# SECTION-1

# Insert the following documents into a movies collection

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
test    0.000GB
> db
test
> show collections
log
movies
> db.movies.insert([{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']},
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

```
> db.movies.insert([{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'},
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
> db.movies.insert({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 against an
WriteResult({ "nInserted" : 1 })
```

```
> db.movies.insert({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 against an array
WriteResult({ "nInserted" : 1 })
> db.movies.insert([{title:'Pee Wee Hermans Big Adventure'},
... {title:'avatar'},
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
```

## Query Find Documents:-

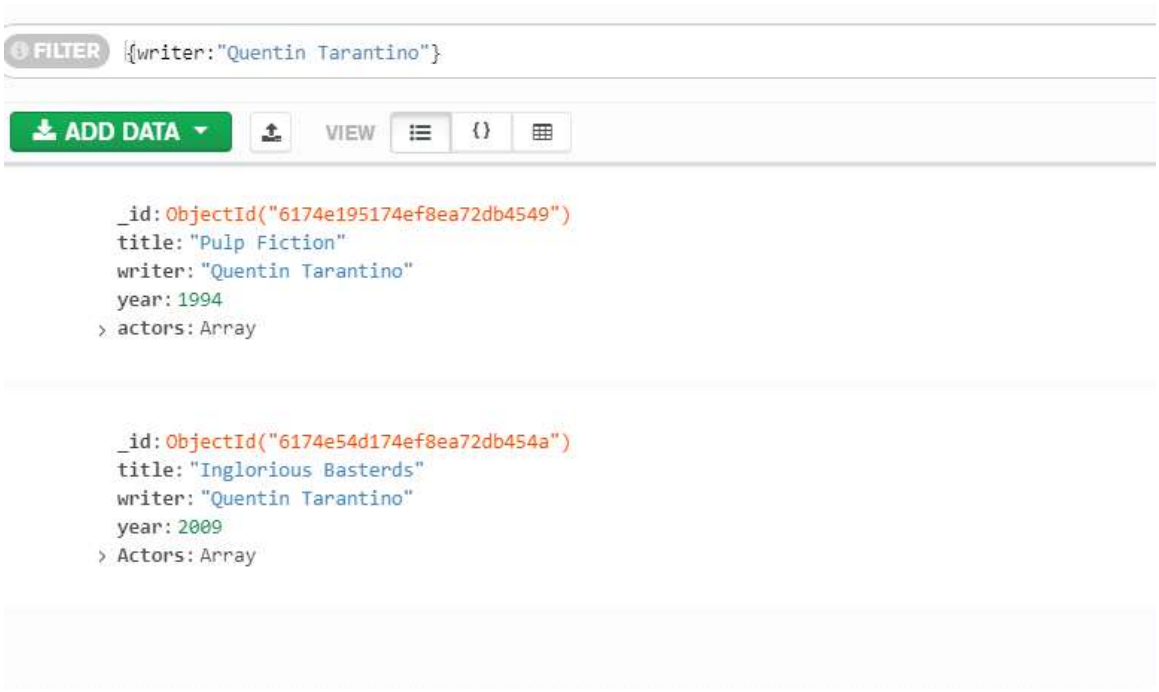
1. Get all documents

```
> db.movies.find().pretty()
{
        "_id" : ObjectId("6174e195174ef8ea72db4548"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("6174e195174ef8ea72db4549"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("6174e54d174ef8ea72db454a"),
        "title" : "Inglorious Basterds",
        "writer" : "Quentin Tarantino",
        "year" : 2009,
        "Actors" : [
                "Brad Pitt",
                "Diane Kruger",
                "Eli Roth"
        ]
}
{
        "_id" : ObjectId("6174e54d174ef8ea72db454b"),
        "title" : "The Hobbit An Unexpected Journey",
        "writer" : "J.R.R.Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit"
```

```
        "_id" : ObjectId("6174e54d174ef8ea72db454c"),
        "title" : "The Hobbit The Desolation of Smaug",
        "writer" : "J.R.R.Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("6174eea6174ef8ea72db454d"),
        "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 against an array"
}
{
        "_id" : ObjectId("6174efa5174ef8ea72db454e"),
        "title" : "Pee Wee Hermans Big Adventure"
}
{ "_id" : ObjectId("6174efa5174ef8ea72db454f"), "title" : "avatar" }
> db.movies.insert([{title:'Pee Wee Hermans Big Adventure'}, {title:'avatar'}, ])
```

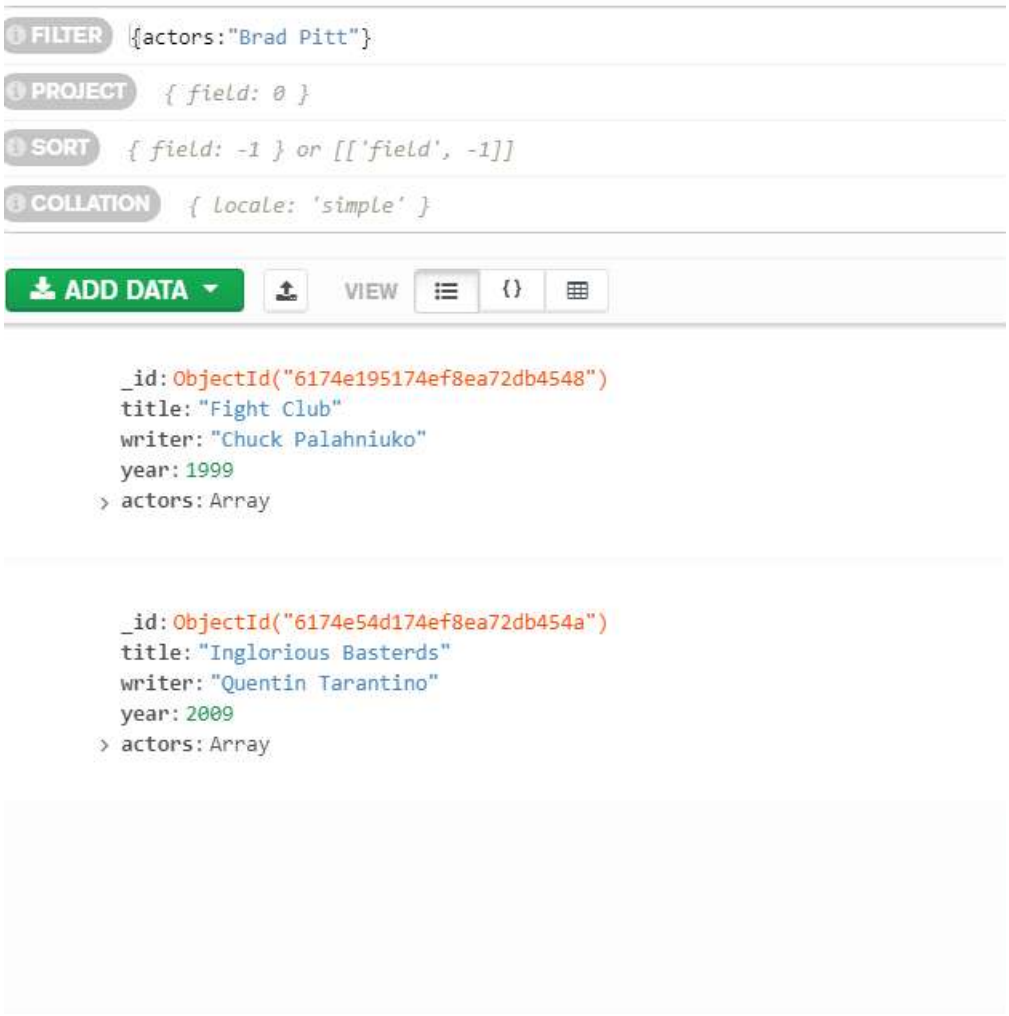
2.Get all documents with writer set to "Quentin Tarantino".

```
> db.movies.find({writer:'Quentin Tarantino'}).pretty()
{
        "_id" : ObjectId("6174e195174ef8ea72db4549"),
        "title" : "Pulp Fiction",
        "writer" : "Quentin Tarantino",
        "year" : 1994,
        "actors" : [
                "John Travolta",
                "Uma Thurman"
        ]
}
{
        "_id" : ObjectId("6174e54d174ef8ea72db454a"),
        "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("6174e195174ef8ea72db4548"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]


        "_id" : ObjectId("6174e54d174ef8ea72db454a"),
        "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("6174e54d174ef8ea72db454b"),
        "title" : "The Hobbit An Unexpected Journey",
        "writer" : "J.R.R.Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("6174e54d174ef8ea72db454c"),
        "title" : "The Hobbit The Desolation of Smaug",
        "writer" : "J.R.R.Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit"
}
{
        "_id" : ObjectId("6174eea6174ef8ea72db454d"),
        "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 against an array"
}
>
```

