4/8/2019

Mongo Database Report

Jonathan McDonagh
20074520

Contents

Introduction 2
Installation of Mongo 2
Setting up the cloud service 5
Original Entity Relationship Diagram 7
Mongo Database 7
Create 7
Customers 7
Menu 8
Waiter 10
Order 11
Queries 14
Aggregation 14
Update 15
Delete 15
Conclusion 16
References 17

Figures

Figure 1: Download for MongoDB 2
Figure 2: MongoDB Setup 3
Figure 3: Setup Finished for MongoDB 3
Figure 4: Mongod.exe from Command Line 4
Figure 5: Mongo.exe to MongoDB Shell 4
Figure 6: mongodump to export DB 5
Figure 7: mongorestore used to put database on the cloud 5
Figure 8: Connecting to database on Cloud and testing a find query on the Cloud 6
Figure 9: ER Diagram 7

Introduction

This report is for my NoSQL Databases module and we will look at the mongo database that I created using MongoDB. The database is for a Restaurant Database. My database has 4 tables put into a MongoDB. “MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need” (Mongo, 2019)

Installation of Mongo

I downloaded MongoDB from https://www.mongodb.com/download-center/community, and installed it onto my laptop. I started the MongoDB server by navigating to the mongo/bin in the command line. Then I typed mongo.exe to start the server. This allowed me to start the shell process and allowed me to create my database by entering my inserts that are in this report.

Here are a few screenshots of how I setup MongoDB.

Step 1: Here is where I downloaded MongoDB:

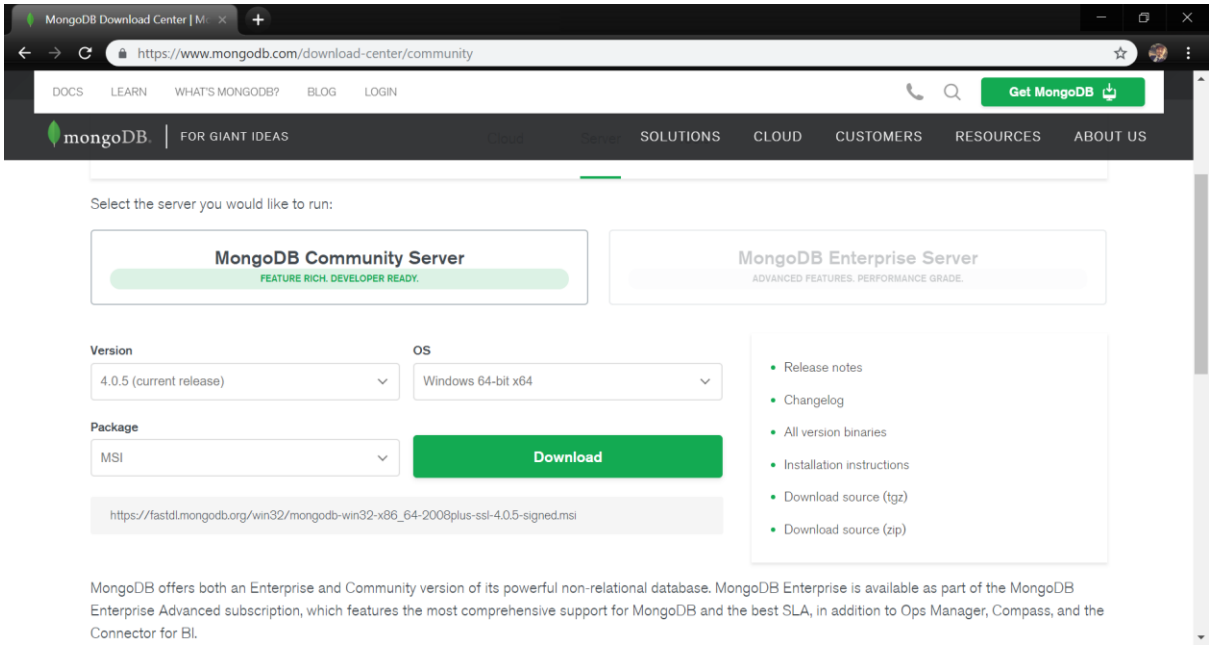

Figure 1: Download for MongoDB

Step 2: Here I started the installation process :

