

MongoDB Schema Design — Online Retail Platform

1. Product Catalog Design

Collection Name: products

Schema Design:

```
{  
  _id: ObjectId,  
  name: String,  
  category: String,  
  description: String,  
  price: Number,  
  stock: Number,  
  brand: String,  
  specifications: Object,  
  createdAt: ISODate,  
  updatedAt: ISODate  
}
```

Sample Documents:

```
// Example 1: Electronics Product  
{  
  
  name: "Samsung Galaxy S23",  
  category: "Mobile Phones",  
  description: "5G smartphone with 256GB storage and 8GB RAM",  
  price: 69999,  
  stock: 45,  
  brand: "Samsung",  
  specifications: {  
    color: "Phantom Black",  
    screenSize: "6.1 inches",  
    battery: "3900 mAh"  
  },  
  createdAt: new Date("2025-02-10T10:00:00Z"),  
  updatedAt: new Date("2025-03-05T14:00:00Z")  
}
```

```
// Example 2: Fashion Product
{
    name: "Men's Cotton T-Shirt",
    category: "Clothing",
    description: "Regular fit 100% cotton t-shirt",
    price: 799,
    stock: 150,
    brand: "H&M",
    specifications: {
        size: ["S", "M", "L", "XL"],
        color: "Navy Blue"
    },
    createdAt: new Date("2025-02-15T09:30:00Z"),
    updatedAt: new Date("2025-02-15T09:30:00Z")
}
```

Sample Output:

The screenshot shows the MongoDB Compass interface with the following details:

- Connections:** onlineRetailDB
- Collection:** products
- Documents:** 2
- Document 1:**

```
_id: ObjectId('690ef43de67d67a79ee44d31')
name : "Samsung Galaxy S23"
category : "Mobile Phones"
description : "5G smartphone with 256GB storage and 8GB RAM"
price : 6999
stock : 100
brand : "Samsung"
specifications : Object
createdAt : 2025-02-10T10:00:00.000+00:00
updatedAt : 2025-03-05T14:00:00.000+00:00
```
- Document 2:**

```
_id: ObjectId('690ef43de67d67a79ee44d32')
name : "Men's Cotton T-Shirt"
category : "Clothing"
description : "Regular fit 100% cotton t-shirt"
price : 799
stock : 150
brand : "H&M"
specifications : Object
createdAt : 2025-02-15T09:30:00.000+00:00
updatedAt : 2025-02-15T09:30:00.000+00:00
```

2. Customer Orders Design

Collection Name: orders

Schema Design:

```
{
```

```
_id: ObjectId,
userId: ObjectId,
orderDate: ISODate,
products: [
  { productId: ObjectId, quantity: Number, priceAtPurchase: Number }
],
totalAmount: Number,
status: String,
paymentMethod: String,
shippingAddress: {
  street: String,
  city: String,
  state: String,
  postalCode: String,
  country: String
}
}
```

Sample Document:

```
{
  _id: ObjectId("6749f3b92b8a3c11a1d4b301"),
  userId: ObjectId("6749f2f72b8a3c11a1d4b501"),
  orderDate: ISODate("2025-03-10T12:00:00Z"),
  products: [
    { productId: ObjectId("6749f2a12b8a3c11a1d4b201"), quantity: 1,
      priceAtPurchase: 69999 },
    { productId: ObjectId("6749f2a12b8a3c11a1d4b202"), quantity: 2,
      priceAtPurchase: 799 }
  ],
  totalAmount: 71597,
  status: "Delivered",
  paymentMethod: "UPI",
  shippingAddress: {
    street: "221B Baker Street",
    city: "Mumbai",
    state: "Maharashtra",
    postalCode: "400001",
    country: "India"
  }
}
```

Sample Output:

The screenshot shows the MongoDB Compass application interface. The left sidebar lists connections and databases. The main area shows the 'orders' collection under the 'onlineRetailDB' database. A specific document is selected, displaying its fields and values. The status bar at the bottom shows system information like temperature, battery level, and date.

