Lab 10 Querying Data (Solution)

DB Query:

- **db.collection.find(query, projection)**: Selects documents in a collection or view and returns a cursor to the selected documents.

query	Optional. Specifies selection filter using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
projection	Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter. For details, see Projection.
Returns:	A cursor to the documents that match the query criteria. When the find() method "returns documents," the method is actually returning a cursor to the documents.

Comparison Operator:

Syntax: { <field> : {<operator> : <value>} }

$eq   = EQual to ,   **$nq**   != Not Equal to,   **$gt**   >   Greater Than,

$gte >= Greater Than or Equal to,   **$lt**   <   Less Than,

$lte   <= Less Than or Equal to

Logical Operator:

Syntax: { <operator> : [{clause1}, {clause2},…] }

Syntax: { $not : {clause1} }

$and   Match all of the specified query clauses

$or   At least one of the query clauses is matched

$nor   Fail to match both of given clauses

$not   Negates the query requirement

Cursor Methods: These methods modify the way that the underlying query is executed.

Some Cursor Methods:

cursor.count()	Modifies the cursor to return the number of documents in the result set rather than the documents themselves.

cursor.limit()	Constrains the size of a cursor's result set. 　　　　db.collection.find(<query>).limit(<number>) You must specify a numeric value for limit(). A limit() value of 0 (i.e. .limit(0)) is equivalent to setting no limit.

cursor.pretty()	Configures the cursor to display results in an easy-to-read format.

cursor.sort()	Returns results ordered according to a sort specification. 　　　　sort({field : value}, {...}) Specify in the sort parameter the field or fields to sort by and a value of 1 or -1 to specify an ascending or descending sort Respectively. Example: sort({ age : -1, posts: 1 })

More cursor methods: https://docs.mongodb.com/v4.2/reference/method/js-cursor/

Example:

```
db.trips.find({ "tripduration": { "$lte" : 70 }).limit(3);
```

```
db.trips.find({"tripduration":{"$lte":70}).sort({"tripduration":-1});
```

Resource:

- **db.collection.aggregate(pipeline):** Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.

    - pipeline array (A sequence of data aggregation operations or stages)

        pipeline = [{ $match : { ... },

                         { $group : { ... },

                         { $sort : { ... },

                    ...   ]

Aggregation Operators

$match	Used to match documents (like SQL where clause)
$project	Used to populate specific field's value(s)
$group	Used to group documents by specific field
$sum	Used to count or sum the values inside a group.
$avg	Calculates average of specific field's value per group.

$min	Finds the minimum value of a field in each group.
$max	Finds the maximum value of a field in each group.
$first, $last	Getting specific field's value from first and last document of each group: Works well when document result is sorted.
$limit	Passes the first n documents unmodified to the pipeline where n is the specified limit. For each input document, outputs either one document (for the first n documents) or zero documents (after the first n documents).
$skip	Skips the first n documents where n is the specified skip number and passes the remaining documents unmodified to the pipeline. For each input document, outputs either zero documents (for the first n documents) or one document (if after the first n documents).
$sort	Reorders the document stream by a specified sort key. Only the order changes; the documents remain unmodified. For each input document, outputs one document.
$lookup	Used to perform an equality match between a field from the input documents with a field from the documents of the "joined" collection

Example:

```
db.employees.aggregate( [
      {"$match": {"totalExp": {$gt: 3}}},
      {"$group": {"_id": "$dept",     "average_age": {"$avg":
"$age"}}},
{"$match": {"average_age": {"$lte": 35}}} ] )
```

```
Output:
{ _id: 'Facilities', average_age: 33.5 }
{ _id: 'Admin', average_age: 32.5 }


db.employees.aggregate([{$group:{"_id":"$dept",
"noOfEmployee":{$sum:1},
"minExp":{$min:"$totalExp"}}}]);

Output:
{ "_id" : "HR", "noOfEmployee" : 2, "minExp" : 3 }
{ "_id" : "Facilities", "noOfEmployee" : 2, "minExp" : 4 }
{ "_id" : "Admin", "noOfEmployee" : 2, "minExp" : 10 }

db.employees.aggregate([{$group:{"_id":"$dept",
"noOfEmployee":{$sum:1},
"maxExp":{$max:"$totalExp"}}}]);

Output:
{ "_id" : "HR", "noOfEmployee" : 2, "maxExp" : 3 }
{ "_id" : "Facilities", "noOfEmployee" : 2, "maxExp" : 14 }
{ "_id" : "Admin", "noOfEmployee" : 2, "maxExp" : 11 }
```

Resource: https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/#mongodb-method-db.collection.aggregate

Task 1: Querying data using an aggregate()/distinct() collection method.

