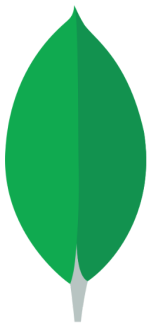
Ejercicios



mongoDB®

2º Desarrollo de Aplicaciones Multiplataforma
Acceso a Datos

Índice

Ejercicios

Base de datos usada: productos

1. Muestra el nombre de cada fabricante junto con el precio medio de la empresa ordenado descendentemente por el precio medio.
2. Muestra el nombre de los productos de manera ordenada, sin repetir y la cantidad de estos.
3. Muestra el nombre y precio de las tablets con un precio mayor a 500€.
4. Inserta un nuevo producto. Si el nombre del producto existe, no debe ser insertarlo. (el tratamiento de errores debe usarse en Java)
5. Inserta una lista de productos.
6. Modificar la categoría de un producto llamado por su nombre de esta forma. **public boolean ej06(String nombre, String categoria)**
7. Modificar el precio de una categoría de productos incrementando su precio 120€.
8. Modifica el nombre del campo ‘fabricante’ por ‘marca’ en todos los productos que su nombre empieza por las letras ‘Sam’.
9. Elimina un producto por su nombre.
10. Elimina una categoría entera sin miedo al éxito.

Base de datos usada: students

1. Saca la media de todas las puntuaciones (score) de cada uno de los alumnos.

Base de datos usada: grades

1. Muestra todos los estudiantes por su ‘student_id’ de forma que no se repitan, junto la suma de sus puntuaciones (scores) y que se muestren de forma ascendente por puntuación.
2. Muestra la puntuación media (score) de todos los tipos (type) sin que se repitan y que estén ordenadas descendentemente.

3. Selecciona la nota más alta. Posteriormente, recoge su tipo (type) e imprime todos sus datos y haz la media de todos los score de ese tipo.

Soluciones

Base de datos usada: productos

1. Solución ej01

```
test> db.productos.aggregate([{$group:{_id:"$fabricante",precioMedio:{$avg:"$precio"}}},{$project:{_id:0,fabricante:"$_id",precioMedio:"$precioMedio"}},{$sort:{precioMedi
:-1}}])
[
  { fabricante: 'Apple', precioMedio: 574 },
  { fabricante: 'Samsung', precioMedio: 507.49 },
  { fabricante: 'Sony', precioMedio: 499 },
  { fabricante: 'Google', precioMedio: 199 },
  { fabricante: 'Amazon', precioMedio: 164 }
]
```

```
db.productos.aggregate(
  {
    /**
     * _id: The id of the group.
     * fieldN: The first field name.
     */
    $group: {
      _id: "$fabricante",
      productos: {
        $sum: 1
      },
      precio_medio: {
        $avg: "$precio"
      }
    }
  },
  {
    /**
     * specifications: The fields to
     *   include or exclude.
     */
    $project: {
      _id:0, fabricante:"$_id", productos:"$productos", precio_medio:"$precio_medio"
    }
  },
  {
    /**
     * Provide any number of field/order pairs.
     */
    $sort: {
      precio_medio: -1
    }
  }
)
```

```
public void ej01() {
    List<Document> lista = new ArrayList<>();
    lista.add(new Document("$group", new Document("_id", "$fabricante").append("productos",
            new Document("$sum", 1)).append("precio_medio", new Document("$avg", "$precio"))));
    lista.add(new Document("$project", new Document("_id", 0).append("fabricante", "$_id")
            .append("productos", "$productos").append("precio_medio", "$precio_medio")));
    lista.add(new Document("$sort", new Document("precio_medio", -1)));

    for (Document d : collection.aggregate(lista)) {
        System.out.println(d);
    }

}
```

2. Solución ej02

```
db.productos.aggregate(
  {
    /**
     * _id: The id of the group.
     * fieldN: The first field name.
     */
    $group: {
      _id: "$nombre",
      numero_productos: {
        $sum: 1
      }
    }
  },
  {
    /**
     * Provide any number of field/order pairs.
     */
    $sort: {
      nombre: 1
    }
  },
  {
    /**
     * specifications: The fields to
     *   include or exclude.
     */
    $project: {
      _id:0, nombre:"$_id", numero_productos:"$numero_productos"
    }
  }
)
```

```
public void ej02() {
    List<Document> lista = new ArrayList<>();
    lista.add(new Document("$group", new Document("_id", "$nombre").append("numero_productos", new Document("$sum", 1))));
    lista.add(new Document("$sort", new Document("nombre", 1)));
    lista.add(new Document("$project", new Document("_id",0).append("nombre", "$_id").append("numero_productos", "$numero_productos")));

    for (Document d : collection.aggregate(lista)) {
        System.out.println(d);
    }

}
```