3. User Authentication Design

Collection Name: users

Schema Design:

```
{  
  _id: ObjectId,  
  username: String,  
  email: String,  
  passwordHash: String,  
  role: String,  
  createdAt: ISODate,  
  lastLogin: ISODate  
}
```

Sample Document:

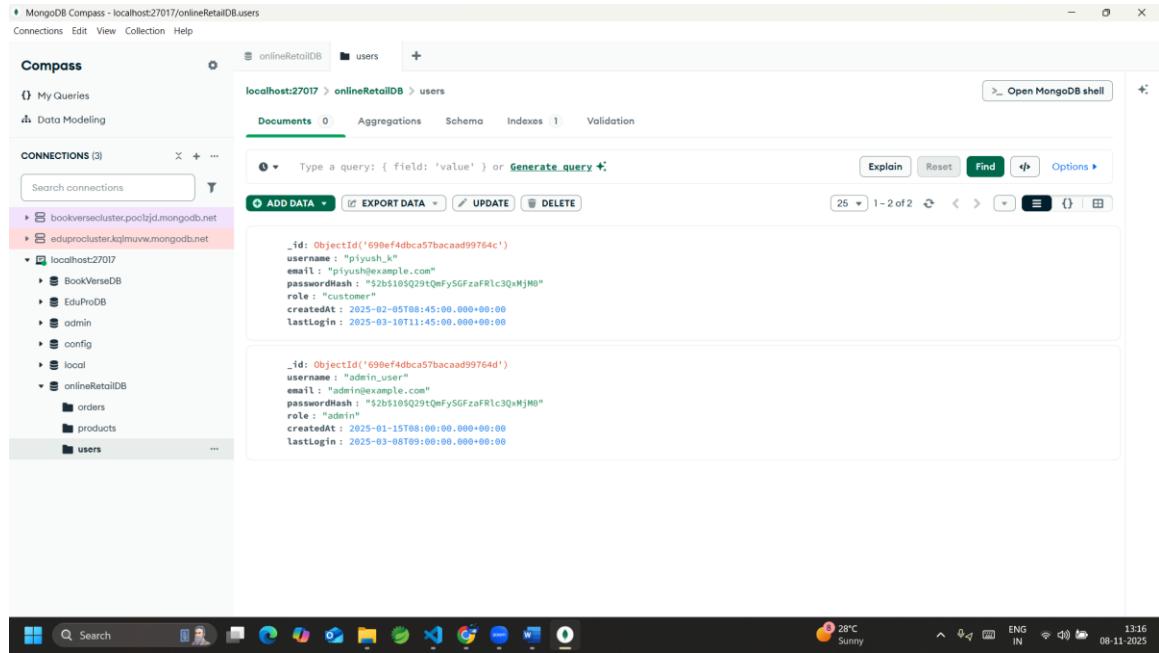
```
{  
  _id: ObjectId("6749f2f72b8a3c11a1d4b501"),  
  username: "piyush_k",  
  email: "piyush@example.com",
```

```

passwordHash: "$2b$10$Q29tQmFySGFzaFRlc3QxMjM0",
role: "customer",
createdAt: ISODate("2025-02-05T08:45:00Z"),
lastLogin: ISODate("2025-03-10T11:45:00Z")
}

```

Sample Document:



The screenshot shows the MongoDB Compass interface. The left sidebar displays connections and collections. The main area shows the 'users' collection with two documents listed. The first document is for a customer user with the ID `_id: ObjectId('690ef4dbca57bacaa99764c')`. The second document is for an admin user with the ID `_id: ObjectId('690ef4dbca57bacaa99764d')`. Both documents include fields for `username`, `email`, `passwordHash`, `role`, `createdAt`, and `lastLogin`.

```

{
  "_id": "690ef4dbca57bacaa99764c",
  "username": "piyush_k",
  "email": "piyush@example.com",
  "passwordHash": "$2b$10$Q29tQmFySGFzaFRlc3QxMjM0",
  "role": "customer",
  "createdAt": "2025-02-05T08:45:00.000+00:00",
  "lastLogin": "2025-03-10T11:45:00.000+00:00"
}

{
  "_id": "690ef4dbca57bacaa99764d",
  "username": "admin_user",
  "email": "admin@example.com",
  "passwordHash": "$2b$10$Q29tQmFySGFzaFRlc3QxMjM0",
  "role": "admin",
  "createdAt": "2025-01-15T08:00:00.000+00:00",
  "lastLogin": "2025-03-08T05:00:00.000+00:00"
}

