

Practical 7

Title:

Create a simple application that connects MongoDB to Power BI using an API to send data from MongoDB to Power BI for reporting

1. Introduction

Business Intelligence (BI) systems transform raw data into meaningful insights for decision-making. Data is often stored in databases like MongoDB in JSON format, whereas visualization tools like Power BI require structured tabular data.

Since **Power BI does not directly integrate efficiently with MongoDB in many local setups, an API is used as an intermediate layer. The API fetches data from MongoDB and sends it in JSON format to Power BI.**

This practical demonstrates a complete pipeline where data is stored, processed, and visualized, supporting analytical decision-making in BI systems

2. Objective

- To connect MongoDB database with Power BI
 - To develop an API using Node.js and Express
 - To transfer data using JSON format
 - To visualize data using Power BI
 - To understand BI data flow architecture
-

3. Tools and Technologies Used

- MongoDB Compass (Database management tool)
 - Node.js (Backend runtime)
 - Express.js (API framework)
 - Mongoose (MongoDB connectivity)
 - Power BI Desktop (Visualization tool)
-

4. System Architecture

Flow:

MongoDB → API (Node.js) → Power BI

Explanation:

- MongoDB stores raw data
- API retrieves and converts data into JSON
- Power BI consumes JSON and converts it into tables
- Visual reports are generated for analysis

5. Procedure (Step-by-Step Implementation)

Step 1: Connect to MongoDB Server

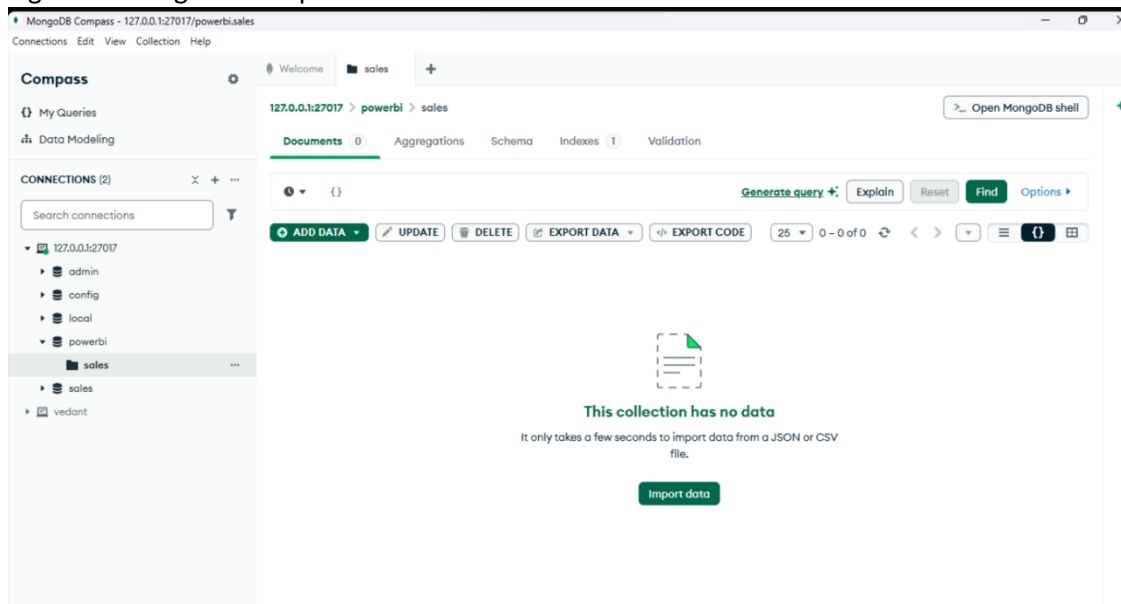
Open MongoDB Compass and connect using the following connection string:

URL : 127.0.0.1:27017

Explanation:

