1 MongoDB

1. Download the file restaurants.json from Canvas and load it into a MongoDB database. The file contains data on 3,772 restaurants in New York City. Write the following queries in MongoDB (show the output from your queries):
    a. Display all the restaurants located in the boroughs Bronx or Brooklyn.

```
restaurantsdb> db.restaurants.find({borough: {$in:['Brooklyn', 'Bronx']}}, {name: 1, _id: 0})
[
  { name: 'Morris Park Bake Shop' },
  { name: "Wendy'S" },
  { name: 'Riviera Caterer' },
  { name: "Wilken'S Fine Food" },
  { name: 'Regina Caterers' },
  { name: 'Taste The Tropics Ice Cream' },
  { name: 'Wild Asia' },
  { name: 'C & C Catering Service' },
  { name: 'May May Kitchen' },
  { name: 'Seuda Foods' },
  { name: 'Carvel Ice Cream' },
  { name: 'Nordic Delicacies' },
  { name: 'The Movable Feast' },
  { name: 'White Castle' },
  { name: "Shashemene Int'L Restaura" },
  { name: 'Carvel Ice Cream' },
  { name: "Dunkin' Donuts" },
  { name: 'Mejlander & Mulgannon' },
  { name: 'Happy Garden' },
  { name: "Sonny'S Heros" }
]
```

    b. Find the restaurant id, name, borough and cuisine for those restaurants whose name starts with the letters 'Mad'.

```
restaurantsdb> db.restaurants.find({name: /^Mad/}, {name: 1, borough: 1, cuisine: 1})
[
  {
    _id: ObjectId("654afe101a2373c469ad5496"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Madison Square'
  },
  {
    _id: ObjectId("654afe101a2373c469ad556b"),
    borough: 'Manhattan',
    cuisine: 'Indian',
    name: 'Madras Mahal'
  },
  {
    _id: ObjectId("654afe101a2373c469ad5816"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Madame X'
  }
```

    c. Find the restaurants that have received a score between 80 and 90.

```
restaurantsdb> db.restaurants.find({grades: {$elemMatch: {score: {$gte: 80, $lte: 90}}}}, {name: 1, grades: 1})
[
  {
    _id: ObjectId("654afe101a2373c469ad59b5"),
    grades: [
      {
        date: ISODate("2014-10-31T00:00:00.000Z"),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate("2014-05-16T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2013-11-19T00:00:00.000Z"),
        grade: 'A',
        score: 5
      },
      {
        date: ISODate("2013-03-25T00:00:00.000Z"),
        grade: 'B',
        score: 19
      },
```

d. Display the restaurant id and name of restaurants which have received a ’C’ grade in year 2014.

```
restaurantsdb> db.restaurants.find({"grades": {$elemMatch: {"grade": "C", "date": {$gte: new Date("2014-01-01"), $lte: n
ew Date("2015-01-01")}}}}, {restaurant_id: 1, name:1, _id: 0})
[
  { name: 'B & M Hot Bagel & Grocery', restaurant_id: '40364299' },
  { name: 'Texas Rotisserie', restaurant_id: '40364304' },
  { name: 'Nyac Main Dining Room', restaurant_id: '40364467' },
  { name: "Mitchell'S Restaurant", restaurant_id: '40366961' },
  { name: "Mcdonald'S", restaurant_id: '40370781' },
  { name: 'Burger King', restaurant_id: '40370916' },
  { name: 'Caffe Dante', restaurant_id: '40373149' },
  { name: 'Omonia Cafe', restaurant_id: '40372445' },
  { name: "Murals On 54/Randolphs'S", restaurant_id: '40372466' },
  {
    name: 'Amsterdam Restaurant & Tapas Lounge',
    restaurant_id: '40377111'
  },
  { name: 'Candle Bar', restaurant_id: '40376031' },
  { name: 'Johns Cafe & Resturant', restaurant_id: '40377630' },
  { name: 'Kang Suh Korean Restaurant', restaurant_id: '40378729' },
  { name: "Mom'S Fried Chicken", restaurant_id: '40382900' },
  { name: "Rosa'S Pizza", restaurant_id: '40389701' },
```

e. Find the cuisine that has the highest number of restaurants.

```
restaurantsdb> db.restaurants.aggregate([{$group: {_id: "$cuisine", count: {$sum: 1 }}}, {$sort: {count: -1}}, {$limit:
1}])
[ { _id: 'American ', count: 1255 } ]
```

f. Find the restaurants that do not prepare an ’American’ cuisine and their average grade score is higher than 30. Display the restaurant ids and their average score.