5 .Get all movies released in the 90s

```
> db.movies.find({year:{$gt:1900,$lt:2000}}).pretty()
{
        "_id" : ObjectId("6174e195174ef8ea72db4548"),
        "title" : "Fight Club",
        "writer" : "Chuck Palahniuko",
        "year" : 1999,
        "actors" : [
                "Brad Pitt",
                "Edward Norton"
        ]
}
{
        "_id" : ObjectId("6174e195174ef8ea72db4549"),
        "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

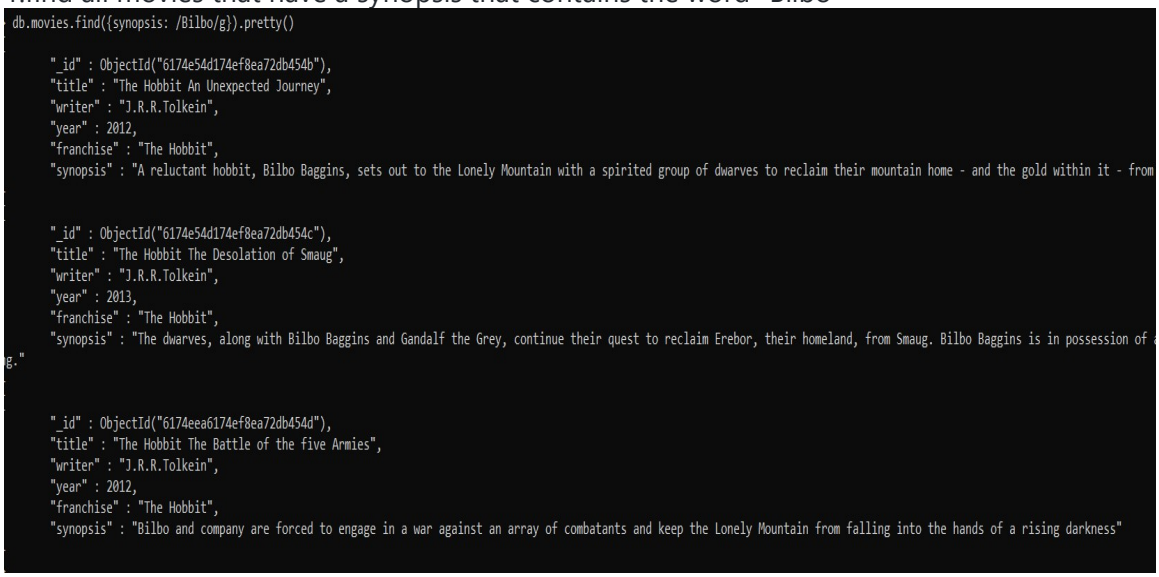
```
db.movies.find({$or:[{year: {$gt:2010}},{year :{$lt:2000}}]}).pretty()

      "_id" : ObjectId("6174e195174ef8ea72db4548"),
      "title" : "Fight Club",
      "writer" : "Chuck Palahniuko",
      "year" : 1999,
      "actors" : [
              "Brad Pitt",
              "Edward Norton"
      ]


      "_id" : ObjectId("6174e195174ef8ea72db4549"),
      "title" : "Pulp Fiction",
      "writer" : "Quentin Tarantino",
      "year" : 1994,
      "actors" : [
              "John Travolta",
              "Uma Thurman"
      ]


      "_id" : ObjectId("6174e54d174ef8ea72db454b"),
      "title" : "The Hobbit An Unexpected Journey",
      "writer" : "J.R.R.Tolkein",
      "year" : 2012,
      "franchise" : "The Hobbit"


      "_id" : ObjectId("6174e54d174ef8ea72db454c"),
      "title" : "The Hobbit The Desolation of Smaug",
      "writer" : "J.R.R.Tolkein",
      "year" : 2013,
      "franchise" : "The Hobbit"


      "_id" : ObjectId("6174eea6174ef8ea72db454d"),
      "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 against an array"
```

## Update Documents:-

```
> db.movies.update(
... {"title" : "The Hobbit An Unexpected Journey" },
... {$set: { "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." }}
);
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movies.update(
... {"title" : "The Hobbit The Desolation of Smaug"  },
... {$set: { "synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magic
al ring."  }});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.movies.update({title: "Pulp Fiction"}, {$push: {actors: "Samuel L. Jackson"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