- Use the "zips" collection of the "sample_training" database to answer the questions 1-6.

1.1. Write a MongoDB query to retrieve the number of cities grouped by state. Arrange the result by the number of cities in descending order. List only the first 5 documents.

```
db.zips.aggregate( [
        {$group: {_id: "$state", "num#city" : {$sum:1} } },
        {$sort: {"num#city": -1}},
        {$limit : 5}
] );
```

```
{ _id: 'TX', 'num#city': 1676 }
```

```
{ _id: 'NY', 'num#city': 1596 }
{ _id: 'CA', 'num#city': 1523 }
{ _id: 'PA', 'num#city': 1458 }
{ _id: 'IL', 'num#city': 1240 }
```

Note: count : {$sum : 1}, //counts the number

1.2. Write a MongoDB query to retrieve the total number of population, grouped by city; retrieve the results only the state "AL" and the first 10 documents.

```
db.zips.aggregate( [
    {$match: {state: "AL"}},
    {$group: { _id: "$city", totalPop : {$sum: "$pop"} }},
    {$limit: 10}
] ) ;
```

```
{ _id: 'TITUS', totalPop: 2683 }
{ _id: 'HAMILTON', totalPop: 11184 }
{ _id: 'BRIERFIELD', totalPop: 1282 }
{ _id: 'MILLERVILLE', totalPop: 863 }
{ _id: 'BLACK', totalPop: 183 }
{ _id: 'LITTLE RIVER', totalPop: 649 }
{ _id: 'WHATLEY', totalPop: 790 }
{ _id: 'BURKVILLE', totalPop: 1478 }
{ _id: 'HURTSBORO', totalPop: 1727 }
{ _id: 'HEADLAND', totalPop: 4595 }
```

1.3. Write a MongoDB query to retrieve the maximum and minimum population, grouped by state. Display the states that the minimum population is greater than 20 and arrange the results by the maximum population in ascending order.

```
db.zips.aggregate( [
{$group: { _id: "$state", maxPop : {$max: "$pop"} , minPop : {$min: "$pop"} }
}
, {$match: {minPop: {$gt: 20}}}, {$sort: {maxPop: 1}}
```

```
]);
```

```
{ _id: 'NH', maxPop: 41438, minPop: 27 }

{ _id: 'DE', maxPop: 50573, minPop: 108 }

{ _id: 'RI', maxPop: 53733, minPop: 45 }

{ _id: 'IN', maxPop: 56543, minPop: 23 }

{ _id: 'CT', maxPop: 60670, minPop: 25 }

{ _id: 'OH', maxPop: 66674, minPop: 38 }
```

1.4. Write a MongoDB query to retrieve the maximum population grouped by state of 'AK', 'AL', 'AR', 'AZ'. Display the states that the maximum population is greater than 30000 and arrange the results by the state in descending order.

```
db.zips.aggregate( [
    {$match: {$or: [{state: "AK"},{state: "AL"},{state: "AR"},{state: "AZ"}] }},
    {$group: { _id: "$state", maxPop : {$max: "$pop"} }},
    {$match: {maxPop: {$gt: 30000}}} ,
    {$sort: {_id: -1}} ] ) ;
```

```
{ _id: 'AZ', maxPop: 57131 }

{ _id: 'AR', maxPop: 53532 }

{ _id: 'AL', maxPop: 44165 }

{ _id: 'AK', maxPop: 32383 }
```

