Universidad de las Américas Puebla

LIS4112 Cloud Computing and Big Data

José Antonio Solís Martínez

ID: 162442

Hands On 1: MongoDB Querying

20/02/2022

Question 1

List all the restaurants in the collection, sorted in increasing order of names

Query:

```
db.RestaurantsCollection.find({"name": {$ne:""}}, {"name": 1}).sort({"name": 1})
```

Execution:

```
{ _id: ObjectId("61f355e3d3284e8b0a6fa347"),
  name: '#1 Garden Chinese' }
{ _id: ObjectId("61f355ecd3284e8b0a6fcb9b"),
  name: '#1 Me. Nick\'S' }
{ _id: ObjectId("61f355ddd3284e8b0a6f9123"),
  name: '#1 Sabor Latino Restaurant' }
{ _id: ObjectId("61f355e6d3284e8b0a6fb160"),
  name: '$1.25 Pizza' }
{ _id: ObjectId("61f355e4d3284e8b0a6fa981"),
  name: '\'\'U\'\' Like Chinese Restaurant' }
{ _id: ObjectId("61f355eed3284e8b0a6fd200"),
  name: '\'\'W\'\' Cafe' }
{ _id: ObjectId("61f355e1d3284e8b0a6f9ebe"),
  name: '\'Wichcraft' }
{ _id: ObjectId("61f355e8d3284e8b0a6fb754"),
  name: '\'Wichcraft' }
{ _id: ObjectId("61f355d6d3284e8b0a6f7c68"),
  name: '(Lewis Drug Store) Locanda Vini E Olii' }
{ _id: ObjectId("61f355ebd3284e8b0a6fc4f4"),
  name: '(Library)  Four & Twenty Blackbirds' }
{ _id: ObjectId("61f355dfd3284e8b0a6f96e8"),
```

```
  name: '(Public Fare) 81St Street And Central Park West (Delacorte Theatre)' }
{ _id: ObjectId("61f355ddd3284e8b0a6f9202"),
  name: '/ L\'Ecole' }
{ _id: ObjectId("61f355edd3284e8b0a6fced9"),
  name: '002 Mercury Tacos Llc' }
{ _id: ObjectId("61f355ddd3284e8b0a6f905d"),
  name: '1 2 3 Burger Shot Beer' }
{ _id: ObjectId("61f355ddd3284e8b0a6f9006"),
  name: '1 Banana Queen' }
{ _id: ObjectId("61f355e5d3284e8b0a6fabbe"),
  name: '1 Buen Sabor' }
{ _id: ObjectId("61f355d7d3284e8b0a6f835b"), name: '1 Darbar' }
{ _id: ObjectId("61f355d2d3284e8b0a6f6fe5"),
  name: '1 East 66Th Street Kitchen' }
{ _id: ObjectId("61f355dbd3284e8b0a6f8ecb"), name: '1 Oak' }
{ _id: ObjectId("61f355dfd3284e8b0a6f99dc"), name: '1 Or 8' }
Type "it" for more
```

Question 2

List all restaurants with "Italian" cuisine and display the name, zip code and geographic coordinates for each of them. Also, make sure the answer is ordered according to the sorting key (increasing postcode, decreasing name).

Query:

```
db.RestaurantsCollection.find( {"cuisine": "Italian"}, {"name": 1, "address.zipcode": 1,
"address.coord": 1}) .sort( {"address.zipcode": 1, "name": -1})
```

Execution:

```
{ _id: ObjectId("61f355d9d3284e8b0a6f852d"),
  address: { coord: [ -73.9927809, 40.7451239 ], zipcode: '10001' },
  name: 'Tre Dici' }
{ _id: ObjectId("61f355e7d3284e8b0a6fb5fd"),
  address: { coord: [ -73.98952609999999, 40.7507917 ], zipcode: '10001' },
  name: 'Stella 34' }
{ _id: ObjectId("61f355e1d3284e8b0a6f9fab"),
  address: { coord: [ -73.990708, 40.750403 ], zipcode: '10001' },
  name: 'Spinelli\'S Pizza/Gyro Ii' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7233"),
  address: { coord: [ -73.99556, 40.748852 ], zipcode: '10001' },
  name: 'Salumeria Beillese/ Biricchino Rest' }
{ _id: ObjectId("61f355d5d3284e8b0a6f79b5"),
  address: { coord: [ -74.00370749999999, 40.7489719 ], zipcode: '10001' },
  name: 'Pepe Giallo' }
{ _id: ObjectId("61f355e7d3284e8b0a6fb2b8"),
  address: { coord: [ -73.99426609999999, 40.7508707 ], zipcode: '10001' },
  name: 'Fb8020 Pizza Concession' }
```

```
{ _id: ObjectId("61f355e3d3284e8b0a6fa312"),
  address: { coord: [ -73.9930147, 40.7524427 ], zipcode: '10001' },
  name: 'Famous Famiglia' }
{ _id: ObjectId("61f355ded3284e8b0a6f93d5"),
  address: { coord: [ -74.0005178, 40.7471561 ], zipcode: '10001' },
  name: 'Company' }
{ _id: ObjectId("61f355d5d3284e8b0a6f78ac"),
  address: { coord: [ -74.003165, 40.748505 ], zipcode: '10001' },
  name: 'Bottino' }
{ _id: ObjectId("61f355e5d3284e8b0a6fab39"),
  address: { coord: [ -73.988411, 40.7196073 ], zipcode: '10002' },
  name: 'Via Tribunali' }
{ _id: ObjectId("61f355dad3284e8b0a6f8b29"),
  address: { coord: [ -73.987714, 40.72184 ], zipcode: '10002' },
  name: 'Tre' }
{ _id: ObjectId("61f355e0d3284e8b0a6f9add"),
  address: { coord: [ -73.988809, 40.7216582 ], zipcode: '10002' },
  name: 'The Meatball Shop' }
{ _id: ObjectId("61f355e3d3284e8b0a6fa638"),
  address: { coord: [ -73.98768199999999, 40.721902 ], zipcode: '10002' },
  name: 'Taverna Di Bacco' }
{ _id: ObjectId("61f355e4d3284e8b0a6fa6c4"),
  address: { coord: [ -73.9893263, 40.7204716 ], zipcode: '10002' },
  name: 'Sauce' }
{ _id: ObjectId("61f355d9d3284e8b0a6f85a2"),
  address: { coord: [ -73.98857989999999, 40.7223506 ], zipcode: '10002' },
  name: 'Pala' }
{ _id: ObjectId("61f355ebd3284e8b0a6fc2cc"),
  address: { coord: [ -73.9915814, 40.7159183 ], zipcode: '10002' },
  name: 'Nibbles' }
{ _id: ObjectId("61f355ead3284e8b0a6fc01c"),
  address: { coord: [ -73.9923807, 40.718652 ], zipcode: '10002' },
  name: 'Louie And Chan' }
{ _id: ObjectId("61f355e3d3284e8b0a6fa47f"),
  address: { coord: [ -73.9889294, 40.7203785 ], zipcode: '10002' },
  name: 'Goodfella\'S Brick Oven Pizza' }
{ _id: ObjectId("61f355ecd3284e8b0a6fcd7d"),
  address: { coord: [ -73.988413, 40.720246 ], zipcode: '10002' },
  name: 'Galli Restaurant' }
{ _id: ObjectId("61f355e1d3284e8b0a6fa103"),
  address: { coord: [ -73.9853261, 40.7216921 ], zipcode: '10002' },
  name: 'Gaia Italian Cafe' }
Type "it" for more
```

Question 3

List all Italian restaurants with the postcode "10075" for which the telephone number is provided in the database. Display their name, zip code and phone number.