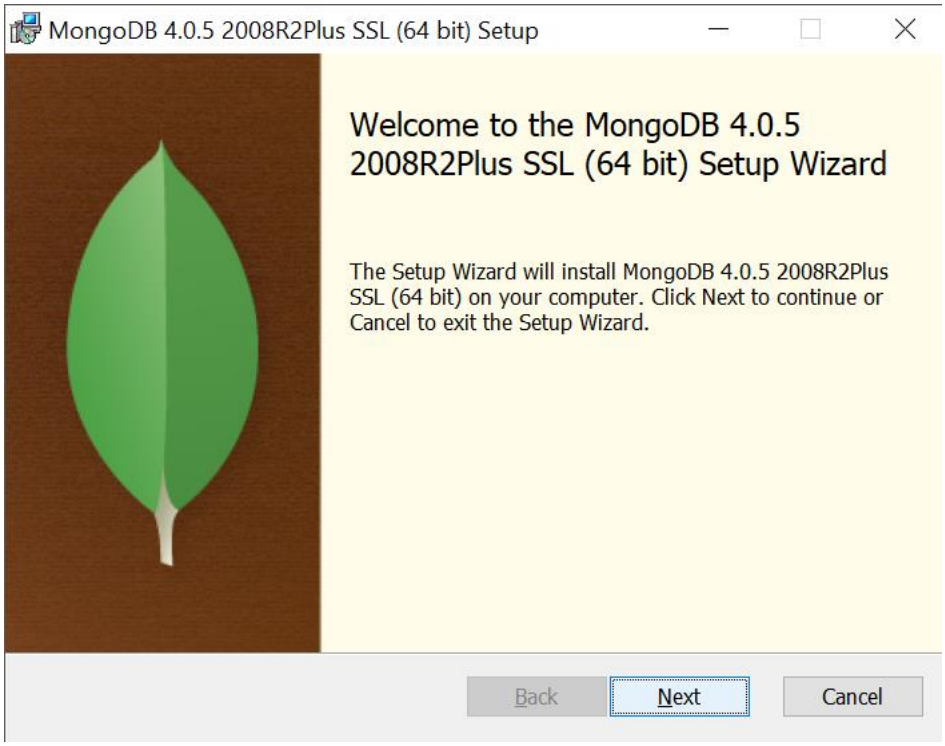

Figure 2: MongoDB Setup

Step 3: Here I completed the installation:

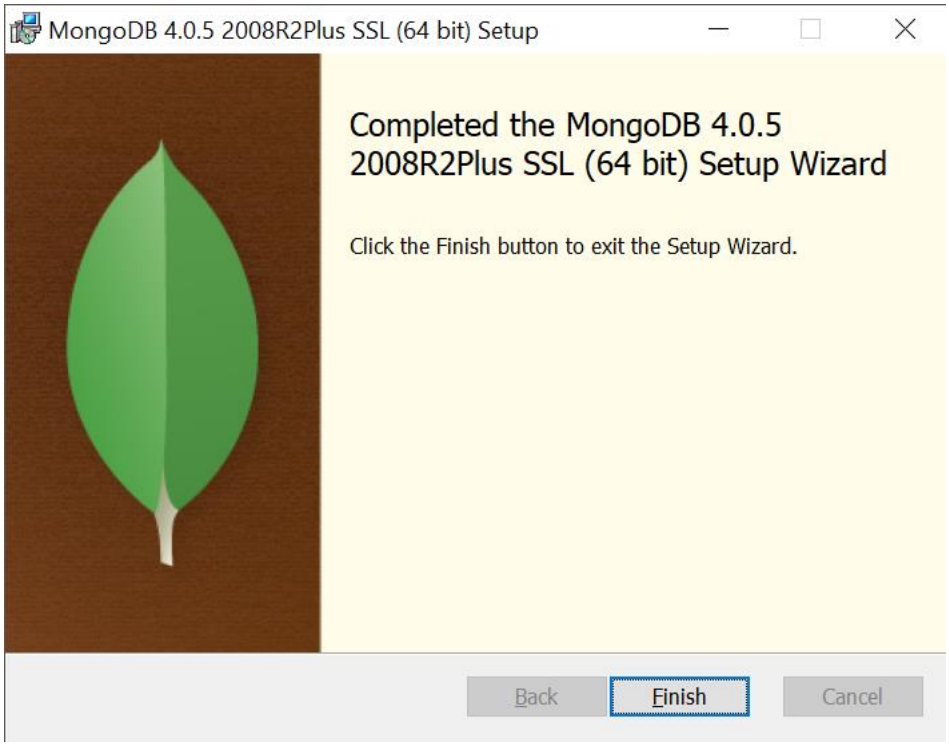

Figure 3: Setup Finished for MongoDB

Step 4: Here I executed mongod.exe from the MongoDB\4.0\bin\ directory

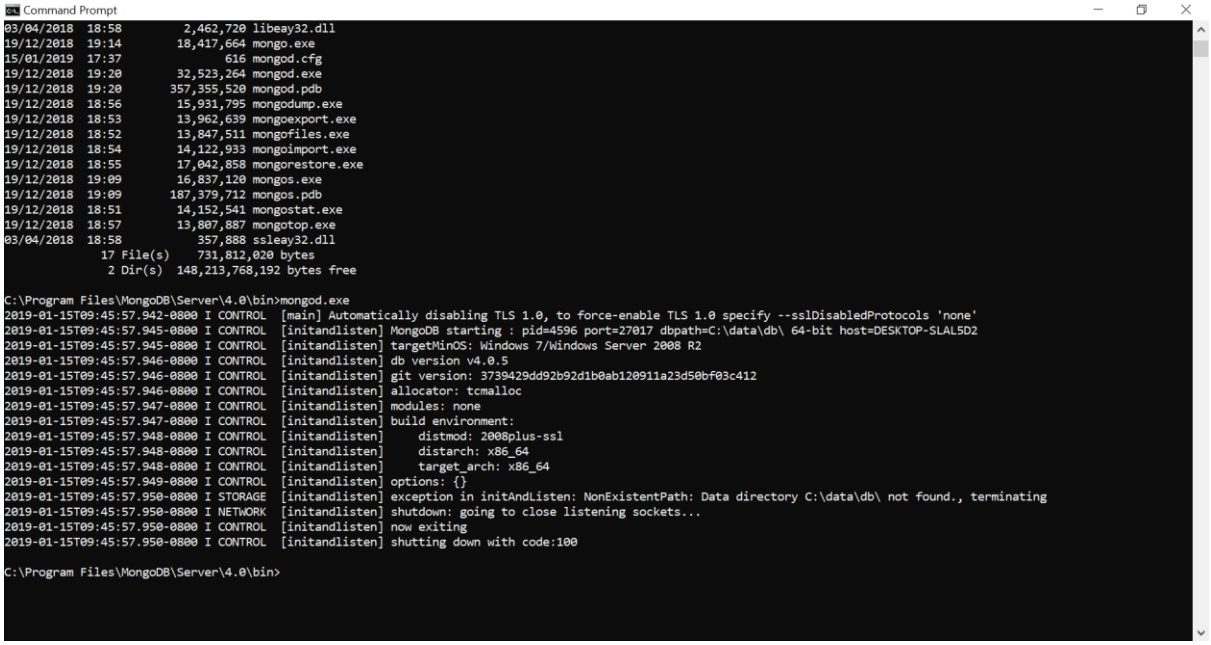

Figure 4: Mongod.exe from Command Line

Step 5: I then changed the directory to the bin folder of the MongoDB I then ran mongo.exe which brought me into the MongoDB shell