```

4. Querying and Indexing

Indexes:

```

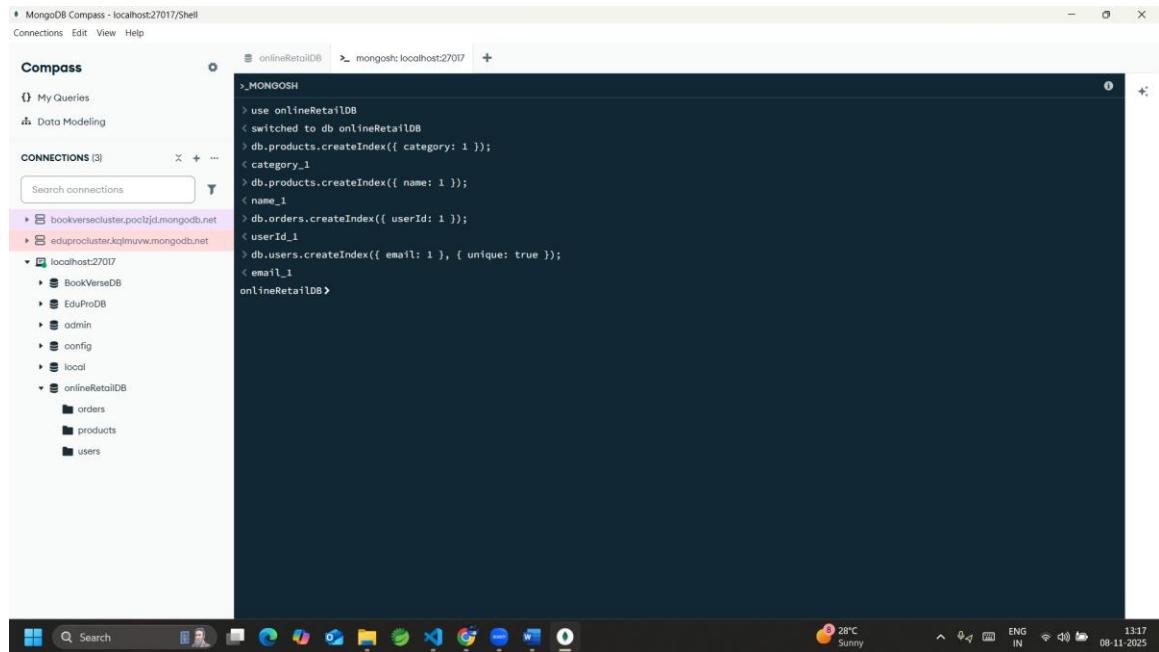
db.products.createIndex({ category: 1 });

db.products.createIndex({ name: 1 });

db.orders.createIndex({ userId: 1 });
db.users.createIndex({ email: 1 }, { unique: true });

```

Sample Document:



