Big Data Lab – Week 5: MongoDB and Python [solution]

Task 1: MongoDB aggregation

3. Use the aggregation pipeline to list the years with total movie runtime between **2000** and **4000**

```
> db.movieDetails.aggregate([ { $group: { _id: "$year", totalRuntime: { $sum: "$runtime" }}}, {$match: { totalRuntime: { $gt: 2000, $lt:4000 } }  }])

{ "_id" : 1995, "totalRuntime" : 2188 }
{ "_id" : 1986, "totalRuntime" : 2706 }
{ "_id" : 1997, "totalRuntime" : 2325 }
{ "_id" : 2001, "totalRuntime" : 3826 }
{ "_id" : 1988, "totalRuntime" : 2445 }
{ "_id" : 1998, "totalRuntime" : 2816 }
{ "_id" : 1990, "totalRuntime" : 2073 }
{ "_id" : 1996, "totalRuntime" : 2666 }
{ "_id" : 1989, "totalRuntime" : 2229 }
{ "_id" : 1994, "totalRuntime" : 2636 }
{ "_id" : 1981, "totalRuntime" : 2049 }
{ "_id" : 1993, "totalRuntime" : 2939 }
{ "_id" : 2015, "totalRuntime" : 3583 }
{ "_id" : 1999, "totalRuntime" : 3885 }
```

4. Use the aggregation pipeline to find the average year runtime. Hint – you will need two $group stages

```
> db.movieDetails.aggregate([ { $group: { _id: "$year", totalRuntime: { $sum: "$runtime" }}}, {$group: {_id:null, avgRuntime: { $avg: "$totalRuntime" } }  }])

{ "_id" : null, "avgRuntime" : 1359.0254237288136 }
```

5. Find all restaurants whose name contains at a 3-digit number with spaces before and after the number. Part of the result is shown below, there are 38 matching restaurants. *Hint:* use the *$regex* operator

```
> db.restaurants.find({"name": {$regex: / [0-9]{3} / }},{"name":1, "_id":0})

{ "name" : "Cafe 101 16Th Floor Cafeteria" }
{ "name" : "Jerry'S 637 Diner" }
{ "name" : "Tea Shop 168 & Bakery" }
{ "name" : "Grill 149 Plus" }
{ "name" : "West 190 Street Pizza" }
{ "name" : "Lounge 247 I M O K" }
{ "name" : "Stand 142 The Original Cascarino\u001aS Brante" }
{ "name" : "Stand 140 Beer Island" }
{ "name" : "Stand 139 Blue Smoke" }
{ "name" : "Stand 139 Taco Frites" }
{ "name" : "Stand 139 Shake Shack" }
```

```
{ "name" : "Stands 303 And 301 Pepsi Porch" }
{ "name" : "Stand 335 Beverages & Snacks" }
{ "name" : "Stand 325 Nathan'S Dogs & Burgers" }
{ "name" : "Stand 321 Caesars Club" }
{ "name" : "Stand 320 - Premio" }
{ "name" : "Stand 127 Food Court" }
{ "name" : "Stand 110 A" }
{ "name" : "Stand 334 Beer Room" }
{ "name" : "Stand 110 B" }
```

6. Use the aggregation pipeline to reproduce the same result as in question 10 (names of all restaurants with a grade dated 2011-03-03).

```
> db.restaurants.aggregate ( [
   { $match : {"grades.date":ISODate("2011-03-03T00:00:00Z")} },
   { $project : {"name":1,"_id": 0 } }
   ] )

{ "name" : "The Assembly Bar" }
{ "name" : "Benny'S Famous Pizza" }
{ "name" : "Ray'S Pizza Restuarant" }
{ "name" : "La Piazza Pizzeria & Restaurant" }
{ "name" : "Kennedy Fried Chicken" }
{ "name" : "Barbara Blum Residence / Good Shepherd Services" }
{ "name" : "Le Pain Quotidien" }
{ "name" : "Blimpie" }
```

You could also try to reproduce the result of the *challenge* exercise in question 10 using the aggregation pipeline. This will involve using the *$filter* operator within the *$project* stage, and you may need to do some research to achieve this. As before, only the first document in the result is shown below.

```
> db.restaurants.aggregate ( [ {$match:{"grades.date":ISODate("2011-03-
03T00:00:00Z")}}, {$project: {
     "grades": {$filter:
     {input: '$grades', as: 'grade',
     cond: {$eq: ['$$grade.date', ISODate("2011-03-03T00:00:00Z")]}
     }},
"name":1,
"_id": 0
} }
]).pretty()


{
        "name" : "The Assembly Bar",
        "grades" : [
                {
                        "date" : ISODate("2011-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                }
        ]
}
```

7. Use map-reduce to find the number of Chinese restaurants in each borough. You should find that there are 2418 keys emitted reduced to 6 output.

```
var mapFunction = function() {
    emit(this.borough, 1);
};

var reduceFunction = function(key,values) {
    return Array.sum(values);
};

db.restaurants.mapReduce(
    mapFunction,
    reduceFunction,
    {
        query: {cuisine: "Chinese"},
        out: "map_reduce_exercise"
    }
);

db. map_reduce_exercise.find()
```

```
{ "_id" : "Bronx", "value" : 323 }
{ "_id" : "Brooklyn", "value" : 763 }
{ "_id" : "Manhattan", "value" : 510 }
{ "_id" : "Missing", "value" : 6 }
{ "_id" : "Queens", "value" : 728 }
{ "_id" : "Staten Island", "value" : 88 }
```