1.5. Write a MongoDB query to display the state, the average population and the number of cities grouped by states. Arrange the results by the average population in descending order and display only the first 2 documents.

```
db.zips.aggregate( [
    {$group: { _id: "$state", avgPop : {$avg: "$pop"} , "num#city" : {$sum: 1} }},
    {$sort: {avgPop: -1}},
    {$limit: 2}
]);
```

```
{ _id: 'DC', avgPop: 25287.5, 'num#city': 24 }

{ _id: 'CA', avgPop: 19540.39461588969, 'num#city': 1523 }
```

*** ควรใช้ $sort ก่อน $limit ถ้าใช้ $limit ก่อน $sort ผลลัพธ์ที่ได้จะต่างกัน ***

```
db.zips.aggregate( [
   {$group: { _id: "$state", avgPop : {$avg: "$pop"} , "num#city" : {$sum: 1} }},
   {$limit: 2},
   {$sort: {avgPop: -1}}
] ) ;
```

```
{ _id: 'AL', avgPop: 7126.255731922399, 'num#city': 567 }

{ _id: 'ND', avgPop: 1617.2151898734178, 'num#city': 395 }
```

1.6. Write a MongoDB query to display all distinct values of the states.

```
 db.zips.distinct("state")
 [
   'AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT',
   'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID',
   'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD',
   'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC',
   'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY',
   'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD',
   'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI',
   'WV', 'WY'
 ]
```

- Use the "sample_analytics" database to answer the questions 7-9.

1.7. Evaluate all collections of the "sample_analytics" database, please identify fields that represent the relationship among the collections.

Collections => Fields

accounts : customers => account_id: accounts

accounts: transaction => account_id: account_id

transaction : customers => account_id: accounts

1.8. Write a MongoDB query to display all documents of the "accounts" collection that maps to the documents of the "customers" collection. List only the first 3 documents.

```
db.accounts.aggregate([
  { $lookup:
    {
      from: "customers",
      localField: "account_id",
      foreignField: "accounts",
      as: "account_docs"
    } },
    { $limit: 3}
])
```

{ _id: ObjectId("5ca4bbc7a2dd94ee5816238e"),
  account_id: 198100,
  limit: 10000,
  products: ['Derivatives', 'CurrencyService', 'InvestmentStock'],
  account_docs:
   [{ _id: ObjectId("5ca4bbcea2dd94ee58162aaa"),
      username: 'earroyo',
      name: 'Teresa Smith',
      address: '477 Wall Inlet\nJasonhaven, NM 02213',
      birthdate: 1991-07-15T18:11:02.000Z,
      email: 'melissapatrick@yahoo.com',
      accounts: [198100, 674364, 278603, 383777, 209363],
      tier_and_details:
       { '6715aad3c524498f8d885bd2f428e40c':
          { tier: 'Platinum',
            benefits: ['shopping discounts', 'car rental insurance'],
            active: true,
            id: '6715aad3c524498f8d885bd2f428e40c' } } }] }
{ _id: ObjectId("5ca4bbc7a2dd94ee58162396"),
  account_id: 260499,
  limit: 10000,
  products: ['InvestmentFund', 'Derivatives', 'InvestmentStock'],
  account_docs:
   [{ _id: ObjectId("5ca4bbcea2dd94ee58162a9d"),
      username: 'petergilbert',
      name: 'Angela Campbell',
      address: '4068 Espinoza Mills\nWest Jessica, WV 60790',
      birthdate: 1991-12-15T18:03:16.000Z,
      email: 'jocelyn67@yahoo.com',
```

        accounts: [ 260499, 946116, 588389, 293111, 126444, 678107 ],
        tier_and_details:
         { '1768a44b906c4303b765ea38b1b4bbb2':
           { tier: 'Platinum',
             benefits: [ 'concierge services', 'travel insurance' ],
             active: true,
             id: '1768a44b906c4303b765ea38b1b4bbb2' },
          '1f6047657041457d9a77a3a3cce687ff':
           { tier: 'Gold',
             benefits: [ 'car rental insurance', 'concert tickets' ],
             active: true,
             id: '1f6047657041457d9a77a3a3cce687ff' },
          '3e05a76a2fc24390b1fb34b0148433f9':
           { tier: 'Gold',
             benefits: [ 'concierge services' ],
             active: true,
             id: '3e05a76a2fc24390b1fb34b0148433f9' } } } ] }
 { _id: ObjectId("5ca4bbc7a2dd94ee58162392"),
   account_id: 794875,
   limit: 9000,
   products: [ 'InvestmentFund', 'InvestmentStock' ],
   account_docs:
    [ { _id: ObjectId("5ca4bbcea2dd94ee58162ab2"),
        username: 'lyoung',
        name: 'Kaitlin Miller',
        address: '546 Tyler Prairie\nMarybury, AL 54175',
        birthdate: 1985-05-27T08:39:47.000Z,
        email: 'mariahmcpherson@gmail.com',
        accounts: [ 794875, 931314, 557378 ],
        tier_and_details:
         { '622f472651c649ba97d374415773b1db':
           { tier: 'Gold',
             benefits: [ 'sports tickets' ],
             active: true,
             id: '622f472651c649ba97d374415773b1db' },
          ab752fd305b14edaa110f8e9dbff401a:
           { tier: 'Silver',
             benefits: [ 'financial planning assistance' ],
             active: true,
             id: 'ab752fd305b14edaa110f8e9dbff401a' },
          be6f1b091ec24a87b92e7e63e472c9c6:
           { tier: 'Silver',
             benefits: [ 'shopping discounts' ],
             active: true,

id: 'be6f1b091ec24a87b92e7e63e472c9c6' } } } ] }

1.9. Write a MongoDB query to display all documents of the "transactions" collection that maps to the documents of the "customers" collection. List only one document with three fields: account_id, transaction_count and the output mapping field.

