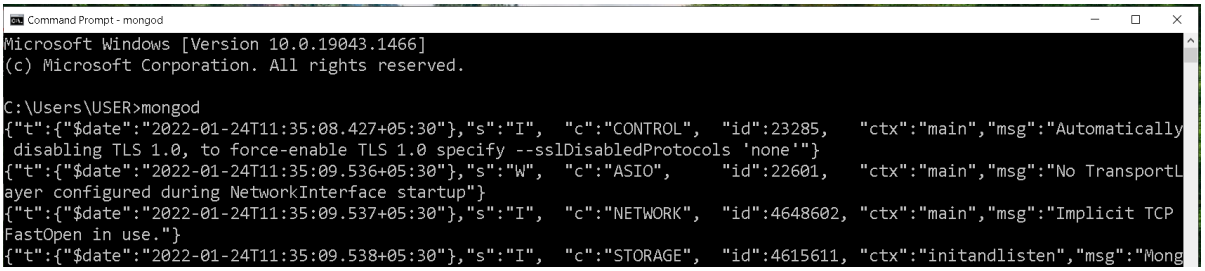
Assignment 1

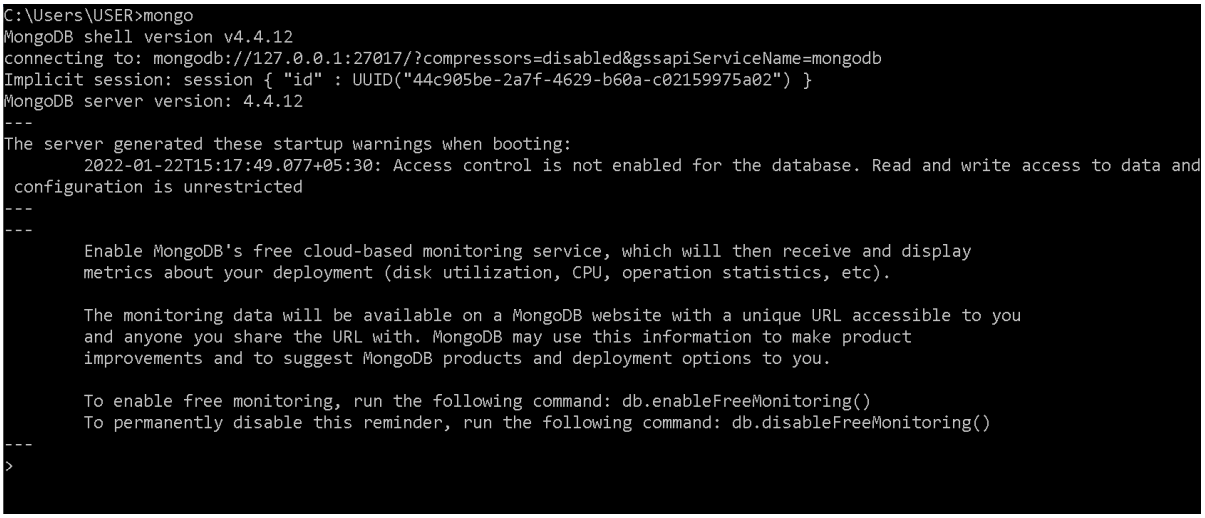
MongoDB Exercise in mongo shell

Connect to a running mongo instance, use a database named mongo_practice.

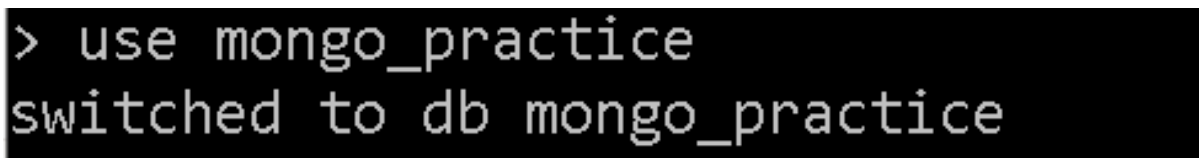
1. Started mongodb server



2. Started mongodb shell



3. Created database mongo_practice



Document all your queries in a javascript file to use as a reference.

Insert Documents

Insert the following documents into a movies collection.

title : Fight Club

writer : Chuck Palahniuko

year : 1999

actors : [Brad Pitt Edward Norton]

title : Pulp Fiction

writer : Quentin Tarantino

year : 1994

actors : [John Travolta Uma Thurman]

title : Inglorious Basterds

writer : Quentin Tarantino

year : 2009

actors : [Brad Pitt Diane Kruger Eli Roth]

title : The Hobbit: An Unexpected Journey

writer : J.R.R. Tolkein

year : 2012

franchise : The Hobbit

title : The Hobbit: The Desolation of Smaug

writer : J.R.R. Tolkein

year : 2013

franchise : The Hobbit

title : The Hobbit: The Battle of the Five Armies

writer : J.R.R. Tolkein

year : 2012

franchise : The Hobbit

synopsis : Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness.

title : Pee Wee Herman's Big Adventure

title : Avatar

Inserted all the entries in the movies collection

```
> db.movies.insertMany(
... [
... {
... title : "Fight Club", writer : "Chuck Palahniuko", year : 1999, actors : [ "Brad Pitt", "Edward Norton" ]
... },
... {
... title : "Pulp Fiction", writer : "Quentin Tarantino", year : 1994, actors : [ "John Travolta", "Uma Thurman" ]
... },
... {
... title : "Inglorious Basterds", writer : "Quentin Tarantino", year : 2009, actors : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ]
... },
... {
... title : "The Hobbit: An Unexpected Journey", writer : "J.R.R. Tolkein", year : 2012, franchise : "The Hobbit"
... },
... {
... title : "The Hobbit: The Desolation of Smaug", writer : "J.R.R. Tolkein", year : 2013, franchise: "The Hobbit"
... },
... {
... title : "The Hobbit: The Battle of the Five Armies", writer : "J.R.R. Tolkein", year : 2012, franchise : "The Hobbit", synopsis : "
Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands
 of a rising darkness."
... },
... {
... title : "Pee Wee Herman's Big Adventure"
... },
... {
... title : "Avatar"
... }
... ]
... )
```

```
... )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61ee497862f85eb1e724b0a7"),
                ObjectId("61ee497862f85eb1e724b0a8"),
                ObjectId("61ee497862f85eb1e724b0a9"),
                ObjectId("61ee497862f85eb1e724b0aa"),
                ObjectId("61ee497862f85eb1e724b0ab"),
                ObjectId("61ee497862f85eb1e724b0ac"),
                ObjectId("61ee497862f85eb1e724b0ad"),
                ObjectId("61ee497862f85eb1e724b0ae")
        ]
}
```

Query / Find Documents

query the movies collection to

1. get all documents

```
> db.movies.find().pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a7"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a8"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a9"),
        "title" : "Inglorious Basterds",
        "writer" : "Quentin Tarantino",
        "year" : 2009,
        "actors" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
```

```
{
        "_id" : ObjectId("61ee497862f85eb1e724b0aa"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ab"),
        "title" : "The Hobbit: The Desolation of Smaug",
        "writer" : "J.R.R. Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ac"),
        "title" : "The Hobbit: The Battle of the Five Armies",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from
falling into the hands of a rising darkness."
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ad"),
        "title" : "Pee Wee Herman's Big Adventure"
}
{ "_id" : ObjectId("61ee497862f85eb1e724b0ae"), "title" : "Avatar" }
```

2. get all documents with writer set to "Quentin Tarantino"

```
> db.movies.find({writer:"Quentin Tarantino"}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a8"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a9"),
        "title" : "Inglorious Basterds",
        "writer" : "Quentin Tarantino",
        "year" : 2009,
        "actors" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
```

3. get all documents where actors include "Brad Pitt"

```
> db.movies.find({actors:"Brad Pitt"}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a7"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a9"),
        "title" : "Inglorious Basterds",
        "writer" : "Quentin Tarantino",
        "year" : 2009,
        "actors" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
```

4. get all documents with franchise set to "The Hobbit"

```
> db.movies.find({franchise: "The Hobbit"}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0aa"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ab"),
        "title" : "The Hobbit: The Desolation of Smaug",
        "writer" : "J.R.R. Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ac"),
        "title" : "The Hobbit: The Battle of the Five Armies",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from
falling into the hands of a rising darkness."
}
```

5. get all movies released in the 90s

```
> db.movies.find(
... {
... $and: [
...           {year: {$gte:1990}}, {year: {$lte:1999}}
...       ]
... }
... ).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a7"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a8"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
```