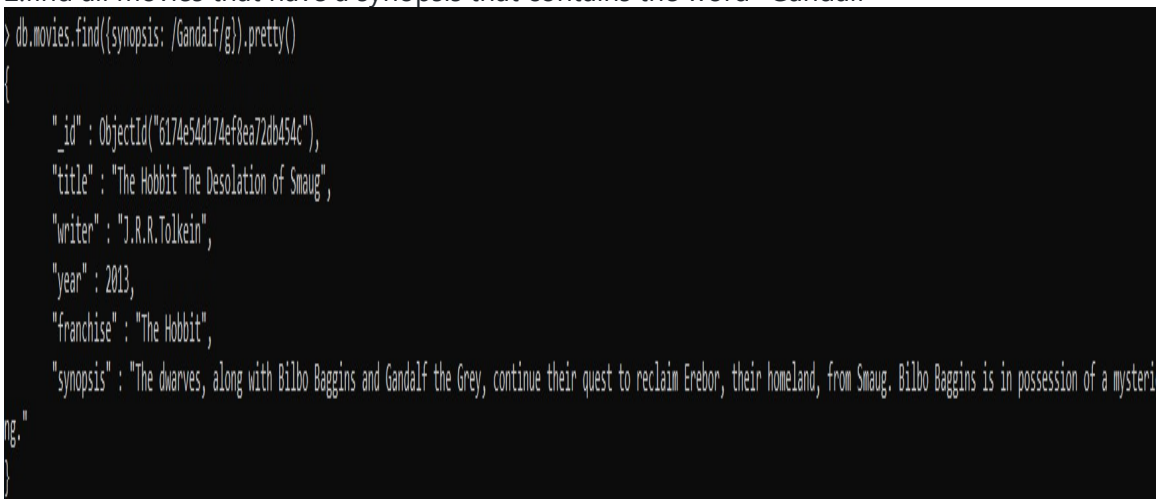
## Text Search:-

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



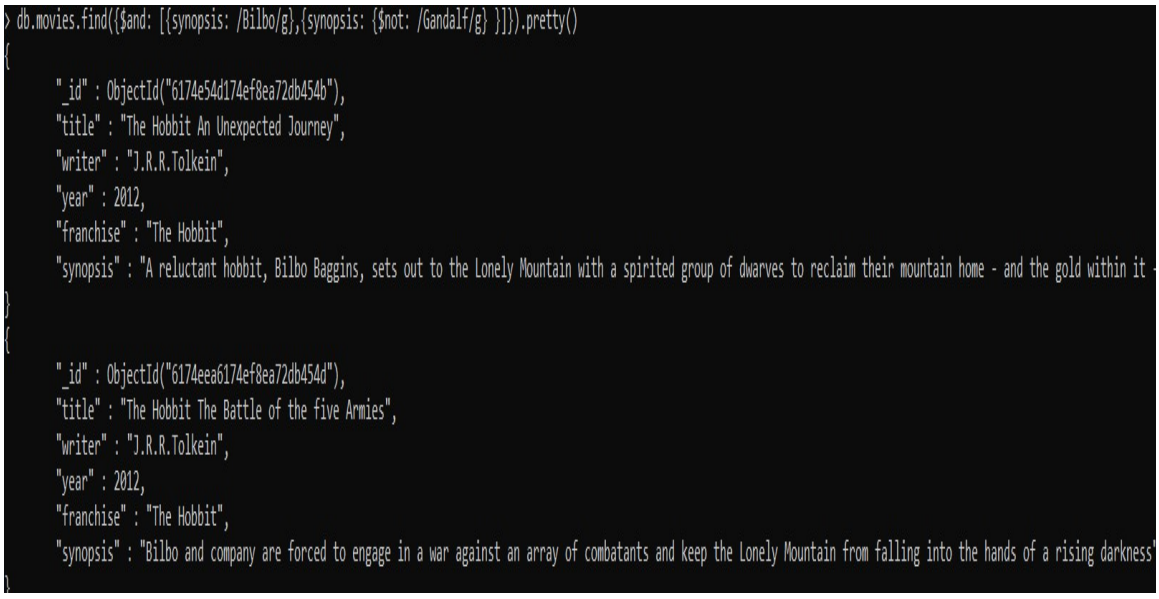
```
> db.movies.find({synopsis: /Bilbo/g}).pretty()
{
        "_id" : ObjectId("6174e54d174ef8ea72db454b"),
        "title" : "The Hobbit An Unexpected Journey",
        "writer" : "J.R.R.Tolkein",
        "year" : 2012,
        "franchise" : "The Hobbit",
        "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from
{
        "_id" : ObjectId("6174e54d174ef8ea72db454c"),
        "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 homeland, from Smaug. Bilbo Baggins is in possession of a
g."
{
        "_id" : ObjectId("6174eea6174ef8ea72db454d"),
        "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"

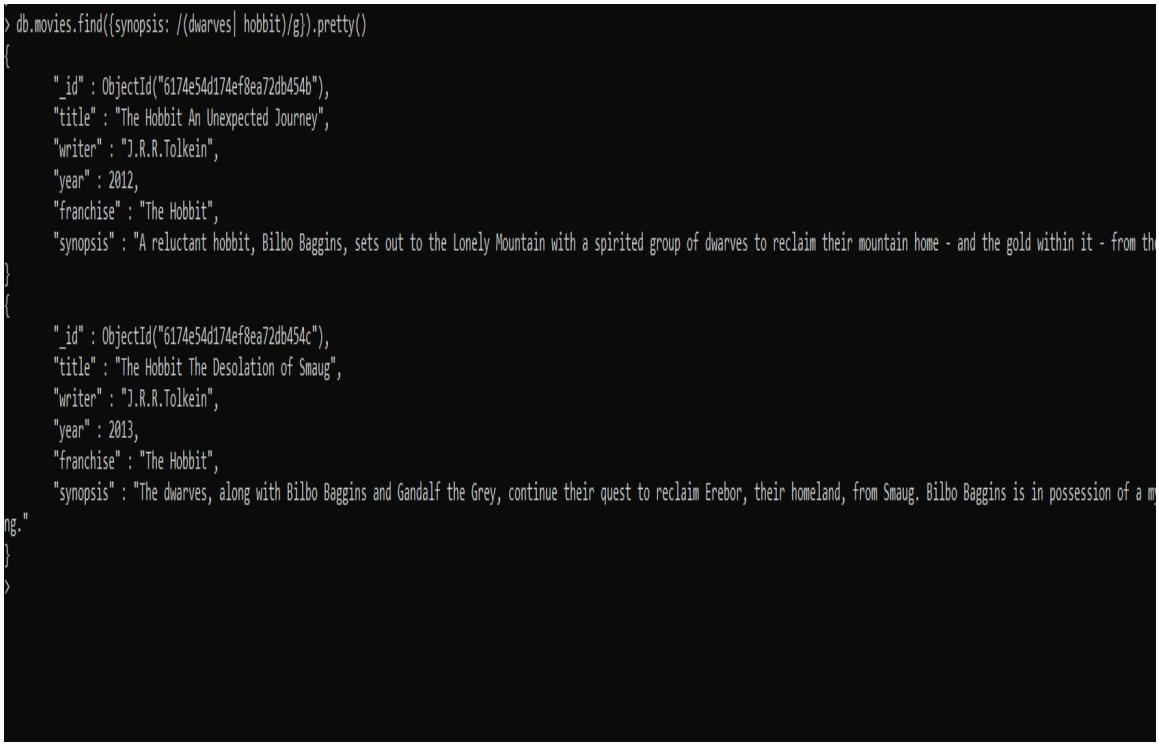


```
> db.movies.find({synopsis: /Gandalf/g}).pretty()
{
        "_id" : ObjectId("6174e54d174ef8ea72db454c"),
        "title" : "The Hobbit The Desolation of Smaug",
        "writer" : "J.R.R.Tolkein",
        "year" : 2013,
        "franchise" : "The Hobbit",
        "synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysteri
g."
}
```

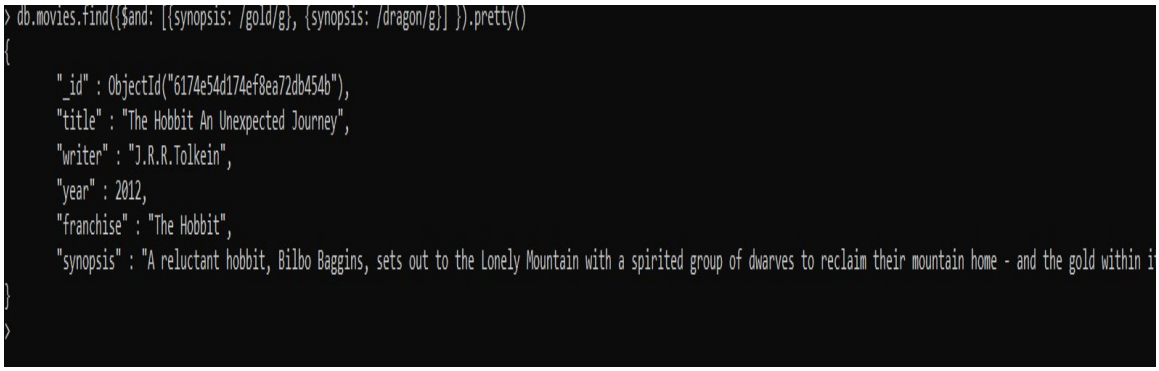
3.find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"




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

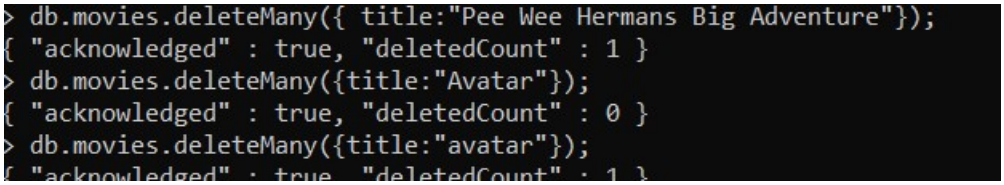



5.find all movies that have a synopsis that contains the word "gold" and "dragon"




Delete Documents

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




# Relationships:-

Insert the following documents into a users collection

```
> db.users.insert([{username: 'GoodGuyGreg',
... first_name:'Good Guy',
... last_name:'Greg'
... },
... {
... username: 'ScumbagSteve',
... full_name: {
... first :'Scumbag',
... last :'Steve'
... }
... }])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

Insert the following documents into a posts collection

```
> db.posts.insert([{username: 'GoodGuyGreg',
... title:'passes out at parrty',
... body:'wakes up early and cleans house'
... },
... {
... username:'GoodGuyGreg',
... title:'Steals your identity',
... body:'Raises your credit score'
... },
... {
... username:'ScumbagSteve',
... title:'Borrows something',
... body:'Sells it'
... },
... {
... username:'ScumbagSteve',
... title:'Borrows everything',
... body:'The end'
... },
... {
... username:'ScumbagSteve',
... title:'Forks you repo on github',
... body: 'Sets to private'
... }])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Insert the following documents into a comments collection