Query:

```
db.RestaurantsCollection.find(
 {$and: [
 {"address.zipcode" : "10075"},
{"telephoneNumber" : {$exists : true}}
 ]},
 {"name" : 1, "address.zipcode" : 1, "telephoneNumber" : 1}
)
```

Execution:

```
< { _id: ObjectId("61f355d2d3284e8b0a6f725e"),
    address: { zipcode: '10075' },
    name: 'Due',
    telephoneNumber: '24680356' }
  { _id: ObjectId("61f355ebd3284e8b0a6fc2de"),
    address: { zipcode: '10075' },
    name: 'Spigolo',
    telephoneNumber: '67895432' }
```

Question 4

Find all restaurants with at least a score of 50.

Query:

```
db.RestaurantsCollection.find(
 {"grades.score" : {$gte: 50}},
 {"name" : 1, "grades.score" : 1}
)
```

Execution:

```
{ _id: ObjectId("61f355d2d3284e8b0a6f6fe4"),
  grades:
   [ { score: 21 },
     { score: 7 },
     { score: 56 },
     { score: 27 },
     { score: 27 } ],
  name: 'May May Kitchen' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7018"),
  grades: [ { score: 2 }, { score: 3 }, { score: 6 }, { score: 54 } ],
  name: 'Polish National Home' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7039"),
  grades: [ { score: 10 }, { score: 10 }, { score: 6 }, { score: 60 } ],
```

```
  name: 'Como Pizza' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7063"),
  grades:
   [ { score: 11 },
     { score: 53 },
     { score: 12 },
     { score: 45 },
     { score: 34 },
     { score: 18 },
     { score: 52 } ],
  name: 'Nanni Restaurant' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7111"),
  grades:
   [ { score: 10 },
     { score: 12 },
     { score: 25 },
     { score: 26 },
     { score: 54 },
     { score: 14 } ],
  name: 'Cafe Espanol' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7136"),
  grades:
   [ { score: 11 },
     { score: 131 },
     { score: 11 },
     { score: 25 },
     { score: 11 },
     { score: 13 } ],
  name: 'Murals On 54/Randolphs\'S' }
{ _id: ObjectId("61f355d2d3284e8b0a6f715d"),
  grades:
   [ { score: 12 },
     { score: 24 },
     { score: 27 },
     { score: 23 },
     { score: 68 } ],
  name: 'Victoria Pizza' }
{ _id: ObjectId("61f355d2d3284e8b0a6f71ae"),
  grades:
   [ { score: 7 },
     { score: 5 },
     { score: 12 },
     { score: 56 },
     { score: 10 } ],
  name: 'Kosher Hut Of Brooklyn' }
{ _id: ObjectId("61f355d2d3284e8b0a6f71d7"),
  grades:
   [ { score: 5 },
     { score: 8 },
     { score: 12 },
     { score: 2 },
     { score: 9 },
     { score: 92 },
     { score: 41 } ],
  name: 'Gandhi' }
{ _id: ObjectId("61f355d2d3284e8b0a6f71d9"),
  grades:
```

```
    [ { score: 17 },
      { score: 12 },
      { score: 62 },
      { score: 10 },
      { score: 7 } ],
    name: 'Village Yokocho' }
  { _id: ObjectId("61f355d2d3284e8b0a6f7227"),
    grades:
    [ { score: 5 },
      { score: 12 },
      { score: 2 },
      { score: 2 },
      { score: 52 } ],
    name: 'De Robertis Pastry Shop' }
  { _id: ObjectId("61f355d2d3284e8b0a6f725b"),
    grades:
    [ { score: 11 },
      { score: 17 },
      { score: 16 },
      { score: 53 },
      { score: 14 },
      { score: 13 } ],
    name: 'El Azteca Mexican Restaurant' }
  { _id: ObjectId("61f355d2d3284e8b0a6f7265"),
    grades:
    [ { score: 9 },
      { score: 12 },
      { score: 58 },
      { score: 13 },
      { score: 71 } ],
    name: 'Live Bait Bar & Restaurant' }
  { _id: ObjectId("61f355d2d3284e8b0a6f726a"),
    grades:
    [ { score: 50 },
      { score: 10 },
      { score: 12 },
      { score: 3 },
      { score: 6 } ],
    name: 'Cafe Lalo' }
  { _id: ObjectId("61f355d2d3284e8b0a6f728b"),
    grades:
    [ { score: 5 },
      { score: 12 },
      { score: 52 },
      { score: 12 },
      { score: 9 } ],
    name: 'Lee\'S Villa Chinese Restaurant' }
  { _id: ObjectId("61f355d2d3284e8b0a6f72ed"),
    grades:
    [ { score: 12 },
      { score: 4 },
      { score: 11 },
      { score: 7 },
      { score: 11 },
      { score: 66 } ],
    name: 'Richer\'S Bakery' }
  { _id: ObjectId("61f355d2d3284e8b0a6f733a"),
```

```
  grades:
   [ { score: 31 },
     { score: 98 },
     { score: 32 },
     { score: 21 },
     { score: 11 } ],
  name: 'Bella Napoli' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7345"),
  grades:
   [ { score: 10 },
     { score: 6 },
     { score: 25 },
     { score: 12 },
     { score: 12 },
     { score: 14 },
     { score: 26 },
     { score: 76 } ],
  name: 'El Molino Rojo Restaurant' }
{ _id: ObjectId("61f355d2d3284e8b0a6f73ad"),
  grades:
   [ { score: 52 },
     { score: 12 },
     { score: 22 },
     { score: 23 },
     { score: 3 },
     { score: 12 } ],
  name: 'Tequilla Sunrise' }
{ _id: ObjectId("61f355d2d3284e8b0a6f73b5"),
  grades:
   [ { score: 9 },
     { score: 12 },
     { score: 50 },
     { score: 11 },
     { score: 15 },
     { score: 12 } ],
  name: 'Golden Pizza' }
Type "it" for more
```

Question 5

Find all restaurants that are either Italian or have the postcode "10075".

```
db.RestaurantsCollection.find(
         { $or: [
                 {"cuisine": "Italian"},
                 {"address.zipcode": "10075"}
                         ]
        }
)
```