6. get all movies released before the year 2000 or after 2010

```
> db.movies.find(
... {
...       $or:  [
...                   {year: {$lt:2000}}, {year: {$gt:2010}}
...             ]
... }
... ).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a7"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0a8"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
```

```
{
       "_id" : ObjectId("61ee497862f85eb1e724b0aa"),
       "title" : "The Hobbit: An Unexpected Journey",
       "writer" : "J.R.R. Tolkein",
       "year" : 2012,
       "franchise" : "The Hobbit"
}
{
       "_id" : ObjectId("61ee497862f85eb1e724b0ab"),
       "title" : "The Hobbit: The Desolation of Smaug",
       "writer" : "J.R.R. Tolkein",
       "year" : 2013,
       "franchise" : "The Hobbit"
}
{
       "_id" : ObjectId("61ee497862f85eb1e724b0ac"),
       "title" : "The Hobbit: The Battle of the Five Armies",
       "writer" : "J.R.R. Tolkein",
       "year" : 2012,
       "franchise" : "The Hobbit",
       "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness."
}
```

Update Documents

1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."

```
> db.movies.updateOne(
... { title: "The Hobbit: An Unexpected Journey"},
... { $set: { synopsis: "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim
 their mountain home - and the gold within it - from the dragon Smaug."}}
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

2. add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

```
> db.movies.updateOne(
... { title: "The Hobbit: The Desolation of Smaug"},
... { $set: { synopsis: "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their home
land, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}}
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

3. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction"

```
> db.movies.updateOne(
... { title: "Pulp Fiction"},
... { $push: { actors: "Samuel L. Jackson"}}
... )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

Text Search

1. find all movies that have a synopsis that contains the word "Bilbo"

```
> db.movies.find({synopsis:{$regex:"Bilbo"}}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0aa"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim th
eir mountain home - and the gold within it - from the dragon Smaug."
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ab"),
        "title" : "The Hobbit: The Desolation of Smaug",
        "writer" : "J.R.R. Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit",
        "synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homelan
d, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ac"),
        "title" : "The Hobbit: The Battle of the Five Armies",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from
falling into the hands of a rising darkness."
}
```

2. find all movies that have a synopsis that contains the word "Gandalf"

```
> db.movies.find({synopsis:{$regex:"Gandalf"}}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ab"),
        "title" : "The Hobbit: The Desolation of Smaug",
        "writer" : "J.R.R. Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit",
        "synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homelan
d, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."
}
```

3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"

```
> db.movies.find({$and:[{synopsis:{$regex:"Bilbo"}}, {synopsis:{$not:/Gandalf/}}]}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0aa"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim th
eir mountain home - and the gold within it - from the dragon Smaug."
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ac"),
        "title" : "The Hobbit: The Battle of the Five Armies",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from
falling into the hands of a rising darkness."
}
```

4. find all movies that have a synopsis that contains the word "dwarves" or "hobbit"

```
> db.movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0aa"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim th
eir mountain home - and the gold within it - from the dragon Smaug."
}
{
        "_id" : ObjectId("61ee497862f85eb1e724b0ab"),
        "title" : "The Hobbit: The Desolation of Smaug",
        "writer" : "J.R.R. Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit",
        "synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homelan
d, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."
}
```

5. find all movies that have a synopsis that contains the word "gold" and "dragon"

```
> db.movies.find({$and:[{synopsis:{$regex:"gold"}}, {synopsis:{$regex:"dragon"}}]}).pretty()
{
        "_id" : ObjectId("61ee497862f85eb1e724b0aa"),
        "title" : "The Hobbit: An Unexpected Journey",
        "writer" : "J.R.R. Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim th
eir mountain home - and the gold within it - from the dragon Smaug."
}
```

Delete Documents

1. delete the movie "Pee Wee Herman's Big Adventure"

```
> db.movies.remove({ title: "Pee Wee Herman's Big Adventure"})
WriteResult({ "nRemoved" : 1 })
```

2. delete the movie "Avatar"

```
> db.movies.remove({ title:"Avatar"})
WriteResult({ "nRemoved" : 1 })
```

Relationships

Insert the following documents into a users collection

username : GoodGuyGreg

first_name : "Good Guy"

last_name : "Greg"

username : ScumbagSteve

full_name :

first : "Scumbag"

last : "Steve"

```
> db.users.insertMany(
... [
... {
... username : "GoodGuyGreg", first_name : "Good Guy", last_name : "Greg"
... },
... {
... username : "ScumbagSteve", full_name : {first : "Scumbag", last : "Steve"}
... }
... ]
... )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61eeb68962f85eb1e724b0af"),
                ObjectId("61eeb68962f85eb1e724b0b0")
        ]
}
```

Insert the following documents into a posts collection

username : GoodGuyGreg

title : Passes out at party

body : Wakes up early and cleans house

username : GoodGuyGreg

title : Steals your identity

body : Raises your credit score

username : GoodGuyGreg

title : Reports a bug in your code

body : Sends you a Pull Request

username : ScumbagSteve

title : Borrows something

body : Sells it

username : ScumbagSteve

title : Borrows everything

body : The end

username : ScumbagSteve

title : Forks your repo on github

body : Sets to private

```
> db.posts.insertMany(
... [
... {
... username : "GoodGuyGreg", title : "Passes out at party", body : "Wakes up early and cleans house"
... },
... {
... username : "GoodGuyGreg", title : "Steals your identity", body : "Raises your credit score"
... },
... {
... username : "GoodGuyGreg", title : "Reports a bug in your code", body : "Sends you a Pull Request"
... },
... {
... username : "ScumbagSteve", title : "Borrows something", body : "Sells it"
... },
... {
... username : "ScumbagSteve", title : "Borrows everything", body : "The end"
... },
... {
... username : "ScumbagSteve", title : "Forks your repo on github", body : "Sets to private"
... }
... ]
... )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61eeb8dc62f85eb1e724b0b1"),
                ObjectId("61eeb8dc62f85eb1e724b0b2"),
                ObjectId("61eeb8dc62f85eb1e724b0b3"),
                ObjectId("61eeb8dc62f85eb1e724b0b4"),
                ObjectId("61eeb8dc62f85eb1e724b0b5"),
                ObjectId("61eeb8dc62f85eb1e724b0b6")
        ]
}
```

Insert the following documents into a comments collection

username : GoodGuyGreg

comment : Hope you got a good deal!

post : [post_obj_id]

where [post_obj_id] is the ObjectId of the posts document: "Borrows something"

username : GoodGuyGreg

comment : What's mine is yours!

post : [post_obj_id]

where [post_obj_id] is the ObjectId of the posts document: "Borrows everything"

username : GoodGuyGreg

comment : Don't violate the licensing agreement!

post : [post_obj_id]

where [post_obj_id] is the ObjectId of the posts document: "Forks your repo on github

username : ScumbagSteve

comment : It still isn't clean

post : [post_obj_id]

where [post_obj_id] is the ObjectId of the posts document: "Passes out at party"

username : ScumbagSteve

comment : Denied your PR cause I found a hack

post : [post_obj_id]

where [post_obj_id] is the ObjectId of the posts document: "Reports a bug in your code"

```
> db.comments.insertMany(
... [
... {
... username : "GoodGuyGreg", comment : "Hope you got a good deal!", post :ObjectId("61eeb8dc62f85eb1e724b0b4")
... },
... {
... username : "GoodGuyGreg", comment : "What's mine is yours!", post :ObjectId("61eeb8dc62f85eb1e724b0b5")
... },
... {
... username : "GoodGuyGreg", comment : "Don't violate the licensing agreement!", post :ObjectId("61eeb8dc62f85eb1e724b0b6")
... },
... {
... username : "ScumbagSteve", comment : "It still isn't clean", post :ObjectId("61eeb8dc62f85eb1e724b0b1")
... },
... {
... username : "ScumbagSteve", comment : "Denied your PR cause I found a hack", post :ObjectId("61eeb8dc62f85eb1e724b0b3")
... }
... ]
... )
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61eebf7162f85eb1e724b0bd"),
                ObjectId("61eebf7162f85eb1e724b0be"),
                ObjectId("61eebf7162f85eb1e724b0bf"),
                ObjectId("61eebf7162f85eb1e724b0c0"),
                ObjectId("61eebf7162f85eb1e724b0c1")
        ]
}
```

Querying related collections

1. find all users

```
> db.users.find().pretty()
{
        "_id" : ObjectId("61eeb68962f85eb1e724b0af"),
        "username" : "GoodGuyGreg",
        "first_name" : "Good Guy",
        "last_name" : "Greg"
}
{
        "_id" : ObjectId("61eeb68962f85eb1e724b0b0"),
        "username" : "ScumbagSteve",
        "full_name" : {
                "first" : "Scumbag",
                "last" : "Steve"
        }
}
```

2. find all posts

```
> db.posts.find().pretty()
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b1"),
        "username" : "GoodGuyGreg",
        "title" : "Passes out at party",
        "body" : "Wakes up early and cleans house"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b2"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b3"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a Pull Request"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b4"),
        "username" : "ScumbagSteve",
        "title" : "Borrows something",
        "body" : "Sells it"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b5"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
        "body" : "The end"
}
```

```
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b6"),
        "username" : "ScumbagSteve",
        "title" : "Forks your repo on github",
        "body" : "Sets to private"
}
```

3. find all posts that was authored by "GoodGuyGreg"