3. Solución ej03

```
db.productos.find({$and:[{categoria:"Tablets"},{precio:{$gte:500}}]},{_id:0,nombre:1,precio:1});
```

```
public void ej03() {
    for (Document d : collection.find(new Document("precio", new Document("$gte", 500)))
            .sort(new Document("nombre", 1))
            .projection(new Document("_id", 0).append("nombre", 1).append("precio", 1))) {
        System.out.println(d);
    }
}
```

4. Solución ej04

```
db.productos.insertMany({
  nombre: "Xiaomi Redmi Note 9",
  categoria: "Smartphones",
  fabricante: "Xiaomi",
  precio: 265.99
})
```

```
public boolean ej04(Document d) {
    for (String s : collection.distinct("nombre", String.class)) {
        if (d.getString("nombre").equalsIgnoreCase(s)) {
            return false;
        }
    }
    collection.insertOne(d);
    return true;
}
```

5. Solución ej05

```
db.productos.insertMany([{
  nombre: "Xiaomi Redmi Note 11",
  categoria: "Smartphones",
  fabricante: "Xiaomi",
  precio: 1065.99
},{
  nombre: "Xiaomi Redmi Note 10",
  categoria: "Smartphones",
  fabricante: "Xiaomi",
  precio: 865.99
}])
```

```
public boolean ej05() {
    try {
        List<Document> lista = new ArrayList<>();
        lista.add(new Document("nombre", "Samsung Galaxy S7").append("categoria", "Tablets")
                .append("fabricante", "Samsung").append("precio", 240));
        lista.add(new Document("nombre", "Samsung Galaxy Tab A8").append("categoria", "Tablets")
                .append("fabricante", "Samsung").append("precio", 203));
        collection.insertMany(lista);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

6. Solución ej06

```
db.productos.updateOne({nombre:"Samsung Galaxy S11"},{$set:{categoria:"Tablet"}})
```

```
public boolean ej06(String nombre, String categoria) {
        long modificaciones = collection.updateOne(new Document("nombre", nombre),
                new Document("$set", new Document("categoria", categoria))).getModifiedCount();

        if (modificaciones > 0) {
            return true;
        } else {
            return false;
        }
}
```

7. Solución ej07

```
db.productos.updateMany({categoria:"Tablet"},{$inc:{precio:120}})
```

```
public boolean ej07(String categoria) {
    long modificaciones = collection.updateMany(new Document("categoria", categoria),
            new Document("$inc", new Document("precio", 120))).getModifiedCount();
    if (modificaciones > 0) {
        return true;
    } else {
        return false;
    }
}
```

8. Solución ej08

```
db.productos.updateMany({nombre:{$regex:/^Sam/i}},{$rename:{fabricante:"marca"}})
```

```
public boolean ej08() {
    long modificaciones = collection.updateMany(new Document("nombre", new Document("$regex", "^Sam")),
            new Document("$rename", new Document("fabricante", "marca"))).getModifiedCount();
    if (modificaciones > 0) {
        return true;
    } else {
        return false;
    }
}
```

9. Solución ej09

```
db.productos.deleteOne({name:"Samsung Galaxy S10"})
```

```
public boolean ej09(String nombre) {
    try {
        System.out.println(collection.deleteOne(new Document("nombre", nombre)));
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

10. Solución ej10

```
db.productos.deleteMany({categoria:"Tablet"})
```

```
public boolean ej10(String categoria) {
    try {
        System.out.println(collection.deleteMany(new Document("categoria", categoria)));
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

Base de datos usada: students

1. Solución ej01

