db.products.find({ price: { $gte: 700, $lte: 900 } })

import com.mongodb.client.FindIterable;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

public class ProductPriceFilter {

public static void main(String[] args) {

// Create MongoDB client (localhost by default)

MongoClient mongoClient = MongoClients.create(); // or provide connection string

// Connect to the 'vit' database

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

// Get the 'products' collection

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

// Create a query to filter products with price between 700 and 900 (inclusive)

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

// Run the query

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

// Print matching products

for (Document product : products) {

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

}

// Close the client

mongoClient.close();

}

}

db.products.aggregate([

{

$group: {

\_id: null,

averagePrice: { $avg: "$price" }

}

}

]);

import com.mongodb.client.AggregateIterable;

import com.mongodb.client.MongoClients;

import com.mongodb.client.MongoClient;

import com.mongodb.client.MongoDatabase;

import com.mongodb.client.MongoCollection;

import org.bson.Document;

import java.util.Arrays;

public class AveragePriceAggregation {

public static void main(String[] args) {

MongoClient mongoClient = MongoClients.create(); // adjust connection string if needed

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

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

// Aggregation pipeline to compute average price

AggregateIterable<Document> avgResult = collection.aggregate(Arrays.asList(

new Document("$group", new Document("\_id", null)

.append("averagePrice", new Document("$avg", "$price")))

));

// Print the result

for (Document doc : avgResult) {

System.out.println("Average Price: " + doc.getDouble("averagePrice"));

}

mongoClient.close();

}

}

// Find the size with the highest price for each item in the sales collection.

db.sales.aggregate([

{

$group: {

\_id: { item: "$item", size: "$size" },

maxPrice: { $max: "$price" }

}

},

{

$sort: { "\_id.item": 1, "maxPrice": -1 }

},

{

$group: {

\_id: "$\_id.item",

sizeWithMaxPrice: { $first: "$\_id.size" },

maxPrice: { $first: "$maxPrice" }

}

}

])

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));

});

}

}

}