```
db.transactions.aggregate([
        { $lookup:
        {
                from: "customers",
                localField: "account_id",
                foreignField: "accounts",
                as: "transaction_docs"
        }},         {$project        :{account_id:1,        transaction_count:1,
transaction_docs:1}},
        { $limit: 1}
])
```

```
{ _id: ObjectId("5ca4bbc1a2dd94ee58161cb3"),
  account_id: 557378,
  transaction_count: 56,
  transaction_docs:
   [ { _id: ObjectId("5ca4bbcea2dd94ee58162ab2"),
       username: 'lyoung',
       name: 'Kaitlin Miller',
       address: '546 Tyler Prairie\nMarybury, AL 54175',
       birthdate: 1985-05-27T08:39:47.000Z,
       email: 'mariahmcpherson@gmail.com',
       accounts: [ 794875, 931314, 557378 ],
       tier_and_details:
        { '622f472651c649ba97d374415773b1db':
           { tier: 'Gold',
             benefits: [ 'sports tickets' ],
             active: true,
             id: '622f472651c649ba97d374415773b1db' },
          ab752fd305b14edaa110f8e9dbff401a:
           { tier: 'Silver',
```

```
        benefits: [ 'financial planning assistance' ],

        active: true,

        id: 'ab752fd305b14edaa110f8e9dbff401a' },

    be6f1b091ec24a87b92e7e63e472c9c6:

     { tier: 'Silver',

        benefits: [ 'shopping discounts' ],

        active: true,

        id: 'be6f1b091ec24a87b92e7e63e472c9c6' } } } ] }
```

1.10. Select one database and create your own questions and answer those by writing MongoDB queries. **Do at least 3 questions** with different types of answers.

Task 2: Querying data using a find() collection method.

- Use the "restaurants" collection of the "sample_restaurants" database to answer the questions 1-10.

2.1.   Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant.

db.restaurants.find({},
    {"restaurant_id" : 1,"name":1,"borough":1,"address.zipcode" :1,"_id":0});

