MongoDB Data Analytics
Tutorial – 4, Advanced queries

CSP 595 – Enterprise Web Application
Dr. Atef Bader
Illinois Institute of Technology

Mongo DB Find/Query Data:

- Use ‘find’ method to query MongoDB to retrieve data from a collection
- ‘find’ can be used with a single collection.
- Using queries, you can either return;
  - All the documents in a collection
  - Only the documents that match a certain filter/criteria

Find all documents in a collection:

- In order to find all the documents in a collection, use the ‘find’ query without any parameters
- dbCursor - is a variable of type DBCursor, this variable can be used as an iterator and print all the values from the collection
- myReviews – Collection that has been used in the Application.
- find() – is an empty query, this will return all the documents from the given collection
- Example:
  - DBCursor dbCursor = myReviews.find();

Create a ‘query’

- A query can be created in the following way;

```
BasicDBObject query = new BasicDBObject();
```

- Now, once the query object is created, you can add multiple conditions in the following way;

```
query.put(Key, Value);
```

  In order to find specific documents in a collection, use find along with some query values

- Example : Return the documents where the product name is ‘XBOX_ONE’ :

```
query.put(“productName”, “XBOX_ONE”);
DBCursor dbCursor = myReviews.find(query);
```

Operators

- You can use different operators to conveniently filter the data based on different requirements

- Suppose, we want to filter only those reviews from our collection which have a rating of more than 3, we need to use ‘$gt’ (greater than)

- Example Return the documents where the review rating is above 3

```
BasicDBObject query = new BasicDBObject();
query.put("reviewRating", new BasicDBObject("$gt", 3));
DBCursor dbCursor = myReviews.find(query);
```

- Please refer this link for more information: https://docs.mongodb.org/manual/reference/operator/query/

Limit and Sort

- ‘limit()’ accepts an integer value

- ‘sort()’ accepts an object of type DBObject

- Example: Return top 5 products based on maximum rating

```
int returnLimit = 5;
Created a new sort object
DBObject sort = new BasicDBObject();
Specify the field that you want to sort on, and the direction of the sort
sort.put("reviewRating",-1);

dbCursor = myReviews.find(query).limit(returnLimit).sort(sort);
```

Aggregation in MongoDB:

- The aggregate method accepts as its argument an array of stages, where each stage, processed sequentially, describes a data processing step.

- More information on aggregation can be found here: https://docs.mongodb.org/getting-started/java/aggregation/

Stages in Aggregation – $match

- $match – This is similar to ‘Where’ in SQL
- Example Match the documents where rating is 5:
  - DBObject match = new BasicDBObject("$match", new BasicDBObject("reviewRating", 5));
- Matching stage is optional

Stages in Aggregation – $group

- $group – This is similar to SQL's 'GROUP BY' clause

- Example grouping based on retailer city:

```
DBObject groupFields = new BasicDBObject("_id", 0);
groupFields.put("_id", "$retailerCity");
groupFields.put("count", new BasicDBObject("$sum", 1))
DBObject group = new BasicDBObject("$group", groupFields);
```

Group by is done on retailer city fields using _id as key to group by

Increment the count by 1 using $sum command

Stages in Aggregation – $project

- $project – This is similar to ‘SELECT’ in SQL

- Vertically Slicing Data from the Original Database.

- Example Getting count based on retailer city :

```
DBObject projectFields projectFields = new BasicDBObject("_id", 0);
projectFields.put(“city", "$_id");
projectFields.put("Review Count", "$count");
DBObject project = new BasicDBObject("$project", projectFields);
```

Project Fields which we want to dis play in the output

Stages in Aggregation – $limit and $sort

Example: Return top 5 products based on maximum rating;

```
DBObject sort = new BasicDBObject();

Specify the field that you want to sort on, and the direction of the sort
sort.put("reviewRating",-1);

DBObject limit=new BasicDBObject();
DBObject orderby=new BasicDBObject();

Adding sort object in  DbObject
orderby=new BasicDBObject("$sort",sort);

limit=new BasicDBObject("$limit",5);
aggregate = myReviews.aggregate(group,project,orderby,limit);
```

Final Stage in Aggregation:

- Now that we are done with the different stages, it is time to run the query

- Example;

```
AggregationOutput aggregate = myReviews.aggregate(match,group,project,orderby,limit);
for (DBObject result : aggregate.results()) {
        BasicDBObject bobj = (BasicDBObject) result;
        System.out.println(bobj.getString("City"));
        System.out.println(bobj.getString("Review Count"));
}
```

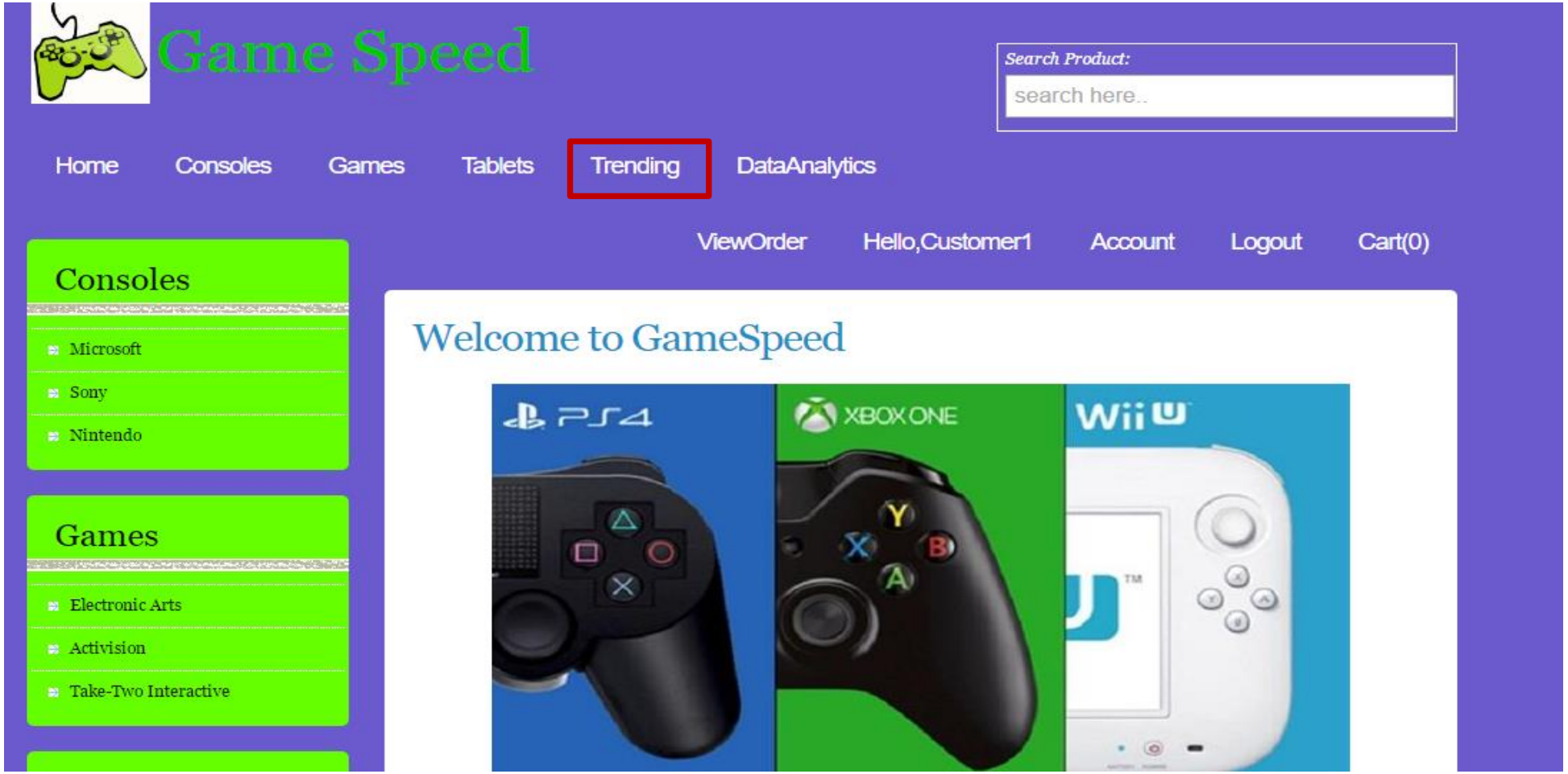
- Once the aggregate function is run, you can iterate through the result and print the required fields

