- connection
  - db.js U
- model
  - usermodel.js U
  - userschem... U
- node_modu...
- .env U
- package-loc... U
- package.json U

connection > db.js > ...

```javascript
import mongoose from "mongoose";
import chalk from "chalk";
// Use import
import dotenv from "dotenv";
dotenv.config();

const connectDB = async () => {
  try {
    const conn = await mongoose.connect(process.env.MONGO_URL);
    console.log(
      chalk.bgMagenta.white(`Connected to MongoDB ${conn.connection.host}`)
    );
  } catch (error) {
    console.error(chalk.bgRed.white(`Error in connection ${error}`));
  }
};
export default connectDB;
```

```javascript
import mongoose from "mongoose";

const productSchema = new mongoose.Schema({
  id: Number,
  title: String,
  description: String,
  price: Number,
  discountPercentage: Number,
  rating: Number,
  stock: Number,
  brand: String,
  category: String,
  thumbnail: String,
  images: [String],
});

const Product = mongoose.model("Product", productSchema);

export default Product;
```

javascript Copy code

```javascript
const getProducts = async () => {
  try {
    // Connect to the database
    await connectDB();

    // Use the find() method to retrieve all documents from the Products col
    // Sort the products by the "rating" field in descending order (-1 for d
    const Products = await Product.find().sort({ rating: -1 });

    // Log the retrieved Products
    console.log("Retrieved Products (sorted by rating in descending order):"
    console.log(Products);
  } catch (error) {
    console.error("Error occurred:", error);
  } finally {
    // Close the database connection
    mongoose.connection.close();
  }
};

// Call the getProducts function to retrieve and log the sorted Products
getProducts();
```

```javascript
import mongoose from "mongoose";
import Product from "./userschema.js";
import connectDB from "../connection/db.js";

const createDocument = async () => {
  try {
    // Connect to the database
    await connectDB();

    const newItem = new Products({ ...
    });

    const result = await newItem.save();
    console.log("Document saved successfully:", result);
  } catch (error) {
    console.error("Error occurred:", error);
  } finally {
    // Close the database connection
    mongoose.connection.close();
  }
};
createDocument();
```

```javascript
const newItem = new Products({
  id: 31,
  title: "iPhone 99",
  description: "An apple mobile which is in space",
  price: 54990,
  discountPercentage: 88.96,
  rating: 9.69,
  stock: 9,
  brand: "Apple",
  category: "smartphones",
  thumbnail: "https://i.dummyjson.com/data/products/1/thumbnail.jpg",
  images: [
    "https://i.dummyjson.com/data/products/1/1.jpg",
    "https://i.dummyjson.com/data/products/1/2.jpg",
    "https://i.dummyjson.com/data/products/1/3.jpg",
    "https://i.dummyjson.com/data/products/1/4.jpg",
    "https://i.dummyjson.com/data/products/1/thumbnail.jpg",
  ],
});
```

```
146  const createDocuments = async () => {
147    try {
148      // Connect to the database
149      await connectDB();
150
151      const newProducts = [ ···
189      ];
190
191      const result = await Product.insertMany(newProducts);
192      console.log("Documents saved successfully:", result);
193    } catch (error) {
194      console.error("Error occurred:", error);
195    } finally {
196      // Close the database connection
197      mongoose.connection.close();
198    }
199  };
200
201  createDocuments();
202
```

```
const newProducts = [
  {
    id: 31,
    title: "iPhone 99",
    description: "An apple mobile which is in space",
    price: 54990,
    discountPercentage: 88.96,
    rating: 9.69,
    stock: 9,
    brand: "Apple",
    category: "smartphones",
    thumbnail: "https://i.dummyjson.com/data/products/1/thumbnail.jpg",
    images: [
      "https://i.dummyjson.com/data/products/1/1.jpg",
      "https://i.dummyjson.com/data/products/1/2.jpg",
      "https://i.dummyjson.com/data/products/1/3.jpg",
      "https://i.dummyjson.com/data/products/1/4.jpg",
      "https://i.dummyjson.com/data/products/1/thumbnail.jpg",
    ],
  },
```

```
  {
    id: 32,
    title: "Samsung Galaxy S22",
    description: "A flagship smartphone from Samsung",
    price: 49999,
    discountPercentage: 10.5,
    rating: 8.5,
    stock: 15,
    brand: "Samsung",
    category: "smartphones",
    thumbnail: "https://i.dummyjson.com/data/products/2/thumbnail.jpg",
    images: [
      "https://i.dummyjson.com/data/products/2/1.jpg",
      "https://i.dummyjson.com/data/products/2/2.jpg",
      "https://i.dummyjson.com/data/products/2/3.jpg",
    ],
  },
];
```

```javascript
const getProducts = async () => {
  try {
    // Connect to the database
    await connectDB();

    // Use the find() method to retrieve all documents from the Products collection
    const Products = await Product.find();

    // Log the retrieved Products
    console.log("Retrieved Products:");
    console.log(Products);
  } catch (error) {
    console.error("Error occurred:", error);
  } finally {
    // Close the database connection
    mongoose.connection.close();
  }
};

// Call the getProducts function to retrieve and log the Products
getProducts();
```

```
const calculateAverageDiscountPercentageByBrand = async () => {
  try {
    // Connect to the database
    await connectDB();

    // Use the aggregation framework to calculate the average discount percentage by brand
    const result = await Product.aggregate([
      {
        $group: {
          _id: "$brand",
