 1 node
);
```

- This means: "Only tell me it's successful **after 1 server saves it**."

Another Example: Acknowledge on multiple servers

```
await db.collection('logs').insertOne(
 { message: 'Backup complete' },
 { writeConcern: { w: 'majority' } }
);
```

- Waits until **most nodes in a replica set** confirm the insert.

7. Quick Recap with Examples

Task	Method	Example Document(s)
Insert one doc	insertOne()	{ item: 'canvas', qty: 100 }
Insert many docs	insertMany()	[{item: 'journal'}, {item: 'mat'}]

Primary Insert Methods

1. db.collection.insertOne() - Inserting One Document

To insert **one single document** (record) into a collection (like a row in a table).

```
db.collection.insertOne({ key: "value" });
```

Example

```
db.users.insertOne({
 name: "Alice",
 age: 25,
 email: "alice@example.com"
});
```

- This adds **one user** to the users collection.

Returns

```
{
 acknowledged: true,
 insertedId: ObjectId("64f...") // Automatically created _id
}
```

- acknowledged: true → MongoDB says: “Yep, I saved it!”
- insertedId → The _id of the document

Use Case

Use insertOne() when:

You want to **add just one document**
For example: Register a new user, add a product, log one event

Example 1: Add a Book

```
db.books.insertOne({
 title: "The Alchemist",
 author: "Paulo Coelho",
 pages: 208
});
```

Example 2: Add a Product

```
db.products.insertOne({
 name: "Headphones",
 price: 49.99,
 inStock: true
});
```

Recap Table

Part	Description
Method	insertOne()
Inserts	One document
Returns	Acknowledgment + the new document's _id
Use When	You need to insert a single record

2. db.collection.insertMany() - Insert Multiple Documents

Insert **multiple documents at once** into a collection.

```
db.collection.insertMany([
 { name: "Bob", age: 30 },
 { name: "Eve", age: 28 }
]);
```

You provide an **array of objects** to insert many records in a single operation.

Returns

```
{
 acknowledged: true,
 insertedIds: {
   '0': ObjectId("..."),
   '1': ObjectId("...")
 }

}
```

- acknowledged: true → MongoDB confirms success
- insertedIds → IDs generated for each inserted document.

Use Case

Use insertMany() when you want to add **lots of documents efficiently** (bulk insert).

Example 1: Insert Multiple Users

```
db.users.insertMany([
 { name: "Bob", age: 30 },
 { name: "Eve", age: 28 }
]);
```

Example 2: Insert Multiple Products

```
db.products.insertMany([
 { name: "Chair", price: 49.99 },
 { name: "Table", price: 149.99 }
]);
```

Additional Note: Upsert

Some methods (like updateOne() or updateMany()) support **upsert: true** option:

- **Update if document exists**
- **Insert if document does not exist**

✓ Summary Table

Method	Inserts	Updates	Upsert Option	Atomic	Use Case
insertOne()	Yes	No	No	Yes	Insert a single document
insertMany()	Yes	No	No	Yes	Insert multiple documents