```
//1) (5 points) list earthquakes with magnitude of 2.0 or more.

//Expected output
{
    "_id" : ObjectId("625751f3f3ed753c1850fbb9"),
    "place" : "15 km SW of Leilani Estates, Hawaii",
    "mag" : 2.44,
    "sig" : 92
}
{
    "_id" : ObjectId("625751f3f3ed753c1850fbbc"),
    "place" : "0 km WSW of Magas Arriba, Puerto Rico",
    "mag" : 3.16,
    "sig" : 154
}
{
    "_id" : ObjectId("625751f3f3ed753c1850fbbe"),
    "place" : "12km NNE of Lake Hughes, CA",
    "mag" : 2.07,
    "sig" : 66
}

//solution:
db.earthquakes.find({"properties.mag": {"$gte": 2.0 }}, {_id:1, place:
"$properties.place", mag: "$properties.mag", sig: "$properties.sig"}).pretty()

//2) (7 points) List earthquakes with longitude (coordinates[0]) between -60 and -120
and latitude (coordinates[1] between 30 and 35).

//Expected output
{
    "geometry" : {
            "coordinates" : [
                    -116.779,
                    32.9331667,
                    17.25
            ]
    },
    "place" : "8km S of San Diego Country Estates, CA"
}
{
    "geometry" : {
            "coordinates" : [
                    -118.4033333,
                    34.7781667,
                    0.69
            ]
    },
```

```
    "place" : "12km NNE of Lake Hughes, CA"
}

//solution:
db.earthquakes.find({$and:[{"geometry.coordinates.0": {"$gt": -120.0, "$lt": -60.0}},
{"geometry.coordinates.1": {"$gte": 30.0, "$lte": 35.0}}]}, {_id:0, place:
"$properties.place", "geometry.coordinates": 1}).pretty()

// 3) (5 points) Use the aggregate pipeline to list the average magnitude (mag
attribute) for each status type (status attribute)

//Expected output
{ "_id" : "reviewed", "avg_mag" : 2.31 }
{ "_id" : "automatic", "avg_mag" : 1.5633333333333332 }

//solution:
db.earthquakes.aggregate([
    { $group: {_id: "$properties.status", avg_mag: {$avg:"$properties.mag"} } },
    { $sort: {avg_mag: -1} },
    { $project: {_id: 1, avg_mag: 1} }
 ])

//4) (8 points) Redo the previous question using mapreduce

//Expected output

{
    "results" : [
            {
                    "_id" : "reviewed",
                    "value" : 2.31
            },
            {
                    "_id" : "automatic",
                    "value" : 1.5633333333333332
            }
    ],
    "ok" : 1
}

//solution:
var mapFunction1 = function() {
    emit(this.properties.status, this.properties.mag);
}

var reduceFunction1 = function(key, values) {
    if(values.length > 0){
        return (Array.sum(values)/values.length);
```

```
    }else{
        return 0;
    }
}

db.earthquakes.mapReduce(mapFunction1, reduceFunction1,
{ out: { inline: 1 }
})

//5) (13 points)
//Using the aggregate pipeline, find pairs of earthquakes that have the same network
type (net attribute)

//Expected output
{
    "_id" : ObjectId("625751f3f3ed753c1850fbb9"),
    "place" : "15 km SW of Leilani Estates, Hawaii",
    "net" : "hv",
    "other_id" : ObjectId("625751f3f3ed753c1850fbbd"),
    "other_place" : "8 km E of Pāhala, Hawaii",
    "other_net" : "hv"
}
{
    "_id" : ObjectId("625751f3f3ed753c1850fbbb"),
    "place" : "8km S of San Diego Country Estates, CA",
    "net" : "ci",
    "other_id" : ObjectId("625751f3f3ed753c1850fbbe"),
    "other_place" : "12km NNE of Lake Hughes, CA",
    "other_net" : "ci"
}

//solution:
 db.earthquakes.aggregate([
    { $project: {_id: 1, place: "$properties.place" , net: "$properties.net"} },
    { $lookup: {
        from: "earthquakes",
        localField: "net",
        foreignField: "properties.net",
        as:"matching_nets"} },
    { $unwind: "$matching_nets" },
    { $match: {$expr: {$lt: ["$_id", "$matching_nets._id"]}}},
    { $project: {_id: 1, place: 1, net: 1, other_id: "$matching_nets._id",
other_place: "$matching_nets.properties.place", other_net:
"$matching_nets.properties.net"}}]).pretty()

//6) (10 points) Using the aggregate pipeline, compute the natural join between
earthquakes and networks collections. Use the net attribute for the join condition.