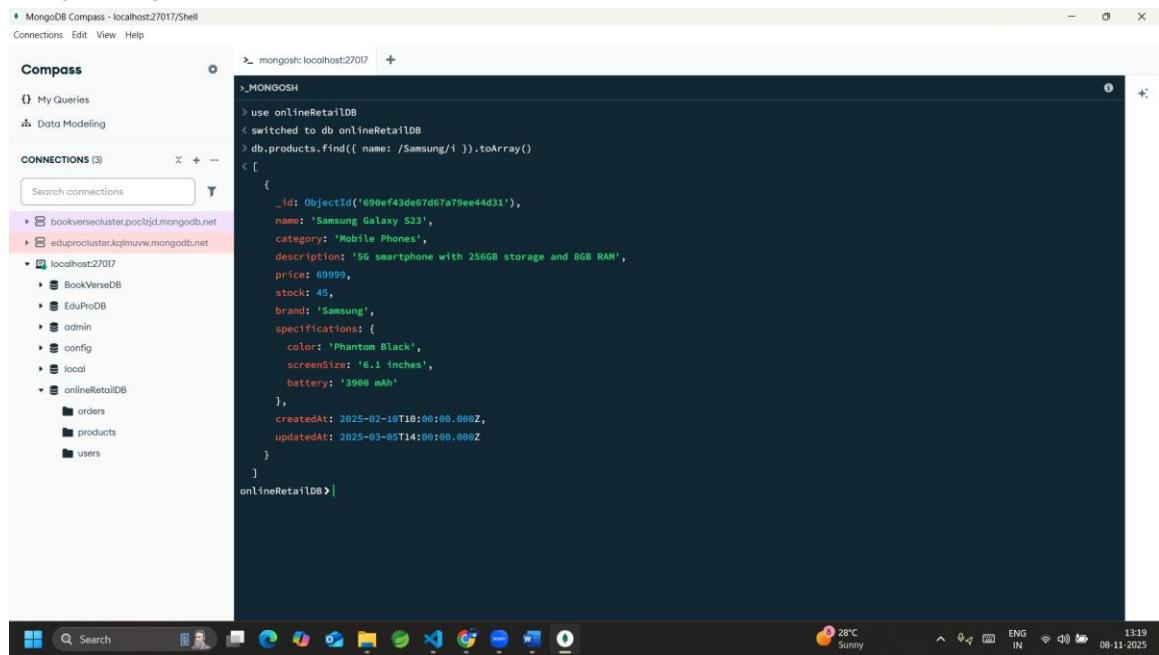
The screenshot shows the MongoDB Compass interface. On the left, the 'Compass' sidebar includes sections for 'My Queries' and 'Data Modeling'. The 'CONNECTIONS (3)' section lists three connections: 'bookversecluster.poc1zd.mongodb.net', 'eduprocluster.kqjmuvw.mongodb.net', and 'localhost:27017'. The 'localhost:27017' connection is expanded, showing databases 'BookVerseDB', 'EduProDB', 'admin', 'config', 'local', and 'onlineRetailDB'. The 'onlineRetailDB' database is selected, revealing collections 'orders', 'products', and 'users'. In the main panel, a terminal window displays the following MongoDB shell session:

```
>_MONGOOSH
> use onlineRetailDB
< switched to db onlineRetailDB
> db.products.createIndex({ category: 1 });
< category_1
> db.products.createIndex({ name: 1 });
< name_1
> db.orders.createIndex({ userId: 1 });
< userId_1
> db.users.createIndex({ email: 1 }, { unique: true });
< email_1
onlineRetailDB >
```

5. Sample Queries

```
//1. Retrieve all products from a category:
db.products.find({ category: "Mobile Phones" }).pretty();
```

Sample Output:



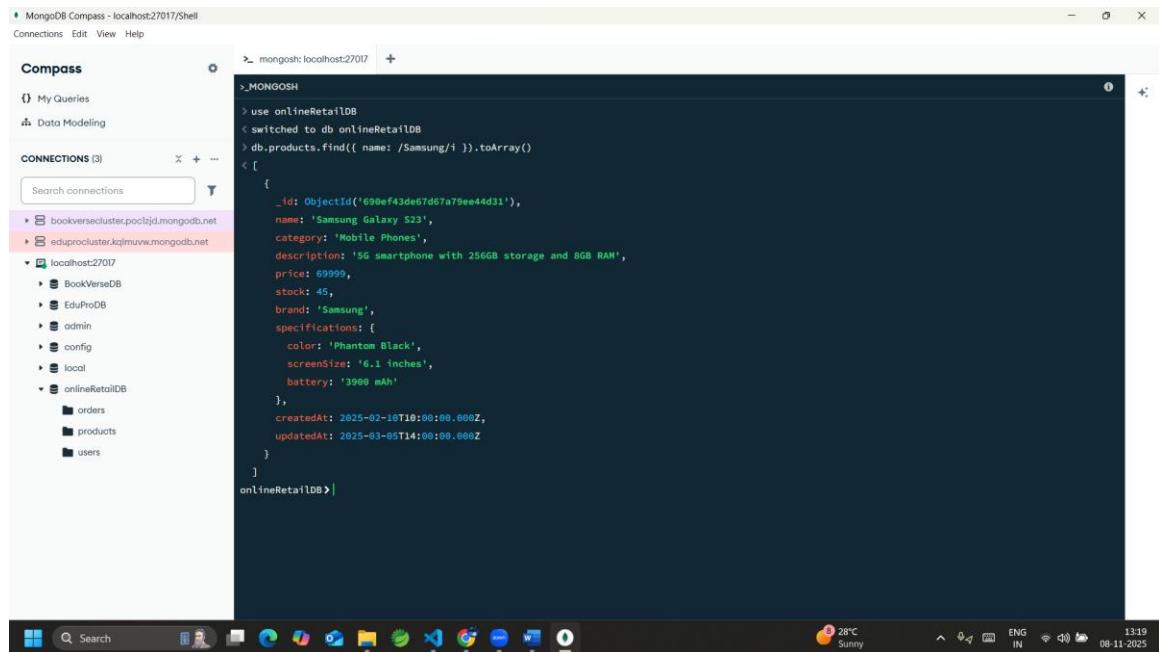
The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' sidebar lists three connections: 'bookversecluster.pocljzd.mongodb.net', 'eduprocluster.kclmuvw.mongodb.net', and 'localhost:27017'. The 'localhost:27017' connection is selected and expanded, showing databases 'BookVerseDB', 'EduProDB', 'admin', 'config', 'local', and 'onlineRetailDB'. The 'products' collection under 'onlineRetailDB' is selected. In the main query editor window, the command `db.products.find({ name: /Samsung/i }).toArray()` is run, and the result is displayed as a single document:

```
_id: ObjectId('690ef43de67d67a79ee44d31'),
name: 'Samsung Galaxy S23',
category: 'Mobile Phones',
description: '5G smartphone with 256GB storage and 8GB RAM',
price: 6999,
stock: 45,
brand: 'Samsung',
specifications: {
  color: 'Phantom Black',
  screenSize: '6.1 inches',
  battery: '3900 mAh'
},
createdAt: 2025-02-10T10:00:00.000Z,
updatedAt: 2025-03-05T14:00:00.000Z
```

//2. Find a specific product by name:

```
db.products.find({ name: /Samsung/i }, { name: 1, price: 1, stock: 1 });
```

Sample Output:



The screenshot shows the MongoDB Compass interface, identical to the previous one, but with a different query. The command `db.products.find({ name: /Samsung/i }, { name: 1, price: 1, stock: 1 })` is run, resulting in the same document as before:

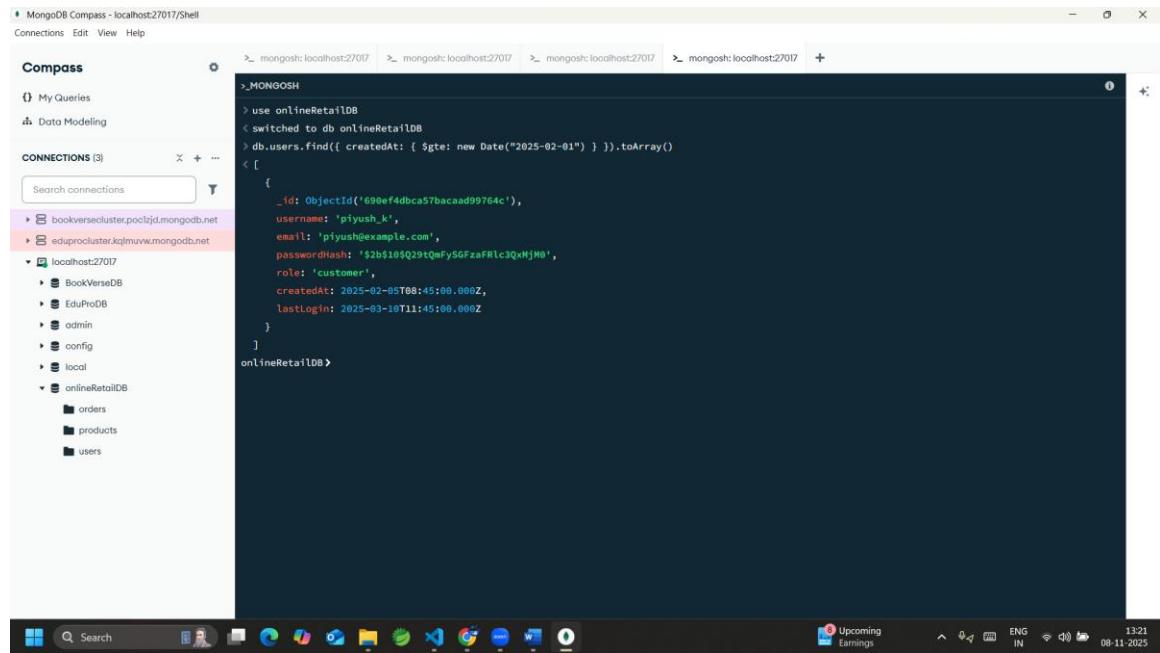
```
_id: ObjectId('690ef43de67d67a79ee44d31'),
name: 'Samsung Galaxy S23',
category: 'Mobile Phones',
description: '5G smartphone with 256GB storage and 8GB RAM',
price: 6999,
stock: 45,
brand: 'Samsung',
specifications: {
  color: 'Phantom Black',
  screenSize: '6.1 inches',
  battery: '3900 mAh'
},
createdAt: 2025-02-10T10:00:00.000Z,
updatedAt: 2025-03-05T14:00:00.000Z
```

```
//3. Retrieve all orders for a specific user:  
db.orders.find({ userId: ObjectId("690ef4dbca57bacaad99764c") });
```

Sample Output:

```
//4. Find all users registered after a date:  
db.users.find({ createdAt: { $gte: ISODate("2025-02-01") } });
```

Sample Output:

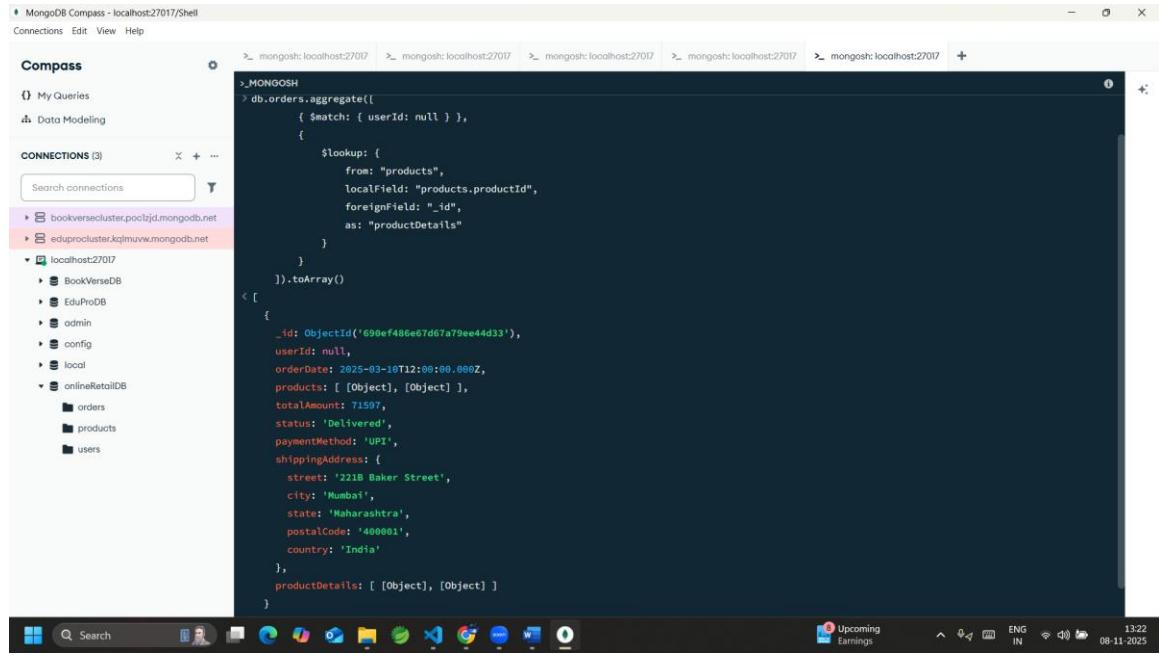


The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' sidebar lists three connections: 'bookversecluster.pocizjd.mongodb.net' (selected), 'localhost:27017' (with sub-databases BookVerseDB, EduProDB, admin, config, local, onlineRetailDB containing orders, products, and users), and 'eduprocluster.kalmuvu.mongodb.net'. The main area shows the MongoDB shell with the command: `> db.users.find({ createdAt: { $gte: new Date("2025-02-01") } }).toArray()`. The output displays one user document:

```
_id: ObjectId('690ef4dbca57bacaad99764c'),  
username: 'piyush_k',  
email: 'piyush@example.com',  
passwordHash: '$2b$10$Q29tQmFySGFzaFRlc3QxMjM0',  
role: 'customer',  
createdAt: 2025-02-05T08:45:00.000Z,  
lastLogin: 2025-03-10T11:45:00.000Z
```

```
//5. Retrieve orders with product details(Aggregation):  
db.orders.aggregate([  
    { $match: { userId: ObjectId("6749f2f72b8a3c11a1d4b501") } },  
    {  
        $lookup: {  
            from: "products",  
            localField: "products.productId",  
            foreignField: "_id",  
            as: "productDetails"  
        }  
    }  
]);
```

Sample Output:



The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' sidebar lists several databases: bookversecluster.poolzjd.mongodb.net (selected), eduprocluster.kqlmuuw.mongodb.net, localhost:27017, BookProDB, EduProDB, admin, config, local, and onlineRetailDB. The 'orders' collection under onlineRetailDB is selected. The main panel displays a MONGOSH shell query and its results. The query uses an aggregate pipeline to find orders where the userId is null, then performs a \$lookup operation to join with the 'products' collection based on the productId field. The resulting document includes fields like _id, userId, orderDate, products, totalAmount, status, paymentMethod, shippingAddress, and productDetails.

```
>_MONGOSH
> db.orders.aggregate([
  { $match: { userId: null } },
  {
    $lookup: {
      from: "products",
      localField: "products.productId",
      foreignField: "_id",
      as: "productDetails"
    }
  }
]).toArray()
< [
  {
    _id: ObjectId('690ef486e67d67a79ee44d33'),
    userId: null,
    orderDate: 2025-03-10T12:00:00.000Z,
    products: [ [Object], [Object] ],
    totalAmount: 71597,
    status: 'Delivered',
    paymentMethod: 'UPI',
    shippingAddress: {
      street: '221B Baker Street',
      city: 'Mumbai',
      state: 'Maharashtra',
      postalCode: '400001',
      country: 'India'
    },
    productDetails: [ [Object], [Object] ]
  }
]
```

6. Benefits of MongoDB for This Use Case

- Scalability: Horizontal scaling using sharding for millions of products.
- Flexibility: Supports dynamic attributes for different product types.
- Performance: Indexing optimizes search and retrieval.
- Aggregation: Efficient analytics and order summaries.
- Integration: Works seamlessly with Node.js and Mongoose ORM.