```
> db.posts.find({username: "GoodGuyGreg"}).pretty()
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b1"),
        "username" : "GoodGuyGreg",
        "title" : "Passes out at party",
        "body" : "Wakes up early and cleans house"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b2"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b3"),
        "username" : "GoodGuyGreg",
        "title" : "Reports a bug in your code",
        "body" : "Sends you a Pull Request"
}
```

4. find all posts that was authored by "ScumbagSteve"

```
> db.posts.find({username: "ScumbagSteve"}).pretty()
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b4"),
        "username" : "ScumbagSteve",
        "title" : "Borrows something",
        "body" : "Sells it"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b5"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
        "body" : "The end"
}
{
        "_id" : ObjectId("61eeb8dc62f85eb1e724b0b6"),
        "username" : "ScumbagSteve",
        "title" : "Forks your repo on github",
        "body" : "Sets to private"
}
```

5. find all comments

```
> db.comments.find().pretty()
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0bd"),
        "username" : "GoodGuyGreg",
        "comment" : "Hope you got a good deal!",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b4")
}
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0be"),
        "username" : "GoodGuyGreg",
        "comment" : "What's mine is yours!",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b5")
}
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0bf"),
        "username" : "GoodGuyGreg",
        "comment" : "Don't violate the licensing agreement!",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b6")
}
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0c0"),
        "username" : "ScumbagSteve",
        "comment" : "It still isn't clean",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b1")
}
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0c1"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b3")
}
```

6. find all comments that was authored by "GoodGuyGreg"

```
> db.comments.find({username: "GoodGuyGreg"}).pretty()
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0bd"),
        "username" : "GoodGuyGreg",
        "comment" : "Hope you got a good deal!",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b4")
}
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0be"),
        "username" : "GoodGuyGreg",
        "comment" : "What's mine is yours!",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b5")
}
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0bf"),
        "username" : "GoodGuyGreg",
        "comment" : "Don't violate the licensing agreement!",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b6")
}
```

7. find all comments that was authored by "ScumbagSteve"

```
> db.comments.find({username: "ScumbagSteve"}).pretty()
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0c0"),
        "username" : "ScumbagSteve",
        "comment" : "It still isn't clean",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b1")
}
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0c1"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b3")
}
```

8. find all comments belonging to the post "Reports a bug in your code"

```
> db.comments.find({post: ObjectId("61eeb8dc62f85eb1e724b0b3")}).pretty()
{
        "_id" : ObjectId("61eebf7162f85eb1e724b0c1"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("61eeb8dc62f85eb1e724b0b3")
}
```

Assignment 2

MongoDB -Aggregation Exercises

Import the zips.json file into your MongoDB. Database name is "population" and collection name is "zipcodes".

mongoimport --db population --collection zipcodes --file zips.json

```
C:\Users\USER>mongoimport --db population --collection zipcodes --file C:\Users\USER\Desktop\json\zip.json
2022-01-24T22:32:06.269+0530    connected to: mongodb://localhost/
2022-01-24T22:32:07.093+0530    29353 document(s) imported successfully. 0 document(s) failed to import.
```

Atlanta Population

1. use db.zipcodes.find() to filter results to only the results where city is ATLANTA and state is GA.

```
> db.zipcodes.find(
... {
... $and: [
...             {city: "ATLANTA"},  {state: "GA"}
...       ]
... }
... ).pretty()
{
        "_id" : "30303",
        "city" : "ATLANTA",
        "loc" : [
                -84.388846,
                33.752504
        ],
        "pop" : 1845,
        "state" : "GA"
}
{
        "_id" : "30305",
        "city" : "ATLANTA",
        "loc" : [
                -84.385145,
                33.831963
        ],
        "pop" : 19122,
        "state" : "GA"
}
```

2. use db.zipcodes.aggregate with $match to do the same as above.

```
> db.zipcodes.aggregate([
... {
...     $match:  {
...           $and: [
...               {city:"ATLANTA"},
...               {state:"GA"}
...           ]
...     }
... }
... ])
{ "_id" : "30303", "city" : "ATLANTA", "loc" : [ -84.388846, 33.752504 ], "pop" : 1845, "state" : "GA" }
{ "_id" : "30305", "city" : "ATLANTA", "loc" : [ -84.385145, 33.831963 ], "pop" : 19122, "state" : "GA" }
{ "_id" : "30306", "city" : "ATLANTA", "loc" : [ -84.351418, 33.786027 ], "pop" : 20081, "state" : "GA" }
{ "_id" : "30307", "city" : "ATLANTA", "loc" : [ -84.335957, 33.769138 ], "pop" : 16330, "state" : "GA" }
{ "_id" : "30309", "city" : "ATLANTA", "loc" : [ -84.388338, 33.798407 ], "pop" : 14766, "state" : "GA" }
{ "_id" : "30310", "city" : "ATLANTA", "loc" : [ -84.423173, 33.727849 ], "pop" : 34017, "state" : "GA" }
{ "_id" : "30311", "city" : "ATLANTA", "loc" : [ -84.470219, 33.722957 ], "pop" : 34880, "state" : "GA" }
{ "_id" : "30312", "city" : "ATLANTA", "loc" : [ -84.378125, 33.746749 ], "pop" : 17683, "state" : "GA" }
{ "_id" : "30313", "city" : "ATLANTA", "loc" : [ -84.39352, 33.76825 ], "pop" : 8038, "state" : "GA" }
{ "_id" : "30314", "city" : "ATLANTA", "loc" : [ -84.425546, 33.756103 ], "pop" : 26649, "state" : "GA" }
{ "_id" : "30308", "city" : "ATLANTA", "loc" : [ -84.375744, 33.771839 ], "pop" : 8549, "state" : "GA" }
{ "_id" : "30315", "city" : "ATLANTA", "loc" : [ -84.380771, 33.705062 ], "pop" : 41061, "state" : "GA" }
{ "_id" : "30316", "city" : "ATLANTA", "loc" : [ -84.333913, 33.721686 ], "pop" : 34668, "state" : "GA" }
{ "_id" : "30317", "city" : "ATLANTA", "loc" : [ -84.31685, 33.749788 ], "pop" : 16395, "state" : "GA" }
{ "_id" : "30318", "city" : "ATLANTA", "loc" : [ -84.445432, 33.786454 ], "pop" : 53894, "state" : "GA" }
{ "_id" : "30319", "city" : "ATLANTA", "loc" : [ -84.335091, 33.868728 ], "pop" : 32138, "state" : "GA" }
{ "_id" : "30324", "city" : "ATLANTA", "loc" : [ -84.354867, 33.820609 ], "pop" : 15044, "state" : "GA" }
{ "_id" : "30326", "city" : "ATLANTA", "loc" : [ -84.358232, 33.848168 ], "pop" : 125, "state" : "GA" }
{ "_id" : "30327", "city" : "ATLANTA", "loc" : [ -84.419966, 33.862723 ], "pop" : 18467, "state" : "GA" }
{ "_id" : "30329", "city" : "ATLANTA", "loc" : [ -84.321402, 33.823555 ], "pop" : 17013, "state" : "GA" }
Type "it" for more
```

3. use $group to count the number of zip codes in Atlanta.

```
> db.zipcodes.aggregate( [
... {  $group: { _id: { city: "$city"}, total: { $sum: 1 } } },
... {  $match: { "_id.city": "ATLANTA"} }
... ] )
{ "_id" : { "city" : "ATLANTA" }, "total" : 41 }
```

4. use $group to find the total population in Atlanta.

```
> db.zipcodes.aggregate( [
... {  $group: { _id: { city: "$city"}, totalPopulation: { $sum: "$pop" } } },
... {  $match: { "_id.city": "ATLANTA"} }
... ] )
{ "_id" : { "city" : "ATLANTA" }, "totalPopulation" : 630046 }
```

Populations By State

1. use aggregate to calculate the total population for each state

```
> db.zipcodes.aggregate( [
...    { $group: { _id: "$state", totalPopulation: { $sum: "$pop" } } }
... ] )
{ "_id" : "OH", "totalPopulation" : 10846517 }
{ "_id" : "TX", "totalPopulation" : 16984601 }
{ "_id" : "ID", "totalPopulation" : 1006749 }
{ "_id" : "MT", "totalPopulation" : 798948 }
{ "_id" : "IN", "totalPopulation" : 5544136 }
{ "_id" : "UT", "totalPopulation" : 1722850 }
{ "_id" : "VT", "totalPopulation" : 562758 }
{ "_id" : "GA", "totalPopulation" : 6478216 }
{ "_id" : "AL", "totalPopulation" : 4040587 }
{ "_id" : "MS", "totalPopulation" : 2573216 }
{ "_id" : "WI", "totalPopulation" : 4891769 }
{ "_id" : "SD", "totalPopulation" : 695397 }
{ "_id" : "OK", "totalPopulation" : 3145585 }
{ "_id" : "NV", "totalPopulation" : 1201833 }
{ "_id" : "WY", "totalPopulation" : 453528 }
{ "_id" : "MO", "totalPopulation" : 5110648 }
{ "_id" : "AZ", "totalPopulation" : 3665228 }
{ "_id" : "ND", "totalPopulation" : 638272 }
{ "_id" : "CO", "totalPopulation" : 3293755 }
{ "_id" : "FL", "totalPopulation" : 12686644 }
Type "it" for more
```

2. sort the results by population, highest first