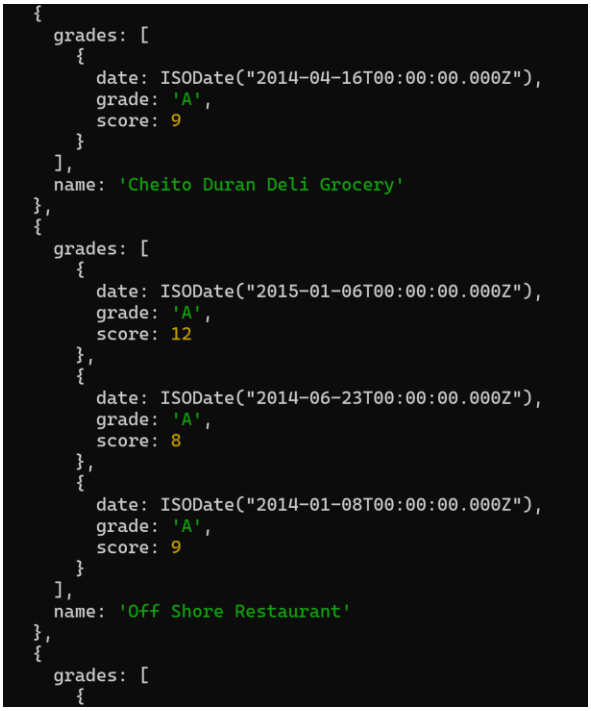
```
restaurantsdb> db.restaurants.aggregate([{$match: {cuisine: {$ne: 'American'}}}, {$unwind: '$grad
es'}, {$group: {_id: '$restaurant_id', averageScore: {$avg: '$grades.score'}}}, {$match: {average
Score: {$gt: 30}}}, {$project: {averageScore: 1, _id: 1}}])
[
  { _id: '40372466', averageScore: 33.666666666666664 },
  { _id: '40624470', averageScore: 30.6 },
  { _id: '40393488', averageScore: 38.6 },
  { _id: '40387237', averageScore: 32.6 },
  { _id: '40825993', averageScore: 30.8 },
  { _id: '40374268', averageScore: 30.8 },
  { _id: '40756344', averageScore: 36 },
  { _id: '40366157', averageScore: 32.142857142857146 }
]
```

g. For each restaurant display only the grades that were recorded from the year 2014 onwards.