Data Analytics:

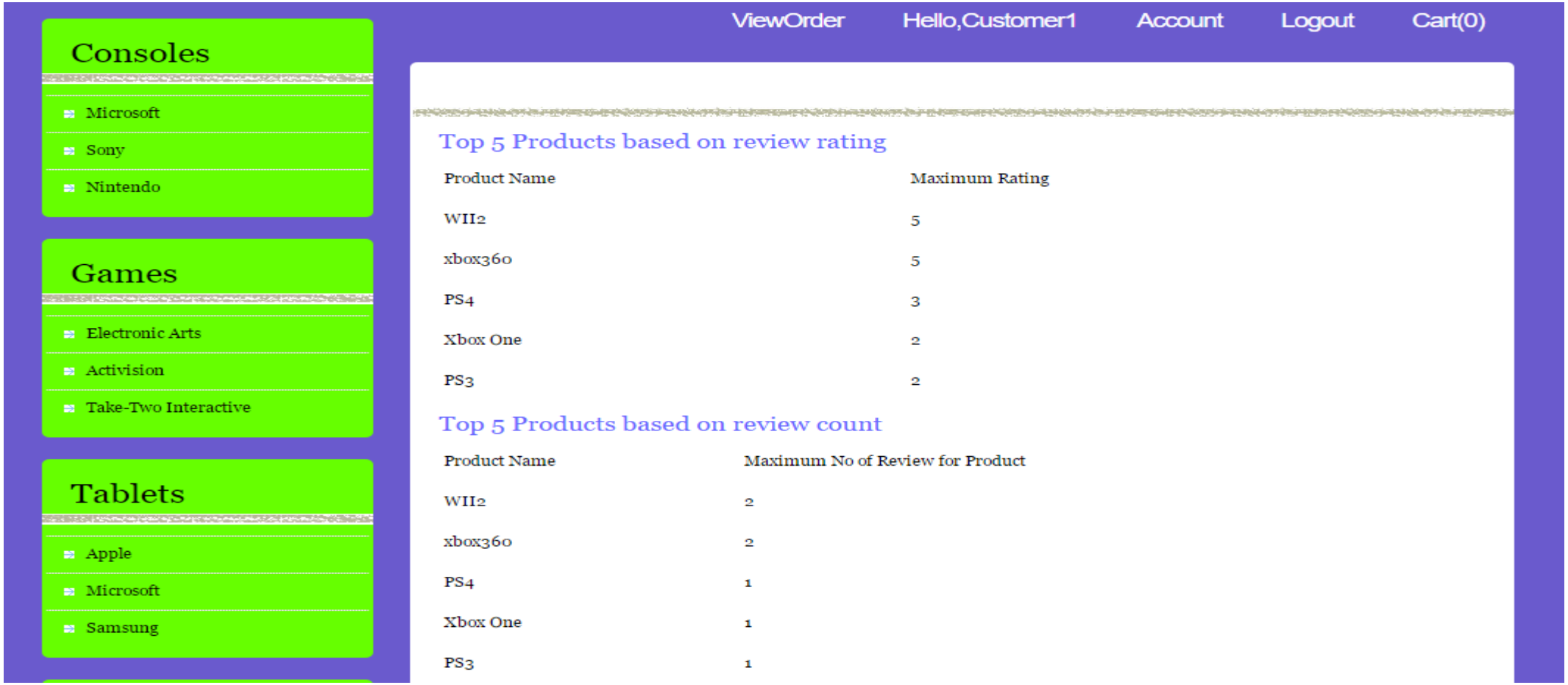
- We can use complex queries in Mongo DB to perform data analysis on the collection.
- This tutorial will demonstrate a few scenarios where you can construct dynamic queries and display the result.
- All the queries are created dynamically based on the filters selected on the screen.

Trending link:

Clicking on the trending button will take us to the page where we will display
Top five most liked products,
Top five most reviewed products regardless of the rating
Top 5 zip code based on no of products reviewed



Trending link for user – Query Outputs :



Trending link for user – Query Outputs :