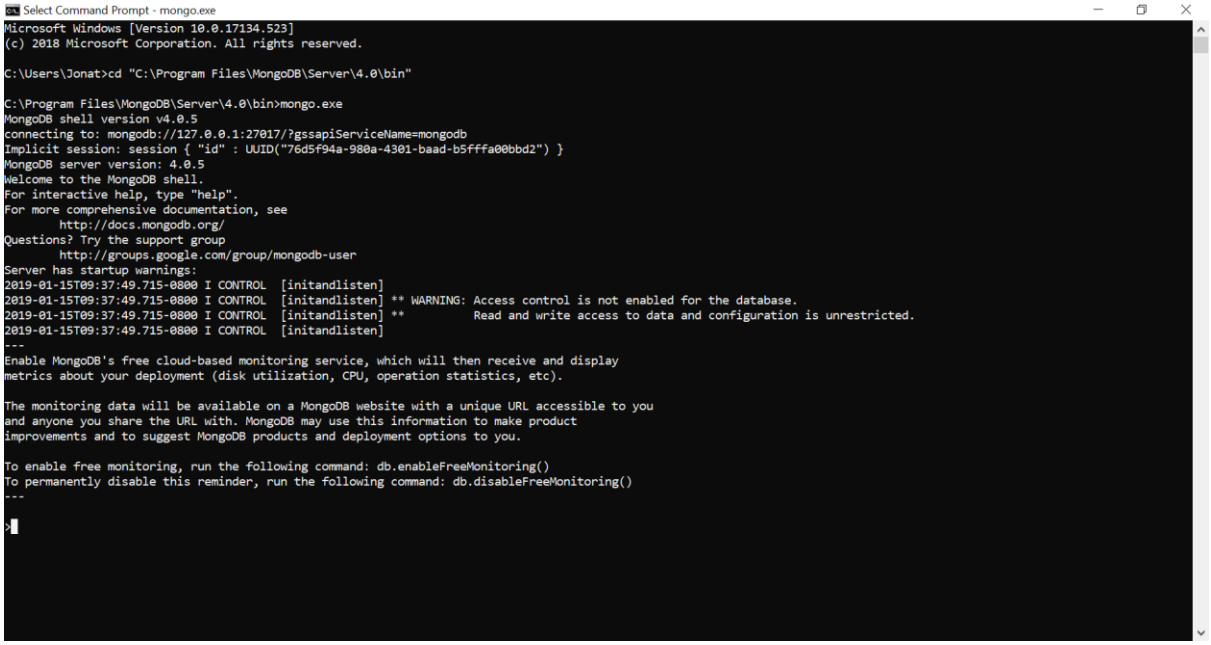

Figure 5: Mongo.exe to MongoDB Shell

Setting up the cloud service

To setup the cloud for my database I used Clever-Cloud. I started of by using mongodump in the command line to export the database. Once exported I used the mongorestore which I got of Clever-Cloud which put the database onto the cloud. Once I completed those two steps I used the mongo –host command which I got of Clever-Cloud. I could then run queries on the cloud from the command line.

Below is the mongodump command I used to export the database

```
C:\Program Files\MongoDB\Server\4.0\bin>mongodump --db Restaurant_DB --gzip --archive > dump_Restaurant_DB.gz
2019-04-02T19:21:40.383+0100    writing Restaurant_DB.Waiter to archive on stdout
2019-04-02T19:21:40.393+0100    writing Restaurant_DB.customer to archive on stdout
2019-04-02T19:21:40.393+0100    writing Restaurant_DB.menu to archive on stdout
2019-04-02T19:21:40.396+0100    writing Restaurant_DB.Order to archive on stdout
2019-04-02T19:21:40.423+0100    done dumping Restaurant_DB.Waiter (5 documents)
2019-04-02T19:21:41.427+0100    done dumping Restaurant_DB.menu (5 documents)
2019-04-02T19:21:41.428+0100    done dumping Restaurant_DB.customer (5 documents)
2019-04-02T19:21:41.429+0100    done dumping Restaurant_DB.Order (4 documents)
```

