Lab 13: Intro to MongoDB

Objective

The students should be able to understand the usage of:

- Schema Validation
- Update
- Delete
- Aggregation Pipeline

Submission Requirements

Compile your queries and screenshot of the output in a word document and upload it on LMS. If the output is large, you may only take the screenshot such that it covers one screen.

Restaurants Collection

You have already created this collection in the last lab.

Aggregation Pipelines

1. Find the average score for each restaurant and print along with its name. hint: $addFields
2. Find the average score, max score and min score for each cuisine.
3. Display the count of restaurants in each borough.
4. Find the count of restaurants for each cuisine and borough combination.
5. Find top 2 cuisines popular in Brooklyn. The Popularity is guaged by the average score for each cuisine. The final output should show averages that are less than equal to 99! *

Insert

6. Insert following into the restaurants collection.

```
{
  "address": {
    "building": "1007",
    "coord": [-73.856077, 40.848447],
    "street": "IBA st.",
    "zipcode": "10001"
  },
  "borough": "Karachi",
  "cuisine": "All",
  "grades": [
    {
      "date": new Date("2014-03-03T00:00:00Z"),
      "grade": "A",
      "score": 10
    },
    {
      "date": new Date("2011-03-10T00:00:00Z"),
      "grade": "B",
      "score": 14
    }
  ],
  "name": "IBA Students Canteen",
  "restaurant_id": "12340000"
}
```

Update

7. All bakeries in brooklyn have partnered with an ice-cream company. These bakeries will now also sell ice-creams. Modify the cuisine to “Bakery and Icecream”.

Delete

8. The restaurants in Queens have shut down. Modify the database to reflect the change.

Retrieve all

9. Find the restaurants that have all grades with a score greater than 5.

Order Collection

10. Create another collection named orders. Insert data via MongoDB Compass using the order.csv file provided on LMS. Use mongo shell to write queries for the following:
    a) Total amount of each Rep
    b) List all unique items.
    c) Total units of each item sold.
    d) Number of orders in each Region
    e) Total units sold of each item in the Central Region
    f) Total amount of each item sorted in descending order.

Scenario

Refer to the scenario create collections with appropriate validation rules.

- Each SalesRep is assigned to one department. Each department has many sales representatives.
- Each department is headed by a manager who is also a SalesRep
- Each customer can be dealt by multiple SalesRep and different SalesRep can deal with different customers

The Sales Rep collection stores the s_name, s_email and s_performance score. The customer collection contains the cust_name, cust_add and cust_email. The department collection contains the dept_name, dept_region, and grade.

Limitations:

- A SalesRep can only deal with maximum 10 customers
- A customer is dealt by maximum 5 SalesRep.
- The performance_score is given as a number between 1 and 10.
- The dept_region must be from among East, West, Central
- Grade is a single character string.
- The fields in red are mandatory fields.
- To implement the M-M relation, both sides will keep embedded documents. We only require name and email address in the array. *Hint: sub-schema*