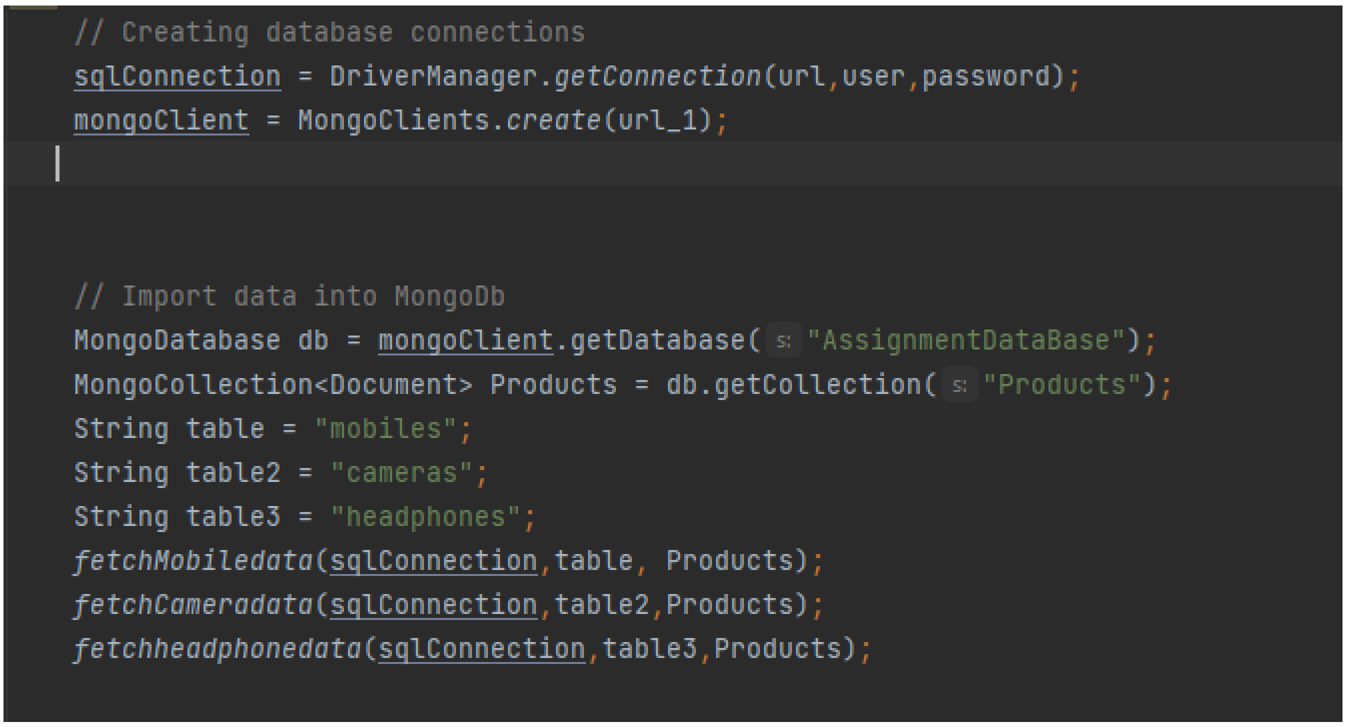
LOGICS

Importing Database: Database Present on Rds Server can be imported to Mongo DB using the Following steps:

1. Create a Connection with both MYSQL and MongoDB. Create a database and Collection to store the data.
2. Create a Statement which will help you to execute the required Query which will read the data from MYSQL database.
3. Now create a method which will take 3 parameters namely a SQL connection, Table name which you want to import and Mongo Db Collection in which you want to import your data.
4. Invoke this method for different tables