Figure 6: mongodump to export DB

Below is the mongorestore command I used to put the database on the cloud

```
C:\Program Files\MongoDB\Server\4.0\bin>mongorestore --host=bqqaneknatuqube-mongodb.services.clever-cloud.com --port=27017 --username=u7xmcvvmt5lqlfmjbikc --password=ndwSdQ
YqF5mrUj79HiZh --nsFrom="Restaurant_DB.*" --nsTo="bqqaneknatuqube.*" --authenticationDatabase="bqqaneknatuqube" --archive=C:\data\dump_Restaurant_DB.gz --gzip
2019-04-02T19:24:24.685+0100    preparing collections to restore from
2019-04-02T19:24:26.377+0100    reading metadata for bqqaneknatuqube.Waiter from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:27.500+0100    restoring bqqaneknatuqube.Waiter from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:29.309+0100    no indexes to restore
2019-04-02T19:24:29.310+0100    finished restoring bqqaneknatuqube.Waiter (5 documents)
2019-04-02T19:24:29.860+0100    reading metadata for bqqaneknatuqube.menu from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:31.191+0100    restoring bqqaneknatuqube.menu from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:31.193+0100    reading metadata for bqqaneknatuqube.customer from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:32.727+0100    no indexes to restore
2019-04-02T19:24:32.728+0100    finished restoring bqqaneknatuqube.menu (5 documents)
2019-04-02T19:24:32.731+0100    restoring bqqaneknatuqube.customer from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:32.738+0100    reading metadata for bqqaneknatuqube.Order from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:33.391+0100    bqqaneknatuqube.customer  541B
2019-04-02T19:24:33.548+0100    restoring bqqaneknatuqube.Order from archive 'C:\data\dump_Restaurant_DB.gz'
2019-04-02T19:24:33.549+0100    bqqaneknatuqube.customer  541B
2019-04-02T19:24:33.555+0100    no indexes to restore
2019-04-02T19:24:33.556+0100    finished restoring bqqaneknatuqube.customer (5 documents)
2019-04-02T19:24:34.162+0100    no indexes to restore
2019-04-02T19:24:34.162+0100    finished restoring bqqaneknatuqube.Order (4 documents)
2019-04-02T19:24:34.166+0100    done
```

Figure 7: mongorestore used to put database on the cloud

Below I connected to the cloud server and ran a simple find query for the database on the cloud

```
C:\Program Files\MongoDB\Server\4.0\bin>mongo --host bqqaneknatuqube-mongodb.services.clever-cloud.com --port 27017 --username=u7xmcvvmt5lqlfmjbikc --password=ndwSdQYqF5mrUj79HiZh bqqaneknatuqube
MongoDB shell version v4.0.5
connecting to: mongodb://bqqaneknatuqube-mongodb.services.clever-cloud.com:27017/bqqaneknatuqube?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7a938242-8db5-4a61-906f-63653c80d06b") }
MongoDB server version: 4.0.3
rs0:PRIMARY> db.menu.find().pretty();
{
        "_id" : ObjectId("5ca3a62a62afb83fad9e3733"),
        "items_ID" : 1,
        "Items" : "Any Burger and Fries",
        "Price" : 12
}
{
        "_id" : ObjectId("5ca3a63362afb83fad9e3734"),
        "items_ID" : 2,
        "Items" : "Pizza and Fries",
        "Price" : 15
}
{
        "_id" : ObjectId("5ca3a63a62afb83fad9e3735"),
        "item_ID" : 3,
        "Items" : "Sunday Roast and a pint",
        "Price" : 14
}
{
        "_id" : ObjectId("5ca3a63f62afb83fad9e3736"),
        "item_ID" : 4,
        "Items" : "Any desert",
        "Price" : 6
}
{
        "_id" : ObjectId("5ca3a65362afb83fad9e3738"),
        "item_ID" : 6,
        "Items" : "Pints",
        "Price" : 5
}
rs0:PRIMARY>
```

Figure 8: Connecting to database on Cloud and testing a find query on the Cloud

Original Entity Relationship Diagram

Below are the four tables that I focused on for my MongoDB Report.