```
> db.zipcodes.aggregate( [
... { $group: { _id: "$state", totalPopulation: { $sum: "$pop" } } },
... { $sort:  { totalPopulation: -1 } }
... ] )
{ "_id" : "CA", "totalPopulation" : 29754890 }
{ "_id" : "NY", "totalPopulation" : 17990402 }
{ "_id" : "TX", "totalPopulation" : 16984601 }
{ "_id" : "FL", "totalPopulation" : 12686644 }
{ "_id" : "PA", "totalPopulation" : 11881643 }
{ "_id" : "IL", "totalPopulation" : 11427576 }
{ "_id" : "OH", "totalPopulation" : 10846517 }
{ "_id" : "MI", "totalPopulation" : 9295297 }
{ "_id" : "NJ", "totalPopulation" : 7730188 }
{ "_id" : "NC", "totalPopulation" : 6628637 }
{ "_id" : "GA", "totalPopulation" : 6478216 }
{ "_id" : "VA", "totalPopulation" : 6181479 }
{ "_id" : "MA", "totalPopulation" : 6016425 }
{ "_id" : "IN", "totalPopulation" : 5544136 }
{ "_id" : "MO", "totalPopulation" : 5110648 }
{ "_id" : "WI", "totalPopulation" : 4891769 }
{ "_id" : "TN", "totalPopulation" : 4876457 }
{ "_id" : "WA", "totalPopulation" : 4866692 }
{ "_id" : "MD", "totalPopulation" : 4781379 }
{ "_id" : "MN", "totalPopulation" : 4372982 }
Type "it" for more
```

3. limit the results to just the first 3 results. What are the top 3 states in population?

```
> db.zipcodes.aggregate( [
... { $group: { _id: "$state", totalPopulation: { $sum: "$pop" } } },
... { $sort:  { totalPopulation: -1 } },
... { $limit: 3}
... ] )
{ "_id" : "CA", "totalPopulation" : 29754890 }
{ "_id" : "NY", "totalPopulation" : 17990402 }
{ "_id" : "TX", "totalPopulation" : 16984601 }
```

Populations by City

1. use aggregate to calculate the total population for each city (you have to use city/state combination). You can use a combination for the _id of the $group: { city: '$city', state: '$state' }

```
> db.zipcodes.aggregate( [
... { $group: { _id: { city: "$city", state: "$state" }, totalPopulation: { $sum: "$pop" } } }
... ] )
{ "_id" : { "city" : "FINDLAY", "state" : "OH" }, "totalPopulation" : 48109 }
{ "_id" : { "city" : "WILLOW RIVER", "state" : "MN" }, "totalPopulation" : 968 }
{ "_id" : { "city" : "INTERNATIONAL FA", "state" : "MN" }, "totalPopulation" : 11124 }
{ "_id" : { "city" : "WESCO", "state" : "MO" }, "totalPopulation" : 460 }
{ "_id" : { "city" : "DURANGO", "state" : "CO" }, "totalPopulation" : 23506 }
{ "_id" : { "city" : "PRINCESS ANNE", "state" : "MD" }, "totalPopulation" : 10343 }
{ "_id" : { "city" : "GUTTENBERG", "state" : "NJ" }, "totalPopulation" : 44735 }
{ "_id" : { "city" : "FAIRMOUNT", "state" : "GA" }, "totalPopulation" : 3405 }
{ "_id" : { "city" : "GREAT BEND", "state" : "ND" }, "totalPopulation" : 235 }
{ "_id" : { "city" : "NEWCASTLE", "state" : "OK" }, "totalPopulation" : 4653 }
{ "_id" : { "city" : "CRAIG", "state" : "CO" }, "totalPopulation" : 10242 }
{ "_id" : { "city" : "BALD KNOB", "state" : "AR" }, "totalPopulation" : 5132 }
{ "_id" : { "city" : "MOUNTAIN HOME A", "state" : "ID" }, "totalPopulation" : 6304 }
{ "_id" : { "city" : "LOCUST GROVE", "state" : "AR" }, "totalPopulation" : 750 }
{ "_id" : { "city" : "MONACA", "state" : "PA" }, "totalPopulation" : 13344 }
{ "_id" : { "city" : "QUINLAN", "state" : "TX" }, "totalPopulation" : 13826 }
{ "_id" : { "city" : "PLAINVIEW", "state" : "SD" }, "totalPopulation" : 153 }
{ "_id" : { "city" : "DEL VALLE", "state" : "TX" }, "totalPopulation" : 5635 }
{ "_id" : { "city" : "CAMDEN", "state" : "AL" }, "totalPopulation" : 4948 }
{ "_id" : { "city" : "GLENOMA", "state" : "WA" }, "totalPopulation" : 657 }
Type "it" for more
```

2. sort the results by population, highest first

```
> db.zipcodes.aggregate( [
... { $group: { _id: { city: "$city", state: "$state" }, totalPopulation: { $sum: "$pop" } } },
... { $sort:  { totalPopulation: -1 } },
... ] )
{ "_id" : { "city" : "CHICAGO", "state" : "IL" }, "totalPopulation" : 2452177 }
{ "_id" : { "city" : "BROOKLYN", "state" : "NY" }, "totalPopulation" : 2300504 }
{ "_id" : { "city" : "LOS ANGELES", "state" : "CA" }, "totalPopulation" : 2102295 }
{ "_id" : { "city" : "HOUSTON", "state" : "TX" }, "totalPopulation" : 2095918 }
{ "_id" : { "city" : "PHILADELPHIA", "state" : "PA" }, "totalPopulation" : 1610956 }
{ "_id" : { "city" : "NEW YORK", "state" : "NY" }, "totalPopulation" : 1476790 }
{ "_id" : { "city" : "BRONX", "state" : "NY" }, "totalPopulation" : 1209548 }
{ "_id" : { "city" : "SAN DIEGO", "state" : "CA" }, "totalPopulation" : 1049298 }
{ "_id" : { "city" : "DETROIT", "state" : "MI" }, "totalPopulation" : 963243 }
{ "_id" : { "city" : "DALLAS", "state" : "TX" }, "totalPopulation" : 940191 }
{ "_id" : { "city" : "PHOENIX", "state" : "AZ" }, "totalPopulation" : 890853 }
{ "_id" : { "city" : "MIAMI", "state" : "FL" }, "totalPopulation" : 825232 }
{ "_id" : { "city" : "SAN JOSE", "state" : "CA" }, "totalPopulation" : 816653 }
{ "_id" : { "city" : "SAN ANTONIO", "state" : "TX" }, "totalPopulation" : 811792 }
{ "_id" : { "city" : "BALTIMORE", "state" : "MD" }, "totalPopulation" : 733081 }
{ "_id" : { "city" : "SAN FRANCISCO", "state" : "CA" }, "totalPopulation" : 723993 }
{ "_id" : { "city" : "MEMPHIS", "state" : "TN" }, "totalPopulation" : 632837 }
{ "_id" : { "city" : "SACRAMENTO", "state" : "CA" }, "totalPopulation" : 628279 }
{ "_id" : { "city" : "JACKSONVILLE", "state" : "FL" }, "totalPopulation" : 610160 }
{ "_id" : { "city" : "ATLANTA", "state" : "GA" }, "totalPopulation" : 609591 }
Type "it" for more
```

3. limit the results to just the first 3 results. What are the top 3 cities in population?

```
> db.zipcodes.aggregate( [
... { $group: { _id: { city: "$city", state: "$state" }, totalPopulation: { $sum: "$pop" } } },
... { $sort: { totalPopulation: -1 } },
... { $limit: 3 }
... ] )
{ "_id" : { "city" : "CHICAGO", "state" : "IL" }, "totalPopulation" : 2452177 }
{ "_id" : { "city" : "BROOKLYN", "state" : "NY" }, "totalPopulation" : 2300504 }
{ "_id" : { "city" : "LOS ANGELES", "state" : "CA" }, "totalPopulation" : 2102295 }
```

4. What are the top 3 cities in population in Texas?

```
> db.zipcodes.aggregate( [
... { $group: { _id: { city: "$city", state: "$state" }, totalPopulation: { $sum: "$pop" } } },
... { $match: { "_id.state": "TX" } },
... { $sort: { totalPopulation: -1 } },
... { $limit: 3 }
... ] )
{ "_id" : { "city" : "HOUSTON", "state" : "TX" }, "totalPopulation" : 2095918 }
{ "_id" : { "city" : "DALLAS", "state" : "TX" }, "totalPopulation" : 940191 }
{ "_id" : { "city" : "SAN ANTONIO", "state" : "TX" }, "totalPopulation" : 811792 }
```

Bonus

1. Write a query to get the average city population for each state.