```
> db.restaurants.find( {},
                       {"restaurant_id" : 1,"name":1,"borough":1,"address.zipcode" :1,"_id":0}  );
< { address: { zipcode: '11226' },
    borough: 'Brooklyn',
    name: 'Taste The Tropics Ice Cream',
    restaurant_id: '40356731' }
  { address: { zipcode: '10314' },
    borough: 'Staten Island',
    name: 'Kosher Island',
    restaurant_id: '40356442' }
  { address: { zipcode: '11004' },
    borough: 'Queens',
    name: 'Carvel Ice Cream',
    restaurant_id: '40361322' }
  { address: { zipcode: '11214' },
    borough: 'Brooklyn',
    name: 'C & C Catering Service',
    restaurant_id: '40357437' }
  { address: { zipcode: '11368' },
    borough: 'Queens',
    name: 'Ho Mei Restaurant',
    restaurant_id: '40362432' }
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.2.   Write a MongoDB query to display the first 3 restaurants which are in the borough Bronx.

db.restaurants.find({"borough": "Bronx"}).limit(3);

```
db.restaurants.find( {},
                      {"restaurant_id" : 1,"name":1,"borough":1,"address.zipcode" :1,"_id":0}  );
{ address: { zipcode: '11226' },
  borough: 'Brooklyn',
  name: 'Taste The Tropics Ice Cream',
  restaurant_id: '40356731' }
{ address: { zipcode: '10314' },
  borough: 'Staten Island',
  name: 'Kosher Island',
  restaurant_id: '40356442' }
{ address: { zipcode: '11004' },
  borough: 'Queens',
  name: 'Carvel Ice Cream',
  restaurant_id: '40361322' }
{ address: { zipcode: '11214' },
  borough: 'Brooklyn',
  name: 'C & C Catering Service',
  restaurant_id: '40357437' }
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.3.　　　Write a MongoDB query to display the next 5 restaurants after skipping the first 10 which are in the borough Bronx.

db.restaurants.find({"borough": "Bronx"}).skip(10).limit(5);

```
db.restaurants.find({"borough": "Bronx"}).skip(10).limit(5);
{ _id: ObjectId("5eb3d668b31de5d588f42b05"),
  address:
   { building: '1014',
     coord: [ -73.89687649999999, 40.8166811 ],
     street: 'Longwood Avenue',
     zipcode: '10459' },
  borough: 'Bronx',
  cuisine: 'Pizza',
  grades:
   [ { date: 2014-09-25T00:00:00.000Z, grade: 'Z', score: 24 },
     { date: 2014-03-19T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2014-01-21T00:00:00.000Z, grade: 'P', score: 10 },
     { date: 2013-06-14T00:00:00.000Z, grade: 'C', score: 40 },
     { date: 2012-11-29T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2012-06-22T00:00:00.000Z, grade: 'B', score: 22 },
     { date: 2012-01-19T00:00:00.000Z, grade: 'B', score: 22 } ],
  name: 'Tony\'S Pizza',
  restaurant_id: '40378347' }
{ _id: ObjectId("5eb3d668b31de5d588f4292e"),
  address:
   { building: '1007',
     coord: [ -73.856077, 40.848447 ],
     street: 'Morris Park Ave',
     zipcode: '10462' },
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

    2.4.   Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.

```
db.restaurants.find(
              { "cuisine" : {$ne : "American "},
                "grades.score" :{$gt: 70},
                "address.coord" : {$lt : -65.754168} }
         );
```

14

```
db.restaurants.find(
                    { "cuisine" : {$ne : "American "},
                      "grades.score" :{$gt: 70},
                      "address.coord.0" : {$lt : -65.754168}  }
                );
```

```
> db.restaurants.find(
                      { "cuisine" : {$ne : "American "},
                        "grades.score" :{$gt: 70},
                        "address.coord.0" : {$lt : -65.754168}  }
                  );
< { _id: ObjectId("5eb3d668b31de5d588f42a92"),
    address:
     { building: '65',
       coord: [ -73.9782725, 40.7624022 ],
       street: 'West   54 Street',
       zipcode: '10019' },
    borough: 'Manhattan',
    cuisine: 'American',
    grades:
     [ { date: 2014-08-22T00:00:00.000Z, grade: 'A', score: 11 },
       { date: 2014-03-28T00:00:00.000Z, grade: 'C', score: 131 },
       { date: 2013-09-25T00:00:00.000Z, grade: 'A', score: 11 },
       { date: 2013-04-08T00:00:00.000Z, grade: 'B', score: 25 },
       { date: 2012-10-15T00:00:00.000Z, grade: 'A', score: 11 },
       { date: 2011-10-19T00:00:00.000Z, grade: 'A', score: 13 } ],
    name: 'Murals On 54/Randolphs\'S',
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.5.   Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belonging to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.