Execution:

```
{ _id: ObjectId("61f355d2d3284e8b0a6f7010"),
  address:
   { building: '10004',
     coord: [ -74.03400479999999, 40.6127077 ],
     street: '4 Avenue',
     zipcode: '11209' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-02-25T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-06-27T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2012-12-03T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2011-11-09T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: 'Philadelhia Grille Express',
  restaurant_id: '40364305' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7016"),
  address:
   { building: '1028',
     coord: [ -73.966032, 40.762832 ],
     street: '3 Avenue',
     zipcode: '10065' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-09-16T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2014-02-24T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2013-05-03T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2012-08-20T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2012-02-13T00:00:00.000Z, grade: 'A', score: 9 } ],
  name: 'Isle Of Capri Resturant',
  restaurant_id: '40364373' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7024"),
  address:
   { building: '251',
     coord: [ -73.9775552, 40.7432016 ],
     street: 'East   31 Street',
     zipcode: '10016' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-04-22T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-06-19T00:00:00.000Z, grade: 'C', score: 32 },
     { date: 2012-05-22T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: 'Marchis Restaurant',
  restaurant_id: '40364668' }
{ _id: ObjectId("61f355d2d3284e8b0a6f702f"),
  address:
   { building: '67',
     coord: [ -74.0707363, 40.59321569999999 ],
     street: 'Olympia Boulevard',
     zipcode: '10305' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-04-24T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-04-04T00:00:00.000Z, grade: 'A', score: 2 },
     { date: 2012-02-02T00:00:00.000Z, grade: 'A', score: 5 },
     { date: 2011-07-23T00:00:00.000Z, grade: 'A', score: 11 } ],
```

```
  name: 'Crystal Room',
  restaurant_id: '40365013' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7033"),
  address:
   { building: '93',
     coord: [ -73.99950489999999, 40.7169224 ],
     street: 'Baxter Street',
     zipcode: '10013' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-12-15T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2013-12-06T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2012-10-23T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-06-04T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2012-01-12T00:00:00.000Z, grade: 'A', score: 13 } ],
  name: 'Forlinis Restaurant',
  restaurant_id: '40365098' }
{ _id: ObjectId("61f355d2d3284e8b0a6f703b"),
  address:
   { building: '146',
     coord: [ -73.9973041, 40.7188698 ],
     street: 'Mulberry Street',
     zipcode: '10013' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-05-02T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2013-03-14T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2012-09-26T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2012-02-15T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2011-09-15T00:00:00.000Z, grade: 'A', score: 11 } ],
  name: 'Angelo Of Mulberry St.',
  restaurant_id: '40365293' }
{ _id: ObjectId("61f355d2d3284e8b0a6f703d"),
  address:
   { building: '7201',
     coord: [ -74.0166091, 40.6284767 ],
     street: '8 Avenue',
     zipcode: '11228' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-12-04T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2014-02-19T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2013-07-09T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2012-06-06T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2011-12-19T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: 'New Corner',
  restaurant_id: '40365355' }
{ _id: ObjectId("61f355d2d3284e8b0a6f703f"),
  address:
   { building: '106',
     coord: [ -74.0003315, 40.7274874 ],
     street: 'West Houston Street',
     zipcode: '10012' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-03-31T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-10-08T00:00:00.000Z, grade: 'A', score: 12 },
```

```
      { date: 2013-03-29T00:00:00.000Z, grade: 'A', score: 12 },
      { date: 2012-09-05T00:00:00.000Z, grade: 'A', score: 7 },
      { date: 2012-03-05T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: 'Arturo\'S',
  restaurant_id: '40365387' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7049"),
  address:
   { building: '1024',
     coord: [ -73.96392089999999, 40.8033908 ],
     street: 'Amsterdam Avenue',
     zipcode: '10025' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-06-12T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2014-01-09T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-06-25T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-06-01T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2011-12-15T00:00:00.000Z, grade: 'A', score: 10 } ],
  name: 'V & T Restaurant',
  restaurant_id: '40365577' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7050"),
  address:
   { building: '351',
     coord: [ -73.96117869999999, 40.7619226 ],
     street: 'East   62 Street',
     zipcode: '10065' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-11-13T00:00:00.000Z, grade: 'B', score: 24 },
     { date: 2014-02-28T00:00:00.000Z, grade: 'B', score: 19 },
     { date: 2013-06-10T00:00:00.000Z, grade: 'B', score: 27 },
     { date: 2012-05-09T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: 'Il Vagabondo Restaurant',
  restaurant_id: '40365709' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7051"),
  address:
   { building: '319321',
     coord: [ -73.988948, 40.760337 ],
     street: '323 W. 46Th St.',
     zipcode: '10036' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-05-13T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-11-12T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-04-27T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2011-12-07T00:00:00.000Z, grade: 'A', score: 7 } ],
  name: 'Barbetta Restaurant',
  restaurant_id: '40365726' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7052"),
  address:
   { building: '2911',
     coord: [ -73.982241, 40.576366 ],
     street: 'West   15 Street',
     zipcode: '11224' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-12-18T00:00:00.000Z, grade: 'A', score: 13 },
```

```
      { date: 2014-05-15T00:00:00.000Z, grade: 'A', score: 12 },
      { date: 2013-06-12T00:00:00.000Z, grade: 'A', score: 9 },
      { date: 2012-02-06T00:00:00.000Z, grade: 'A', score: 9 } ],
  name: 'Gargiulo\'S Restaurant',
  restaurant_id: '40365784' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7053"),
  address:
   { building: '236',
     coord: [ -73.9827418, 40.7655827 ],
     street: 'West   56 Street',
     zipcode: '10019' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-05-05T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2013-08-12T00:00:00.000Z, grade: 'A', score: 6 },
     { date: 2012-08-13T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-02-28T00:00:00.000Z, grade: 'A', score: 13 } ],
  name: 'Patsy\'S Italian Restaurant',
  restaurant_id: '40365789' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7054"),
  address:
   { building: '10701',
     coord: [ -73.856132, 40.743841 ],
     street: 'Corona Avenue',
     zipcode: '11368' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-07-17T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2014-02-25T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2013-03-27T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2012-02-07T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2011-12-28T00:00:00.000Z, grade: 'A', score: 13 } ],
  name: 'Parkside Restaurant',
  restaurant_id: '40365841' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7062"),
  address:
   { building: '2929',
     coord: [ -73.942849, 40.6076256 ],
     street: 'Avenue R',
     zipcode: '11229' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-03-13T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2013-10-02T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-01-22T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-06-12T00:00:00.000Z, grade: 'A', score: 8 },
     { date: 2011-12-01T00:00:00.000Z, grade: 'B', score: 20 },
     { date: 2011-05-25T00:00:00.000Z, grade: 'A', score: 9 } ],
  name: 'Michael\'S Restaurant',
  restaurant_id: '40366154' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7063"),
  address:
   { building: '146',
     coord: [ -73.9736776, 40.7535755 ],
     street: 'East   46 Street',
     zipcode: '10017' },
  cuisine: 'Italian',
```

```
  grades:
   [ { date: 2014-03-11T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2013-07-31T00:00:00.000Z, grade: 'C', score: 53 },
     { date: 2012-12-19T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-06-04T00:00:00.000Z, grade: 'C', score: 45 },
     { date: 2012-01-18T00:00:00.000Z, grade: 'C', score: 34 },
     { date: 2011-09-28T00:00:00.000Z, grade: 'B', score: 18 },
     { date: 2011-05-24T00:00:00.000Z, grade: 'C', score: 52 } ],
  name: 'Nanni Restaurant',
  restaurant_id: '40366157' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7067"),
  address:
   { building: '13558',
     coord: [ -73.8216767, 40.6689548 ],
     street: 'Lefferts Boulevard',
     zipcode: '11420' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-06-20T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2013-06-07T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2012-06-28T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2012-01-13T00:00:00.000Z, grade: 'C', score: 28 } ],
  name: 'Don Peppe',
  restaurant_id: '40366230' }
{ _id: ObjectId("61f355d2d3284e8b0a6f707d"),
  address:
   { building: '220-20',
     coord: [ -73.74292179999999, 40.7305714 ],
     street: 'Hillside Avenue',
     zipcode: '11427' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-10-22T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2014-03-07T00:00:00.000Z, grade: 'B', score: 15 },
     { date: 2013-01-09T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2012-07-12T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2012-01-24T00:00:00.000Z, grade: 'A', score: 11 } ],
  name: 'Cara Mia',
  restaurant_id: '40366812' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7098"),
  address:
   { building: '302',
     coord: [ -73.985535, 40.730605 ],
     street: 'East 12 Street',
     zipcode: '10003' },
  cuisine: 'Italian',
  grades:
   [ { date: 2014-07-22T00:00:00.000Z, grade: 'A', score: 3 },
     { date: 2013-07-23T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-01-23T00:00:00.000Z, grade: 'A', score: 6 },
     { date: 2011-12-01T00:00:00.000Z, grade: 'A', score: 7 } ],
  name: 'John\'S Restaurant',
  restaurant_id: '40367407' }
{ _id: ObjectId("61f355d2d3284e8b0a6f709d"),
  address:
   { building: '42-01',
     coord: [ -73.911784, 40.764766 ],
```

```
      street: '28 Avenue',
      zipcode: '11103' },
   cuisine: 'Italian',
   grades:
    [ { date: 2014-12-18T00:00:00.000Z, grade: 'Z', score: 26 },
      { date: 2014-04-03T00:00:00.000Z, grade: 'A', score: 11 },
      { date: 2013-09-16T00:00:00.000Z, grade: 'C', score: 28 },
      { date: 2013-02-05T00:00:00.000Z, grade: 'A', score: 8 },
      { date: 2012-08-22T00:00:00.000Z, grade: 'A', score: 13 },
      { date: 2012-01-30T00:00:00.000Z, grade: 'A', score: 12 } ],
   name: 'Piccola Venezia',
   restaurant_id: '40367540' }
Type "it" for more
```