```
> db.zipcodes.aggregate( [
... { $group: { _id: { city: "$city", state: "$state" }, pop: { $sum: "$pop" } } },
... { $group: { _id: "$_id.state", avgCityPop: { $avg: "$pop" } } }
... ] )
{ "_id" : "DE", "avgCityPop" : 14481.91304347826 }
{ "_id" : "TN", "avgCityPop" : 9656.350495049504 }
{ "_id" : "DC", "avgCityPop" : 303450 }
{ "_id" : "MT", "avgCityPop" : 2593.987012987013 }
{ "_id" : "IN", "avgCityPop" : 9271.130434782608 }
{ "_id" : "UT", "avgCityPop" : 9518.508287292818 }
{ "_id" : "WA", "avgCityPop" : 12258.670025188916 }
{ "_id" : "SD", "avgCityPop" : 1839.6746031746031 }
{ "_id" : "AL", "avgCityPop" : 7907.2152641878665 }
{ "_id" : "MS", "avgCityPop" : 7524.023391812865 }
{ "_id" : "WI", "avgCityPop" : 7323.00748502994 }
{ "_id" : "OK", "avgCityPop" : 6155.743639921722 }
{ "_id" : "GA", "avgCityPop" : 11547.62210338681 }
{ "_id" : "NV", "avgCityPop" : 18209.590909090908 }
{ "_id" : "WY", "avgCityPop" : 3384.5373134328356 }
{ "_id" : "MO", "avgCityPop" : 5672.195338512764 }
{ "_id" : "AZ", "avgCityPop" : 20591.16853932584 }
{ "_id" : "CO", "avgCityPop" : 9981.075757575758 }
{ "_id" : "ND", "avgCityPop" : 1645.0309278350514 }
{ "_id" : "AR", "avgCityPop" : 4175.355239786856 }
Type "it" for more
```

2. What are the top 3 states in terms of average city population?

```
> db.zipcodes.aggregate( [
... { $group: { _id: { city: "$city", state: "$state" }, pop: { $sum: "$pop" } } },
... { $group: { _id: "$_id.state", avgCityPop: { $avg: "$pop" } } },
... { $sort: { avgCityPop: -1 } },
... { $limit: 3 } ] )
{ "_id" : "DC", "avgCityPop" : 303450 }
{ "_id" : "CA", "avgCityPop" : 27756.42723880597 }
{ "_id" : "FL", "avgCityPop" : 27400.958963282937 }
```

Assignment 3

MongoDB – Complex Queries

Mongo DB Exercises - With the Restaurants Data Set

1. Download the restaurants.zip file

2. Unzip the file, you will see restaurants.json file

3. Run the mongod server

4. Run the following command to import the json file provided. It will load the json file into the mongodb with database name - restaurants, collections name - addresses

mongoimport --db restaurants --collection addresses --file restaurants.json

```
C:\Users\USER>mongoimport --db restaurants --collection addresses --file C:\Users\USER\Desktop\json\restaurants.json
2022-01-25T12:16:50.454+0530    connected to: mongodb://localhost/
2022-01-25T12:16:51.010+0530    3772 document(s) imported successfully. 0 document(s) failed to import.
```

5. Run mongo shell command

6. show databases

```
> show dbs
admin           0.000GB
ani             0.000GB
config          0.000GB
local           0.000GB
mongo_practice  0.000GB
population      0.002GB
restaurants     0.001GB
```

7. use restaurants

8. db.addresses.find() should print entire json data

9. Then start working on the following exercises and submit your queries as the answers to the questions

Exercise Questions

1. Write a MongoDB query to display all the documents in the collection restaurants.

```
> db.addresses.find().pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00c"),
        "address" : {
                "building" : "469",
                "coord" : [
                        -73.961704,
                        40.662942
                ],
                "street" : "Flatbush Avenue",
                "zipcode" : "11225"
        },
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "grades" : [
                {
                        "date" : ISODate("2014-12-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2014-07-01T00:00:00Z"),
                        "grade" : "B",
                        "score" : 23
                },
                {
                        "date" : ISODate("2013-04-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2012-05-08T00:00:00Z"),
                        "grade" : "A",
```

2. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

```
> db.addresses.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00c"),
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00d"),
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00e"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Riviera Caterer",
        "restaurant_id" : "40356018"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00f"),
        "borough" : "Queens",
        "cuisine" : "Jewish/Kosher",
        "name" : "Tov Kosher Kitchen",
        "restaurant_id" : "40356068"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b010"),
        "borough" : "Queens",
        "cuisine" : "American ",
```

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

```
> db.addresses.find({},{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1,"_id":0}).pretty()
{
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Riviera Caterer",
        "restaurant_id" : "40356018"
}
{
        "borough" : "Queens",
        "cuisine" : "Jewish/Kosher",
        "name" : "Tov Kosher Kitchen",
        "restaurant_id" : "40356068"
}
{
        "borough" : "Queens",
        "cuisine" : "American ",
        "name" : "Brunos On The Boulevard",
        "restaurant_id" : "40356151"
}
{
        "borough" : "Staten Island",
```

4. 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.addresses.find({},{"restaurant_id" : 1,"name":1,"borough":1,"address.zipcode" :1,"_id":0}).pretty()
{
        "address" : {
                "zipcode" : "11225"
        },
        "borough" : "Brooklyn",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "address" : {
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "address" : {
                "zipcode" : "11224"
        },
        "borough" : "Brooklyn",
        "name" : "Riviera Caterer",
        "restaurant_id" : "40356018"
}
{
        "address" : {
                "zipcode" : "11374"
        },
        "borough" : "Queens",
        "name" : "Tov Kosher Kitchen",
        "restaurant_id" : "40356068"
}
```

5. Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.

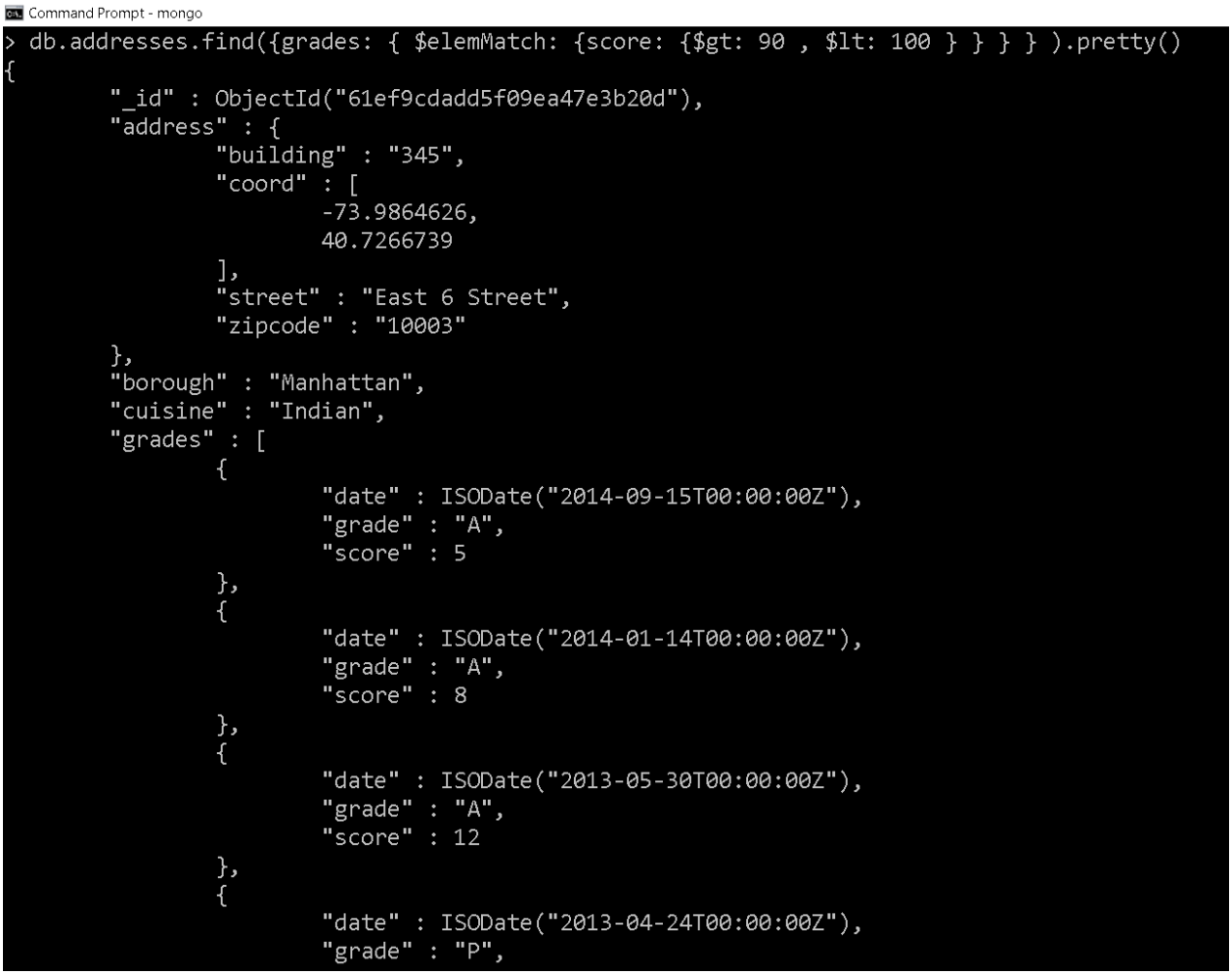
```
> db.addresses.find({borough:"Bronx"}).limit(5).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00d"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2011-11-23T00:00:00Z"),
                        "grade" : "A",
```

6. Write a MongoDB query to display all the restaurant which is in the borough Bronx.

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

8. Write a MongoDB query to find the restaurants who achieved a score more than 90.

9. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.



