1. **Delete documents** (single and multiple deletion):  
  
import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.model.Filters;

import org.bson.Document;

public class DeleteDocuments {

public static void main(String[] args) {

MongoClient client = MongoClients.create("mongodb://localhost:27017");

MongoDatabase db = client.getDatabase("operators"); // or "test"

MongoCollection<Document> sales = db.getCollection("sales");

sales.deleteOne(Filters.eq("item", "Cappuccino"));

System.out.println("One document deleted.");

sales.deleteMany(Filters.eq("item", "Cappuccino"));

System.out.println("All matching documents deleted.");

client.close();

}

}

2. **Find with skip & limit**:

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.FindIterable;

import org.bson.Document;

public class PaginationExample {

public static void main(String[] args) {

MongoClient client = MongoClients.create("mongodb://localhost:27017");

MongoDatabase db = client.getDatabase("operators");

MongoCollection<Document> coll = db.getCollection("sales");

// Skip the first document, then fetch the next two

FindIterable<Document> docs = coll.find()

.skip(1)

.limit(2);

for (Document doc : docs) {

System.out.println(doc.toJson());

}

client.close();

}

}

3. **Sorting via server-side**:

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.FindIterable;

import org.bson.Document;

import com.mongodb.client.model.Sorts;

public class ServerSideSort {

public static void main(String[] args) {

MongoClient client = MongoClients.create("mongodb://localhost:27017");

MongoDatabase db = client.getDatabase("operators");

MongoCollection<Document> coll = db.getCollection("sampleCollection");

FindIterable<Document> sorted = coll.find()

.sort(Sorts.descending("First\_Name"));

for (Document doc : sorted) {

System.out.println(doc.toJson());

}

client.close();

}

}

4. **Sorting using Java logic (after fetching)**:

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoCollection;

import java.util.List;

import java.util.ArrayList;

import java.util.Collections;

import org.bson.Document;

public class ClientSideSort {

public static void main(String[] args) {

MongoClient client = MongoClients.create("mongodb://localhost:27017");

MongoDatabase db = client.getDatabase("operators");

MongoCollection<Document> coll = db.getCollection("sampleCollection");

// Load all into a list

List<Document> docs = new ArrayList<>();

for (Document d : coll.find()) {

docs.add(d);

}

// Sort descending by "price"

Collections.sort(docs, (d1, d2) -> {

Double p1 = d1.getDouble("price");

Double p2 = d2.getDouble("price");

return p2.compareTo(p1);

});

// Print

for (Document d : docs) {

System.out.println("Sorted Document: " + d.toJson());

}

client.close();

}

}

5. **Filtering only even \_id documents**:

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.FindIterable;

import org.bson.Document;

public class EvenIdFilter {

public static void main(String[] args) {

MongoClient client = MongoClients.create("mongodb://localhost:27017");

MongoDatabase db = client.getDatabase("operators");

MongoCollection<Document> coll = db.getCollection("sampleCollection");

FindIterable<Document> all = coll.find();

for (Document doc : all) {

Integer id = doc.getInteger("\_id");

if (id != null && id % 2 == 0) {

System.out.println(doc.toJson());

}

}

client.close();

}

}

6)necessary java imports

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class SortAsc {

public static void main(String[] args) {

MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");

MongoDatabase database = mongoClient.getDatabase("vit");

// Get the collection

MongoCollection<Document> collection = database.getCollection("sales");

// Retrieve the documents and store them in a list

List<Document> documentList = new ArrayList<>();

for (Document doc : collection.find()) {

documentList.add(doc);

}

// Sort documents in descending order by pice

Collections.sort(documentList, (doc1, doc2) -> {

Double price1 = doc1.getDouble("price");

Double price2 = doc2.getDouble("price");

return price2.compareTo(price1); // Descending order

// for ascending order: return price1.compareTo(price2);

});

// Printing

for (Document doc : documentList) {

System.out.println("Sorted Document: " + doc);

}

}

}