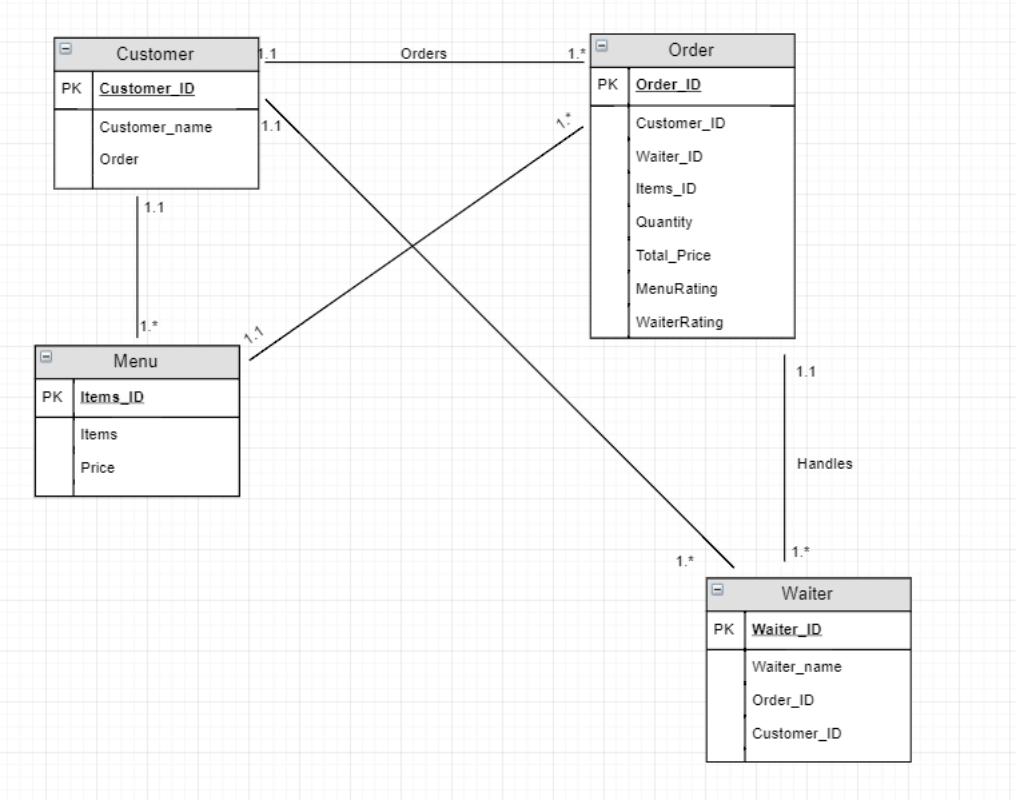

Figure 9: ER Diagram

Mongo Database

I created a database for a series of collections, Customer, Menu, Waiter and Order and I inserted multiple documents into each one. Below I will display all the code I used to create the CRUD (Create, Read, Update and Delete). The customer can have multiple items from the menu assigned to them using an array. I also used included some queries, here is the code below.

Create

Customers

```
db.customer.insert({

        customer_ID:1,

        customer_name:'Jonathan',

        order: 'Pizza meal and desert and drink'

        });
```

```
db.customer.insert({
        customer_ID:2,
        customer_name:'James',
        order: 'Burger meal and desert'
        });

db.customer.insert({
        customer_ID:3,
        customer_name:'Lauren',
        order: 'Pizza meal and desert and drink'
        });

db.customer.insert({
        customer_ID:4,
        customer_name:'Richie',
        order: 'Burger meal and desert'
        });

db.customer.insert({
        customer_ID:5,
        customer_name:'Nick',
        order: 'Burger meal and desert'
        });
```

Menu

```
db.menu.insert({
        items_ID:1,
        Items:'Any Burger and Fries',
        Price:'12'
        });
```

```
db.menu.insert({
        items_ID:2,
        Items:'Pizza and Fries',
        Price:'15'
        });

db.menu.insert({
        item_ID:3,
        Items:'Sunday Roast and a pint',
        Price:'14'
        });

db.menu.insert({
        item_ID:4,
        Items:'Any desert',
        Price:'6'
        });

db.menu.insert({
        item_ID:5,
        Items:'Curry with rice and chips',
        Price:'12'
        });

db.menu.insert({
        item_ID:6,
        Items:'Pints',
        Price:'5'
        });
```

Waiter

```
db.Waiter.insert({
        Waiter_ID:1,
        Waiter_name: 'Andy',
        Order_ID: 1,
        customer_ID: 1
        });

db.Waiter.insert({
        Waiter_ID:2,
        Waiter_name: 'Jessica',
        Order_ID: 2,
        customer_ID: 2
        });

db.Waiter.insert({
        Waiter_ID:3,
        Waiter_name: 'Mick',
        Order_ID: 3,
        customer_ID: 3
        });

db.Waiter.insert({
        Waiter_ID:4,
        Waiter_name: 'Jake',
        Order_ID: 2,
        customer_ID: 4,
        });
```

```
db.Waiter.insert({
        Waiter_ID:5,
        Waiter_name: 'Lisa',
        Order_ID: 1,
        customer_ID: 5
        });
```