```

```
//Expected output
{
    "place" : "15 km SW of Leilani Estates, Hawaii",
    "netCode" : "hv",
    "description" : "Hawaii Volcano Observatory"
}
{
    "place" : "22 km NNW of Nelchina, Alaska",
    "netCode" : "ak",
    "description" : "Alaska Earthquake Center"
}
{
    "place" : "8km S of San Diego Country Estates, CA",
    "netCode" : "ci",
    "description" : "California Integrated Seismic Network: Southern California
Seismic Network (Caltech/USGS Pasadena and Partners) and Southern California
Earthquake Data Center"
}
{
    "place" : "0 km WSW of Magas Arriba, Puerto Rico",
    "netCode" : "pr",
    "description" : "Puerto Rico Seismic Network"
}
{
    "place" : "8 km E of Pāhala, Hawaii",
    "netCode" : "hv",
    "description" : "Hawaii Volcano Observatory"
}
{
    "place" : "12km NNE of Lake Hughes, CA",
    "netCode" : "ci",
    "description" : "California Integrated Seismic Network: Southern California
Seismic Network (Caltech/USGS Pasadena and Partners) and Southern California
Earthquake Data Center"
}
//solution:
db.earthquakes.aggregate([
    { $project: {_id: 1, place: "$properties.place" , netCode: "$properties.net"} },
    { $lookup: {
        from: "networks",
        localField: "netCode",
        foreignField: "net",
        as:"matching_nets"} },
    {$match: {matching_nets: {$size:1}}},
    { $unwind: "$matching_nets" },
    { $project: {_id: 0, place: 1, netCode: 1, description:
"$matching_nets.description"}}]).pretty()
```

```
 //7) (15 points) Using the aggregate pipeline, compute earthquakes natural join
networks natural join magnitude_types. Use the earthquakes.json, networks.json, and
magnitude_types.json files.

//Expected output
{
    "place" : "15 km SW of Leilani Estates, Hawaii",
    "netCode" : "hv",
    "magType" : "ml",
    "description" : "Hawaii Volcano Observatory",
    "magDescription" : "ml  (local)",
    "magRange" : "~2.0 to ~6.5",
    "distanceRange" : "0 - 600 km"
}
{
    "place" : "22 km NNW of Nelchina, Alaska",
    "netCode" : "ak",
    "magType" : "ml",
    "description" : "Alaska Earthquake Center",
    "magDescription" : "ml  (local)",
    "magRange" : "~2.0 to ~6.5",
    "distanceRange" : "0 - 600 km"
}
{
    "place" : "8km S of San Diego Country Estates, CA",
    "netCode" : "ci",
    "magType" : "ml",
    "description" : "California Integrated Seismic Network: Southern California
Seismic Network (Caltech/USGS Pasadena and Partners) and Southern California
Earthquake Data Center",
    "magDescription" : "ml  (local)",
    "magRange" : "~2.0 to ~6.5",
    "distanceRange" : "0 - 600 km"
}
{
    "place" : "0 km WSW of Magas Arriba, Puerto Rico",
    "netCode" : "pr",
    "magType" : "md",
    "description" : "Puerto Rico Seismic Network",
    "magDescription" : "md  (duration)",
    "magRange" : "~4 or smaller",
    "distanceRange" : "0 - 400 km"
}
{
    "place" : "8 km E of Pāhala, Hawaii",
    "netCode" : "hv",
    "magType" : "md",
    "description" : "Hawaii Volcano Observatory",
```

```
    "magDescription" : "md  (duration)",
    "magRange" : "~4 or smaller",
    "distanceRange" : "0 - 400 km"
}
{
    "place" : "12km NNE of Lake Hughes, CA",
    "netCode" : "ci",
    "magType" : "ml",
    "description" : "California Integrated Seismic Network: Southern California
Seismic Network (Caltech/USGS Pasadena and Partners) and Southern California
Earthquake Data Center",
    "magDescription" : "ml  (local)",
    "magRange" : "~2.0 to ~6.5",
    "distanceRange" : "0 - 600 km"
}

 //solution:
 db.earthquakes.aggregate([
    { $project: {_id: 1, place: "$properties.place" , netCode: "$properties.net",
magType: "$properties.magType"} },
    { $lookup: {
        from: "networks",
        localField: "netCode",
        foreignField: "net",
        as:"matching_nets"} },
    {$match: {matching_nets: {$size:1}}},
    { $unwind: "$matching_nets" },
    { $project: {_id: 0, place: 1, netCode: 1, description:
"$matching_nets.description", magType: 1}},
    { $lookup: {
        from: "magnitude_types",
        localField: "magType",
        foreignField: "magType",
        as:"matching_magTypes"} },
        {$match: {matching_magTypes: {$size:1}}},
        { $unwind: "$matching_magTypes" },
    { $project: {_id: 0, place: 1, netCode: 1, description: 1, magType: 1,
magDescription: "$matching_magTypes.magDescription", magRange:
"$matching_magTypes.Magnitude Range", distanceRange: "$matching_magTypes.Distance
Range"}}]).pretty()

//8) (7 points) Using the aggregate pipeline, compute earthquakes natural join
networks natural join magnitude_types. Use the earthquakes.json, networks.json, and
magnitude_types.json files.