Question 6

List all restaurants with a zip code of "10075" or "10098" with either Italian or American cuisine with a score of at least 50.

Query:

```
db.RestaurantsCollection.find(
 {$and: [
 {$or: [  {"address.zipcode" : "10075"}, {"address.zipcode" : "10098"} ]}, {$or: [
{"cuisine" : "American"}, {"cuisine" : "Italian"} ]},
 {"grades.score" : {$gte : 50}}
 ]})
```

Execution:


```
{ _id: ObjectId("61f355d9d3284e8b0a6f851e"),
  address:
   { building: '1462',
     coord: [ -73.953769, 40.77037 ],
     street: '1 Avenue',
     zipcode: '10075' },
  cuisine: 'American',
  grades:
   [ { date: 2015-01-16T00:00:00.000Z, grade: 'Z', score: 19 },
     { date: 2014-09-02T00:00:00.000Z, grade: 'C', score: 57 },
     { date: 2014-03-20T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-09-12T00:00:00.000Z, grade: 'B', score: 21 },
     { date: 2013-04-05T00:00:00.000Z, grade: 'B', score: 15 },
     { date: 2012-10-22T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2012-05-21T00:00:00.000Z, grade: 'B', score: 25 } ],
  name: 'Three Star Diner',
  restaurant_id: '41097286' }
```

Question 7

List all restaurants with at least one score concerning customer service *(C)*, *price (P)* or *quality (Q)*. Simply display the names, cuisine and zip code.

Query:

```
db.RestaurantsCollection.find(
{ $or : [
{"grades.grade" : "C"},
{"grades.grade" : "P"},
{"grades.grade" : "Q"}
]},
{
"name" : 1,
"cuisine" : 1,
"address.zipcode" : 1
}
)
```

Execution:

```
{ _id: ObjectId("61f355d2d3284e8b0a6f6fe4"),
  address: { zipcode: '11208' },
  cuisine: 'Chinese',
  name: 'May May Kitchen' }
{ _id: ObjectId("61f355d2d3284e8b0a6f6fe7"),
  address: { zipcode: '11218' },
  cuisine: 'Ice Cream, Gelato, Yogurt, Ices',
  name: 'Carvel Ice Cream' }
{ _id: ObjectId("61f355d2d3284e8b0a6f6ff3"),
  address: { zipcode: '11368' },
  cuisine: 'Chinese',
  name: 'Ho Mei Restaurant' }
{ _id: ObjectId("61f355d2d3284e8b0a6f700b"),
  address: { zipcode: '11371' },
  cuisine: 'American',
  name: 'Terminal Cafe/Yankee Clipper' }
{ _id: ObjectId("61f355d2d3284e8b0a6f700e"),
  address: { zipcode: '10308' },
  cuisine: 'Delicatessen',
  name: 'B & M Hot Bagel & Grocery' }
{ _id: ObjectId("61f355d2d3284e8b0a6f700f"),
  address: { zipcode: '10038' },
  cuisine: 'Chicken',
  name: 'Texas Rotisserie' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7017"),
  address: { zipcode: '10003' },
  cuisine: 'American',
  name: 'Old Town Bar & Restaurant' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7018"),
  address: { zipcode: '11222' },
  cuisine: 'Polish',
  name: 'Polish National Home' }
{ _id: ObjectId("61f355d2d3284e8b0a6f701c"),
  address: { zipcode: '10019' },
  cuisine: 'American',
  name: 'Nyac Main Dining Room' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7021"),
  address: { zipcode: '10019' },
  cuisine: 'French',
  name: 'Tout Va Bien' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7024"),
  address: { zipcode: '10016' },
  cuisine: 'Italian',
  name: 'Marchis Restaurant' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7031"),
  address: { zipcode: '11223' },
  cuisine: 'American',
  name: 'Shell Lanes' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7039"),
  address: { zipcode: '10032' },
  cuisine: 'Pizza',
  name: 'Como Pizza' }
{ _id: ObjectId("61f355d2d3284e8b0a6f703e"),
  address: { zipcode: '10036' },
  cuisine: 'American',
  name: 'The Princeton Club' }
```

```
{ _id: ObjectId("61f355d2d3284e8b0a6f7041"),
  address: { zipcode: '10033' },
  cuisine: 'American',
  name: 'Reynold\'S Bar' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7045"),
  address: { zipcode: '10451' },
  cuisine: 'American',
  name: 'Yankee Tavern' }
{ _id: ObjectId("61f355d2d3284e8b0a6f704a"),
  address: { zipcode: '11366' },
  cuisine: 'Chinese',
  name: 'King Yum Restaurant' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7059"),
  address: { zipcode: '10013' },
  cuisine: 'Café/Coffee/Tea',
  name: 'Mee Sum Coffee Shop' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7063"),
  address: { zipcode: '10017' },
  cuisine: 'Italian',
  name: 'Nanni Restaurant' }
{ _id: ObjectId("61f355d2d3284e8b0a6f7065"),
  address: { zipcode: '11368' },
  cuisine: 'Latin (Cuban, Dominican, Puerto Rican, South & Central American)',
  name: 'Emilio Iii Bar' }
Type "it" for more
```