```
> db.comments.insert([{username:'GoodGuyGreg',
... comment:'Hope you got a good deal!',
... post:ObjectId('61767817b71c61ae02743ec7')
... },
... {
... username:'GoodGuyGreg',
... comment:'Don violat the licensing agreement!',
... post:ObjectId('61767817b71c61ae02743ec8')
...
... }])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
 db.comments.insert([{username:'GoodGuyGreg',
.. comment:'Whats mine is yours!',
.. post:ObjectId('61767817b71c61ae02743ec8')
.. },
.. {username:'ScumbagSteve',
.. comment:'It still isn't clean',
ncaught exception: SyntaxError: missing } after property list :
(shell):6:22
 db.comments.insert([{username:'GoodGuyGreg',
.. comment:'Whats mine is yours!',
.. post:ObjectId('61767817b71c61ae02743ec8')
.. },
.. {
.. username:'ScumbagSteve',
.. comment:'It still isnt clean',
.. post:ObjectId('61767817b71c61ae02743ec5')
.. },
.. {
.. username:'ScumbagSteve',
.. comment:'Denied your PR cause I found a hack',
.. post:ObjectId('61767817b71c61ae02743ec9')
.. }])
```

Querying related collections

1. find all users

```
> db.users.find().pretty()
{
        "_id" : ObjectId("617670bdb71c61ae02743ec3"),
        "username" : "GoodGuyGreg",
        "first_name" : "Good Guy",
        "last_name" : "Greg"
}
{
        "_id" : ObjectId("617670bdb71c61ae02743ec4"),
        "username" : "ScumbagSteve",
        "full_name" : {
                "first" : "Scumbag",
                "last" : "Steve"
        }
```

2.Find all posts

```
> db.posts.find().pretty()
{
        "_id" : ObjectId("61767817b71c61ae02743ec5"),
        "username" : "GoodGuyGreg",
        "title" : "passes out at parrty",
        "body" : "wakes up early and cleans house"
}
{
        "_id" : ObjectId("61767817b71c61ae02743ec6"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("61767817b71c61ae02743ec7"),
        "username" : "ScumbagSteve",
        "title" : "Borrows something",
        "body" : "Sells it"
}
{
        "_id" : ObjectId("61767817b71c61ae02743ec8"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
        "body" : "The end"
}
{
        "_id" : ObjectId("61767817b71c61ae02743ec9"),
        "username" : "ScumbagSteve",
        "title" : "Forks you repo on github",
        "body" : "Sets to private"
}
{
        "_id" : ObjectId("617694c2b71c61ae02743ecf"),
        "username" : "GoodGuyGreg",
        "title" : "Report a bug in your code",
        "body" : "Sends me a pull Request"
}
```

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

```
> db.posts.find({username:'GoodGuyGreg'}).pretty()
{
        "_id" : ObjectId("61767817b71c61ae02743ec5"),
        "username" : "GoodGuyGreg",
        "title" : "passes out at parrty",
        "body" : "wakes up early and cleans house"
}
{
        "_id" : ObjectId("61767817b71c61ae02743ec6"),
        "username" : "GoodGuyGreg",
        "title" : "Steals your identity",
        "body" : "Raises your credit score"
}
{
        "_id" : ObjectId("617694c2b71c61ae02743ecf"),
        "username" : "GoodGuyGreg",
        "title" : "Report a bug in your code",
        "body" : "Sends me a pull Request"
}
```

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

```
> db.posts.find({username: 'ScumbagSteve'}).pretty()
{
        "_id" : ObjectId("61767817b71c61ae02743ec7"),
        "username" : "ScumbagSteve",
        "title" : "Borrows something",
        "body" : "Sells it"
}
{
        "_id" : ObjectId("61767817b71c61ae02743ec8"),
        "username" : "ScumbagSteve",
        "title" : "Borrows everything",
        "body" : "The end"
}
{
        "_id" : ObjectId("61767817b71c61ae02743ec9"),
        "username" : "ScumbagSteve",
        "title" : "Forks you repo on github",
        "body" : "Sets to private"
}
```

5.find all comments


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

```
> db.comments.find({username:'GoodGuyGreg'}).pretty()
{
        "_id" : ObjectId("617689deb71c61ae02743eca"),
        "username" : "GoodGuyGreg",
        "comment" : "Hope you got a good deal!",
        "post" : ObjectId("61767817b71c61ae02743ec7")
}
{
        "_id" : ObjectId("617689deb71c61ae02743ecb"),
        "username" : "GoodGuyGreg",
        "comment" : "Don violat the licensing agreement!",
        "post" : ObjectId("61767817b71c61ae02743ec9")
}
{
        "_id" : ObjectId("61769291b71c61ae02743ecc"),
        "username" : "GoodGuyGreg",
        "comment" : "Whats mine is yours!",
        "post" : ObjectId("61767817b71c61ae02743ec8")
}
```

7.find all comments that was authored by "ScumbagSteve

```
> db.comments.find({username:'ScumbagSteve'}).pretty()
{
        "_id" : ObjectId("61769291b71c61ae02743ecd"),
        "username" : "ScumbagSteve",
        "comment" : "It still isnt clean",
        "post" : ObjectId("61767817b71c61ae02743ec5")
}
{
        "_id" : ObjectId("61769291b71c61ae02743ece"),
        "username" : "ScumbagSteve",
        "comment" : "Denied your PR cause I found a hack",
        "post" : ObjectId("617694e6f3f5cb33f9caa96d")
}
```

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

```
db.posts.aggregate([
.. {
.. $match: { title: 'Report a bug in your code' }
.. },
.. {
.. $lookup: {
.. from: 'comments',
.. localField: '_id',
.. foreignField: 'post',
.. as: 'comments'
.. }
.. }
.. ]).pretty();

        "_id" : ObjectId("617694c2b71c61ae02743ecf"),
        "username" : "GoodGuyGreg",
        "title" : "Report a bug in your code",
        "body" : "Sends me a pull Request",
        "comments" : [ ]
```

ASSIGNMENT-2

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"}]})
"_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" : "30308", "city" : "ATLANTA", "loc" : [ -84.375744, 33.771839 ], "pop" : 8549, "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" : "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" }
ype "it" for more
```

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

```
> db.zipcodes.aggregate({$match:{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" : "30308", "city" : "ATLANTA", "loc" : [ -84.375744, 33.771839 ], "pop" : 8549, "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" : "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" }
ype "it" for more
>
```

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

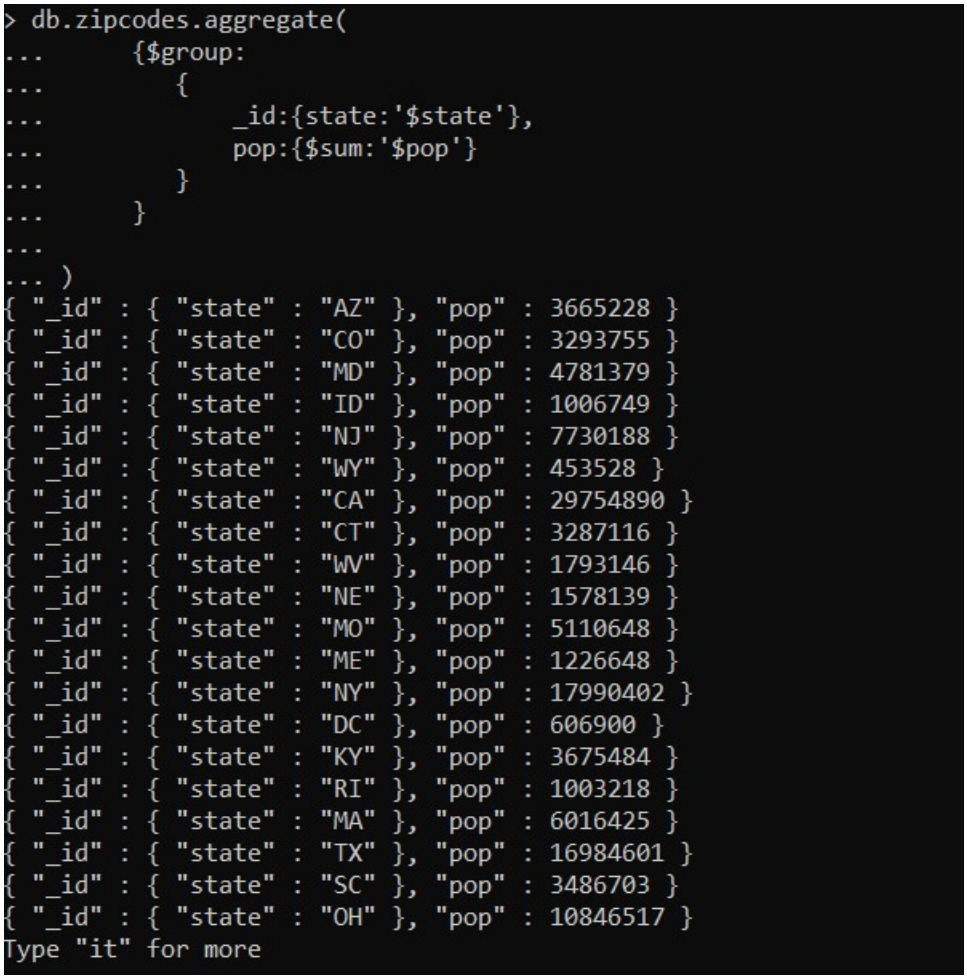
```
> db.zipcodes.aggregate({$match:{city:"ATLANTA"}
...
... },
... {$group:{_id:"$city",
... zipcodes:{$sum:1}}})
{ "_id" : "ATLANTA", "zipcodes" : 41 }
>
```