```
db.restaurants.find( {
                    "cuisine" : {$ne : "American "},
                    "grades.grade" :"A",
                    "borough": {$ne : "Brooklyn"} } ).sort({"cuisine":-1});
```

15

```
db.restaurants.find( {
                              "cuisine" : {$ne : "American "},
                              "grades.grade" :"A",
                              "borough": {$ne : "Brooklyn"} } ).sort({"cuisine":-1});
{ _id: ObjectId("5eb3d668b31de5d588f430eb"),
  address:
   { building: '8278',
     coord: [ -73.88143509999999, 40.7412552 ],
     street: 'Broadway',
     zipcode: '11373' },
  borough: 'Queens',
  cuisine: 'Vietnamese/Cambodian/Malaysia',
  grades:
   [ { date: 2014-06-12T00:00:00.000Z, grade: 'B', score: 21 },
     { date: 2013-05-20T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2012-12-26T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2012-12-03T00:00:00.000Z, grade: 'P', score: 5 },
     { date: 2012-05-04T00:00:00.000Z, grade: 'B', score: 27 } ],
  name: 'Pho Bac Vietnamese Seafood Cuisine',
  restaurant_id: '40578058' }
{ _id: ObjectId("5eb3d668b31de5d588f43060"),
  address:
   { building: '89',
     coord: [ -73.9995899, 40.7168015 ],
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.6.   Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepare either American or Chinese dishes.

db.restaurants.find(
{ "borough": "Bronx" , $or : [{ "cuisine" : "American " },{ "cuisine" : "Chinese" }] });

```
db.restaurants.find(
{ "borough": "Bronx" , $or : [{ "cuisine" : "American " },{ "cuisine" : "Chinese" }] } );
{ _id: ObjectId("5eb3d668b31de5d588f4295c"),
  address:
   { building: '277',
     coord: [ -73.8941893, 40.8634684 ],
     street: 'East Kingsbridge Road',
     zipcode: '10458' },
  borough: 'Bronx',
  cuisine: 'Chinese',
  grades:
   [ { date: 2014-03-03T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2013-09-26T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2013-03-19T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2012-08-29T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2011-08-17T00:00:00.000Z, grade: 'A', score: 13 } ],
  name: 'Happy Garden',
  restaurant_id: '40364296' }
{ _id: ObjectId("5eb3d668b31de5d588f4294b"),
  address:
   { building: '1236',
     coord: [ -73.8893654, 40.81376179999999 ],
     street: '238 Spofford Ave',
     zipcode: '10474' },
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.7. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronx or Brooklyn.

```
db.restaurants.find(
        {"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},
        {"restaurant_id" : 1, "name":1,"borough":1, "cuisine" :1}
);
```