- Take-Two Interactive

Tablets

- Apple
- Microsoft
- Samsung

Accessories

- Microsoft Accessories
- Sony Accessories
- Nintendo Accessories

Top 5 Products based on review count

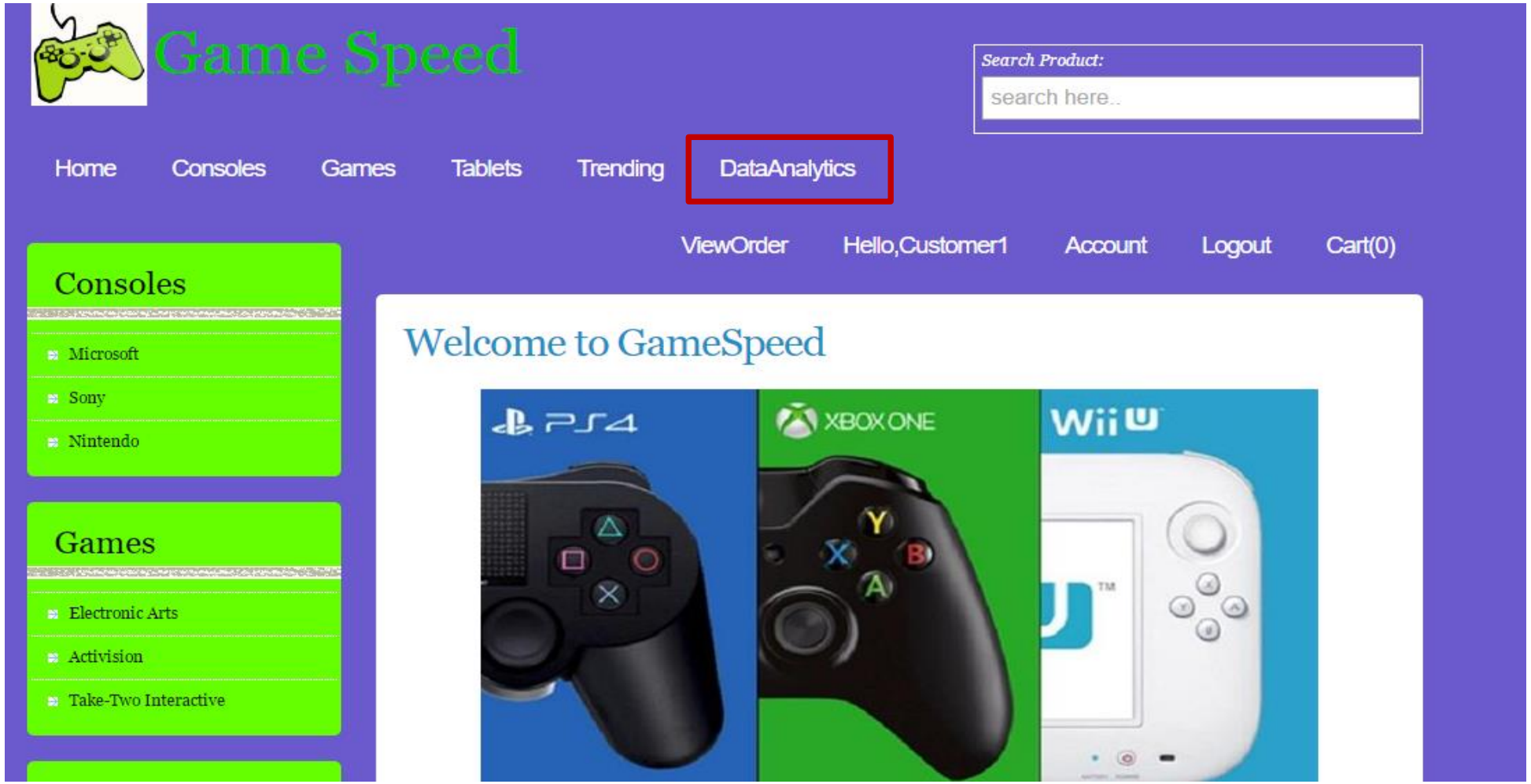
Product Name	Maximum No of Review for Product
WII2	2
xbox360	2
PS4	1
Xbox One	1
PS3	1

Top 5 Zip code based on Maximum no of products reviewed

Zip Code	Maximum no of Products Reviewed in Zip Code
690000	2
60616	2
60117	1
60612	1
60011	1

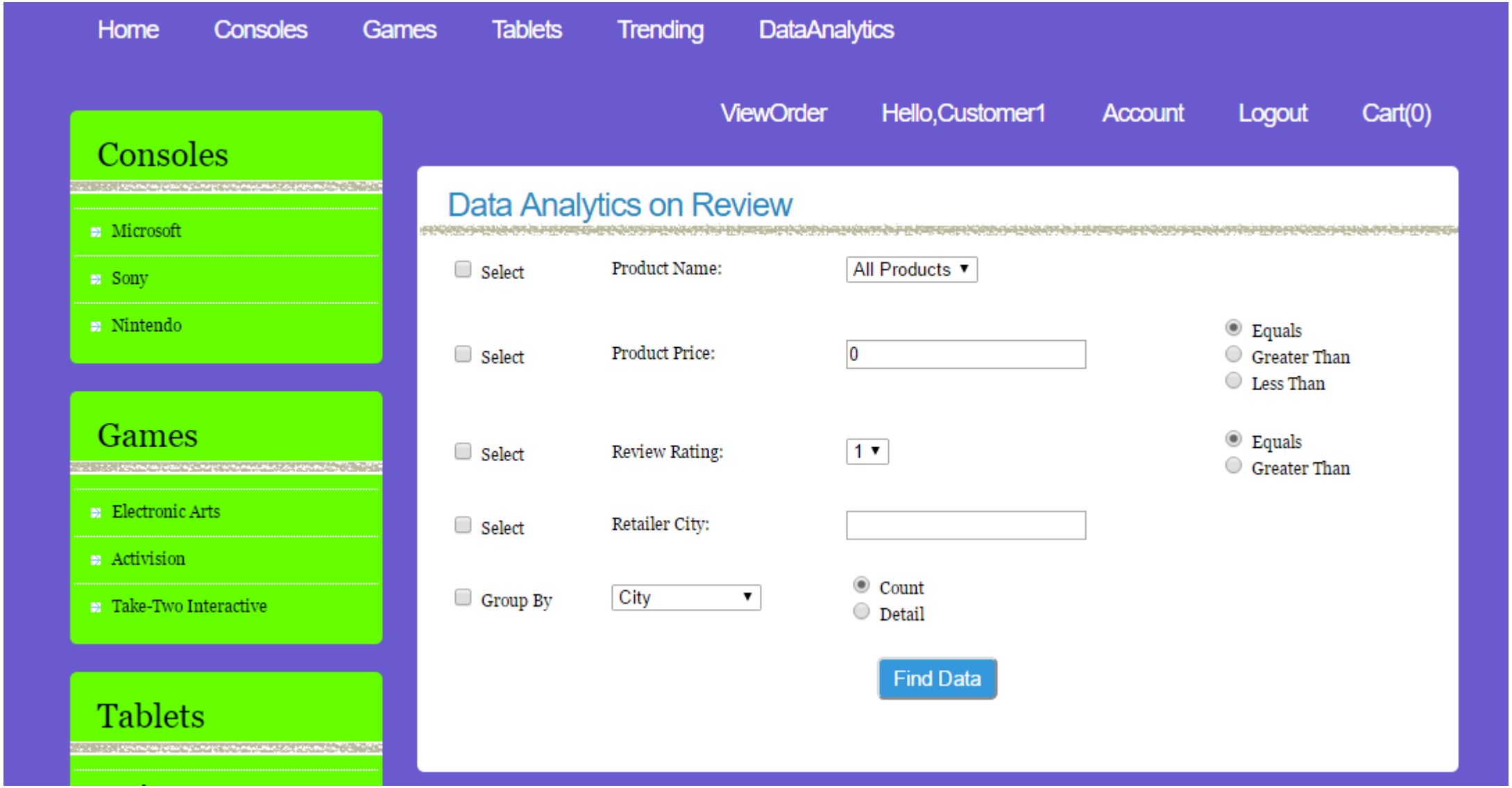
Data Analytics

- Clicking on the data analytics link will take us to data analytics page where we can perform analytics required