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

```
db.zipcodes.aggregate([{$group:{_id:'$ATLANTA',count:{$sum:"$pop"}}}])
"_id" : null, "count" : 248408400 }
```

Populations By State

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




2. sort the results by population, highest first

```
> db.zipcodes.aggregate(
...       {$group:
...          {
...              _id:{state:'$state'},
...              pop:{$sum:'$pop'}
...          }
...       },
...       {$sort:
...          {population:-1}
...       }
...
... )
{ "_id" : { "state" : "AZ" }, "pop" : 3665228 }
{ "_id" : { "state" : "CO" }, "pop" : 3293755 }
{ "_id" : { "state" : "MD" }, "pop" : 4781379 }
{ "_id" : { "state" : "ID" }, "pop" : 1006749 }
{ "_id" : { "state" : "NJ" }, "pop" : 7730188 }
{ "_id" : { "state" : "WY" }, "pop" : 453528 }
{ "_id" : { "state" : "CA" }, "pop" : 29754890 }
{ "_id" : { "state" : "CT" }, "pop" : 3287116 }
{ "_id" : { "state" : "WV" }, "pop" : 1793146 }
{ "_id" : { "state" : "NE" }, "pop" : 1578139 }
{ "_id" : { "state" : "MO" }, "pop" : 5110648 }
{ "_id" : { "state" : "ME" }, "pop" : 1226648 }
{ "_id" : { "state" : "NY" }, "pop" : 17990402 }
{ "_id" : { "state" : "DC" }, "pop" : 606900 }
{ "_id" : { "state" : "KY" }, "pop" : 3675484 }
{ "_id" : { "state" : "RI" }, "pop" : 1003218 }
{ "_id" : { "state" : "MA" }, "pop" : 6016425 }
{ "_id" : { "state" : "TX" }, "pop" : 16984601 }
{ "_id" : { "state" : "SC" }, "pop" : 3486703 }
{ "_id" : { "state" : "OH" }, "pop" : 10846517 }
```

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

```
db.zipcodes.aggregate([{$group:{"_id":"$state",state:{$max:"$pop"}}},{$sort:{state:-1}},{$limit:3}])
"_id" : "IL", "state" : 112047 }
"_id" : "NY", "state" : 111396 }
"_id" : "CA", "state" : 99568 }
```

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:'$state'},
...              population:{$sum:'$pop'}
...          }
...       }
... )
{ "_id" : { "city" : "WILLISTON", "state" : "VT" }, "population" : 5592 }
{ "_id" : { "city" : "GAY", "state" : "WV" }, "population" : 1037 }
{ "_id" : { "city" : "EUREKA", "state" : "NC" }, "population" : 6803 }
{ "_id" : { "city" : "HAMPSTEAD", "state" : "NC" }, "population" : 8159 }
{ "_id" : { "city" : "MOBRIDGE", "state" : "SD" }, "population" : 4099 }
{ "_id" : { "city" : "HIGH RIDGE", "state" : "MO" }, "population" : 12915 }
{ "_id" : { "city" : "HUXLEY", "state" : "IA" }, "population" : 2417 }
{ "_id" : { "city" : "NORMANDY", "state" : "MO" }, "population" : 31649 }
{ "_id" : { "city" : "OVAPA", "state" : "WV" }, "population" : 525 }
{ "_id" : { "city" : "WHEATON", "state" : "MO" }, "population" : 970 }
{ "_id" : { "city" : "LOS OSOS", "state" : "CA" }, "population" : 14648 }
{ "_id" : { "city" : "WEST LOGAN", "state" : "WV" }, "population" : 14143 }
{ "_id" : { "city" : "OGDEN", "state" : "NC" }, "population" : 26744 }
{ "_id" : { "city" : "MONTESANO", "state" : "WA" }, "population" : 10079 }
{ "_id" : { "city" : "DULCE", "state" : "NM" }, "population" : 2936 }
{ "_id" : { "city" : "GRAYS RIVER", "state" : "WA" }, "population" : 209 }
{ "_id" : { "city" : "EXETER", "state" : "RI" }, "population" : 3774 }
{ "_id" : { "city" : "NORTHROP", "state" : "MN" }, "population" : 724 }
{ "_id" : { "city" : "WINDSOR", "state" : "WI" }, "population" : 1825 }
{ "_id" : { "city" : "ASTORIA", "state" : "IL" }, "population" : 2093 }
Type "it" for more
```

2. sort the results by population, highest first

```
> db.zipcodes.aggregate(
...       {$group:
...          {
...              _id:{city:'$city',state:'$state'},
...              population:{$sum:'$pop'}
...          }
...       },
...       {$sort:
...          {population:-1}
...       }
... )
{ "_id" : { "city" : "CHICAGO", "state" : "IL" }, "population" : 2452177 }
{ "_id" : { "city" : "BROOKLYN", "state" : "NY" }, "population" : 2300504 }
{ "_id" : { "city" : "LOS ANGELES", "state" : "CA" }, "population" : 2102295 }
{ "_id" : { "city" : "HOUSTON", "state" : "TX" }, "population" : 2095918 }
{ "_id" : { "city" : "PHILADELPHIA", "state" : "PA" }, "population" : 1610956 }
{ "_id" : { "city" : "NEW YORK", "state" : "NY" }, "population" : 1476790 }
{ "_id" : { "city" : "BRONX", "state" : "NY" }, "population" : 1209548 }
{ "_id" : { "city" : "SAN DIEGO", "state" : "CA" }, "population" : 1049298 }
{ "_id" : { "city" : "DETROIT", "state" : "MI" }, "population" : 963243 }
{ "_id" : { "city" : "DALLAS", "state" : "TX" }, "population" : 940191 }
{ "_id" : { "city" : "PHOENIX", "state" : "AZ" }, "population" : 890853 }
{ "_id" : { "city" : "MIAMI", "state" : "FL" }, "population" : 825232 }
{ "_id" : { "city" : "SAN JOSE", "state" : "CA" }, "population" : 816653 }
{ "_id" : { "city" : "SAN ANTONIO", "state" : "TX" }, "population" : 811792 }
{ "_id" : { "city" : "BALTIMORE", "state" : "MD" }, "population" : 733081 }
{ "_id" : { "city" : "SAN FRANCISCO", "state" : "CA" }, "population" : 723993 }
{ "_id" : { "city" : "MEMPHIS", "state" : "TN" }, "population" : 632837 }
{ "_id" : { "city" : "SACRAMENTO", "state" : "CA" }, "population" : 628279 }
{ "_id" : { "city" : "JACKSONVILLE", "state" : "FL" }, "population" : 610160 }
{ "_id" : { "city" : "ATLANTA", "state" : "GA" }, "population" : 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:{$max:"$pop"}}},{$sort:{city:-1}},{$limit: 3}])
{ "_id" : "CHICAGO", "city" : 112047 }
{ "_id" : "BROOKLYN", "city" : 111396 }
{ "_id" : "NEW YORK", "city" : 106564 }
>
```

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

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

BONUS

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