Question 8

Change the type of cuisine in the restaurant "Juni" to "American (new)". In addition, record the date and time of the system in a "lastModified" field at the time the change is made. If there are several restaurants with the same name, only the first one must be modified.

Query:

```
db.RestaurantsCollection.update(
{ "name": "Juni" },
  {
  $set: { cousine: "American (New)" },
  $currentDate: { "lastModified": true }
  }
)
```

Execution:

```
< 'DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.'
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.RestaurantsCollection.find({name: "Juni"})
< { _id: ObjectId("61f355dad3284e8b0a6f8761"),
    address:
     { building: '12',
       coord: [ -73.9852329, 40.745971 ],
       street: 'East 31 Street',
       zipcode: '10016' },
    cuisine: 'American',
    grades:
     [ { date: 2014-09-19T00:00:00.000Z, grade: 'A', score: 12 },
       { date: 2013-08-05T00:00:00.000Z, grade: 'A', score: 5 },
       { date: 2012-06-07T00:00:00.000Z, grade: 'A', score: 0 } ],
    name: 'Juni',
    restaurant_id: '41156888',
    cousine: 'American (New)',
    lastModified: 2022-02-17T01:25:20.163Z }
```

Question 9

Change the address of the restaurant whose id is "41156888" to "East 31st Street".

Query:

```
db.RestaurantsCollection.update({"restaurant_id": "41156888"}, {$set: {address: "East
31st Street"}});
```

Execution:

```
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

Question 10

Change the type of cuisine of all restaurants with a postcode "100016" and the type of cuisine "Other" into "Cuisine to be determined". In addition, record the date and time of the system in a "lastModified" field at the time the change is made.

Query:

```
db.RestaurantsCollection.update(
 { "address.zipcode": "10016", "cuisine": "Other" },
 {
 $set: { "cuisine": "Cuisine to be determined" },
 $currentDate: { "lastModified": true }
 },
 {multi: true}
)
```

Execution:

```
< { acknowledged: true,
    insertedId: null,
    matchedCount: 20,
    modifiedCount: 20,
    upsertedCount: 0 }
```

Question 11

Replace all information about the restaurant with an ID "41154403" with the following information:

```
"name": "Vella 2",
"address": {
        "city": "1480",
        "street": "2 Avenue",
        "zipcode": "10075"
}
```

Query:

```
db.RestaurantsCollection.updateOne({restaurant_id: "41154403"},
        {$set :
                {"name" : "Vella 2",
                "address" : {"city" : "1480" , "street" : "2 Avenue", "zipcode" :
"10075"}
                }
        });
```

Execution:

```
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

Question 12

List the types of cuisine represented in the database. For each one, display the number of associated restaurants. Order the result by decreasing number of restaurants.

Query:

```
db.RestaurantsCollection.aggregate([
        { $group:
                { "_id": "$cuisine", "restaurants": {$sum: 1}
        }
        },
        {$sort:
                { "restaurants": -1 }
        }
])
```

Execution:

```
{ _id: 'American', restaurants: 6181 }
{ _id: 'Chinese', restaurants: 2417 }
{ _id: 'Café/Coffee/Tea', restaurants: 1214 }
{ _id: 'Pizza', restaurants: 1162 }
{ _id: 'Italian', restaurants: 1070 }
{ _id: 'Other', restaurants: 990 }
{ _id: 'Latin (Cuban, Dominican, Puerto Rican, South & Central American)',
  restaurants: 854 }
{ _id: 'Japanese', restaurants: 759 }
{ _id: 'Mexican', restaurants: 756 }
{ _id: 'Bakery', restaurants: 691 }
{ _id: 'Caribbean', restaurants: 656 }
{ _id: 'Spanish', restaurants: 637 }
{ _id: 'Donuts', restaurants: 477 }
{ _id: 'Pizza/Italian', restaurants: 468 }
{ _id: 'Sandwiches', restaurants: 459 }
{ _id: 'Hamburgers', restaurants: 433 }
{ _id: 'Chicken', restaurants: 410 }
{ _id: 'Ice Cream, Gelato, Yogurt, Ices', restaurants: 348 }
{ _id: 'French', restaurants: 344 }
```

```
{ _id: 'Delicatessen', restaurants: 321 }
Type "it" for more
```

Question 13

Show, for each zip code, the number of Italian restaurants with this zip code. Order the result by decreasing number of restaurants.

Query:

```
db.RestaurantsCollection.aggregate( [
    {
        $match: {"cuisine" : "Italian"}
    },
    {
        $group: {
            _id: "$address.zipcode",
            "cuisines": { $sum : 1}
        }
    },
    {
        $sort: {
            "cuisines":-1
        }
    }
] )
```

Execution:

```
{ _id: '10013', cuisines: 51 }
{ _id: '10012', cuisines: 45 }
{ _id: '10014', cuisines: 43 }
{ _id: '10019', cuisines: 42 }
{ _id: '10003', cuisines: 40 }
{ _id: '10011', cuisines: 37 }
{ _id: '10036', cuisines: 35 }
{ _id: '10022', cuisines: 32 }
{ _id: '11215', cuisines: 22 }
{ _id: '10028', cuisines: 22 }
{ _id: '10016', cuisines: 21 }
{ _id: '10021', cuisines: 20 }
{ _id: '10017', cuisines: 20 }
{ _id: '10024', cuisines: 19 }
{ _id: '11201', cuisines: 19 }
{ _id: '10023', cuisines: 18 }
{ _id: '10458', cuisines: 17 }
{ _id: '10009', cuisines: 17 }
{ _id: '10065', cuisines: 17 }
```

```
{ _id: '10010', cuisines: 17 }
Type "it" for more
```

Question 14

Consider Italian restaurants whose identifier (restaurant_id) is higher or equal to "41205309" and has an "averagePrice" attribute. Calculate the average of these average prices. Then repeat the same operation by calculating the average by zipcode.