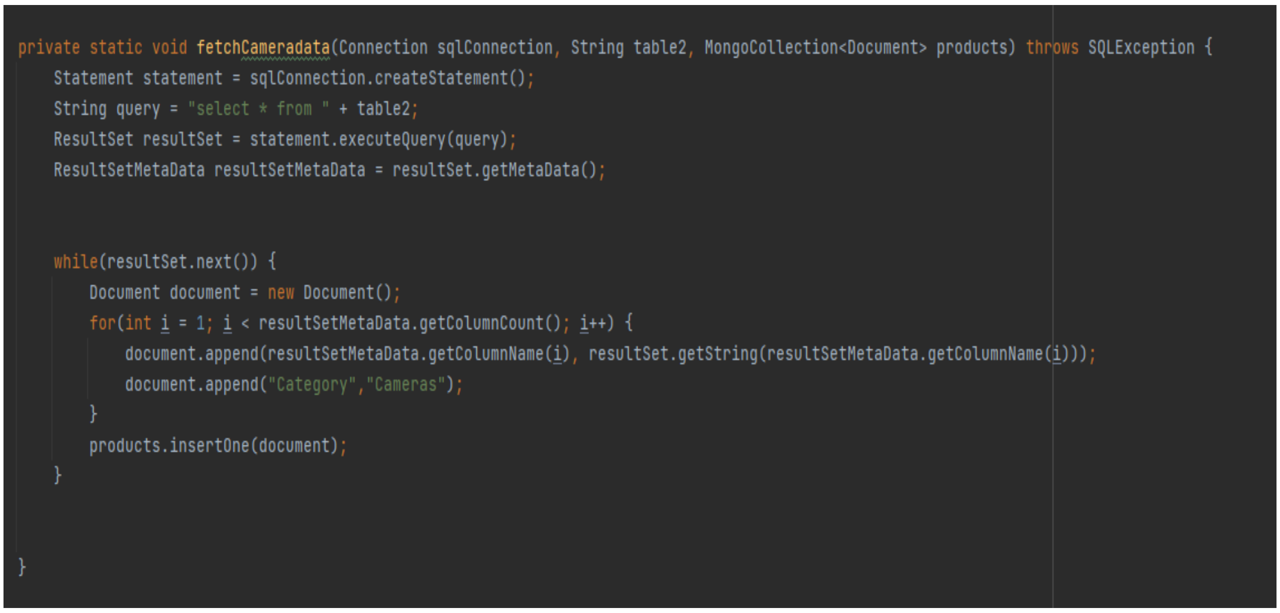


```
// Creating database connections
sqlConnection = DriverManager.getConnection(url,user,password);
mongoClient = MongoClients.create(url_1);



// Import data into MongoDb
MongoDatabase db = mongoClient.getDatabase( s: "AssignmentDataBase");
MongoCollection<Document> Products = db.getCollection( s: "Products");
String table = "mobiles";
String table2 = "cameras";
String table3 = "headphones";
fetchMobiledata(sqlConnection,table, Products);
fetchCameradata(sqlConnection,table2,Products);
fetchheadphonedata(sqlConnection,table3,Products);
```

Fig. Creating Connections, Creating a Database and Collection, Declaring Table names, Calling the methods

Import method: The method created imports the data from my sql to mongo db according to the following steps:

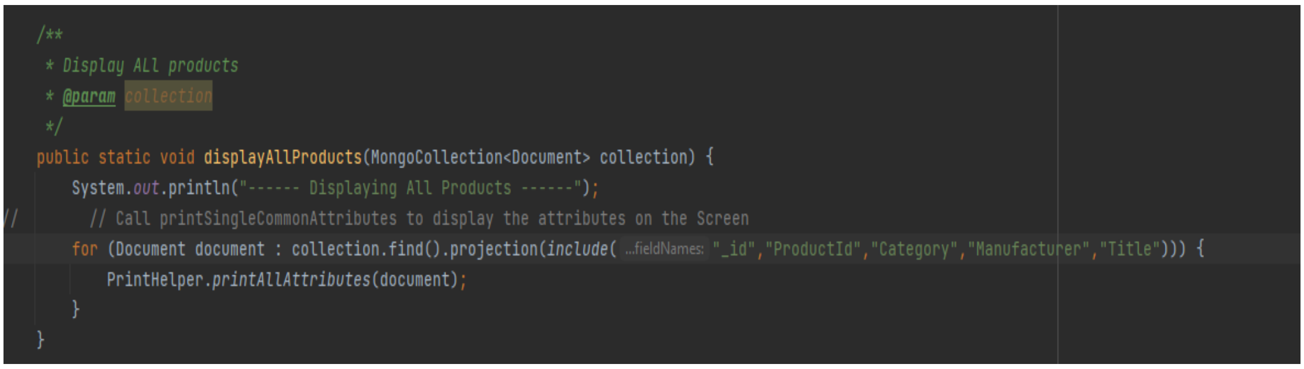
1. The created method takes 3 parameters namely a **SQL Connection, Mongo Db Collection and a table name**.
2. Create a Statement using the Statement "**statement = sqlConnection.createStatement();**" Syntax it will help us to execute the query.
3. Now Create a Find Query "**"select * from "+ table name"**. This query will fetch all the data from the given table.
4. Create a result set to store the data on which you can iterate over.
5. Create a document and now append the data present in the result set using a while loop add a new key value pair **"document.append("Category","Mobile");"** Value will be different for different products.
6. Now use **"products.insertOne(document);"** command to insert the Document in the mongo db collection i.e. products;



```java
private static void fetchCameradata(Connection sqlConnection, String table2, MongoCollection<Document> products) throws SQLException {
    Statement statement = sqlConnection.createStatement();
    String query = "select * from " + table2;
    ResultSet resultSet = statement.executeQuery(query);
    ResultSetMetaData resultSetMetaData = resultSet.getMetaData();


    while(resultSet.next()) {
        Document document = new Document();
        for(int i = 1; i < resultSetMetaData.getColumnCount(); i++) {
            document.append(resultSetMetaData.getColumnName(i), resultSet.getString(resultSetMetaData.getColumnName(i)));
            document.append("Category","Cameras");
        }
        products.insertOne(document);
    }


}
```

