**MONGO DB**

**Assignment-2**

**Name:** Gogineni Jaswant

**Reg No:** 21BCE1199

**Questions:**

**1.Find products released in the same year as "SmartTablet".**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import com.mongodb.MongoClient;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server

MongoClient mongoClient = new MongoClient("localhost", 27017); System.out.println("Connected to MongoDB server successfully");

// Accessing the database

MongoDatabase database mongoClient.getDatabase("mongodbjava");

System.out.println("Database 'mongodbjava' accessed successfully");

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

Document smartTablet = collection.find(new Document("name", "SmartTablet")) .projection(new Document("releaseDate", 1))

.first();

String releaseYear smartTablet.getString("releaseDate").substring(0, 4);

Document query = new Document("releaseDate", new Document("$regex", releaseYear));

MongoCursor<Document> cursor = collection.find(query).iterator();

while (cursor.hasNext()) {

Document product = cursor.next();

}

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

cursor.close();

}

**2.Find products where the price is between 600 and 900, inclusive**

package Jmongo;

import com.mongodb.client.MongoCollection; import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import com.mongodb.MongoClient;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server MongoClient mongoClient = new MongoClient("localhost", 27017); System.out.println("Connected to MongoDB server successfully");

// Accessing the database

MongoDatabase database mongoClient.getDatabase("mongodbjava"); System.out.println("Database 'mongodbjava accessed successfully");

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

Document priceRangeQuery new Document("price", new Document("$gte", 600).append("$lte", 900));

MongoCursor<Document> cursor collection.find(priceRangeQuery).iterator();

while (cursor.hasNext()) (

Document product cursor.next();

System.out.println(product.to]son()); // Output or process the matching products

}

cursor.close();

}

}

**3.Find all products that have "white" as a color and a CPU speed greater than the average CPU speed of all products.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import com.mongodb.MongoClient;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server

MongoClient mongoClient = new MongoClient("localhost", 27017); System.out.println("Connected to MongoDB server successfully");

// Accessing the database

MongoDatabase database mongoClient.getDatabase("mongodbjava");

System.out.println("Database 'mongodbjava' accessed successfully");

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

// Calculate average CPU speed

Document avgCpuQuery new Document("$group", new Document("\_id", null).append("avgCpu", new Document("$avg", "$spec.cpu")));

Document avgCpuResult collection.aggregate(Arrays.asList(avgCpuQuery)).first();

double avgCpuSpeed avgCpuResult.getDouble("avgCpu");

// Query for products with "white" color and CPU speed greater than average Document whiteColorQuery = new Document("color", "white");

Document cpuGreater ThanAvgQuery = new Document("spec.cpu", new Document("$gt", avgCpuSpeed));

Document query = new Document("$and", Arrays.asList(whiteColorQuery, cpuGreater ThanAvgQuery));

MongoCursor<Document> cursor collection.find(query).iterator();

while (cursor.hasNext()) {

Document product cursor.next();

System.out.println(product.toJson()); // Output or process the matching products

}

cursor.close();

}

}

**4.find name,price,storage if price is 699 or storage is 1024.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import import com.mongodb.client.MongoDatabase;

com.mongodb.client.MongoCursor;

import org.bson.Document;

import com.mongodb.MongoClient;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server

MongoClient mongoClient = new MongoClient("localhost", 27017); System.out.println("Connected to MongoDB server successfully");

// Accessing the database MongoDatabase database mongoClient.getDatabase("mongodbjava"); System.out.println("Database 'mongodbjava' accessed successfully");

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

Document query= new Document("$or", Arrays.asList( new Document("price", 699), new Document("storage", 1024) ));

Document projection = new Document("name", 1) .append("price", 1) .append("storage", 1);

MongoCursor<Document> cursor collection.find(query) .projection(projection) .iterator();

while (cursor.hasNext()) {

Document product = cursor.next(); System.out.println(product.toJson()); // Output or process the matching products

}

}

cursor.close();

}

}

**5.Find products with either cpu is greater than 2gb or ram is between 4 and 12.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import com.mongodb.MongoClient;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server

MongoClient mongoClient = new MongoClient("localhost", 27017); System.out.println("Connected to MongoDB server successfully");

// Accessing the database

MongoDatabase database mongoClient.getDatabase("mongodbjava"); System.out.println("Database 'mongodbjava' accessed successfully");

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

Document query = new Document("$or", Arrays.asList( new Document("spec.cpu", new Document("$gt", 2)), new Document("spec.ram", new Document("$gte", 4).append("$lte", 12)) ));

MongoCursor<Document> cursor collection.find(query).iterator();

while (cursor.hasNext()) {

Document product = cursor.next();

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

}

cursor.close();

}

}

**6. Find products with niether price is between 600 and 900 nor is it white or black.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import com.mongodb.MongoClient;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server

MongoClient mongoClient = new MongoClient("localhost", 27017); System.out.println("Connected to MongoDB server successfully");

// Accessing the database

MongoDatabase database mongoClient.getDatabase("mongodbjava"); System.out.println("Database 'mongodbjava' accessed successfully");