```
db.zipcodes.aggregate({$group:{_id:{city:'$city', state:'$state'},population:{$avg:'$pop'}}},{$sort:{population:-1}})
{ "_id" : { "city" : "BELL GARDENS", "state" : "CA" }, "population" : 99568 }
{ "_id" : { "city" : "NORWALK", "state" : "CA" }, "population" : 94188 }
{ "_id" : { "city" : "ARLETA", "state" : "CA" }, "population" : 88114 }
{ "_id" : { "city" : "SOUTH GATE", "state" : "CA" }, "population" : 87026 }
{ "_id" : { "city" : "RIDGEWOOD", "state" : "NY" }, "population" : 85732 }
{ "_id" : { "city" : "WESTLAND", "state" : "MI" }, "population" : 84712 }
{ "_id" : { "city" : "HOLLY PARK", "state" : "CA" }, "population" : 78511 }
{ "_id" : { "city" : "WESTMINSTER", "state" : "CA" }, "population" : 77965 }
{ "_id" : { "city" : "INDUSTRY", "state" : "CA" }, "population" : 77114 }
{ "_id" : { "city" : "COAST GUARD ISLA", "state" : "CA" }, "population" : 76110 }
{ "_id" : { "city" : "GWYNN OAK", "state" : "MD" }, "population" : 76002 }
{ "_id" : { "city" : "WOODSIDE", "state" : "NY" }, "population" : 75894 }
{ "_id" : { "city" : "CORONA", "state" : "NY" }, "population" : 75746 }
{ "_id" : { "city" : "GLEN BURNIE", "state" : "MD" }, "population" : 75692 }
{ "_id" : { "city" : "RIALTO", "state" : "CA" }, "population" : 75341 }
{ "_id" : { "city" : "JACKSON HEIGHTS", "state" : "NY" }, "population" : 72983.5 }
{ "_id" : { "city" : "HUNTINGTON PARK", "state" : "CA" }, "population" : 72139 }
{ "_id" : { "city" : "LAKE LOS ANGELES", "state" : "CA" }, "population" : 71024 }
{ "_id" : { "city" : "TAYLOR", "state" : "MI" }, "population" : 70811 }
{ "_id" : { "city" : "STAR CITY", "state" : "WV" }, "population" : 70185 }
```

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

```
db.zipcodes.aggregate([{$group:{"_id": "$state", state:{$avg:"$pop"}}},{$sort:{state:-1}},{$limit: 3}])
{ "_id" : "DC", "state" : 25287.5 }
{ "_id" : "CA", "state" : 19627.236147757256 }
{ "_id" : "FL", "state" : 15779.407960199005 }
```

ASSIGNMENT-3

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


2. Write a MongoDB query to display the fields restaurant_id, name, borough


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 collectio

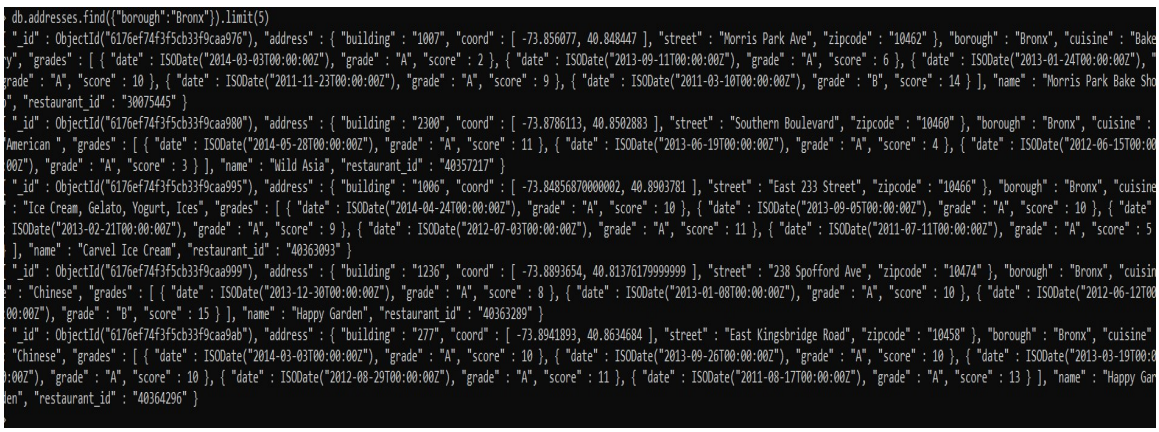
```
> db.addresses.find({}, {restaurant_id:1, _id:0, name:1,borough:1, cuisine:1 })
{ "borough" : "Bronx", "cuisine" : "Bakery", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "borough" : "Manhattan", "cuisine" : "Irish", "name" : "Dj Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{ "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", "cuisine" : "Jewish/Kosher", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{ "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{ "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "borough" : "Bronx", "cuisine" : "American ", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "borough" : "Brooklyn", "cuisine" : "American ", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{ "borough" : "Brooklyn", "cuisine" : "Chinese", "name" : "May May Kitchen", "restaurant_id" : "40358429" }
{ "borough" : "Manhattan", "cuisine" : "American ", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{ "borough" : "Brooklyn", "cuisine" : "Jewish/Kosher", "name" : "Seuda Foods", "restaurant_id" : "40360045" }
{ "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "borough" : "Queens", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "borough" : "Manhattan", "cuisine" : "American ", "name" : "Glorious Food", "restaurant_id" : "40361521" }
{ "borough" : "Brooklyn", "cuisine" : "American ", "name" : "The Movable Feast", "restaurant_id" : "40361606" }
```

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

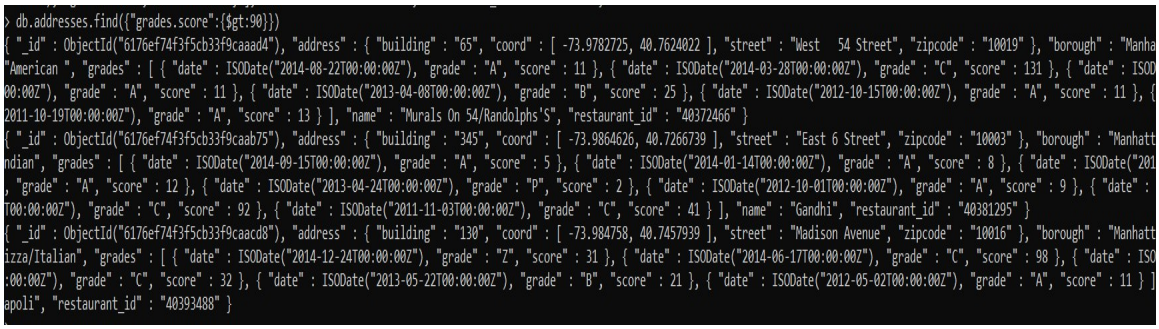
```
> db.addresses.find({}, {restaurant_id:1, _id:0, name:1,borough:1, "address.zipcode":1 })
{ "address" : { "zipcode" : "10462" }, "borough" : "Bronx", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "address" : { "zipcode" : "11225" }, "borough" : "Brooklyn", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "address" : { "zipcode" : "10019" }, "borough" : "Manhattan", "name" : "Dj Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{ "address" : { "zipcode" : "11224" }, "borough" : "Brooklyn", "name" : "Riviera Caterer", "restaurant_id" : "40356018" }
{ "address" : { "zipcode" : "11374" }, "borough" : "Queens", "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{ "address" : { "zipcode" : "11369" }, "borough" : "Queens", "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151" }
{ "address" : { "zipcode" : "10314" }, "borough" : "Staten Island", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{ "address" : { "zipcode" : "11234" }, "borough" : "Brooklyn", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "address" : { "zipcode" : "11219" }, "borough" : "Brooklyn", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{ "address" : { "zipcode" : "11226" }, "borough" : "Brooklyn", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "address" : { "zipcode" : "10460" }, "borough" : "Bronx", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "address" : { "zipcode" : "11214" }, "borough" : "Brooklyn", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{ "address" : { "zipcode" : "11208" }, "borough" : "Brooklyn", "name" : "May May Kitchen", "restaurant_id" : "40358429" }
{ "address" : { "zipcode" : "10065" }, "borough" : "Manhattan", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{ "address" : { "zipcode" : "11223" }, "borough" : "Brooklyn", "name" : "Seuda Foods", "restaurant_id" : "40360045" }
{ "address" : { "zipcode" : "11218" }, "borough" : "Brooklyn", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "address" : { "zipcode" : "11004" }, "borough" : "Queens", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "address" : { "zipcode" : "11209" }, "borough" : "Brooklyn", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "address" : { "zipcode" : "10021" }, "borough" : "Manhattan", "name" : "Glorious Food", "restaurant_id" : "40361521" }
{ "address" : { "zipcode" : "11215" }, "borough" : "Brooklyn", "name" : "The Movable Feast", "restaurant_id" : "40361606" }
```

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


6Write 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

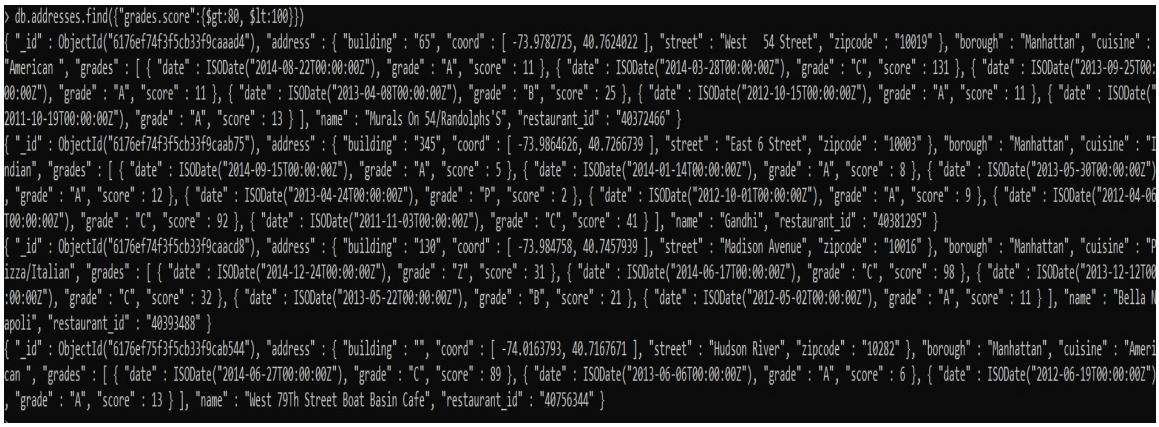



8.Write a MongoDB query to find the restaurants who achieved a score more

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.


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(
                            {
                              "cuisine" : {$ne : "American "},
                              "grades.score" :{$gt: 70},
                              "address.coord" : {$lt : -65.754168}
                             }
                      );
{ _id: ObjectId("6176ef74f3f5cb33f9caab75"),
  address:
   { building: '345',
     coord: [ -73.9864626, 40.7266739 ],
     street: 'East 6 Street',
     zipcode: '10003' },
  borough: 'Manhattan',
  cuisine: 'Indian',
  grades:
   [ { date: 2014-09-15T00:00:00.000Z, grade: 'A', score: 5 },
     { date: 2014-01-14T00:00:00.000Z, grade: 'A', score: 8 },
     { date: 2013-05-30T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2013-04-24T00:00:00.000Z, grade: 'P', score: 2 },
     { date: 2012-10-01T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2012-04-06T00:00:00.000Z, grade: 'C', score: 92 },
     { date: 2011-11-03T00:00:00.000Z, grade: 'C', score: 41 } ],
  name: 'Gandhi',
  restaurant_id: '40381295' }
{ _id: ObjectId("6176ef74f3f5cb33f9caacd8"),
  address:
   { building: '130',
     coord: [ -73.984758, 40.7457939 ],
```