10. Write a MongoDB query to find the restaurants which locate in latitude value less than -95.754168.

11. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.

Command Prompt - mongo

```
> db.addresses.find(
... { $and:
...       [
...        { cuisine: { $ne: "American " } },
...        {"grades.score": {$gt : 70} },
...        {"address.coord.0": { $lt : -65.754168 } }
...       ]
... }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b20d"),
        "address" : {
                "building" : "345",
                "coord" : [
                        -73.9864626,
                        40.7266739
                ],
                "street" : "East 6 Street",
                "zipcode" : "10003"
        },
        "borough" : "Manhattan",
        "cuisine" : "Indian",
        "grades" : [
                {
                        "date" : ISODate("2014-09-15T00:00:00Z"),
                        "grade" : "A",
                        "score" : 5
                },
                {
                        "date" : ISODate("2014-01-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
```

12. 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.addresses.find(
... { $and:
...       [
...         { cuisine: { $ne: "American " } },
...         {"grades.score": {$gt : 70} },
...         {"address.coord.1": { $lt : -65.754168 } }
...       ]
... }
... ).pretty()
```

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

Command Prompt - mongo

```
> db.addresses.find(
... { $and:
...       [
...         { cuisine: { $ne: "American "} },
...         {"grades.grade": "A" },
...         {"borough": { $ne: "Brooklyn" } }
...       ]
... }
... ).sort({ "cuisine": -1}).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b718"),
        "address" : {
                "building" : "89",
                "coord" : [
                        -73.9995899,
                        40.7168015
                ],
                "street" : "Baxter Street",
                "zipcode" : "10013"
        },
        "borough" : "Manhattan",
        "cuisine" : "Vietnamese/Cambodian/Malaysia",
        "grades" : [
                {
                        "date" : ISODate("2014-08-21T00:00:00Z"),
                        "grade" : "A",
                        "score" : 13
                },
                {
                        "date" : ISODate("2013-08-31T00:00:00Z"),
                        "grade" : "A",
                        "score" : 13
                },
```

14. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.

```
> db.addresses.find(
... { name: { $regex: /^Wil/ } },
... {"restaurant_id":1, "name":1 ,"borough":1 ,"cuisine":1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b014"),
        "borough" : "Bronx",
        "cuisine" : "American ",
        "name" : "Wild Asia",
        "restaurant_id" : "40357217"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b016"),
        "borough" : "Brooklyn",
        "cuisine" : "Delicatessen",
        "name" : "Wilken'S Fine Food",
        "restaurant_id" : "40356483"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3be1b"),
        "borough" : "Bronx",
        "cuisine" : "Pizza",
        "name" : "Wilbel Pizza",
        "restaurant_id" : "40871979"
}
```

15. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

Command Prompt - mongo

```
> db.addresses.find(
... { name: { $regex: /ces$/ } },
... { "restaurant_id":1, "name":1, "borough":1 ,"cuisine":1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b49e"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Pieces",
        "restaurant_id" : "40399910"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b55e"),
        "borough" : "Queens",
        "cuisine" : "American ",
        "name" : "S.M.R Restaurant Services",
        "restaurant_id" : "40403857"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b567"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Good Shepherd Services",
        "restaurant_id" : "40403989"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3ba17"),
        "borough" : "Queens",
        "cuisine" : "Ice Cream, Gelato, Yogurt, Ices",
        "name" : "The Ice Box-Ralph'S Famous Italian Ices",
        "restaurant_id" : "40690899"
}
{
```

16. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

Command Prompt - mongo

```
> db.addresses.find(
... {name: { $regex:/Reg/ } },
... {"restaurant_id":1 ,"name":1 ,"borough":1 ,"cuisine":1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b012"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Regina Caterers",
        "restaurant_id" : "40356649"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b111"),
        "borough" : "Manhattan",
        "cuisine" : "Café/Coffee/Tea",
        "name" : "Caffe Reggio",
        "restaurant_id" : "40369418"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b21f"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Regency Hotel",
        "restaurant_id" : "40382679"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b53d"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Regency Whist Club",
        "restaurant_id" : "40402377"
}
{
```

17. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronx or Brooklyn.

Command Prompt - mongo

```
> db.addresses.find(
... { "borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"] } },
... {"restaurant_id" : 1, "name": 1, "borough": 1, "cuisine": 1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00c"),
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00d"),
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00e"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "Riviera Caterer",
        "restaurant_id" : "40356018"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00f"),
        "borough" : "Queens",
        "cuisine" : "Jewish/Kosher",
        "name" : "Tov Kosher Kitchen",
        "restaurant_id" : "40356068"
}
{
```

19. 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 Bronxor Brooklyn.

Command Prompt - mongo

```
> db.addresses.find(
... { "borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"] } },
... {"restaurant_id" : 1, "name": 1, "borough": 1, "cuisine": 1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b013"),
        "borough" : "Manhattan",
        "cuisine" : "Irish",
        "name" : "Dj Reynolds Pub And Restaurant",
        "restaurant_id" : "30191841"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b017"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "1 East 66Th Street Kitchen",
        "restaurant_id" : "40359480"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b020"),
        "borough" : "Manhattan",
        "cuisine" : "Delicatessen",
        "name" : "Bully'S Deli",
        "restaurant_id" : "40361708"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b021"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Glorious Food",
        "restaurant_id" : "40361521"
}
{
```

20. 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.

Command Prompt - mongo

```
> db.addresses.find(
... { "grades.score":{ $not: {$gt : 10} } },
... {"restaurant_id" :1,"name" :1,"borough":1,"cuisine":1}
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b015"),
        "borough" : "Brooklyn",
        "cuisine" : "American ",
        "name" : "C & C Catering Service",
        "restaurant_id" : "40357437"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b017"),
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "1 East 66Th Street Kitchen",
        "restaurant_id" : "40359480"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b01e"),
        "borough" : "Brooklyn",
        "cuisine" : "Delicatessen",
        "name" : "Nordic Delicacies",
        "restaurant_id" : "40361390"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b02d"),
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "name" : "White Castle",
        "restaurant_id" : "40362344"
}
{
```

21. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

Command Prompt - mongo

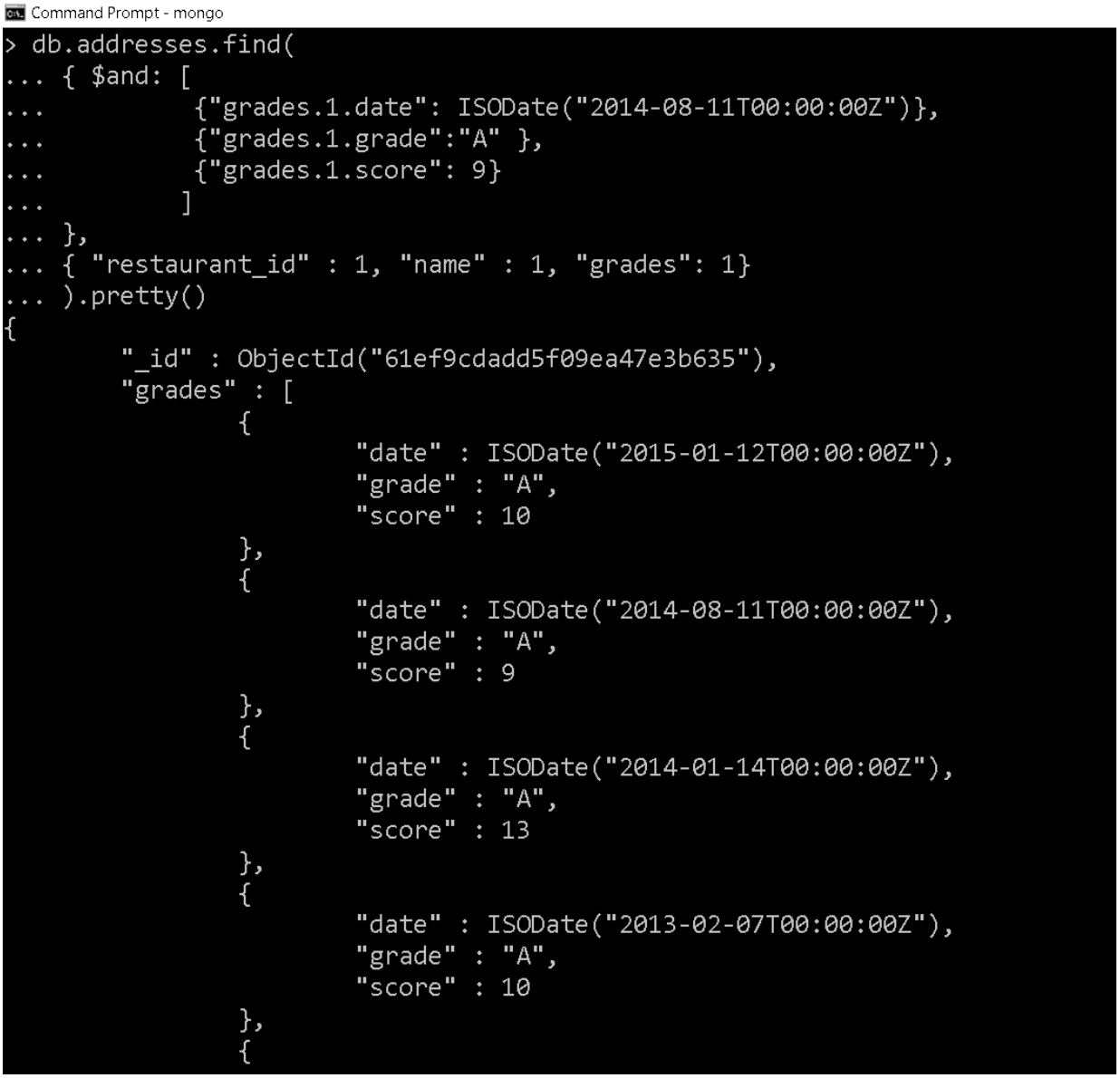
```
> db.addresses.find(
... {$or: [
...         {name: /^Wil/},
...         {"$and": [
...                   {"cuisine" : {$ne :"American "}},
...                   {"cuisine" : {$ne :"Chinese"}}
...                  ]
...         }
...        ]
... },
... {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00c"),
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "name" : "Wendy'S",
        "restaurant_id" : "30112340"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00d"),
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00f"),
        "borough" : "Queens",
        "cuisine" : "Jewish/Kosher",
        "name" : "Tov Kosher Kitchen",
        "restaurant_id" : "40356068"
}
```

22. 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..

Command Prompt - mongo