          averageDiscountPercentage: { $avg: "$discountPercentage" },
        },
      },
      {
        $sort: { averageDiscountPercentage: -1 }, // Sort by average discount percentage in descending order
      },
    ]);

    if (result.length > 0) {
      console.log("Average Discount Percentage by Brand:");
      result.forEach((brandData) => {
        console.log(
          `${brandData._id}: ${brandData.averageDiscountPercentage.toFixed(2)}%`
        );
      });

      // Determine the brand with the highest average discount percentage
      const highestDiscountBrand = result[0]._id;
      console.log(
        `Brand with the highest average discount percentage: ${highestDiscountBrand}`
      );
    } else {
      console.log("No products found.");
    }
```

```
    } catch (error) {
      console.error("Error occurred:", error);
    } finally {
      // Close the database connection
      mongoose.connection.close();
    }
  };

  // Call the calculateAverageDiscountPercentageByBrand function to calculate and log the results
  calculateAverageDiscountPercentageByBrand();
```

```javascript
const listProductsInRangeAndRating = async () => {
  try {
    // Connect to the database
    await connectDB();

    // Define the query criteria
    const query = {
      discountPercentage: { $gte: 10, $lte: 20 }, // Discount percentage between 10% and 20%
      rating: { $gt: 4.2 }, // Rating greater than 4.2
    };

    // Use the find() method with the query criteria to retrieve matching products
    const products = await Product.find(query);

    if (products.length > 0) {
      console.log(
        "Products with discount between 10% and 20% and rating > 4.2:"
      );
      products.forEach((product) => {
        console.log(`Product Name: ${product.title}`);
        console.log(`Discount Percentage: ${product.discountPercentage}%`);
        console.log(`Rating: ${product.rating}`);
        console.log("------------------------");
      });
    } else {
      console.log("No products found matching the criteria.");
    }
  } catch (error) {
    console.error("Error occurred:", error);
  } finally {
    // Close the database connection
    mongoose.connection.close();
  }
};
```

```javascript
const updatePriceByTitle = async (title, newPrice) => {
  try {
    // Connect to the database
    await connectDB();

    // Update the price for all documents with the specified title
    const result = await Products.updateMany(
      { title: title }, // The condition to match the documents
      { $set: { price: newPrice } } // The update to apply
    );

    console.log("Documents updated successfully:", result.nModified, "documents modified");
  } catch (error) {
    console.error("Error occurred:", error);
  } finally {
    // Close the database connection
    mongoose.connection.close();
  }
};

// Example usage: Update the price of all "iPhone 9" products to a new price
updatePriceByTitle("iPhone 9", 599.99);
```

```
const updateManyDocuments = async (title, newPrice) => {
  try {
    // Connect to the database
    await connectDB();

    // Update all documents that match the condition
    const updateResult = await Product.updateMany(
      { title: title }, // The condition to match the documents
      { $set: { price: newPrice } } // The update to apply
    );

    // Find and retrieve the updated documents
    const updatedDocuments = await Product.find({ title: title });

    if (updateResult.nModified === 0) {
      console.log("No documents were updated.");
    } else {
      console.log("documents updated successfully.");
      console.log("Updated documents:", updatedDocuments);
    }
  } catch (error) {
    console.error("Error occurred:", error);
  } finally {
    // Close the database connection
    mongoose.connection.close();
  }
};

// Example usage: Update the price of all "iPhone 9" products and retrieve the updated documents
updateManyDocuments("iPhone 9", 599.99);
```

```
234
235  const deleteDocumentsByTitle = async (title) => {
236    try {
237      // Connect to the database
238      await connectDB();
239
240      // Delete all documents that match the condition
241      const deleteResult = await Product.deleteMany({ title: title });
242
243      if (deleteResult.deletedCount === 0) {
244        console.log("No documents matching the condition were found.");
245      } else {
246        console.log(deleteResult.deletedCount, "documents deleted successfully.");
247      }
248    } catch (error) {
249      console.error("Error occurred:", error);
250    } finally {
251      // Close the database connection
252      mongoose.connection.close();
253    }
254  };
255
256  // Example usage: Delete all documents with the title "iPhone 9"
257  deleteDocumentsByTitle("iPhone 9");
258
```

```
import mongoose from "mongoose";

const productSchema = new mongoose.Schema({
  id: {
    type: Number,
    required: true,
    unique: true,
  },
  title: {
    type: String,
    required: true,
  },
  description: String,
  price: {
    type: Number,
    required: true,
    min: 0, // Minimum price value
  },
  discountPercentage: {
    type: Number,
    min: 0,
    max: 100, // Percentage should be between 0 and 100
  },
  rating: {
    type: Number,
    min: 0,
    max: 10, // Rating should be between 0 and 10
  },
  stock: {
    type: Number,
    min: 0,
  },
  brand: String,
  category: String,
  thumbnail: String,
  images: [String],
});

const Product = mongoose.model("Product", productSchema);

export default Product;
```