Fig. Method to import Data from SQL to MongoDB

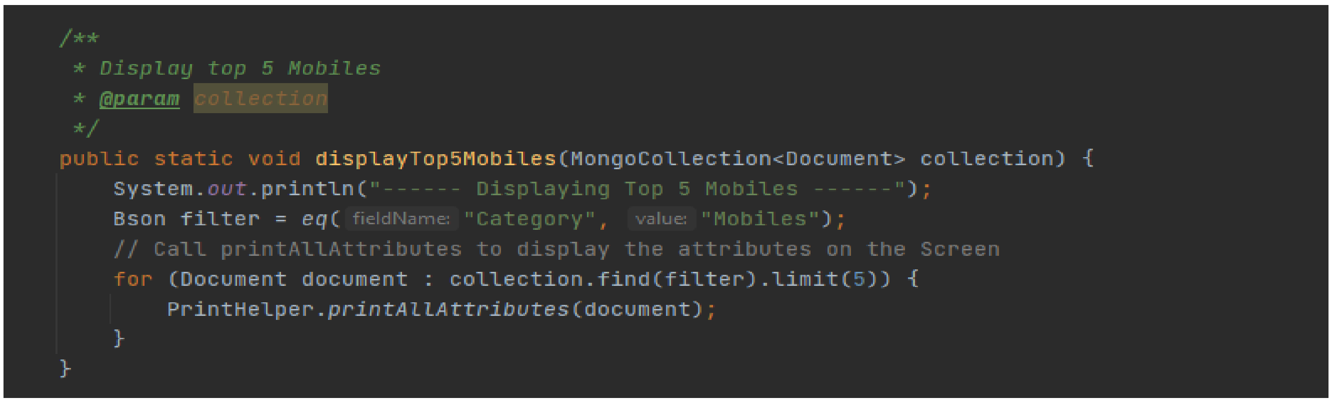
Query 1:



```java
/**
 * Display ALL products
 * @param collection
 */
public static void displayAllProducts(MongoCollection<Document> collection) {
    System.out.println("------ Displaying All Products ------");
//      // Call printSingleCommonAttributes to display the attributes on the Screen
    for (Document document : collection.find().projection(include( ...fieldNames: "_id","ProductId","Category","Manufacturer","Title"))) {
        PrintHelper.printAllAttributes(document);
    }
}
```

Using For each loop Define a document and fire query **“collection.find().projection(include("_id","ProductId","Category","Manufacturer","Title")))”** Now inside loop Call Print Helper method and pass the document to print all the projected fields I.e. the common fields across the database. Projection function will only fetch the specified fields and exclude the rest fields.