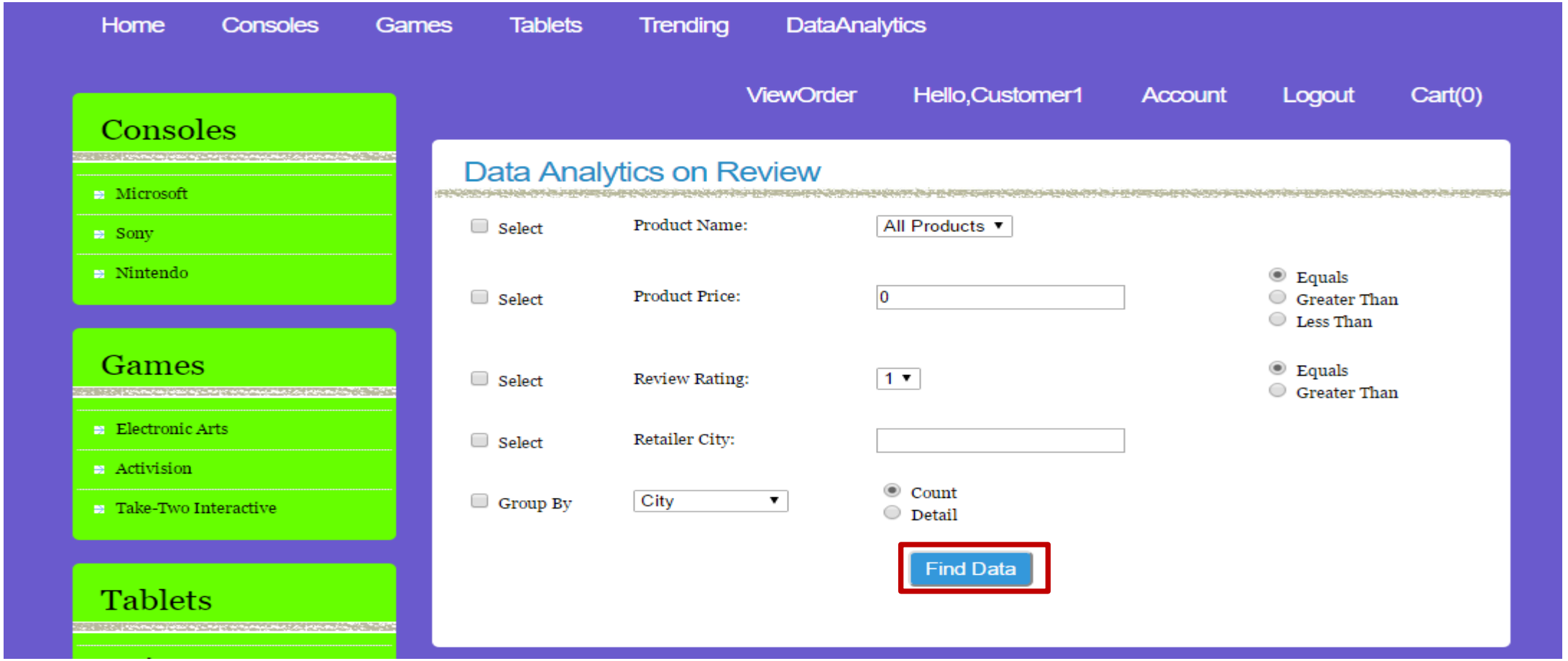
Data Analytics – Store Manager

- This page will contain all fields for selection by which we can perform data analytics

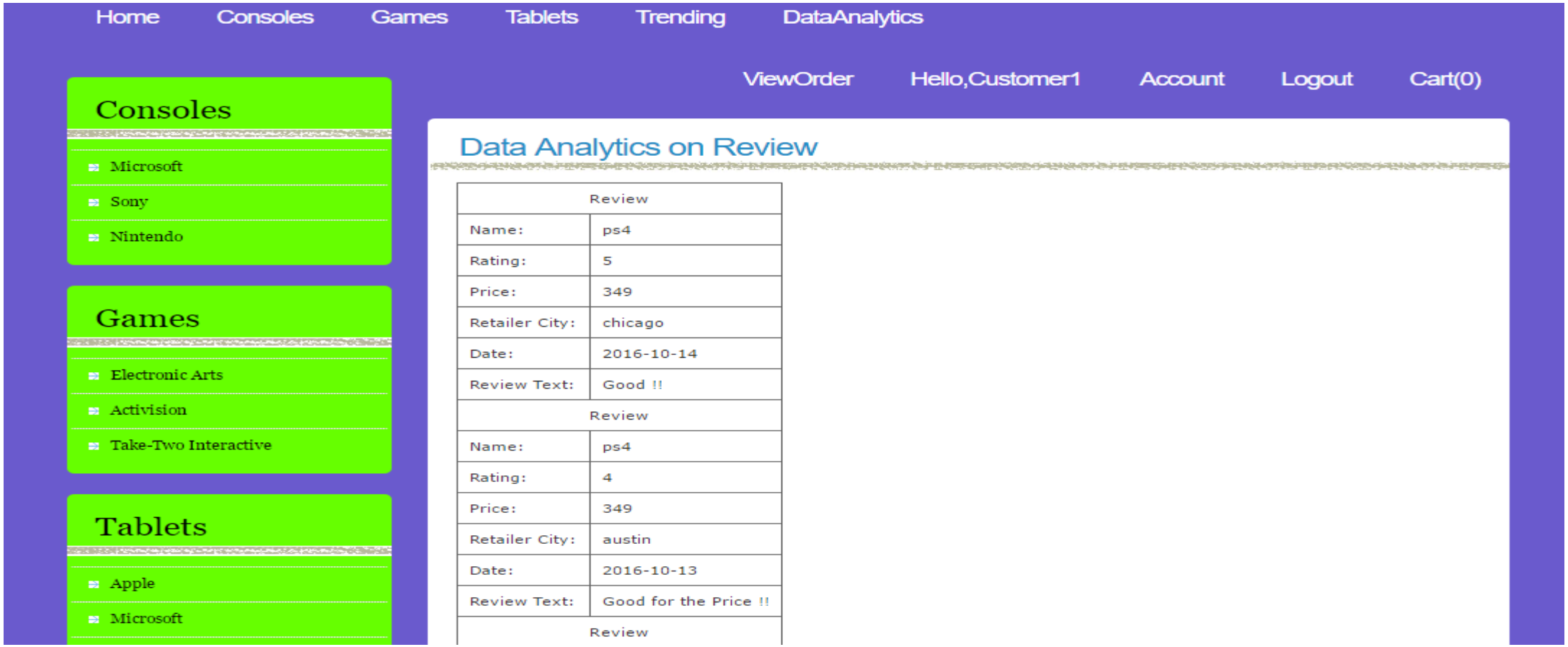


Query 1 – Print the list of all the reviews

- Press the find data button without any selection and you will get list of all reviews