```
@SuppressWarnings("unchecked")
public void ej01() {
    FindIterable<Document> lista = collection.find();
    List<Document> puntuaciones = new ArrayList<>();
    double sumaPuntuaciones = 0;

    for (Document d : lista) {
        sumaPuntuaciones = 0;
        puntuaciones = (List<Document>) d.get("scores");
        for (Document d2 : puntuaciones) {
            sumaPuntuaciones += d2.getDouble("score");
        }
        if (!d.getString("name").isEmpty()) {
            System.out.println(
                    d.getString("name") + "\tMedia puntuaciones: " + sumaPuntuaciones / puntuaciones.size());
        }

    }
}
```

Base de datos usada: grades

1. Solución ej01

```
db.grades.aggregate(
    {
        /**
         * _id: The id of the group.
         * fieldN: The first field name.
         */
        $group: {
          _id: "$student_id",
          puntuacion_total: {
            $sum: "$score"
          }
        }
    },
    {
        /**
         * specifications: The fields to
         *   include or exclude.
         */
        $project: {
          _id:0, id_estudiante:"$_id", puntuacion_total:"$puntuacion_total"
        }
    },
    {
        /**
         * Provide any number of field/order pairs.
         */
        $sort: {
          puntuacion_total: 1
        }
    }
)
```

```
public void ej01() {
    List<Document> lista = new ArrayList<>();
    lista.add(new Document("$group", new Document("_id", "$student_id").append("puntuacion_total", new Document("$sum", "$score"))));
    lista.add(new Document("$project", new Document("_id",0).append("id_estudiante", "$_id").append("puntuacion_total", "$puntuacion_total")));
    lista.add(new Document("$sort", new Document("puntuacion_total",1)));

    for (Document d : collection.aggregate(lista)) {
        System.out.println(d);
    }
}
```

2. Solución ej02

```
db.grades.aggregate(
    {
        /**
         * _id: The id of the group.
         * fieldN: The first field name.
         */
        $group: {
          _id: "$type",
          puntuacion: {
            $avg: "$score"
          }
        }
    },
    {
        /**
         * specifications: The fields to
         *   include or exclude.
         */
        $project: {
          _id:0, tipo: "$_id", media_puntuacion:"$puntuacion"
        }
    },
    {
        /**
         * Provide any number of field/order pairs.
         */
        $sort: {
          media_puntuacion: -1
        }
    }
)
```

```
public void ej02() {
    List<Document> lista = new ArrayList<>();
    lista.add(new Document("$group", new Document("_id", "$type").append("puntuacion", new Document("$avg", "$score"))));
    lista.add(new Document("$project", new Document("_id", 0).append("tipo", "$_id").append("media_puntuacion", "$puntuacion")));
    lista.add(new Document("$sort", new Document("media_puntuacion", -1)));

    for (Document d : collection.aggregate(lista)) {
        System.out.println(d);
    }

}
```

3. Solución ej03

```java
public void ej03() {
    String type = collection.find().sort(new Document("score", -1)).first().getString("type");
    double puntuacion_total = 0;
    int cantidad = 0;
    FindIterable<Document> lista = collection.find(new Document("type", type)).sort(new Document("score", -1));
    System.out.println("Todos los registros del tipo: " + type.toUpperCase() + "\n");

    for (Document d : lista) {
        System.out.println("Estudiante nº " + d.getInteger("student_id") + ", puntuación: " + d.getDouble("score"));
        puntuacion_total += d.getDouble("score");
        cantidad++;
    }

    System.out.println("\nMedia de puntuaciónes: " + puntuacion_total / cantidad);
}
```

BUSCAR UNA CIUDAD PASANDO EL NOMBRE Y EL PAÍS COMO PARÁMETRO Y QUE DEVUELVA EL OBJETO CIUDAD

```java
private static Ciudad buscarCiudad(String nombre,String pais) {
    Ciudad c= new Ciudad();
    Document d=collection.find(new Document("name", nombre).append("country", pais)).first();

    c.setCountry(d.getString("country"));
    c.setName(d.getString("name"));
    c.setPopulation(d.getLong("population"));
    c.setTimezone(d.getString("timezone"));
    Document loc=(Document) d.get("location");
    c.setLongitude(Float.parseFloat(String.valueOf(loc.getDouble("longitude"))));
    c.setLatitude(Float.parseFloat(String.valueOf(loc.getDouble("latitude"))));

    return c;
}
```

```java
private static boolean borrarProducto(String nombre) {

    long num = collection.deleteOne(Filters.eq("nombre", nombre)).getDeletedCount();
    if (num == 1)
        return true;
    else

        return false;