Order

```
db.Order.insert({
        Order_ID: 1,
        customer_ID: 1,
        Waiter_ID: 1,
        item_ID: [
                2, 4, 6,
                ],
        Quantity: 1,
        Total_Price: 26,
        MenuRating: '5 out of 5',
        WaiterRating: '5 out of 5',
        });
```

```
db.Order.insert({
        Order_ID: 2,
        customer_ID: 2,
        Waiter_ID: 2,
        item_ID: [
                1, 4,
                ],
        Quantity: 1,
        Total_Price: 18,
        MenuRating: '5 out of 5',
        WaiterRating: '5 out of 5'
        });

db.Order.insert({
        Order_ID: 3,
        customer_ID: 3,
        Waiter_ID: 3,
        item_ID: [
                2, 4, 6,
                ],
        Quantity: 1,
        Total_Price: 26,
        MenuRating: '4 out of 5',
        WaiterRating: '5 out of 5'
        });
```

```
db.Order.insert({
        Order_ID: 4,
        customer_ID: 4,
        Waiter_ID: 4,
        item_ID: [
                1, 4,
                ],
        Quantity: 1,
        Total_Price: 18,
        MenuRating: '5 out of 5',
        WaiterRating: '5 out of 5'
        });

db.Order.insert({
        Order_ID: 5,
        customer_ID: 5,
        Waiter_ID: 5,
        item_ID: [
                1, 4,
                ],
        Quantity: 1,
        Total_Price: 18,
        MenuRating: '3 out of 5',
        WaiterRating: '4 out of 5'
        });
```

Queries

This query displays all customers who had item 2 from the menu (Pizza and Fries):

```
db.Order.find({item_ID:2,}).pretty();
```

These four find queries below just displays all the records in each table:

```
db.customer.find().pretty();
```

```
db.Waiter.find().pretty();
```

```
db.menu.find().pretty();
```

```
db.Order.find().pretty();
```

Here I have a query to count all the customers and waiters:

```
db.customer.find().count();
```

```
db.Waiter.find().count();
```

Here is a query that I used to find any items on the menu that are lower than 10 euro:

```
db.menu.find({"Price":{$lt:10}}).pretty();
```

and here is the query to find any items that are greater than 10 euro:

```
db.menu.find({"Price":{$gt:10}}).pretty();
```

Aggregation

This query $unwind deconstructs an array field from the input documents to output a document for each element. Each output document is the input document with the value of the array field replaced by the element.:

```
db.Order.aggregate([{ $unwind : "$item_ID" }]).pretty();
```

This query $project passes along the documents with the requested fields to the next stage in the pipeline. The specified fields can be existing fields from the input documents or newly computed fields. The code below displays all the customer names and their order:

```
db.customer.aggregate([{ $project : { customer_name: 1, order: 1} }]).pretty();
```

Update

Here I update the order for customer 5 (Nick), the update changes the order from ‘Burger meal and desert’ to ‘Burger meal and desert and pint’.

```
db.customer.update({
        customer_ID:5},
        {$set: { order: 'Burger meal and desert and pint'
        }
});
```

Here I update the item_ID for customer 5 (Nick), I updated the item_ID from ‘1, 4’ to ‘1, 4, 6’ so it includes the Burger meal (1) the desert (4) and the pint (6).

```
db.Order.update({
        Order_ID:5},
        {
        $set: {Order_ID: [1,4,6,], Total_Price: 23,}
        });
```

Delete

I created a simple remove statement to remove the only item on the menu that was not ordered the Chicken Curry. Its item_ID is 5.

```
db.menu.remove({
        item_ID:5,
        });
```

Conclusion

After completing my assignment I feel that I learned a lot of new skills regarding NoSQL and MongoDB. I found the initial design of my Mongo database a bit difficult but I quickly learned. I found the insertion of data to sometimes be a bit tedious. But saying that, I did enjoy learning the syntax and performing some queries.

I found Mongo to be extremely user friendly. For example I liked the way Exceptions are not thrown if a query contains an unknown field. I found MongoDB interesting due to its flexibility and its scalability. This definitely is a major advantage over other traditional databases.

I used Clever-Cloud to store my data on the cloud. Which I found extremely easy to setup.

Overall I really enjoyed this assignment and I'm happy that I was able to develop my skills in this area.

References

MongoDB. 2019. MongoDB. [Online] Available at: https://www.mongodb.com/what-is-mongodb [Accessed 05 March 2019].