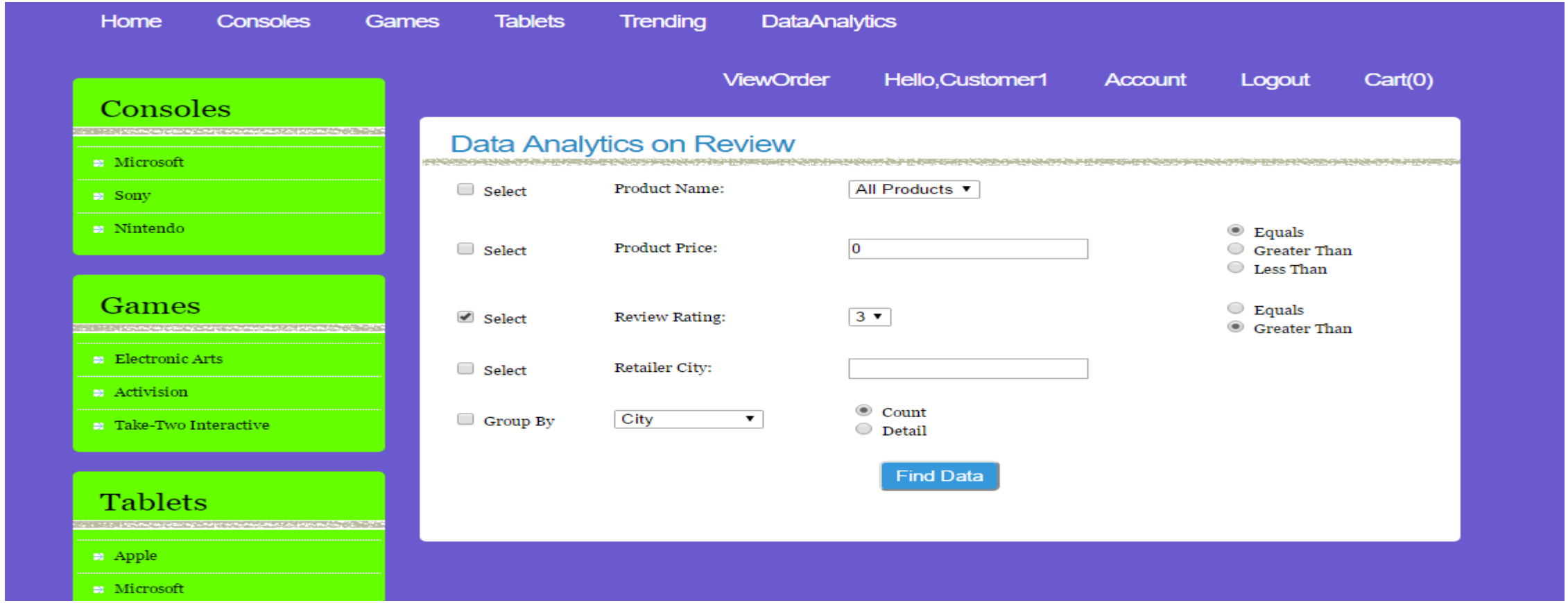


Result 1 - Print the list of all the reviews



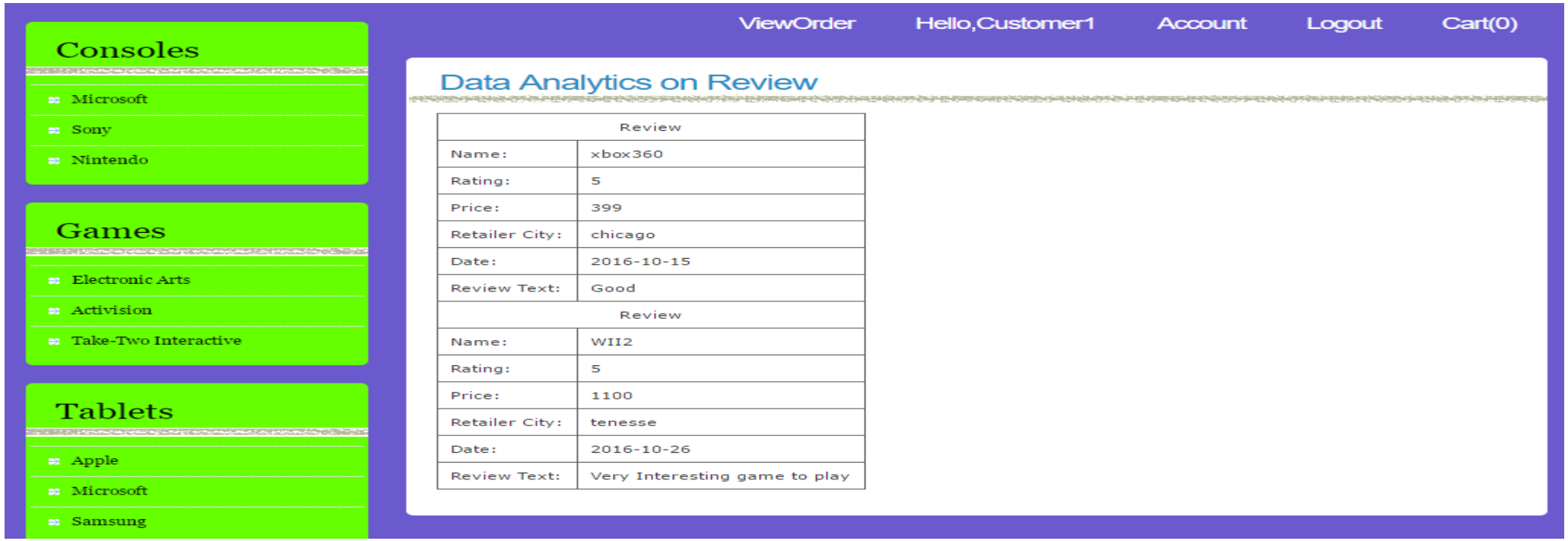
Query 2 – Print a list of reviews where rating is more than 3