```
db.restaurants.find(
{"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},
{"restaurant_id" : 1, "name":1,"borough":1, "cuisine" :1}
);
{ _id: ObjectId("5eb3d668b31de5d588f42967"),
  borough: 'Manhattan',
  cuisine: 'American',
  name: 'Metropolitan Club',
  restaurant_id: '40364347' }
{ _id: ObjectId("5eb3d668b31de5d588f4299b"),
  borough: 'Manhattan',
  cuisine: 'Spanish',
  name: 'Spain Restaurant & Bar',
  restaurant_id: '40365472' }
{ _id: ObjectId("5eb3d668b31de5d588f429a9"),
  borough: 'Manhattan',
  cuisine: 'Café/Coffee/Tea',
  name: 'Mee Sum Coffee Shop',
  restaurant_id: '40365904' }
{ _id: ObjectId("5eb3d668b31de5d588f42a6c"),
  borough: 'Manhattan',
  cuisine: 'Japanese',
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.8.   Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10. Arrange the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

```
db.restaurants.find(
{"grades.score" :
{ $not: {$gt : 10}} },
{"restaurant_id" : 1,"name":1,"borough":1, "cuisine" :1}
). sort({"cuisine":1,"borough" : -1});
```

```
db.restaurants.find(
{"grades.score" :
{ $not: {$gt : 10}} },
{"restaurant_id" : 1,"name":1,"borough":1, "cuisine" :1}
). sort({"cuisine":1,"borough" : -1});
{ _id: ObjectId("5eb3d669b31de5d588f467e9"),
  borough: 'Queens',
  cuisine: 'Afghan',
  name: 'Bakhtar Kabab',
  restaurant_id: '41661199' }
{ _id: ObjectId("5eb3d668b31de5d588f445d6"),
  borough: 'Queens',
  cuisine: 'African',
  name: 'Aso Rock Lounge And Restaurant',
  restaurant_id: '41260089' }
{ _id: ObjectId("5eb3d669b31de5d588f47222"),
  borough: 'Manhattan',
  cuisine: 'African',
  name: 'Cross Culture Kitchen',
  restaurant_id: '50000133' }
{ _id: ObjectId("5eb3d669b31de5d588f47df0"),
  borough: 'Manhattan',
  cuisine: 'African',
  name: 'Le Baobab-Gouygui Restaurant',
  restaurant_id: '50007470' }
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.9. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates. Arrange the date in descending order.

```
db.restaurants.find(
        { "grades.date": ISODate("2014-08-11T00:00:00Z"),
         "grades.grade":"A" ,
         "grades.score" : 11
        },
        {"restaurant_id" : 1,"name":1,"grades":1}
      ).sort({date: -1});
```

```
db.restaurants.find(
                {
                  "grades.date": ISODate("2014-08-11T00:00:00Z"),
                  "grades.grade":"A" ,
                  "grades.score" : 11
                },
                {"restaurant_id" : 1,"name":1,"grades":1}
            ).sort({date: -1});
{ _id: ObjectId("5eb3d668b31de5d588f429a5"),
  grades:
   [ { date: 2014-08-11T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-07-22T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2013-03-14T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-07-02T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2012-02-02T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2011-08-24T00:00:00.000Z, grade: 'A', score: 11 } ],
  name: 'Neary\'S Pub',
  restaurant_id: '40365871' }
{ _id: ObjectId("5eb3d668b31de5d588f42a85"),
  grades:
   [ { date: 2014-08-11T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2013-12-10T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2013-06-10T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-06-08T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2012-01-25T00:00:00.000Z, grade: 'A', score: 8 },
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***

2.10.   Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of the coord array contains a value which is more than 42 and upto 52.

```
db.restaurants.find(
        { "address.coord.1": {$gt : 42, $lte : 52}},
        {"restaurant_id" : 1,"name":1,"address":1,"coord":1});
```

```
> db.restaurants.find(  { "address.coord.1": {$gt : 42, $lte : 52}},
                        {"restaurant_id" : 1,"name":1,"address":1,"coord":1}
                );
{ _id: ObjectId("5eb3d668b31de5d588f42bf6"),
  address:
   { building: '1',
     coord: [ -0.7119979, 51.6514664 ],
     street: 'Pennplaza E, Penn Sta',
     zipcode: '10001' },
  name: 'T.G.I. Fridays',
  restaurant_id: '40388936' }
{ _id: ObjectId("5eb3d668b31de5d588f42bc7"),
  address:
   { building: '47',
     coord: [ -78.877224, 42.89546199999999 ],
     street: 'Broadway @ Trinity Pl',
     zipcode: '10006' },
  name: 'T.G.I. Friday\'S',
  restaurant_id: '40387990' }
{ _id: ObjectId("5eb3d668b31de5d588f42e4c"),
  address:
   { building: '3000',
     coord: [ -87.86567699999999, 42.61150920000001 ],
     street: '47 Avenue',
     zipcode: '11101' },
```

*** แสดงผลลัพธ์บางส่วน ไม่ได้แสดงผลลัพธ์ทั้งหมด***