**1. Create Operation: Inserting a New User Record**

**Scenario:**  
A social media application needs to add a new user when they sign up.

**Solution:**

db.users.insertOne({

username: "johndoe",

email: "johndoe@example.com",

password: "encryptedPassword123",

createdAt: new Date()

});

**2. Create Operation: Inserting Multiple Documents**

**Scenario:**  
A product manager wants to add multiple products to the inventory at once.

**Solution:**

db.products.insertMany([

{ name: "Laptop", price: 1200, stock: 30 },

{ name: "Smartphone", price: 800, stock: 50 },

{ name: "Tablet", price: 300, stock: 20 }

]);

**3. Read Operation: Finding a Specific User by Username**

**Scenario:**  
Retrieve user details based on the username for login validation.

**Solution:**

db.users.findOne({ username: "johndoe" });

**4. Read Operation: Filtering Products by Price Range**

**Scenario:**  
Display products within a certain price range on an e-commerce site.

**Solution:**

db.products.find({ price: { $gte: 100, $lte: 500 } });

**5. Read Operation: Sorting Data**

**Scenario:**  
Sort all blog posts by the most recent first.

**Solution:**

db.posts.find().sort({ createdAt: -1 });

**6. Update Operation: Updating User Email**

**Scenario:**  
A user requests to update their email address.

**Solution:**

db.users.updateOne(

{ username: "johndoe" },

{ $set: { email: "john.doe@newmail.com" } }

);

**7. Update Operation: Incrementing Product Stock**

**Scenario:**  
Restocking products after receiving a new shipment.

**Solution:**

db.products.updateOne(

{ name: "Laptop" },

{ $inc: { stock: 20 } }

);

**8. Update Operation: Updating Multiple Records**

**Scenario:**  
Applying a discount on all products under a specific category.

**Solution:**

db.products.updateMany(

{ category: "electronics" },

{ $mul: { price: 0.9 } }

);

**9. Delete Operation: Removing a User Account**

**Scenario:**  
User requests account deletion.

**Solution:**

db.users.deleteOne({ username: "johndoe" });

**10. Delete Operation: Deleting Multiple Records**

**Scenario:**  
Remove discontinued products from the inventory.

**Solution:**

db.products.deleteMany({ status: "discontinued" });

**11. Aggregation: Calculating Average Product Price**

**Scenario:**  
Management wants to know the average price of all products.

**Solution:**

db.products.aggregate([

{ $group: { \_id: null, averagePrice: { $avg: "$price" } } }

]);

**12. Lookup Operation: Joining Data from Two Collections**

**Scenario:**  
Display order details along with customer information.

**Solution:**

db.orders.aggregate([

{

$lookup: {

from: "customers",

localField: "customerId",

foreignField: "\_id",

as: "customerDetails"

}

}

]);

**13. Text Search: Full-Text Search on Blog Posts**

**Scenario:**  
Allow users to search for blog posts by keywords.

**Solution:**

db.posts.createIndex({ content: "text" });

db.posts.find({ $text: { $search: "MongoDB tutorial" } });

**14. Projection: Returning Specific Fields Only**

**Scenario:**  
Retrieve user information but exclude password for security reasons.

**Solution:**

db.users.find(

{ username: "johndoe" },

{ password: 0 }

);

**15. Bulk Operations: Bulk Insertion**

**Scenario:**  
Bulk import of user data from a CSV file.

**Solution:**

db.users.insertMany(usersArray);

**16. Bulk Operations: Bulk Update**

**Scenario:**  
Mass update of product prices during a seasonal sale.

**Solution:**

db.products.bulkWrite([

{

updateMany: {

filter: { category: "electronics" },

update: { $mul: { price: 0.85 } }

}

},

{

updateMany: {

filter: { category: "fashion" },

update: { $mul: { price: 0.9 } }

}

}

]);

**17. Indexing: Improving Query Performance**

**Scenario:**  
Speed up search by product name.

**Solution:**

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

**18. Atomic Transactions: Ensuring Data Integrity**

**Scenario:**  
Handle a bank transfer where money is deducted from one account and credited to another.

**Solution:**

const session = db.getMongo().startSession();

session.startTransaction();

try {

db.accounts.updateOne(

{ accountNumber: "12345" },

{ $inc: { balance: -500 } },

{ session }

);

db.accounts.updateOne(

{ accountNumber: "67890" },

{ $inc: { balance: 500 } },

{ session }

);

session.commitTransaction();

} catch (e) {

session.abortTransaction();

} finally {

session.endSession();

}

**19. Array Operations: Adding to an Array**

**Scenario:**  
Add a new comment to a blog post.

**Solution:**

db.posts.updateOne(

{ \_id: postId },

{ $push: { comments: { user: "johndoe", text: "Great post!" } } }

);

**20. Array Operations: Removing from an Array**

**Scenario:**  
Remove a specific item from a user's wishlist.

**Solution:**

db.users.updateOne(

{ username: "johndoe" },

{ $pull: { wishlist: { productId: "abc123" } } }

);