- Select the filter for rating and option greater than

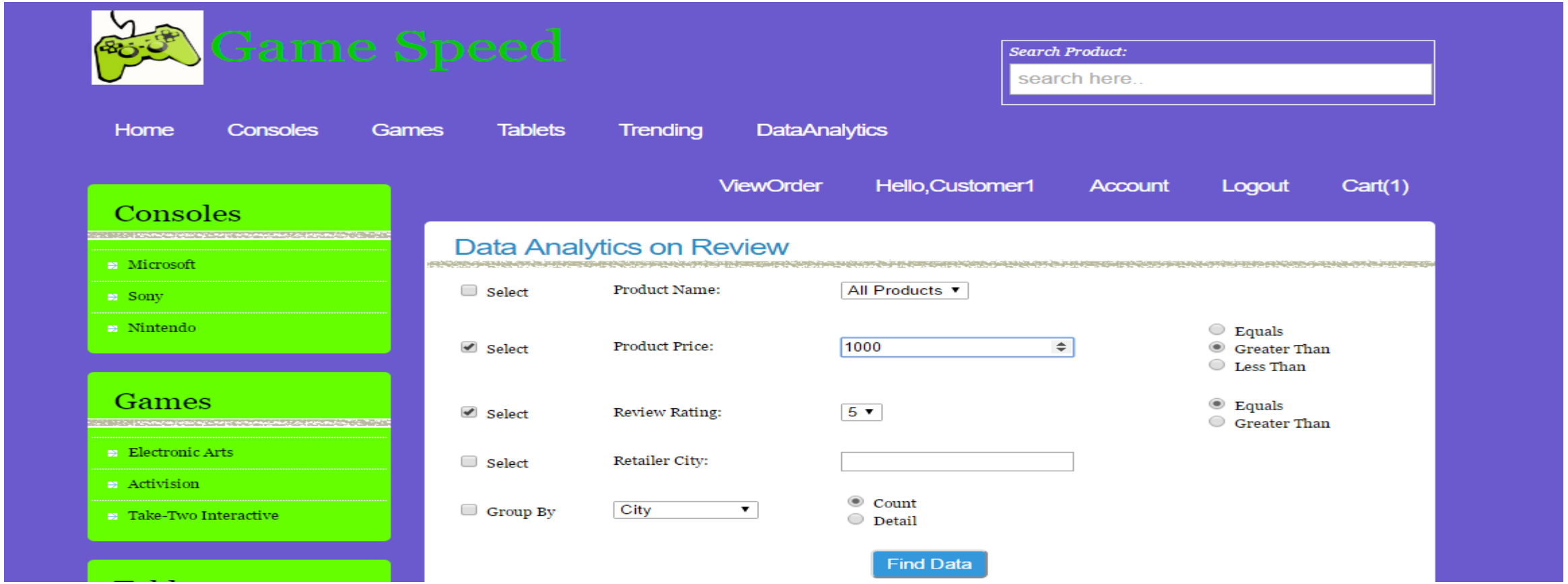


Result 2 - Print a list of reviews where rating is more than 3

Only reviews with rating greater than 3 will be displayed

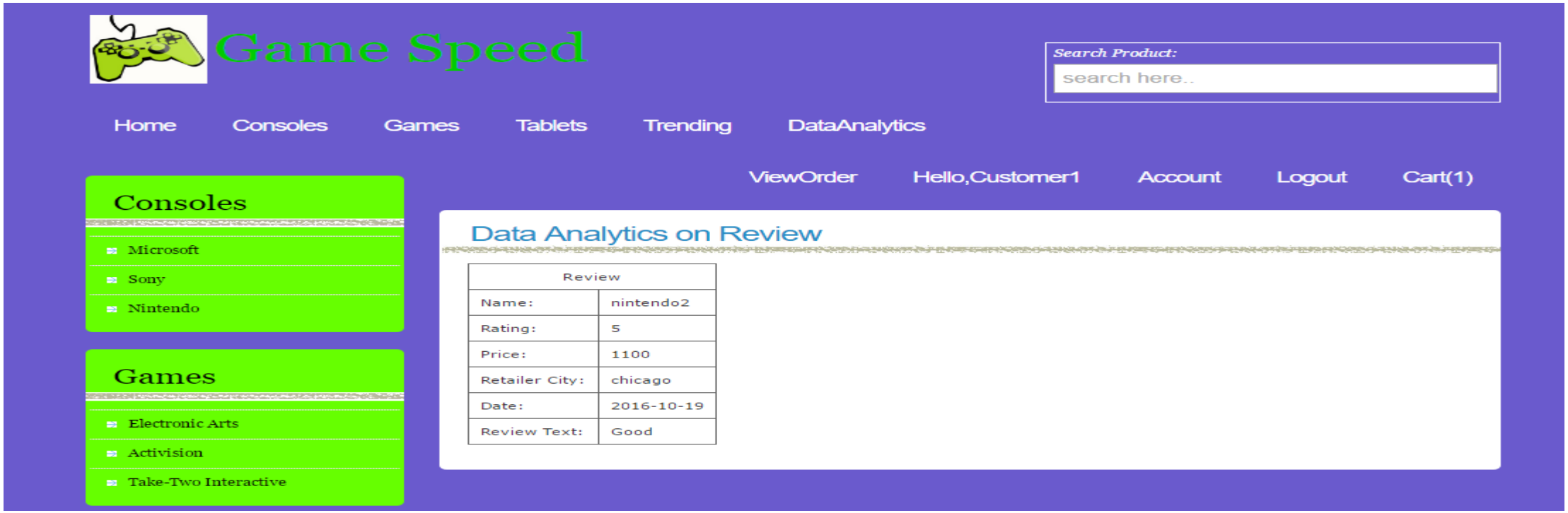


Query 3 - Get a list of products that got review rating 5 and price more than thousand



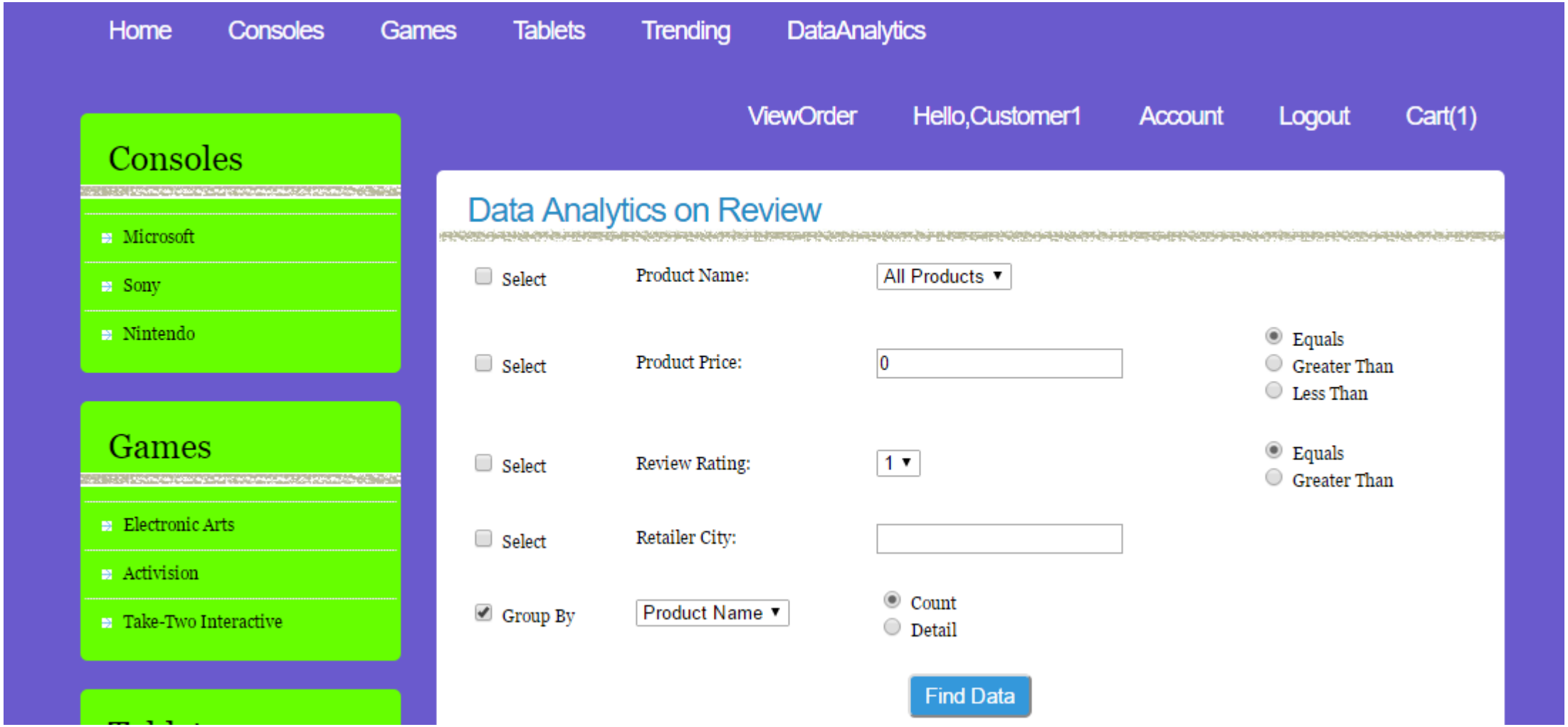
Result 3 - Get a list of products that got review rating 5 and price more than thousand

Data will be displayed with the particular review which we added that has price greater than 1000 and rating 5