```
> db.addresses.find(
... { "grades.date": ISODate("2014-08-11T00:00:00Z"),
...   "grades.grade":"A" ,
...   "grades.score" : 11
... },
... {"restaurant_id" : 1,"name":1,"grades":1}
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b089"),
        "grades" : [
                {
                        "date" : ISODate("2014-08-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 13
                },
                {
                        "date" : ISODate("2013-07-22T00:00:00Z"),
                        "grade" : "A",
                        "score" : 9
                },
                {
                        "date" : ISODate("2013-03-14T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2012-07-02T00:00:00Z"),
                        "grade" : "A",
                        "score" : 11
                },
                {
                        "date" : ISODate("2012-02-02T00:00:00Z"),
                        "grade" : "A",
```

23. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z"



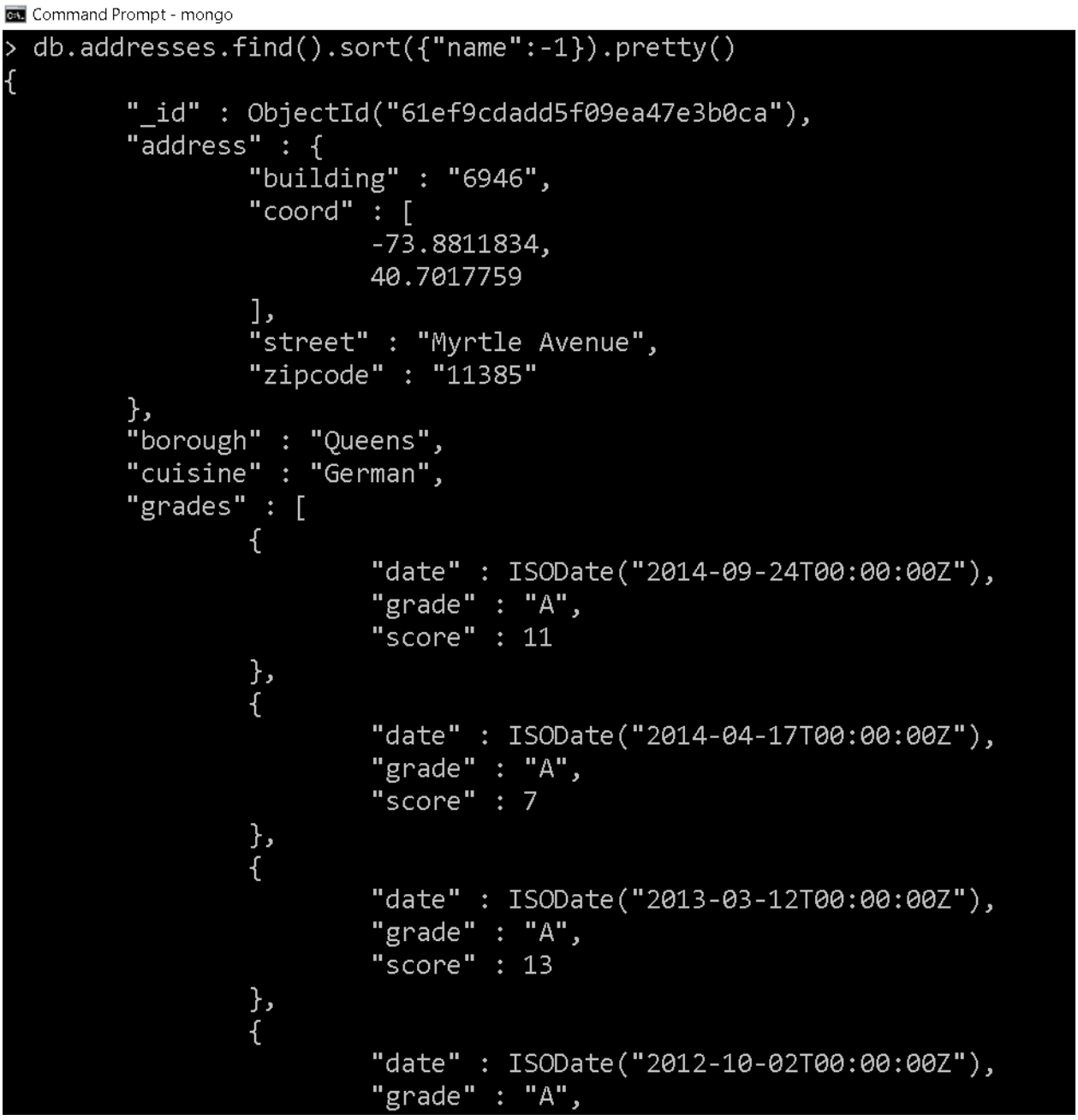
24. Write a MongoDB query to find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..

Command Prompt - mongo

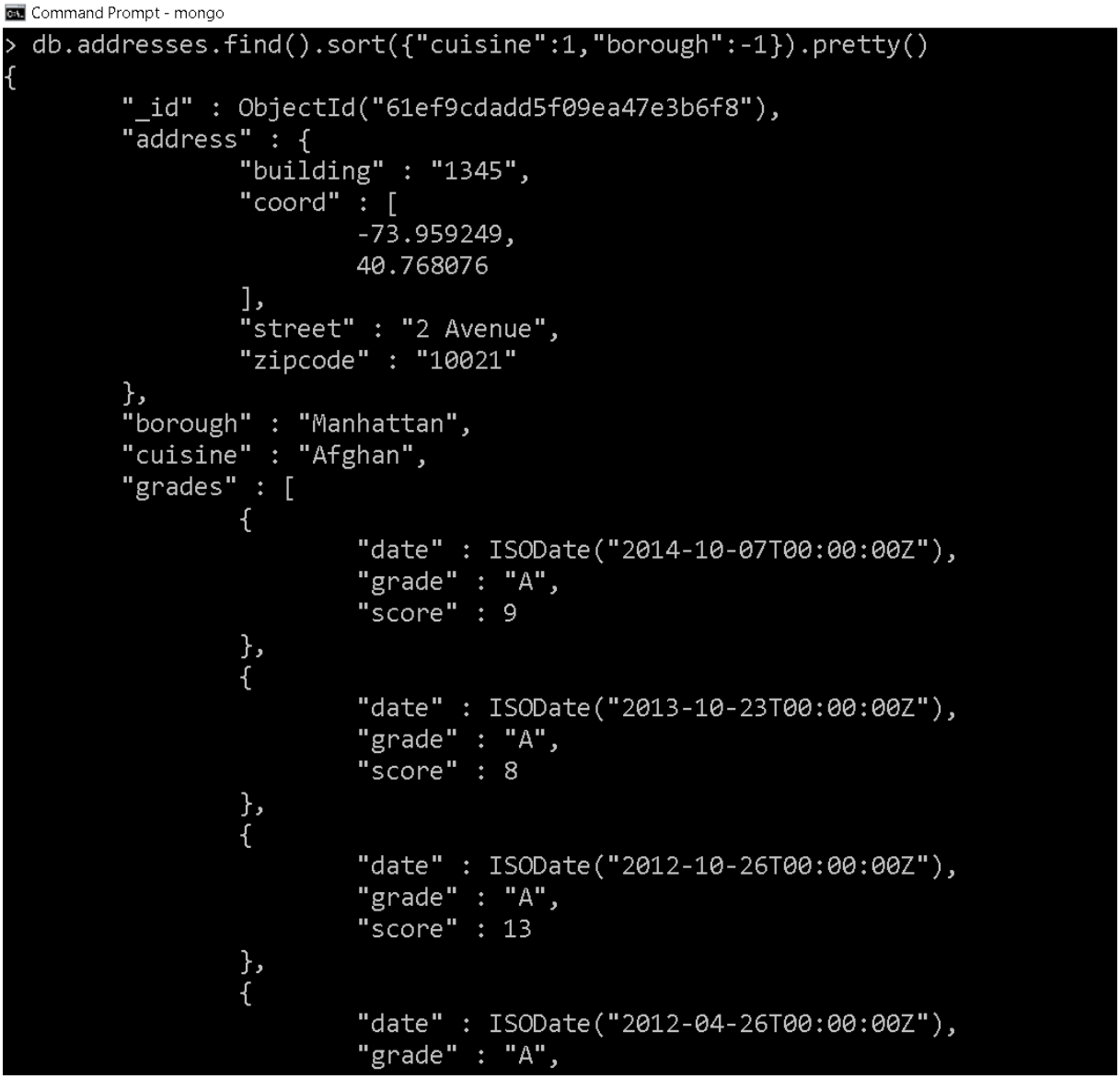
```
> db.addresses.find(
... { "address.coord.1": {$gt : 42, $lte : 52} },
... { "restaurant_id" : 1, "name" : 1, "address": 1, "coord": 1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b2ae"),
        "address" : {
                "building" : "47",
                "coord" : [
                        -78.877224,
                        42.89546199999999
                ],
                "street" : "Broadway @ Trinity Pl",
                "zipcode" : "10006"
        },
        "name" : "T.G.I. Friday'S",
        "restaurant_id" : "40387990"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b2da"),
        "address" : {
                "building" : "1",
                "coord" : [
                        -0.7119979,
                        51.6514664
                ],
                "street" : "Pennplaza E, Penn Sta",
                "zipcode" : "10001"
        },
        "name" : "T.G.I. Fridays",
        "restaurant_id" : "40388936"
}
{
```

25. Write a MongoDB query to arrange the name of the restaurants in ascending order along with all the columns.

26. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.



27. Write a MongoDB query to arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.



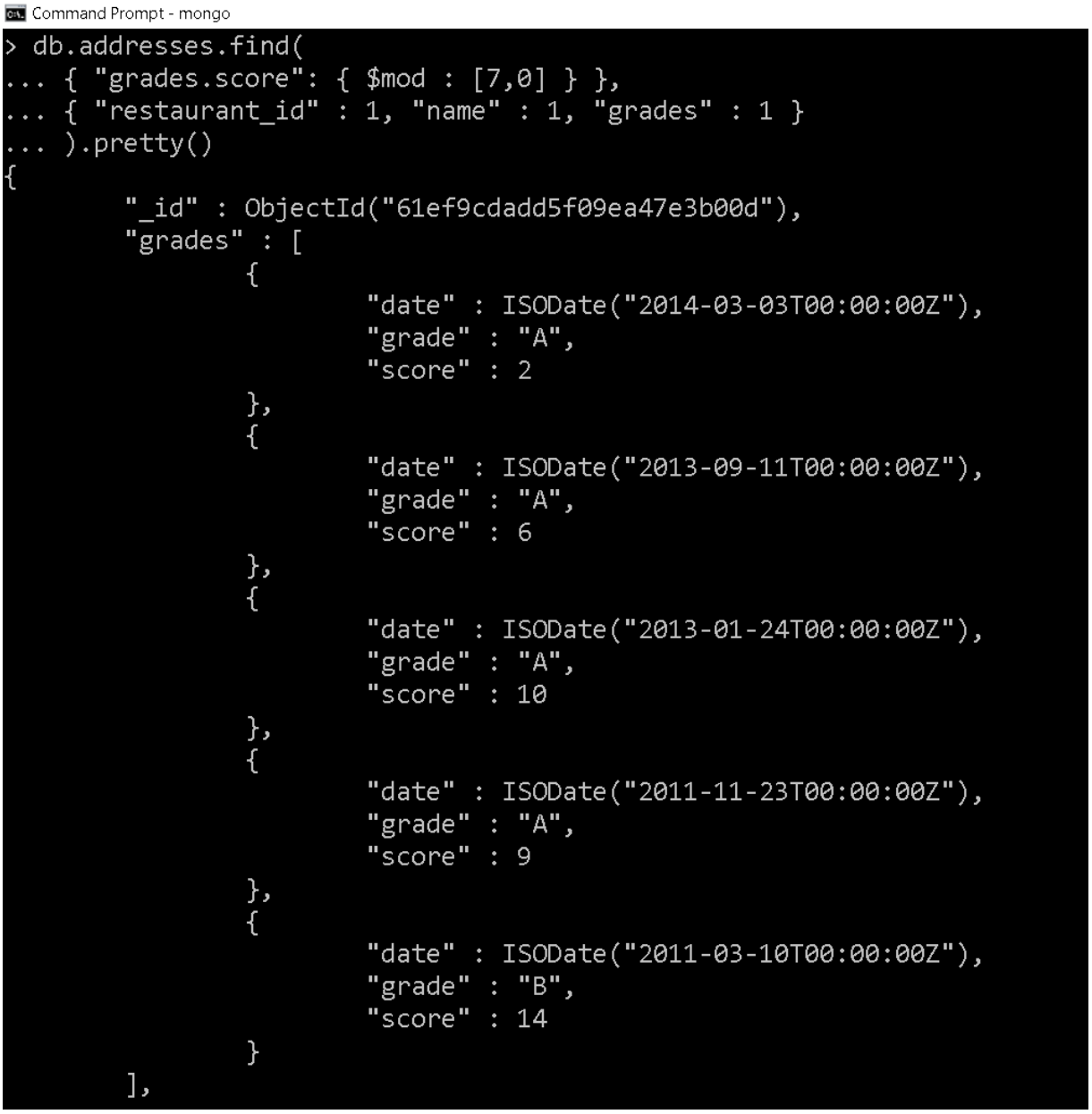
28. Write a MongoDB query to know whether all the addresses contains the street or not.

29. Write a MongoDB query which will select all documents in the restaurants collection where the coord field value is Double.

Command Prompt - mongo