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

```
db.addresses.find({$and : [{"cuisine" : {$ne : "American "}}, {"grades.grade" : "A"}, {"borough" : {$ne : "Brooklyn "}}]}).sort({cuisine : -1})
{ _id: ObjectId("6176ef74f3f5cb33f9cab082"),
  address:
   { building: '89',
     coord: [ -73.9995899, 40.7168015 ],
     street: 'Baxter Street',
     zipcode: '10013' },
  borough: 'Manhattan',
  cuisine: 'Vietnamese/Cambodian/Malaysia',
  grades:
   [ { date: 2014-08-21T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-08-31T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-04-11T00:00:00.000Z, grade: 'C', score: 3 },
     { date: 2012-10-17T00:00:00.000Z, grade: 'A', score: 4 },
     { date: 2012-05-15T00:00:00.000Z, grade: 'A', score: 10 } ],
  name: 'Thai Son',
  restaurant_id: '40559606' }
{ _id: ObjectId("6176ef74f3f5cb33f9cab13b"),
  address:
   { building: '8278',
     coord: [ -73.88143509999999, 40.7412552 ],
     street: 'Broadway',
     zipcode: '11373' },
  borough: 'Queens',
  cuisine: 'Vietnamese/Cambodian/Malaysia',
```

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: /^Wil/},
  {
  "restaurant_id" : 1,
  "name":1,"borough":1,
  "cuisine" :1
  }
  );
< { _id: ObjectId("6176ef74f3f5cb33f9caa97d"),
    borough: 'Brooklyn',
    cuisine: 'Delicatessen',
    name: 'Wilken\'S Fine Food',
    restaurant_id: '40356483' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa980"),
    borough: 'Bronx',
    cuisine: 'American ',
    name: 'Wild Asia',
    restaurant_id: '40357217' }
  { _id: ObjectId("6176ef75f3f5cb33f9cab785"),
    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.

```
> db.addresses.find(
  {name: /ces$/},
  {
  "restaurant_id" : 1,
  "name":1,"borough":1,
  "cuisine" :1
  }
  )
< { _id: ObjectId("6176ef74f3f5cb33f9caae09"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Pieces',
    restaurant_id: '40399910' }
  { _id: ObjectId("6176ef74f3f5cb33f9caaec8"),
    borough: 'Queens',
    cuisine: 'American ',
    name: 'S.M.R Restaurant Services',
    restaurant_id: '40403857' }
  { _id: ObjectId("6176ef74f3f5cb33f9caaece"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Good Shepherd Services',
    restaurant_id: '40403989' }
  { _id: ObjectId("6176ef74f3f5cb33f9cab381"),
    borough: 'Queens',
    cuisine: 'Ice Cream, Gelato, Yogurt, Ices',
    name: 'The Ice Box-Ralph\'S Famous Italian Ices',
    restaurant_id: '40690899' }
  { _id: ObjectId("6176ef75f3f5cb33f9cab583"),
```

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.

```
> db.addresses.find(
  {"name": /.*Reg.*/},
  {
  "restaurant_id" : 1,
  "name":1,"borough":1,
  "cuisine" :1
  }
  );
< { _id: ObjectId("6176ef74f3f5cb33f9caa97e"),
    borough: 'Brooklyn',
    cuisine: 'American ',
    name: 'Regina Caterers',
    restaurant_id: '40356649' }
  { _id: ObjectId("6176ef74f3f5cb33f9caaa7b"),
    borough: 'Manhattan',
    cuisine: 'Café/Coffee/Tea',
    name: 'Caffe Reggio',
    restaurant_id: '40369418' }
  { _id: ObjectId("6176ef74f3f5cb33f9caab8a"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Regency Hotel',
    restaurant_id: '40382679' }
  { _id: ObjectId("6176ef74f3f5cb33f9caaea7"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Regency Whist Club',
    restaurant_id: '40402377' }
  { _id: ObjectId("6176ef74f3f5cb33f9caaf8a"),
```