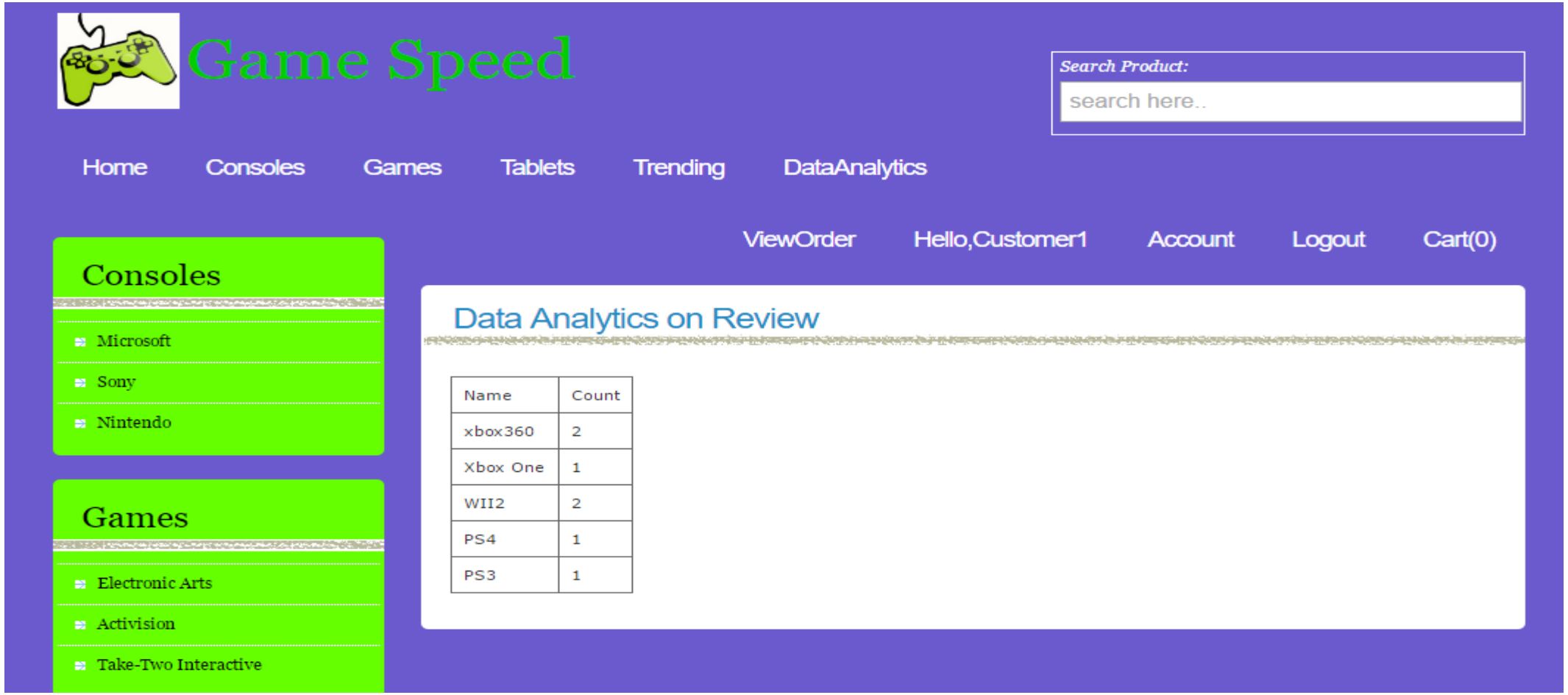


Query 4 - Print a list of how many reviews for every product

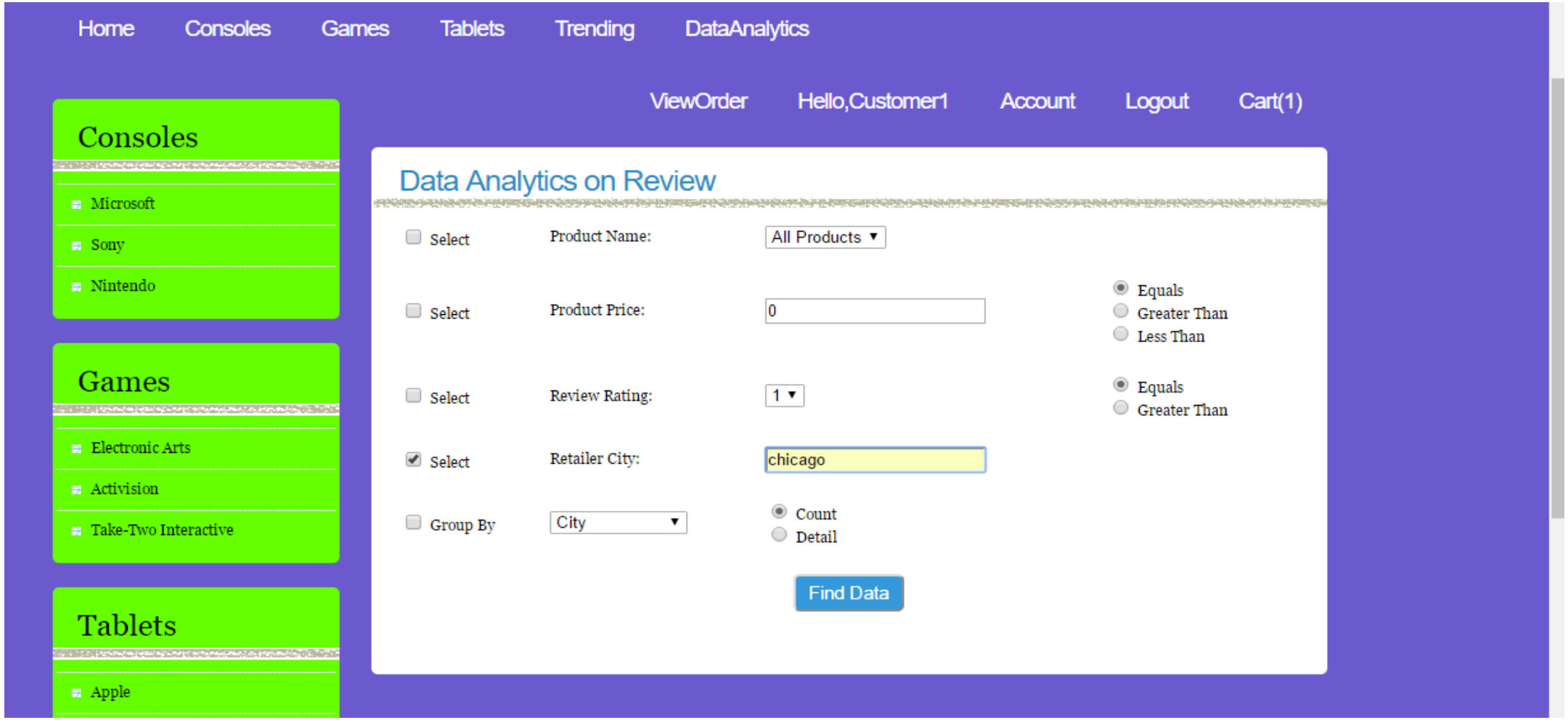
Select the group by filter to get count based on products