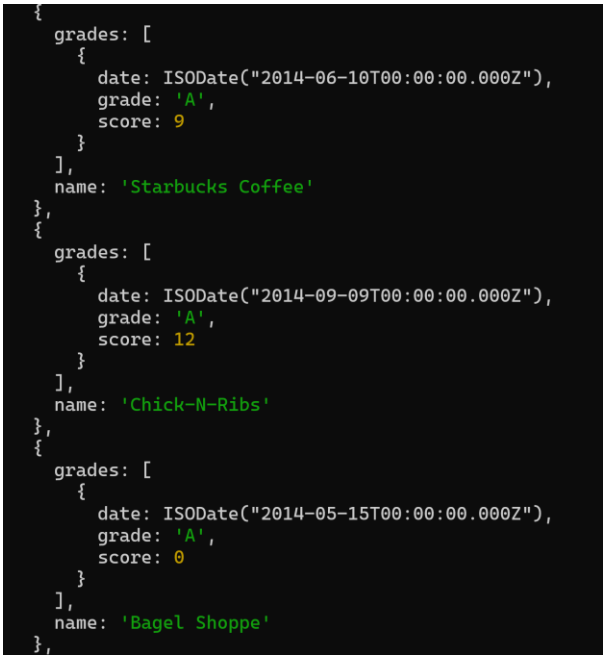
```
restaurantsdb> db.restaurants.aggregate([{$unwind: '$grades'}, {$match: {'grades.date': {$gte: new Date('2014/01/01')}}}, {$g
roup: {_id: {restaurant_id: '$restaurant_id', name:'$name'}, grades: {$push: '$grades'}}}, {$project: {_id: 0, name:'$_id.nam
e', grades:1}}])
[
  {
    grades: [
      {
        date: ISODate("2014-07-30T00:00:00.000Z"),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate("2014-02-19T00:00:00.000Z"),
        grade: 'A',
        score: 11
      }
    ],
    name: 'Diamante Poblano Restaurant'
  },
  {
    grades: [
      {
        date: ISODate("2014-12-11T00:00:00.000Z"),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate("2014-05-14T00:00:00.000Z"),
        grade: 'A',
        score: 13
      }
    ],
    name: 'Le Singe Vert'
```

```
  grades: [
    {
      date: ISODate("2014-10-16T00:00:00.000Z"),
      grade: 'A',
      score: 13
    },
    {
      date: ISODate("2014-05-29T00:00:00.000Z"),
      grade: 'A',
      score: 4
    }
  ],
  name: 'Boulevard Tavern'
},
{
  grades: [
    {
      date: ISODate("2014-06-18T00:00:00.000Z"),
      grade: 'B',
      score: 17
    }
  ],
  name: 'Corato Pizza Ii'
},
{
  grades: [
    {
      date: ISODate("2014-05-13T00:00:00.000Z"),
      grade: 'A',
      score: 7
    }
  ],
  name: 'New Wave Cafe & Restaurant'
```

```
{
  grades: [
    {
      date: ISODate("2014-06-06T00:00:00.000Z"),
      grade: 'A',
      score: 12
    }
  ],
  name: 'Bowery Eletric'
},
{
  grades: [
    {
      date: ISODate("2014-06-03T00:00:00.000Z"),
      grade: 'A',
      score: 11
    }
  ],
  name: 'The Triad'
},
{
  grades: [
    {
      date: ISODate("2014-04-21T00:00:00.000Z"),
      grade: 'A',
      score: 6
    }
  ],
  name: 'Pax Wholesome Foods'
},
```



```
{
  grades: [
    {
      date: ISODate("2014-04-16T00:00:00.000Z"),
      grade: 'A',
      score: 9
    }
  ],
  name: 'Cheito Duran Deli Grocery'
},
{
  grades: [
    {
      date: ISODate("2015-01-06T00:00:00.000Z"),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate("2014-06-23T00:00:00.000Z"),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate("2014-01-08T00:00:00.000Z"),
      grade: 'A',
      score: 9
    }
  ],
  name: 'Off Shore Restaurant'
},
{
  grades: [
    {
```



```
{
  grades: [
    {
      date: ISODate("2014-06-10T00:00:00.000Z"),
      grade: 'A',
      score: 9
    }
  ],
  name: 'Starbucks Coffee'
},
{
  grades: [
    {
      date: ISODate("2014-09-09T00:00:00.000Z"),
      grade: 'A',
      score: 12
    }
  ],
  name: 'Chick-N-Ribs'
},
{
  grades: [
    {
      date: ISODate("2014-05-15T00:00:00.000Z"),
      grade: 'A',
      score: 0
    }
  ],
  name: 'Bagel Shoppe'
},
```

```
{
  grades: [
    {
      date: ISODate("2015-01-15T00:00:00.000Z"),
      grade: 'Z',
      score: 19
    },
    {
      date: ISODate("2014-07-31T00:00:00.000Z"),
      grade: 'A',
      score: 10
    }
  ],
  name: 'Keens Steakhouse'
},
{
  grades: [
    {
      date: ISODate("2014-03-03T00:00:00.000Z"),
      grade: 'A',
      score: 7
    }
  ],
  name: 'Flik International'
},
```

```
{
  grades: [
    {
      date: ISODate("2014-08-20T00:00:00.000Z"),
      grade: 'A',
      score: 7
    },
    {
      date: ISODate("2014-03-11T00:00:00.000Z"),
      grade: 'A',
      score: 8
    }
  ],
  name: "Lenny & John'S Pizza"
},
{
  grades: [
    {
      date: ISODate("2014-11-26T00:00:00.000Z"),
      grade: 'A',
      score: 13
    },
    {
      date: ISODate("2014-04-25T00:00:00.000Z"),
      grade: 'A',
      score: 12
    }
  ],
  name: 'Renaissance Hotel'
},
```

```
  {
    grades: [
      {
        date: ISODate("2014-03-11T00:00:00.000Z"),
        grade: 'A',
        score: 11
      }
    ],
    name: 'Nanni Restaurant'
  },
  {
    grades: [
      {
        date: ISODate("2014-03-06T00:00:00.000Z"),
        grade: 'A',
        score: 2
      }
    ],
    name: "Molly'S"
  },
  {
    grades: [
      {
        date: ISODate("2014-03-04T00:00:00.000Z"),
        grade: 'A',
        score: 7
      }
    ],
    name: 'Park Bar'
  }
```

h. Calculate the average score across all the restaurants in the collection

```
restaurantsdb> db.restaurants.aggregate([{$unwind: '$grades'}, {$group: {_id: null,
averageRating: {$avg: '$grades.score'}}}])
[ { _id: null, averageRating: 11.427736743468195 } ]
```

