1)  
MongoDatabase database = mongoClient.getDatabase("vit");

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

Document priceRangeQuery = new Document("price", new Document("$gte", 700).append("$lte", 900));

FindIterable<Document> products = collection.find(priceRangeQuery);

2) Find Products in a Price Range ($gte, $lte)

package connection;

import com.mongodb.client.\*;

import org.bson.Document;

public class PriceRangeQuery {

public static void main(String[] args) {

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

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

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

Document priceRangeQuery = new Document("price", new Document("$gte", 700).append("$lte", 900));

FindIterable<Document> products = collection.find(priceRangeQuery);

for (Document doc : products) {

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

}

}

} }

Find Products with Specific Brands ($in)

package connection;

import com.mongodb.client.\*;

import org.bson.Document;

import java.util.Arrays;

public class BrandFilterQuery {

public static void main(String[] args) {

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

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

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

Document brandQuery = new Document("brand", new Document("$in", Arrays.asList("Apple", "Samsung")));

FindIterable<Document> result = collection.find(brandQuery);

for (Document doc : result) {

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

}

}

}

}

Find Products with Stock Less Than 10 ($lt)  
package connection;

import com.mongodb.client.\*;

import org.bson.Document;

public class LowStockQuery {

public static void main(String[] args) {

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

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

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

Document stockQuery = new Document("stock", new Document("$lt", 10));

FindIterable<Document> result = collection.find(stockQuery);

for (Document doc : result) {

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

}

}

}

}

Find Products with price and rating Conditions (AND Condition)  
package connection;

import com.mongodb.client.\*;

import org.bson.Document;

public class PriceAndRatingQuery {

public static void main(String[] args) {

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

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

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

Document query = new Document("$and", Arrays.asList(

new Document("price", new Document("$gte", 500)),

new Document("rating", new Document("$gte", 4))

));

FindIterable<Document> result = collection.find(query);

for (Document doc : result) {

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

}

}

}

}

CRUD OPERATIONS

package connection;

import org.bson.Document;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import static com.mongodb.client.model.Aggregates.\*;

import static com.mongodb.client.model.Accumulators.\*;

import static com.mongodb.client.model.Sorts.\*;

import java.util.Arrays;

public class MongoDB {

public static void main(String[] args) {

try (MongoClient mongo = MongoClients.create("mongodb://localhost:27017")) {

MongoDatabase db = mongo.getDatabase("vit");

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

sales.aggregate(Arrays.asList(

group("$item",

max("maxPrice", "$price"),

addToSet("sizes", "$size")

),

sort(descending("maxPrice"))

)).forEach((Document doc) -> {

System.out.printf("%-10s: $%2d (Sizes: %s)%n",

doc.getString("\_id"),

doc.getInteger("maxPrice"),

doc.getList("sizes", String.class));

});

}

}

}

package connection;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.FindIterable;

import org.bson.Document;

public class ReadDocuments {

public static void main(String[] args) {

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

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

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

FindIterable<Document> documents = collection.find();

for (Document doc : documents) {

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

}

}

}

}

package connection;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.result.UpdateResult;

import org.bson.Document;

import com.mongodb.client.model.Filters;

import com.mongodb.client.model.Updates;

public class UpdateDocument {

public static void main(String[] args) {

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

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

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

UpdateResult result = collection.updateOne(

Filters.eq("name", "Alice"),

Updates.set("age", 22)

);

System.out.println("Matched: " + result.getMatchedCount());

System.out.println("Modified: " + result.getModifiedCount());

}

}

}

package connection;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.result.DeleteResult;

import org.bson.Document;

import com.mongodb.client.model.Filters;

public class DeleteDocument {

public static void main(String[] args) {

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

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

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

DeleteResult result = collection.deleteOne(Filters.eq("name", "Alice"));

System.out.println("Deleted Count: " + result.getDeletedCount());

}

}

}