}
```

```java
    // subir el precio una cantidad pasada por parametro
    private static boolean actualizaPrecio(String nombre, double cantidad) {

        long num = collection.updateOne(Filters.eq("nombre", nombre), Updates.inc("precio", cantidad))
                .getModifiedCount();

        if (num == 1)
            return true;
        else

            return false;
    }
```

```java
//Muestra los productos de cada fabricante junto con el precio medio de la empresa ordenado descendentemente por el precio medio.
private static void Productos01() {
    Document group = new Document("$group", new Document("_id", "$fabricante")
            .append("precio", new Document("$avg", "$precio"))
            .append("productos", new Document("$addToSet", "$nombre")));
    Document project = new Document("$project", new Document("_id", 0).append("MARCA", "$_id")
            .append("PRECIOMEDIO", "$precio").append("PRODUCTOS", "$productos"));
    Document sort = new Document("$sort", new Document("PRECIOMEDIO", -1));

    List<Document> productostAggregationQuery = new ArrayList<>();
    productostAggregationQuery.add(group);
    productostAggregationQuery.add(project);
    productostAggregationQuery.add(sort);

    AggregateIterable<Document> lista = collection.aggregate(productostAggregationQuery);

    for (Document d : lista) {
        System.out.println(d);
    }
}
```

```
Document{{MARCA=Samsung, PRECIOMEDIO=507.49, PRODUCTOS=[Galaxy Tab 10, Galaxy S3]}}
Document{{MARCA=Sony, PRECIOMEDIO=499.0, PRODUCTOS=[Vaio]}}
Document{{MARCA=Apple, PRECIOMEDIO=469.2, PRODUCTOS=[iPad 16GB Wifi, iPad 64GB Wifi, Macbook Air 13inch, iPad 32GB Wifi, Moussy]}}
Document{{MARCA=Google, PRECIOMEDIO=199.0, PRODUCTOS=[Nexus 7]}}
Document{{MARCA=Amazon, PRECIOMEDIO=164.0, PRODUCTOS=[Kindle Paper White, Kindle Fire]}}
```

```java
//Muestra el nombre de los productos de manera ordenada, sin repetir y la cantidad de estos.
private static void Productos02() {
    TreeSet<String>tree= new TreeSet<>();
    DistinctIterable<String>lista= collection.distinct("nombre",String.class);
    for(String s : lista) {
        tree.add(s);
    }
    for (String s:tree) {
        System.out.println(s);
    }
    System.out.println("total productos: "+tree.size());
        }
```

```
Galaxy S3
Galaxy Tab 10
Kindle Fire
Kindle Paper White
Macbook Air 13inch
Moussy
Nexus 7
Vaio
iPad 16GB Wifi
iPad 32GB Wifi
iPad 64GB Wifi
total productos: 11
```

```java
// Muestra el nombre y precio de las tablets con un precio mayor a 500€.
private static void Productos03() {
    FindIterable<Document>lista= collection.find(new Document("categoria","Tablets")
            .append("precio",new Document("$gte",500)))
            .projection(Projections.exclude("_id","categoria","fabricante"));

    for(Document d:lista) {
        System.out.println(d);
    }

}
```

```
Document{{nombre=iPad 32GB Wifi, precio=599}}
Document{{nombre=iPad 64GB Wifi, precio=699}}
```

```java
private static boolean Productos05() {
    List<Document>lista= new ArrayList<>();
    lista.add(new Document("nombre","pincho").append("categoria", "usb").append("fabricante", "Amazon").append("precio", 20));
    lista.add(new Document("nombre","movil").append("categoria", "telefono").append("fabricante", "Samsung").append("precio", 200));
    long antes =collection.countDocuments();
    collection.insertMany(lista);
    long despues=collection.countDocuments();
    if(despues-antes==lista.size()) {
        return true;
    }else
        return false;
}
```

```java
private static boolean Productos08() {
    if( collection.updateMany(Filters.eq("fabricante",new Document("$regex","^Sam")),Updates.rename("fabricante","marca")).getModifiedCount()>0)
        return true;
    return false;
}
```