```
> db.addresses.find( {"address.coord" : { $type : 1 } } ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b00c"),
        "address" : {
                "building" : "469",
                "coord" : [
                        -73.961704,
                        40.662942
                ],
                "street" : "Flatbush Avenue",
                "zipcode" : "11225"
        },
        "borough" : "Brooklyn",
        "cuisine" : "Hamburgers",
        "grades" : [
                {
                        "date" : ISODate("2014-12-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 8
                },
                {
                        "date" : ISODate("2014-07-01T00:00:00Z"),
                        "grade" : "B",
                        "score" : 23
                },
                {
                        "date" : ISODate("2013-04-30T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2012-05-08T00:00:00Z"),
                        "grade" : "A",
```

30. Write a MongoDB query which will select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.



31. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

Command Prompt - mongo

```
> db.addresses.find(
... { name: {$regex: /mon/ } },
... { "name":1, "borough":1, "address.coord":1, "cuisine":1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b09f"),
        "address" : {
                "coord" : [
                        -73.98306099999999,
                        40.7441419
                ]
        },
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Desmond'S Tavern"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b0a9"),
        "address" : {
                "coord" : [
                        -73.8221418,
                        40.7272376
                ]
        },
        "borough" : "Queens",
        "cuisine" : "Jewish/Kosher",
        "name" : "Shimons Kosher Pizza"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b0b9"),
        "address" : {
                "coord" : [
                        -74.10465599999999,
```

32. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.

```
> db.addresses.find(
... { name: { $regex: /^Mad/ } },
... { "name":1 , "borough":1, "address.coord":1, "cuisine":1 }
... ).pretty()
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b548"),
        "address" : {
                "coord" : [
                        -73.9860597,
                        40.7431194
                ]
        },
        "borough" : "Manhattan",
        "cuisine" : "American ",
        "name" : "Madison Square"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b618"),
        "address" : {
                "coord" : [
                        -73.98302199999999,
                        40.742313
                ]
        },
        "borough" : "Manhattan",
        "cuisine" : "Indian",
        "name" : "Madras Mahal"
}
{
        "_id" : ObjectId("61ef9cdadd5f09ea47e3b8c4"),
        "address" : {
                "coord" : [
                        -74.000002,
```