Query:

```
db.RestaurantsCollection.aggregate([
{
  $match: {
      $and: [
          {"cuisine": 'Italian'},
          {"restaurant_id": { $gte: '41205309' }},
          {"averagePrice": {$exists: true}}
      ]
  },
},
  {
  $group : {
          "_id" : null,
          "averagePrices": { $avg: "$averagePrice" },
          "averageZipcodes": { $avg: { '$toInt': '$address.zipcode' }}
          }
  },
])
```

Execution:

```
{ _id: null, averagePrices: 25, averageZipcodes: 10023.5 }
```

Question 15

Create a new collection called "comments" in the same database.

Query:

```
db.createCollection("comments")
```

Execution:

```
{ ok: 1 }
```

Question 16

Insert three documents into the previously created collection, using the following pattern:

```
{
        "_id": "----",
        "restaurant_id": "----",
        "client_id": "----",
        "comment": "----",
        "date": ISODate("----"),
        "type": "----"
}
```

Some useful details:

- Restaurant identifiers must match existing restaurants in the collection `restaurants`
- You need to provide feedback from different customers, and for different `restaurants`
- The type `attribute` can only take the "positive" or "negative" values.

Query:

```
db.comments.insertMany([
    {"restaurant_id": "30112340", "client_id": "139166537", "comment": "It's ok.",
"date": new Date("2016-10-12"), "type": "positive"},
    {"restaurant_id": "30191841", "client_id": "561478210", "comment": "Nice music, nice
food, nice place indeed.", "date": new Date("2017-05-13"), "type": "positive"},
    {"restaurant_id": "40356018", "client_id": "501862302", "comment": "Food took ages to
arrive, came cold. Awful service.", "date": new Date("2020-07-13"), "type": "negative"}
])
```

Execution:

```
{ acknowledged: true,
  insertedIds:
   { '0': ObjectId("620dace1918e8e2dc8ab5e7e"),
     '1': ObjectId("620dace1918e8e2dc8ab5e7f"),
     '2': ObjectId("620dace1918e8e2dc8ab5e80") } }
```

Question 17

List all the comments in your database. Each comment must also contain all the information about the restaurant to which it relates.

Query:

```
db.RestaurantsCollection.aggregate([{
$lookup:{
from: "comments",
localField: "restaurant_id",
foreignField: "restaurant_id",
as: "restaurant_comments"
}
},
{
$match: {"restaurant_comments": {$ne: []}}
},
{
$project: { restaurant_comments:{comment: 1}, name: 1, cuisine: 1, restaurant_id: 1,
address: 1}
},
])
```

Execution:

```
{ _id: ObjectId("61f355d2d3284e8b0a6f6fd9"),
  address:
   { building: '469',
     coord: [ -73.961704, 40.662942 ],
     street: 'Flatbush Avenue',
     zipcode: '11225' },
  cuisine: 'Hamburgers',
  name: 'Wendy\'S',
  restaurant_id: '30112340',
  restaurant_comments: [ { comment: 'It\'s ok.' } ] }
{ _id: ObjectId("61f355d2d3284e8b0a6f6fda"),
  address:
   { building: '351',
     coord: [ -73.98513559999999, 40.7676919 ],
     street: 'West   57 Street',
     zipcode: '10019' },
  cuisine: 'Irish',
  name: 'Dj Reynolds Pub And Restaurant',
  restaurant_id: '30191841',
  restaurant_comments: [ { comment: 'Nice music, nice food, nice place indeed.' } ] }
{ _id: ObjectId("61f355d2d3284e8b0a6f6fdb"),
  address:
   { building: '2780',
     coord: [ -73.98241999999999, 40.579505 ],
     street: 'Stillwell Avenue',
     zipcode: '11224' },
```

```
  cuisine: 'American',
  name: 'Riviera Caterer',
  restaurant_id: '40356018',
  restaurant_comments: [ { comment: 'Food took ages to arrive, came cold. Awful service.'
} ] }
```

Question 18

Insert seven other documents into the comments collection, following the pattern described above, and following these rules:

- Restaurant identifiers must match existing restaurants in the restaurants collection
- At least one of the restaurants must have several comments
- At least one of the customers must have commented several times
- At least one of the customers must have commented several times on the same restaurant
- The type attribute can only take the "positive" or "negative" values

Query:

```
db.comments.insertMany([
    {"restaurant_id": "30112340", "client_id": "139166537", "comment": "So so, not that
good tbh", "date": new Date("2016-10-12"), "type": "negative"},
    {"restaurant_id": "30112340", "client_id": "139166537", "comment": "Have I reviewed
this place already? I don't know, but food was alright", "date": new Date("2017-05-13"),
"type": "positive"},
        {"restaurant_id": "30112340", "client_id": "240277648", "comment": "Do you have
any discounts for veterans?", "date": new Date("2016-10-12"), "type": "positive"},
    {"restaurant_id": "30191841", "client_id": "561478210", "comment": "Excellent food,
the service is always very good and cleanliness too. An excellent choice to have a nice
meal or a coffee and dessert.", "date": new Date("2017-05-13"), "type": "positive"},
        {"restaurant_id": "40356151", "client_id": "501862302", "comment": "It's ok.",
"date": new Date("2016-10-12"), "type": "positive"},
    {"restaurant_id": "30191841", "client_id": "561478210", "comment": "Excellent food
and service, highly recommended to hang out. Cheap breakfasts and good portions.
Innovative!", "date": new Date("2017-05-13"), "type": "positive"},
    {"restaurant_id": "40356068", "client_id": "501862302", "comment": "Good service ,
good food., good prices, clean.", "date": new Date("2020-07-13"), "type": "positive"}
])
```

Execution:

```
< { acknowledged: true,
    insertedIds:
     { '0': ObjectId("620db0e9918e8e2dc8ab5e84"),
       '1': ObjectId("620db0e9918e8e2dc8ab5e85"),
       '2': ObjectId("620db0e9918e8e2dc8ab5e86"),
       '3': ObjectId("620db0e9918e8e2dc8ab5e87"),
       '4': ObjectId("620db0e9918e8e2dc8ab5e88"),
       '5': ObjectId("620db0e9918e8e2dc8ab5e89"),
       '6': ObjectId("620db0e9918e8e2dc8ab5e8a") } }
```

Question 19

Find the list of restaurants with reviews and display only the name and comments list for each restaurant. Several strategies are possible.