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

Document query = new Document("$nor", Arrays.asList( new Document("price", new Document("$gte", 600).append("$lte", 900)), new Document("color", new Document("$in", Arrays.asList("white", "black"))) ));

MongoCursor<Document> cursor = collection.find(query).iterator(); while (cursor.hasNext()) {

Document product = cursor.next();

}

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

}

cursor.close();

}

**7. Find products where price exists and there is no gray in the colours.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import org.bson.Document;

import com.mongodb.MongoClient;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server

MongoClient mongoClient = new MongoClient("localhost", 27017); System.out.println("Connected to MongoDB server successfully");

// Accessing the database

MongoDatabase database mongoClient.getDatabase("mongodbjava"); System.out.println("Database 'mongodbjava' accessed successfully");

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

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

new Document("price", new Document("$exists", true)), new Document("color", new Document("$not", new Document("$in", Arrays.asList("gray")))) ));

MongoCursor<Document> cursor = collection.find(query).iterator();

while (cursor.hasNext()) {

Document product = cursor.next();

}

System.out.println(product.toJson()); // Output or process the matching products

}

}

**8.Find products that have either "white" or "black" as a color option and are priced below 800.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import com.mongodb.MongoClient;

import com.mongodb.MongoClientURI;

import org.bson.Document;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

// Establish connection to MongoDB server

MongoClient mongoClient = new MongoClient("localhost", 27017);

System.out.println("Connected to MongoDB server successfully");

// Accessing the database

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

System.out.println("Database 'mongodbjava' accessed successfully");

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

// Query to find products with "white" or "black" color and price below 800

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

new Document("$or", Arrays.asList(

new Document("color", "white"),

new Document("color", "black")

)),

new Document("price", new Document("$lt", 800))

));

MongoCursor<Document> cursor = collection.find(query).iterator();

// Iterate over the cursor and print each matching document

while (cursor.hasNext()) {

Document product = cursor.next();

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

}

// Close the cursor and MongoDB connection

cursor.close();

mongoClient.close();

}

}

**9.Find products that do not have "gold" as a color and are priced below 700 or have a storage option of 512GB.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import com.mongodb.MongoClient;

import com.mongodb.MongoClientURI;

import org.bson.Document;

import java.util.Arrays;

public class Jmongoclient {

public static void main(String[] args) {

MongoClient mongoClient = new MongoClient("localhost", 27017);

System.out.println("Connected to MongoDB server successfully");

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

System.out.println("Database 'mongodbjava' accessed successfully");

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

// Query to find products without "gold" color and priced below 700 or with storage of 512GB

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

new Document("price", new Document("$lt", 700)),

new Document("$or", Arrays.asList(

new Document("color", new Document("$ne", "gold")),

new Document("storage", "512GB")

))

));

MongoCursor<Document> cursor = collection.find(query).iterator();

while (cursor.hasNext()) {

Document product = cursor.next();

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

}

cursor.close();

mongoClient.close();

}

}

**10. Find products released before 2019.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import com.mongodb.MongoClient;

import com.mongodb.MongoClientURI;

import org.bson.Document;

public class Jmongoclient {

public static void main(String[] args) {

MongoClient mongoClient = new MongoClient("localhost", 27017);

System.out.println("Connected to MongoDB server successfully");

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

System.out.println("Database 'mongodbjava' accessed successfully");

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

// Query to find products released before 2019

Document query = new Document("release\_year", new Document("$lt", 2019));

MongoCursor<Document> cursor = collection.find(query).iterator();

while (cursor.hasNext()) {

Document product = cursor.next();

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

}

cursor.close();

mongoClient.close();

}

}

**11. List products with a screen size greater than 9.5 inches.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import com.mongodb.MongoClient;

import com.mongodb.MongoClientURI;

import org.bson.Document;

public class Jmongoclient

public static void main(String[] args) {

MongoClient mongoClient = new MongoClient("localhost", 27017);

System.out.println("Connected to MongoDB server successfully");

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

System.out.println("Database 'mongodbjava' accessed successfully");

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

// Query to find products with screen size greater than 9.5 inches

Document query = new Document("screen\_size", new Document("$gt", 9.5));

MongoCursor<Document> cursor = collection.find(query).iterator();

while (cursor.hasNext()) {

Document product = cursor.next();

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

}

cursor.close();

mongoClient.close();

}

}

**12. Retrieve products with storage options including 512 GB.**

package Jmongo;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoCursor;

import com.mongodb.client.MongoDatabase;

import com.mongodb.MongoClient;

import com.mongodb.MongoClientURI;

import org.bson.Document;

public class Jmongoclient {

public static void main(String[] args) {

MongoClient mongoClient = new MongoClient("localhost", 27017);

System.out.println("Connected to MongoDB server successfully");

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

System.out.println("Database 'mongodbjava' accessed successfully");

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

// Query to find products with storage options including 512 GB

Document query = new Document("storage", "512GB");

MongoCursor<Document> cursor = collection.find(query).iterator();

while (cursor.hasNext()) {

Document product = cursor.next();

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

}

cursor.close();

mongoClient.close();

}

}