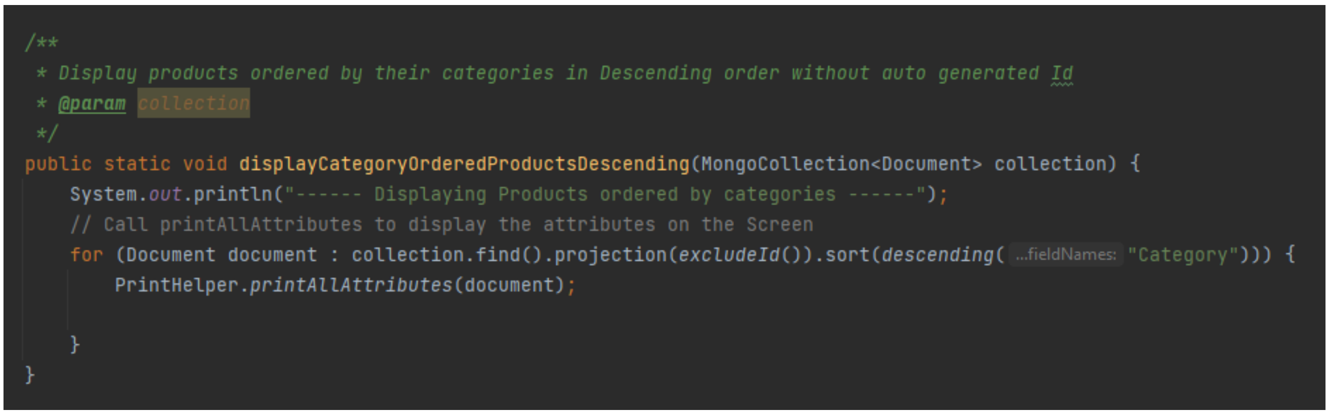
Query 2:



```java
/**
 * Display top 5 Mobiles
 * @param collection
 */
public static void displayTop5Mobiles(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Top 5 Mobiles ------");
    Bson filter = eq( fieldName: "Category",  value: "Mobiles");
    // Call printAllAttributes to display the attributes on the Screen
    for (Document document : collection.find(filter).limit(5)) {
        PrintHelper.printAllAttributes(document);
    }
}
```

Create a Bson filter parameter and assign the filter query to it **“eq("Category", "Mobiles")”** , now using the for each loop again define a document and use command **“collection.find(filter).limit(5)”** , the limit() function will restrict the limit of data fetched from database to only 5. Inside loop Call Print helper method and pass on the document to print all the data.

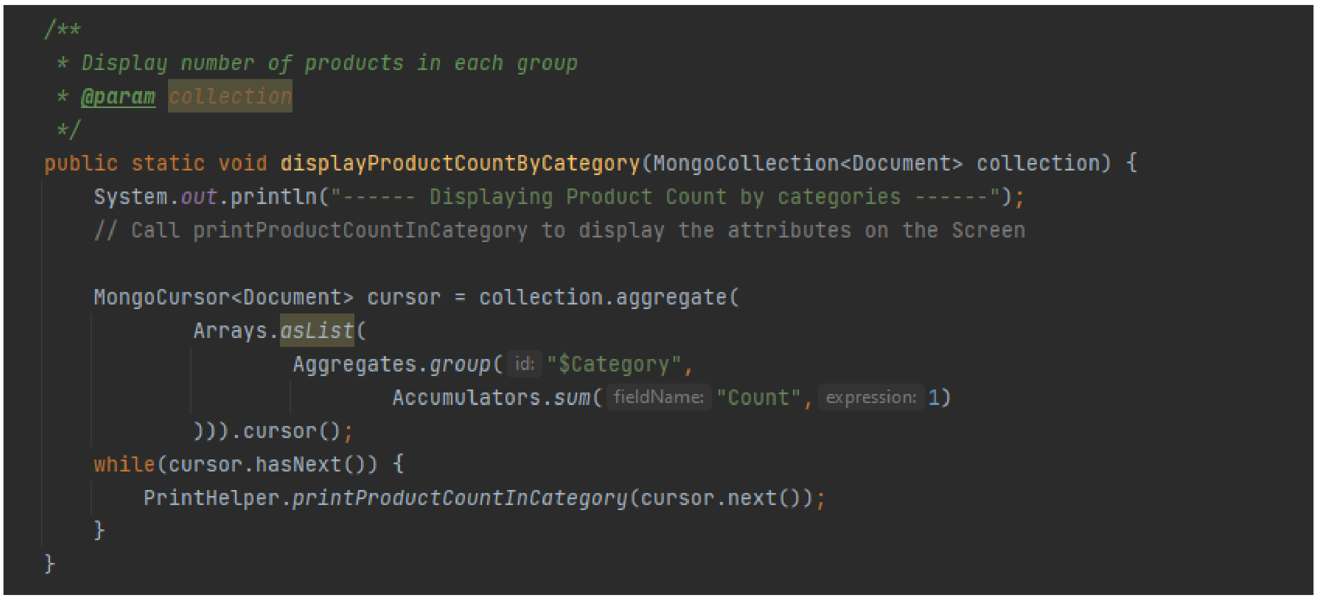
Query 3:



```java
/**
 * Display products ordered by their categories in Descending order without auto generated Id
 * @param collection
 */
public static void displayCategoryOrderedProductsDescending(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Products ordered by categories ------");
    // Call printAllAttributes to display the attributes on the Screen
    for (Document document : collection.find().projection(excludeId()).sort(descending( ...fieldNames: "Category"))) {
        PrintHelper.printAllAttributes(document);

    }
}
```

Using For each loop define a document and pass on the **“collection.find().projection(excludeId()).sort(descending("Category")))”** ,here projection(excludeId()) will exclude the id field and will fetch all the other fields data, sort(descending(“Category”))) method will sort the fetched data in descending order based on the categories of the products.