2. sales collection
   a. Find total quantity of sales for each state and sort list in descending order

```
restaurantsdb> db.sales.aggregate([{$group: {_id: {'state': '$ state '}, 'total_qty'
: {$sum: '$ qty '}}}, {$sort: {total_qty: -1}}])
[
  { _id: { state: ' OR ' }, total_qty: 1741 },
  { _id: { state: ' CA ' }, total_qty: 1455 },
  { _id: { state: ' NV ' }, total_qty: 655 }
]
```

   b. perform following operations
      i. Change quantity of sales in Berkley from 648 to 750

```
restaurantsdb> db.sales.updateOne({' city ': ' Berkeley '} , {$set: {' qty ':  750}}
)
```

      ii. Increase the quantity of all the sales in Oregon by 50

```
restaurantsdb> db.sales.updateMany({' state ': ' OR '}, {$inc: {' qty ': 50}})
{
```

      iii. Add to sales no. 5 the names of the salespeople: David and Martha.

```
restaurantsdb> db.sales.updateOne({' _id ':5}, {$set: {' salespeople ': ['David', 'M
artha']}})
```

      iv. Add James to the list of salespeople of sales no. 5

```
restaurantsdb> db.sales.updateOne({' _id ':5}, {$push: {' salespeople ': 'James'}})
```

      v. Replace Martha with Lisa in the salesperson list of sales no. 5

```
restaurantsdb> db.sales.updateOne({' _id ':5, ' salespeople ': 'Martha'}, {$set: {'
salespeople .$': 'Lisa'}})
```

vi. Delete all the sales from California

```
restaurantsdb> db.sales.deleteMany({ ' state ': ' CA '})
```

c. print final collection

```
restaurantsdb> db.sales.find({})
[
  {
    _id: ObjectId("654eae7399e4eb998ed1ba04"),
    ' _id ': 2,
    ' city ': ' Bend ',
    ' state ': ' OR ',
    ' qty ': 541
  },
  {
    _id: ObjectId("654eae7399e4eb998ed1ba06"),
    ' _id ': 4,
    ' city ': ' Eugene ',
    ' state ': ' OR ',
    ' qty ': 892
  },
  {
    _id: ObjectId("654eae7399e4eb998ed1ba07"),
    ' _id ': 5,
    ' city ': ' Reno ',
    ' state ': ' NV ',
    ' qty ': 655,
    ' salespeople ': [ 'David', 'Lisa', 'James' ]
  },
  {
    _id: ObjectId("654eae7399e4eb998ed1ba08"),
    ' _id ': 6,
    ' city ': ' Portland ',
    ' state ': ' OR ',
    ' qty ': 458
  }
]
```

2 Map Reduce

1. Projection $\pi_S(R)$: From each tuple of relation R produce only the components for the attributes in S

Map Function: Get each tuple in R. Have a tuple (t, t). For each tuple, remove attributes that are not in S, and get (t', t')

Reduce Function: Will be multiple (t', t') for each t'. For each t', there will be (t', [t', t', …]) and reduce will result in one (t', t') pair for t'

Psuedo Code:

Map(key, value):

value_attributes = value.split()

selectedAttributes = [attribute1, attribute2, ….]

resultAttributes = []

for attribute in selectedAttributes:

if attribute in value_attributes:

resultAttributes.append(attribute)

emit(key, resultAttributes)

Reduce(key, values):

value = values[0]

emit(key, value)

2. Intersection R∩S: Return the tuples that are present in both relations R and S. Assume that relations R and S have the same schema (same attributes and same type).

Map Function: turn each tuple into a key, value pair

Reduce Function: If for key t, have value list of [t,t], then output (t,t). If have value list of just [t], then output nothing

Psuedo Code:

Map(key, value):

emit(key, value)

Reduce(key, value):

if len(value) = 2:

emit(key, value)

else:

emit None

3. Grouping $\gamma_{A,\theta(B)}(R)$. Given a relation R(A, B, C), with one grouping attribute A, one aggregated attribute B, and another attribute C, which is neither grouped or aggregated:
    a. Partition the tuples of R according to their values in attribute A
    b. For each group, aggregate the values in attribute B and apply function θ on the aggregated value (θ is an aggregation operation such as SUM, COUNT or MAX)

Map Function: for each tuple (a, b, c) produce a key value pair of (a, b)

Reduce Function: for each key a, will have [b1, b2, b3, ….]. Apply whatever function θ to [b1, b2, b3, ….]. θ ([b1, b2, b3, ….] = x. output (a, x)

Psuedo code:

Map(key, value):

#assume have a variable tuple = (a, b, c)

key = tuple[0]

value = tuple[1]

emit(key, value)

Reduce(key, values):

#assume for each key, have a collection of its associated values, also assume have function θ

x = θ (values)

emit(key, x)

3 Spark

1. Show the total number of missions for each of the countries involved (according to ContryFlyingMission). Write this query (a) With the DataFrame API (b) using Spark SQL (c) with RDD operations. Which of these methods was the most efficient?

2. Plot a bar chart with the number of missions by country.

3. Plot the number of missions per day for each of the countries involved.

4. How many takeoffs were launched to attack North Vietnam on 29 June 1966 from each location?

5. Which campaigns saw the heaviest bombings? (the names of the campaigns are stored in OperationSupported).

6. Which month saw the highest number of missions?

7. What was the most used aircraft type during the war (in terms of number of missions)