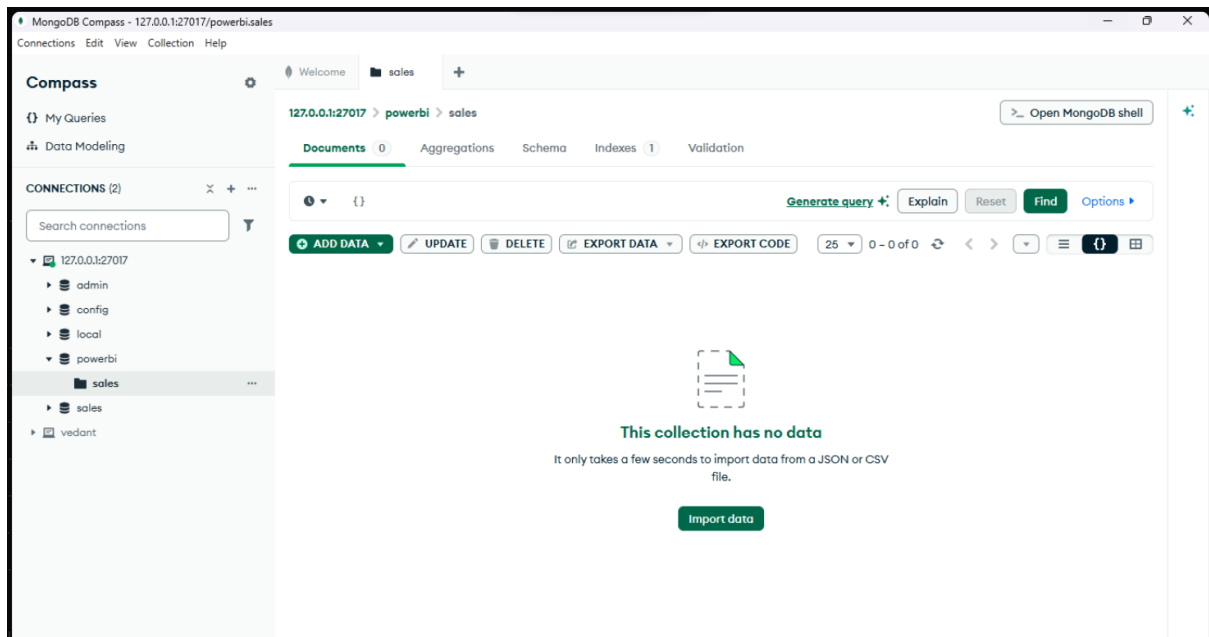
- 127.0.0.1 → Localhost (your system)
- 27017 → Default MongoDB port
- This connects Compass to the local MongoDB server

Figure 1: MongoDB Compass connected to local server



Step 2: Create Database and Collection

1. Click Create Database
2. Enter:
 - Database Name: powerbi
 - Collection Name: sales



Explanation:

The collection is created but initially contains no data.

