Lab 8 – MongoDB – Query

# Objective

In this Lab, you learn to query a database in MongoDB to obtain information.

# Getting Started

In this lab, you will use products.json dataset. Download products.json from Blackboard and store it in a folder named dataset.

You will need to import the product into a MongoDB database. There are 2 ways to do this:

### Method 1 - Compass

1. Use Compass, create a database, named inventory, and a collection, named products, by clicking on Create Database
2. Navigate to the documents page in the created dbase/collection and click on “ADD DATA”
3. Import File
4. JSON
5. Select a file (the products.json file)

### Method 2 – Command Prompt

1. To import data, go to the *bin* directory: C:\Program Files\MongoDB\Server\5.0\**bin**
2. Execute the following command:   
   mongoimport --db inventory --collection products --file ..\dataset\products.json   
   ensuring the path to the products.json file is correct.  
   After executing the command, the data is imported to the inventory database.

To make sure data is imported successfully, use compass to navigate to the database collection and make sure there are product documents present.

Alternatively, show the data using the Mongo Shell. To see the documents inside the database:

> use inventory  
> db.products.find().forEach(printjson)

# Submission

You will submit your answers, along with the output from the MongoShell for each question through the online quiz setup for the lab.

# Tasks

1. Write a query to return *name* and *price* of each product in the *inventory* database.

db.products.find({}, {"name" : 1, "price" : 1, "\_id" : 0});

1. Write a query to return *name* and *price* for products of type *accessory* in the *inventory* database.

db.products.find({"type" : {"$in" : ["accessory"]}}, {"name" : 1, "price" : 1, "\_id" : 0})

1. Write a query to return *name* and *price* for products with price between $13 and $19 (Values *13* and *19* are included).

db.products.find({"price" : {"$gte" : 13, "$lte": 19}}, {"name" : 1, "price" : 1, "\_id" : 0})

1. Write a query to return the *id*, *name*, *price*, and *type* for products that are not of type *accessory*.

db.products.find({"type" : {"$ne" : "accessory"}}, {"name" : 1, "price" : 1, "\_id" : 1, "type" : 1})

1. Write a query to return *id*, *name*, *price*, and type for products with type *accessory* or *service*.

db.products.find({"$or" : [{"type" : {"$in" : ["accessory"]}}, {"type" : {"$in" : ["service"]}}]}, {"name" : 1, "price" : 1, "\_id" : 1, "type" : 1})

1. Write a query to return *id*, *name*, *price*, and *type* for products that do have the *type* key.

db.products.find({"type" : {"$exists" : true}}, {"name" : 1, "price" : 1, "\_id" : 1, "type" : 1})

1. Write a query to return *id*, *name*, *price*, and *type* for products that their type is both *accessory* and *case*.

db.products.find({"$and" : [{"type" : {"$in" : ["accessory"]}}, {"type" : {"$in" : ["case"]}}]}, {"name" : 1, "price" : 1, "\_id" : 1, "type" : 1})

1. In your own words, write about your experience with MongoDB so far and your interpretation of how tasks are completed tasks in MongoDB versus in a relational database such as Oracle. Do you think it is better, equal, or worse than relational databases?

My experience with MongoDB started early last summer, I was taking my WEB322 course and I had some exposure to it. But we didn’t have to run complicated queries so I don’t recall if we used “$in” or “$or” kinda operators. We used more stuff like insertOne, findOneAndUpdate etc.

In my opinion MongoDB is way easier than Oracle SQL. There is no complicated join statements, we don’t have to write lines of code to find a single output. I assume it would be faster to run queries in MongoDB than Oracle SQL. In conclusion, it is better, in terms of ease of learning, workload to create queries, and efficiency of the queries.

# Marking

Questions 1 through 7 are 1 mark each, 1/2 marks may be given for partially correct answers.

Question 8 is worth 3 marks and requires enough detail to justify all 3 marks.