//solution
db.earthquakes_embedded.aggregate([
```

```
    { $project: {_id: 0, place: "$properties.place" , netCode: "$properties.net.net",
magType: "$properties.magType.magType", description: "$properties.net.description",
magDescription: "$properties.magType.magDescription", magRange:
"$properties.magType.Magnitude Range", distanceRange: "$properties.magType.Distance
Range"} }]).pretty()


// 9) (10 points) Use the mapreduce framework to count each type (the individual terms
in the types attribute).

//Expected output
{
    "results" : [
            {
                    "_id" : "origin",
                    "value" : 6
            },
            {
                    "_id" : "phase-data",
                    "value" : 6
            },
            {
                    "_id" : "nearby-cities",
                    "value" : 2
            },
            {
                    "_id" : "scitech-link",
                    "value" : 2
            }
    ],
    "ok" : 1
}

//solution:
// map function
var mapFunction1 = function() {
    var words = this.properties.types.split(",");
    for (var idx=0; idx<words.length; idx++) {
        if(words[idx].length > 0)
            emit(words[idx], 1);
    }
}

// reduce function
var reduceFunction1 = function(word, values) {
    var count=0;
    for (var idx=0; idx<values.length; idx++) {
        count = count + values[idx];
```

```
    }
    return count;
}

db.earthquakes.mapReduce(mapFunction1, reduceFunction1, {out:  {inline:1} })

//10) (20 points) Consider the undirected graph in graph.json. A diagram of the graph
is shown below:
/*
     (G:Student)
     /
    F(Course)
   /         \
  B(Faculty)--D(Course)---\
 /          \     /        \
A(student)   C(Faculty)----E(student)
*/

//The degree of a node is the number of edges it has. Find any node whose degree is
greater than or equal the degree of every adjacent node. For example, the node B has a
degree 4 which is >= the degree of all B's adjacent nodes (A=1, F=2, C=3, D=4)

//Expected Output
{
    "results" : [
            {
                    "_id" : "C",
                    "value" : "this node's degree is smaller than the degree of its
neighbors"
            },
            {
                    "_id" : "D",
                    "value" : "node degree =4 , this node's degree is >= the degree of
all of its neighbors"
            },
            {
                    "_id" : "A",
                    "value" : "this node's degree is smaller than the degree of its
neighbors"
            },
            {
                    "_id" : "G",
                    "value" : "this node's degree is smaller than the degree of its
neighbors"
            },
            {
                    "_id" : "B",
```

```
                    "value" : "node degree =4 , this node's degree is >= the degree of
all of its neighbors"
            },
            {
                    "_id" : "F",
                    "value" : "this node's degree is smaller than the degree of its
neighbors"
            },
            {
                    "_id" : "E",
                    "value" : "this node's degree is smaller than the degree of its
neighbors"
            }
    ],
    "ok" : 1
}

//solution:
var mapFunction2 = function() {
    val ={node2: this.node_id, degree: this.adj_list.length};
    emit(this.node_id, val)

    var adj = this.adj_list;
    for (var idx=0; idx<this.adj_list.length; idx++) {
        val2 = {node2: this.node_id, degree: this.adj_list.length};
        emit(this.adj_list[idx], val2);
    }
}

var reduceFunction2 = function(key, values) {
    var current_node_degree=0;
    for (var idx=0; idx<values.length; idx++) {
        if(values[idx].node2 == key){
            current_node_degree = values[idx].degree;
        }
    }

    var flag = 0;
    for (var idx=0; idx<values.length; idx++) {
        if(values[idx].degree > current_node_degree){
           flag =1;
        }
    }

    if(flag==0){
        return "node degree =" + current_node_degree + " , this node's degree is >=
the degree of all of its neighbors";
    }else{
```

```
        return "this node's degree is smaller than the degree of its neighbors";
    }

}

db.graph2.mapReduce(mapFunction2, reduceFunction2, {out:  {inline:1} })

//11) (Extra Credit, 10% of the homework grade) Consider the graph in the previous
question. Find any subgraphs of three nodes that match the pattern (x1:Faculty)-
(x2:Course)-(x3:Student)

//Expected output
{
    "results" : [
            {
                    "_id" : "A",
                    "value" : "no match found"
            },
            {
                    "_id" : "D",
                    "value" : {
                            "matches_found" : [
                                    "(C:Faculty)-(D:Course)-(E:Student)",
                                    "(B:Faculty)-(D:Course)-(E:Student)"
                            ]
                    }
            },
            {
                    "_id" : "C",
                    "value" : "no match found"
            },
            {
                    "_id" : "E",
                    "value" : "no match found"
            },
            {
                    "_id" : "G",
                    "value" : "no match found"
            },
            {
                    "_id" : "B",
                    "value" : "no match found"
            },
            {
                    "_id" : "F",
                    "value" : {
                            "matches_found" : [
                                    "(B:Faculty)-(F:Course)-(G:Student)"
```

```
                            ]
                    }
            }
    ],
    "ok" : 1
}
```