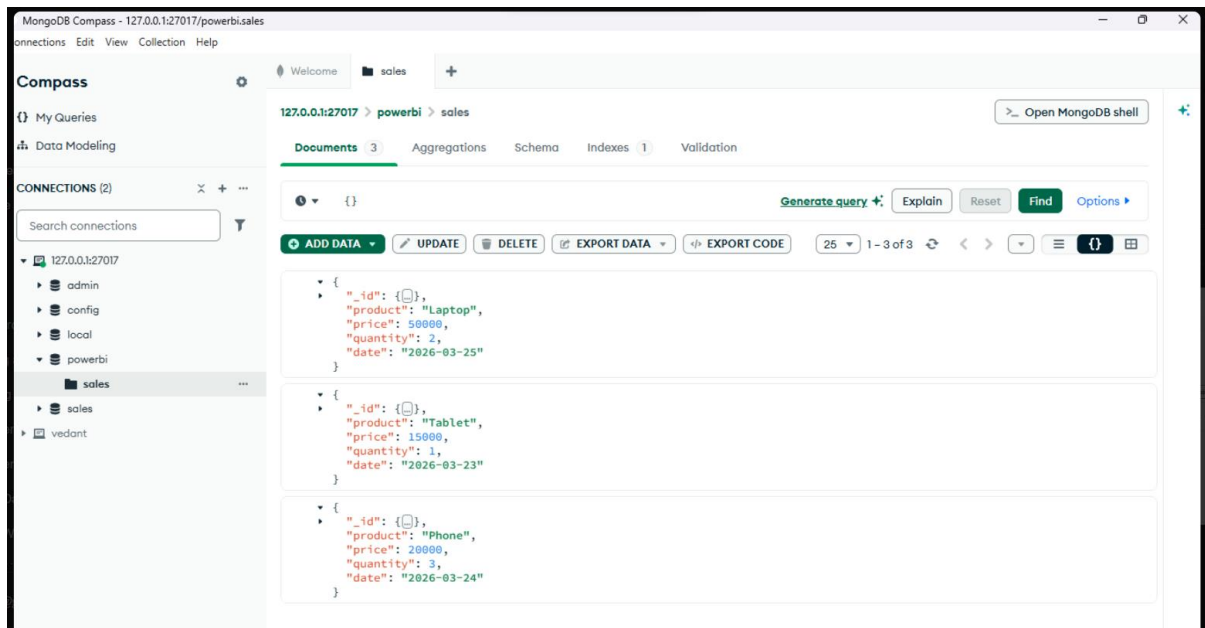
Step 3: Insert Data into MongoDB

Click **Add Data** → **Insert Document** and insert:

```
{
  "product": "Laptop",
  "price": 50000,
  "quantity": 2,
  "date": "2026-03-25"
}

{
  "product": "Tablet",
  "price": 15000,
  "quantity": 1,
  "date": "2026-03-23"
}

{
  "product": "Phone",
  "price": 20000,
  "quantity": 3,
  "date": "2026-03-24"
}
```



Explanation:

The collection now contains multiple records stored in JSON format.

Step 4: Create API Project

1. Create a folder named api
2. Open terminal inside the folder

Run: **npm init -y**

Step 5: Install Required Packages

Run : **npm install express mongoose cors**

```
C:\Windows\System32\cmd.exe X + v
C:\Users\PL2\Desktop\TYA_03>cd api
C:\Users\PL2\Desktop\TYA_03\api>npm init -y
Wrote to C:\Users\PL2\Desktop\TYA_03\api\package.json:

{
  "name": "api",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs"
}

C:\Users\PL2\Desktop\TYA_03\api>npm install express mongoose cors
added 84 packages, and audited 85 packages in 11s

24 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\PL2\Desktop\TYA_03\api>code .
C:\Users\PL2\Desktop\TYA_03\api>node server.js
API running on port 3000
```

Explanation:

- Express → for API creation
- Mongoose → for MongoDB connection
- CORS → to allow external requests (Power BI)

Step 6: Create API (server.js)

Create a file **server.js** and write:

```
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");

const app = express();
app.use(cors());

// Connect MongoDB
mongoose.connect("mongodb://127.0.0.1:27017/powerbi", {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

// Schema
const SalesSchema = new mongoose.Schema({
  product: String,
  price: Number,
  quantity: Number,
  date: String,
```

```
});

const Sales = mongoose.model("sales", SalesSchema);

// API Route
app.get("/api/sales", async (req, res) => {
  const data = await Sales.find();
  res.json(data);
});

// Server
app.listen(3000, () => {
  console.log("API running on port 3000");
});
```

Step 7: Run API Server

Run on CMD : **node server.js**

Step 8: Test API

Open browser and enter:

http://localhost:3000/api/sales

Output: JSON data is displayed.