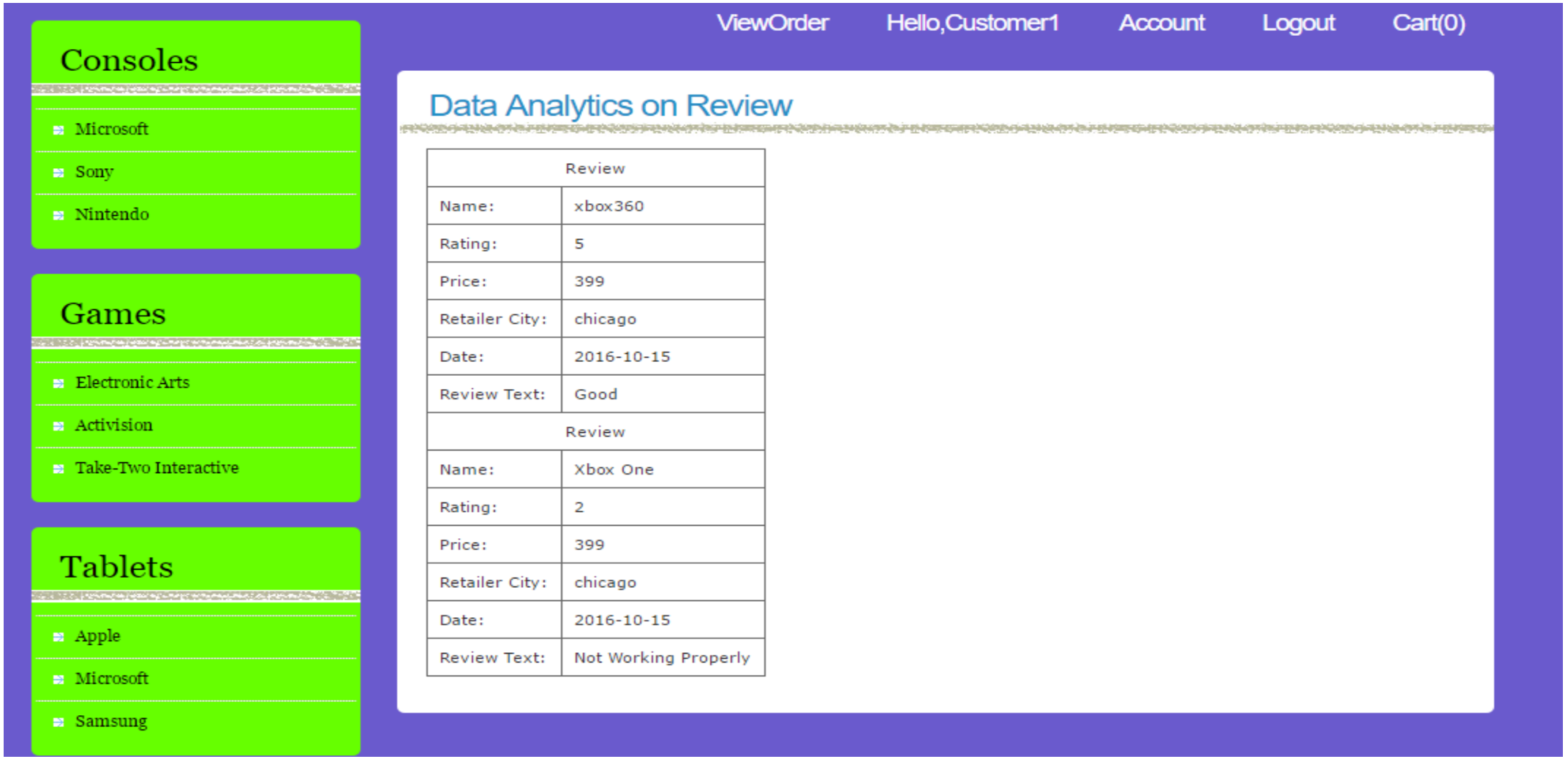
Result 4 - Print a list of how many reviews for every product



Query 5 - Get the list of reviews for shoppers in Chicago



Result 5 - Get the list of reviews for shoppers in Chicago



Sample Code: Top five zip-codes where maximum number of products sold

```
pw.print("<table id='bestseller'>");
        groupFields = new BasicDBObject("_id", 0);
        groupFields.put("count",new BasicDBObject("$sum",1));
        groupFields.put("_id", "$zipCode");
        group = new BasicDBObject("$group", groupFields);
        sort = new BasicDBObject();
        projectFields.put("value", "$_id");
        projectFields.put("ReviewValue","$count");
        project = new BasicDBObject("$project", projectFields);
        sort.put("ReviewValue",-1);
        orderby=new BasicDBObject("$sort",sort);
        limit=new BasicDBObject("$limit",5);
        aggregate = myReviews.aggregate(group,project,orderby,limit);
        constructGroupByContent(aggregate,pw);
  pw.print("</table>");
```

Sample Code: Top five zip-codes where maximum number of products sold

```
public void constructGroupByContent(AggregationOutput aggregate, PrintWriter pw)
{

      for (DBObject result : aggregate.results()) {
            BasicDBObject bobj = (BasicDBObject) result;
      String tableData = "<tr><td> "+bobj.getString("value")+"</td> "
                              +     "<td>"+bobj.getString("ReviewValue")+"</td></tr>";
                              pw.print(tableData);
      }
}
```

Sample Code for list of reviews where rating greater than 3:

```
int reviewRating = Integer.parseInt(request.getParameter("reviewRating"));
String compareRating = request.getParameter("compareRating");
String[] filters = request.getParameterValues("queryCheckBox");
myReviews=MongoDBDataStoreUtilities.getConnection();
BasicDBObject query = new BasicDBObject();
boolean noFilter = false;
boolean filterByRating = false;
if(filters != null){
        for (int i = 0; i < filters.length; i++) {
            //Check what all filters are ON
            //Build the query accordingly
            switch (filters[i]){
                case "reviewRating":
                    filterByRating = true;
                    if (compareRating.equals("EQUALS_TO")) {
                        query.put("reviewRating", reviewRating);
                    }else{
                    query.put("reviewRating", new BasicDBObject("$gt", reviewRating));
                    }
                    break;} }}
 DBCursor dbCursor = myReviews.find(query);
constructTableContent(dbCursor, pw);
```

Sample Code for list of reviews where rating greater than 3

```
public void constructTableContent(DBCursor dbCursor,PrintWriter pw)
{
	String tableData = "";
	pw.print("<table class='gridtable'>");
	while (dbCursor.hasNext())
	{
		BasicDBObject bobj = (BasicDBObject) dbCursor.next();
		tableData =   "<tr><td align='center' colspan='2'>Review</td></tr><tr><td>Name: </td><td>" +
bobj.getString("productName") + "</td></tr>"
				+ "<tr><td>Rating:</td><td>" + bobj.getString("reviewRating") + "</td></tr>"
				+ "<tr><td>Date:</td><td>" + bobj.getString("reviewDate") + "</td></tr>"
				+ "<tr><td>Review Text:</td><td>" + bobj.getString("reviewText")+"</td><tr>";
		pw.print(tableData);
	}
		pw.print("</table>");
	 //No data found
	if(dbCursor.count() == 0)
	{
		tableData = "<h2>No Data Found</h2>";
		pw.print(tableData);
	}
}
```

Questions?