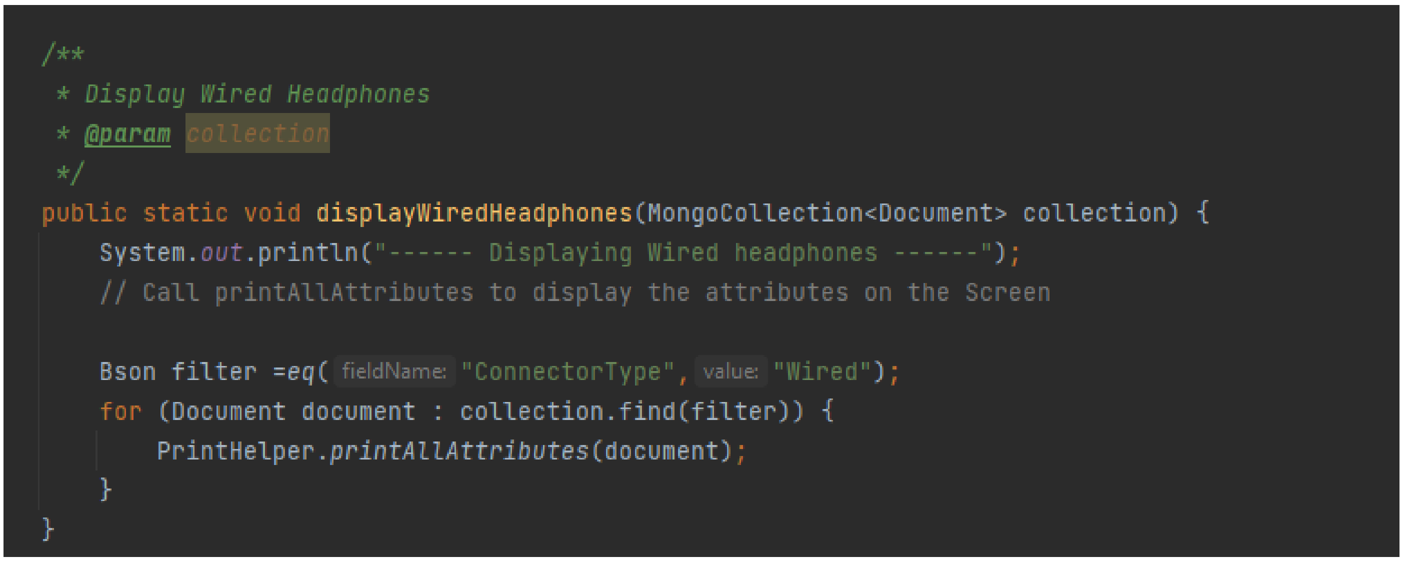
Query 4:



```java
/**
 * Display number of products in each group
 * @param collection
 */
public static void displayProductCountByCategory(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Product Count by categories ------");
    // Call printProductCountInCategory to display the attributes on the Screen

    MongoCursor<Document> cursor = collection.aggregate(
            Arrays.asList(
                    Aggregates.group( id: "$Category",
                            Accumulators.sum( fieldName: "Count", expression: 1)
            ))).cursor();
    while(cursor.hasNext()) {
        PrintHelper.printProductCountInCategory(cursor.next());
    }
}
```

This query is solved using aggregation method. Create a Mongo cursor of document type and then aggregate the field by grouping them based on Category and now find the count of all the products present in all the categories. Now suing while loop on cursor.hasNext() Call Print helper method and print the values till cursor has a next value.

Query 5:



```
/**
 * Display Wired Headphones
 * @param collection
 */
public static void displayWiredHeadphones(MongoCollection<Document> collection) {
    System.out.println("------ Displaying Wired headphones ------");
    // Call printAllAttributes to display the attributes on the Screen

    Bson filter =eq( fieldName: "ConnectorType", value: "Wired");
    for (Document document : collection.find(filter)) {
        PrintHelper.printAllAttributes(document);
    }
}
```

Create a Bson filter **“eq(fieldname: “Connector type”, value: “Wired”)”** , now using for each loop pass on “collection.find(filter))” query and inside call Print Helper method to print all the retrieved values.