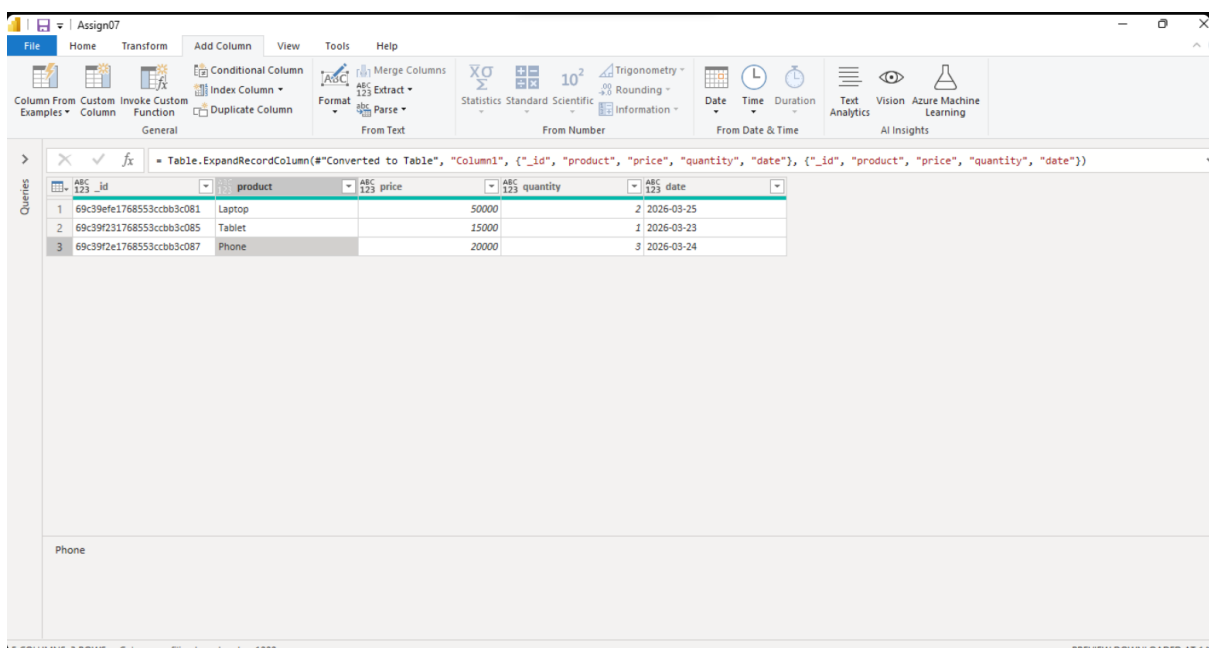
Step 9: Connect API to Power BI

1. Open Power BI Desktop
 2. Click Home → Get Data → Web
 3. Select Basic
 4. Enter: **http://localhost:3000/api/sales**
 5. Click OK
 6. In Authentication window:
 - Select Anonymous
 - Click Connect
 7. Navigator window opens
 - You will see JSON data preview
 - Click Transform Data
-

Step 10: Transform Data in Power BI

Now you are in Power Query Editor:

1. You will see data as List or Record
2. Click To Table
3. Click Expand (↔ icon) on column
4. Select fields:
 - product
 - price
 - quantity
 - date
5. Click OK
6. Click Close & Apply (*important final step*)



Explanation:

JSON data is converted into structured table format with columns.

Step 11: Create Visualizations

Instead of just listing, write like this:

- Select Bar Chart → Drag *product* to axis and *price* to values
- Select Pie Chart → Use *product* as legend
- Select Card → Show total sales (sum of price or quantity)

6. Results

- MongoDB data successfully stored and retrieved
- API fetched and returned JSON data
- Power BI converted JSON into structured format
- Data visualizations created successfully

This shows how BI systems organize and present data effectively

7. Insights

- API acts as a bridge between MongoDB and Power BI
 - JSON enables easy data transfer
 - Visualization improves understanding of data
-

8. Advantages

- Flexible data storage (MongoDB)
 - Real-time data retrieval
 - Easy integration using API
 - Scalable architecture
-

9. Limitations

- Requires backend setup
 - API must be running continuously
 - Performance depends on server
-

10. Conclusion

This practical demonstrates the integration of MongoDB with Power BI using an API. The API enables smooth data transfer from a NoSQL database to a BI tool, allowing efficient data visualization and better decision-making.