Query:

```
db.RestaurantsCollection.aggregate([
   {
     $lookup:
       {
         from: "comments",
         localField: "restaurant_id",
         foreignField: "restaurant_id",
         as: "restaurants_n_comments"
       }
  },
  { $unwind: "$restaurants_n_comments" },
    {
    $match: {"restaurants_n_comments.comment": { $exists: true, $ne: ""}}
  },
  {
    $project: {
        "name": 1,
        "comments": "$restaurants_n_comments.comment"
    }
  }
])
```

Execution:

```
{ _id: ObjectId("61f355d2d3284e8b0a6f6fd9"),
  name: 'Wendy\'S',
  comments: 'It\'s ok.' }
{ _id: ObjectId("61f355d2d3284e8b0a6f6fd9"),
```

```
    name: 'Wendy\'S',
    comments: 'So so, not that good tbh' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fd9"),
    name: 'Wendy\'S',
    comments: 'Have I reviewed this place already? I don\'t know, but food was alright' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fd9"),
    name: 'Wendy\'S',
    comments: 'Do you have any discounts for veterans?' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fda"),
    name: 'Dj Reynolds Pub And Restaurant',
    comments: 'Nice music, nice food, nice place indeed.' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fda"),
    name: 'Dj Reynolds Pub And Restaurant',
    comments: 'Excellent food, the service is always very good and cleanliness too. An
  excellent choice to have a nice meal or a coffee and dessert.' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fda"),
    name: 'Dj Reynolds Pub And Restaurant',
    comments: 'Excellent food and service, highly recommended to hang out. Cheap breakfasts
  and good portions. Innovative!' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fdb"),
    name: 'Riviera Caterer',
    comments: 'Food took ages to arrive, came cold. Awful service.' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fdc"),
    name: 'Tov Kosher Kitchen',
    comments: 'Good service , good food., good prices, clean.' }
  { _id: ObjectId("61f355d2d3284e8b0a6f6fdd"),
    name: 'Brunos On The Boulevard',
    comments: 'It\'s ok.' }
```

Question 20

Create an increasing index on the cuisine attribute of the restaurant collection.

Query:

```
db.RestaurantsCollection.createIndex({cuisine: 1})
```

Execution:

```
>_MONGOSH
> db.RestaurantsCollection.createIndex({cuisine: 1})
< 'cuisine_1'
Atlas atlas-kijdn9-shard-0 [primary] DataTestBigData >
```

Question 21

Create another index for the restaurants collection, consisting of the cuisine attribute (increasing) and the zipcode attribute (decreasing).

Query:

```
db.RestaurantsCollection.createIndex({cuisine: 1, "address.zipcode": -1})
```

Execution:

```
>_MONGOSH
> db.RestaurantsCollection.createIndex({cuisine: 1, "address.zipcode": -1})
< 'cuisine_1_address.zipcode_-1'
Atlas atlas-kijdn9-shard-0 [primary] DataTestBigData >
```

Question 22

List all indexes created on the restaurants collection.

Query:

```
db.RestaurantsCollection.getIndexes()
```

Execution:

```
>_MONGOSH
> db.RestaurantsCollection.getIndexes()
< [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { cuisine: 1 }, name: 'cuisine_1' },
    {
      v: 2,
      key: { cuisine: 1, 'address.zipcode': -1 },
      name: 'cuisine_1_address.zipcode_-1'
    }
  ]
Atlas atlas-kijdn9-shard-0 [primary] DataTestBigData >
```

Question 23

Use the explain method to display the execution plan for the query that returns all Italian restaurants. What information is provided by the system?

Query:

```
# Returns all Italian restaurants
db.RestaurantsCollection.find({cuisine: "Italian"})
```

```
db.RestaurantsCollection.explain().find({cuisine: "Italian"})
```

Execution:

```
>_MONGOSH
> db.RestaurantsCollection.explain().find({cuisine: "Italian"})
< { queryPlanner:
    { plannerVersion: 1,
      namespace: 'DataTestBigData.RestaurantsCollection',
      indexFilterSet: false,
      parsedQuery: { cuisine: { '$eq': 'Italian' } },
      queryHash: '1A247377',
      planCacheKey: '5525FB31',
      winningPlan:
       { stage: 'FETCH',
         inputStage:
          { stage: 'IXSCAN',
            keyPattern: { cuisine: 1 },
            indexName: 'cuisine_1',
            isMultiKey: false,
            multiKeyPaths: { cuisine: [] },
            isUnique: false,
            isSparse: false,
            isPartial: false,
            indexVersion: 2,
            direction: 'forward',
            indexBounds: { cuisine: [ '["Italian", "Italian"]' ] } } },
      rejectedPlans:
```

```
        [ { stage: 'FETCH',
            inputStage:
             { stage: 'IXSCAN',
               keyPattern: { cuisine: 1, 'address.zipcode': -1 },
               indexName: 'cuisine_1_address.zipcode_-1',
               isMultiKey: false,
               multiKeyPaths: { cuisine: [], 'address.zipcode': [] },
               isUnique: false,
               isSparse: false,
               isPartial: false,
               indexVersion: 2,
               direction: 'forward',
               indexBounds:
                { cuisine: [ '["Italian", "Italian"]' ],
                  'address.zipcode': [ '[MaxKey, MinKey]' ] } } } ] },
    serverInfo:
     { host: 'cluster0-shard-00-02.6wsnn.mongodb.net',
       port: 27017,
       version: '4.4.12',
       gitVersion: '51475a8c4d9856eb1461137e7539a0a763cc85dc' },
    ok: 1,
    '$clusterTime':
     { clusterTime: Timestamp({ t: 1644552177, i: 8 }),
       signature:
        { hash: Binary(Buffer.from("58a85fc628a43abe9b87422ce08f7172ccbf549e", "hex"), 0),
          keyId: 6997651294655611000 } },
    operationTime: Timestamp({ t: 1644552177, i: 8 }) }
Atlas atlas-kijdn9-shard-0 [primary] DataTestBigData>
```

Answer:
The system provides the data of the queryPlanner, which shows some general information about the query, like the way it was parsed and some data of the planner, but more importantly it shows which were the plans that won to execute the query and what other plans were considered to fulfill it. Thanks to our created index, we can see that it was used in the winning plan to find our data.
Apart from information of the query planner, it shows the properties of the mongoDB server, the status of the query, and the time.

Question 24

Same question but adding the parameter "executionStats" in the explain method.

Query:

```
db.RestaurantsCollection.explain("executionStats").find({cuisine: "Italian"})
```

Execution

```
{ queryPlanner:
   { plannerVersion: 1,
     namespace: 'DataTestBigData.RestaurantsCollection',
     indexFilterSet: false,
     parsedQuery: { cuisine: { '$eq': 'Italian' } },
     winningPlan:
      { stage: 'FETCH',
        inputStage:
         { stage: 'IXSCAN',
           keyPattern: { cuisine: 1 },
           indexName: 'cuisine_1',
           isMultiKey: false,
           multiKeyPaths: { cuisine: [] },
           isUnique: false,
           isSparse: false,
           isPartial: false,
           indexVersion: 2,
           direction: 'forward',
           indexBounds: { cuisine: [ '["Italian", "Italian"]' ] } } },
     rejectedPlans:
      [ { stage: 'FETCH',
          inputStage:
           { stage: 'IXSCAN',
             keyPattern: { cuisine: 1, 'address.zipcode': -1 },
             indexName: 'cuisine_1_address.zipcode_-1',
             isMultiKey: false,
             multiKeyPaths: { cuisine: [], 'address.zipcode': [] },
             isUnique: false,
             isSparse: false,
             isPartial: false,
             indexVersion: 2,
             direction: 'forward',
             indexBounds:
              { cuisine: [ '["Italian", "Italian"]' ],
                'address.zipcode': [ '[MaxKey, MinKey]' ] } } } ] },
  executionStats:
   { executionSuccess: true,
     nReturned: 1070,
     executionTimeMillis: 11,
     totalKeysExamined: 1070,
     totalDocsExamined: 1070,
     executionStages:
      { stage: 'FETCH',
        nReturned: 1070,
        executionTimeMillisEstimate: 9,
        works: 1071,
        advanced: 1070,
        needTime: 0,
```

```
        needYield: 0,
        saveState: 1,
        restoreState: 1,
        isEOF: 1,
        docsExamined: 1070,
        alreadyHasObj: 0,
        inputStage:
         { stage: 'IXSCAN',
           nReturned: 1070,
           executionTimeMillisEstimate: 0,
           works: 1071,
           advanced: 1070,
           needTime: 0,
           needYield: 0,
           saveState: 1,
           restoreState: 1,
           isEOF: 1,
           keyPattern: { cuisine: 1 },
           indexName: 'cuisine_1',
           isMultiKey: false,
           multiKeyPaths: { cuisine: [] },
           isUnique: false,
           isSparse: false,
           isPartial: false,
           indexVersion: 2,
           direction: 'forward',
           indexBounds: { cuisine: [ '["Italian", "Italian"]' ] },
           keysExamined: 1070,
           seeks: 1,
           dupsTested: 0,
           dupsDropped: 0 } } },
  serverInfo:
   { host: 'cluster0-shard-00-02.6wsnn.mongodb.net',
     port: 27017,
     version: '4.4.12',
     gitVersion: '51475a8c4d9856eb1461137e7539a0a763cc85dc' },
  ok: 1,
  '$clusterTime':
   { clusterTime: Timestamp({ t: 1644552375, i: 6 }),
     signature:
      { hash: Binary(Buffer.from("c7ce42df52497d0b5c2cc3d9630c0ec641f020d0", "hex"), 0),
        keyId: 6997651294655611000 } },
  operationTime: Timestamp({ t: 1644552375, i: 6 }) }
```

Answer:
It has the same information as the previous execution, but a new field called "executionStats" is added. This field has very specific information about the statistics of the execution, like how much time it took, how many documents were reviewed and many more. These may be useful when comparing the performance of different queries that seem to perform the different operations, or when deciding if a given cloud provider has a faster mongoDB server.

Question 25

Drop the two indexes you previously created, and then re-display the statistics on the query execution plan that returns all Italian restaurants. What do you see?

Query:

```
db.RestaurantsCollection.dropIndexes()
```

```
db.RestaurantsCollection.explain().find({cuisine: "Italian"})
```

```
db.RestaurantsCollection.explain("executionStats").find({cuisine: "Italian"})
```

Execution:

- Dropping index:

```
> db.RestaurantsCollection.dropIndexes()
< {
    nIndexesWas: 3,
    msg: 'non-_id indexes dropped for collection',
    ok: 1,
    '$clusterTime': {
      clusterTime: Timestamp({ t: 1644552939, i: 11 }),
      signature: {
        hash: Binary(Buffer.from("786ee90043a30c4e95f4637c57ea9c0dfba97aa1", "hex"), 0),
        keyId: Long("6997651294655610881")
      }
    },
    operationTime: Timestamp({ t: 1644552939, i: 11 })
  }
Atlas atlas-kijdn9-shard-0 [primary] DataTestBigData >
```

- Explain without executionStats:

```
{ queryPlanner:
   { plannerVersion: 1,
     namespace: 'DataTestBigData.RestaurantsCollection',
     indexFilterSet: false,
     parsedQuery: { cuisine: { '$eq': 'Italian' } },
     queryHash: '1A247377',
     planCacheKey: '1A247377',
     winningPlan:
      { stage: 'COLLSCAN',
        filter: { cuisine: { '$eq': 'Italian' } },
        direction: 'forward' },
```

```
    rejectedPlans: [] },
  serverInfo:
   { host: 'cluster0-shard-00-02.6wsnn.mongodb.net',
     port: 27017,
     version: '4.4.12',
     gitVersion: '51475a8c4d9856eb1461137e7539a0a763cc85dc' },
  ok: 1,
  '$clusterTime':
   { clusterTime: Timestamp({ t: 1644553011, i: 3 }),
     signature:
      { hash: Binary(Buffer.from("d13b2ed576fe1deee9b7d081ced5e2130aed916f", "hex"), 0),
        keyId: 6997651294655611000 } },
  operationTime: Timestamp({ t: 1644553011, i: 3 }) }
```

- Explain with executionStats:

```
{ queryPlanner:
   { plannerVersion: 1,
     namespace: 'DataTestBigData.RestaurantsCollection',
     indexFilterSet: false,
     parsedQuery: { cuisine: { '$eq': 'Italian' } },
     winningPlan:
      { stage: 'COLLSCAN',
        filter: { cuisine: { '$eq': 'Italian' } },
        direction: 'forward' },
     rejectedPlans: [] },
  executionStats:
   { executionSuccess: true,
     nReturned: 1070,
     executionTimeMillis: 13,
     totalKeysExamined: 0,
     totalDocsExamined: 25356,
     executionStages:
      { stage: 'COLLSCAN',
        filter: { cuisine: { '$eq': 'Italian' } },
        nReturned: 1070,
        executionTimeMillisEstimate: 2,
        works: 25358,
        advanced: 1070,
        needTime: 24287,
        needYield: 0,
        saveState: 25,
        restoreState: 25,
        isEOF: 1,
        direction: 'forward',
        docsExamined: 25356 } },
  serverInfo:
   { host: 'cluster0-shard-00-02.6wsnn.mongodb.net',
     port: 27017,
     version: '4.4.12',
     gitVersion: '51475a8c4d9856eb1461137e7539a0a763cc85dc' },
  ok: 1,
  '$clusterTime':
   { clusterTime: Timestamp({ t: 1644553237, i: 12 }),
     signature:
```

```
      { hash: Binary(Buffer.from("09c5facff6a2595964cd8d3c89d21c7337b5ffd1", "hex"), 0),
        keyId: 6997651294655611000 } },
  operationTime: Timestamp({ t: 1644553237, i: 12 }) }
```

Answer:
There is a lot to analyze, but what seemed most important to me was the fact that there was only one winning plan which consisted of looking at the columns until a match was found. The execution stats show that the query was 2ms slower, which may not seem like a lot but it surely adds up as our database grows bigger.