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

```
> db.addresses.find(
  {
  "borough": "Bronx" ,
  $or : [
  { "cuisine" : "American " },
  { "cuisine" : "Chinese" }
  ]
  }
  )
< { _id: ObjectId("6176ef74f3f5cb33f9caa980"),
    address:
     { building: '2300',
       coord: [ -73.8786113, 40.8502883 ],
       street: 'Southern Boulevard',
       zipcode: '10460' },
    borough: 'Bronx',
    cuisine: 'American ',
    grades:
     [ { date: 2014-05-28T00:00:00.000Z, grade: 'A', score: 11 },
       { date: 2013-06-19T00:00:00.000Z, grade: 'A', score: 4 },
       { date: 2012-06-15T00:00:00.000Z, grade: 'A', score: 3 } ],
    name: 'Wild Asia',
    restaurant_id: '40357217' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa999"),
    address:
     { building: '1236',
       coord: [ -73.8893654, 40.81376179999999 ],
       street: '238 Spofford Ave',
```

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 Bronxor Brooklyn.

```
Type "it" for more
> db.addresses.find(
{"borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"]}},
{
"restaurant_id" : 1,
"name":1,"borough":1,
"cuisine" :1
}
)
< { _id: ObjectId("6176ef74f3f5cb33f9caa976"),
  borough: 'Bronx',
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa977"),
  borough: 'Brooklyn',
  cuisine: 'Hamburgers',
  name: 'Wendy\'S',
  restaurant_id: '30112340' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa979"),
  borough: 'Brooklyn',
  cuisine: 'American ',
  name: 'Riviera Caterer',
  restaurant_id: '40356018' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa97a"),
  borough: 'Queens',
```

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.

```
> db.addresses.find(
  {"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},
  {
  "restaurant_id" : 1,
  "name":1,"borough":1,
  "cuisine" :1
  }
  )
< { _id: ObjectId("6176ef74f3f5cb33f9caa978"),
    borough: 'Manhattan',
    cuisine: 'Irish',
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa983"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: '1 East 66Th Street Kitchen',
    restaurant_id: '40359480' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa988"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Glorious Food',
    restaurant_id: '40361521' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa98b"),
    borough: 'Manhattan',
    cuisine: 'Delicatessen',
    name: 'Bully\'S Deli',
    restaurant_id: '40361708' }
```

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

```
> db.addresses.find(
  {"grades.score" :
  { $not:
  {$gt : 10}
  }
  },
  {
  "restaurant_id" : 1,
  "name":1,"borough":1,
  "cuisine" :1
  }
  );
< { _id: ObjectId("6176ef74f3f5cb33f9caa981"),
    borough: 'Brooklyn',
    cuisine: 'American ',
    name: 'C & C Catering Service',
    restaurant_id: '40357437' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa983"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: '1 East 66Th Street Kitchen',
    restaurant_id: '40359480' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa987"),
    borough: 'Brooklyn',
    cuisine: 'Delicatessen',
    name: 'Nordic Delicacies',
    restaurant_id: '40361390' }
```

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'.

```
db.addresses.find(
{$or: [
  {name: /^Wil/},
  {"$and": [
       {"cuisine" : {$ne :"American "}},
       {"cuisine" : {$ne :"Chinees"}}
   ]}
]}
,{"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
);
{ _id: ObjectId("6176ef74f3f5cb33f9caa976"),
  borough: 'Bronx',
  cuisine: 'Bakery',
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa977"),
  borough: 'Brooklyn',
  cuisine: 'Hamburgers',
  name: 'Wendy\'S',
  restaurant_id: '30112340' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa978"),
  borough: 'Manhattan',
  cuisine: 'Irish',
  name: 'Dj Reynolds Pub And Restaurant',
  restaurant_id: '30191841' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa97a"),
  borough: 'Queens',
  cuisine: 'Jewish/Kosher',
  name: 'Tov Kosher Kitchen',
```

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..

```
> db.addresses.find(
                {
                 "grades.date": ISODate("2014-08-11T00:00:00Z"),
                 "grades.grade":"A" ,
                 "grades.score" : 11
                },
                {"restaurant_id" : 1,"name":1,"grades":1}
            );
{ _id: ObjectId("6176ef74f3f5cb33f9caa9f4"),
  grades:
   [ { date: 2014-08-11T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-07-22T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2013-03-14T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-07-02T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2012-02-02T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2011-08-24T00:00:00.000Z, grade: 'A', score: 11 } ],
  name: 'Neary\'S Pub',
  restaurant_id: '40365871' }
{ _id: ObjectId("6176ef74f3f5cb33f9caaacf"),
  grades:
   [ { date: 2014-08-11T00:00:00.000Z, grade: 'A', score: 11 },
     { date: 2013-12-10T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2013-06-10T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2012-06-08T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2012-01-25T00:00:00.000Z, grade: 'A', score: 8 },
     { date: 2011-09-13T00:00:00.000Z, grade: 'A', score: 12 } ],
```

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"

```
> db.addresses.find(
                        { "grades.1.date": ISODate("2014-08-11T00:00:00Z"),
                          "grades.1.grade":"A" ,
                          "grades.1.score" : 9
                        },
                         {"restaurant_id" : 1,"name":1,"grades":1}
                    );
{ _id: ObjectId("6176ef74f3f5cb33f9caafa1"),
  grades:
   [ { date: 2015-01-12T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2014-08-11T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2014-01-14T00:00:00.000Z, grade: 'A', score: 13 },
     { date: 2013-02-07T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2012-04-30T00:00:00.000Z, grade: 'A', score: 11 } ],
  name: 'Club Macanudo (Cigar Bar)',
  restaurant_id: '40526406' }
```

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.

```
> db.addresses.find(
                    {
                      "address.coord.1": {$gt : 42, $lte : 52}
                    },
                      {"restaurant_id" : 1,"name":1,"address":1,"coord":1}
                );
{ _id: ObjectId("6176ef74f3f5cb33f9caac18"),
  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("6176ef74f3f5cb33f9caac44"),
  address:
   { building: '1',
     coord: [ -0.7119979, 51.6514664 ],
     street: 'Pennplaza E, Penn Sta',
     zipcode: '10001' },
  name: 'T.G.I. Fridays',
  restaurant_id: '40388936' }
{ _id: ObjectId("6176ef74f3f5cb33f9caae9d"),
  address:
   { building: '3000',
```

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

```
> db.addresses.find().sort({"name":1});
{ _id: ObjectId("6176ef75f3f5cb33f9cab606"),
  address:
   { building: '129',
     coord: [ -73.962943, 40.685007 ],
     street: 'Gates Avenue',
     zipcode: '11238' },
  borough: 'Brooklyn',
  cuisine: 'Italian',
  grades:
   [ { date: 2014-03-06T00:00:00.000Z, grade: 'A', score: 5 },
     { date: 2013-08-29T00:00:00.000Z, grade: 'A', score: 2 },
     { date: 2013-03-08T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2012-06-27T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2011-11-17T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: '(Lewis Drug Store) Locanda Vini E Olii',
  restaurant_id: '40804423' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa983"),
  address:
   { building: '1',
     coord: [ -73.96926909999999, 40.7685235 ],
     street: 'East   66 Street',
     zipcode: '10065' },
  borough: 'Manhattan',
  cuisine: 'American ',
  grades:
```

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

```
> db.addresses.find().sort(
                            {"name":-1}
                            );
< { _id: ObjectId("6176ef74f3f5cb33f9caaa35"),
    address:
     { building: '6946',
       coord: [ -73.8811834, 40.7017759 ],
       street: 'Myrtle Avenue',
       zipcode: '11385' },
    borough: 'Queens',
    cuisine: 'German',
    grades:
     [ { date: 2014-09-24T00:00:00.000Z, grade: 'A', score: 11 },
       { date: 2014-04-17T00:00:00.000Z, grade: 'A', score: 7 },
       { date: 2013-03-12T00:00:00.000Z, grade: 'A', score: 13 },
       { date: 2012-10-02T00:00:00.000Z, grade: 'A', score: 9 },
       { date: 2012-05-09T00:00:00.000Z, grade: 'A', score: 13 },
       { date: 2011-12-28T00:00:00.000Z, grade: 'B', score: 24 } ],
    name: 'Zum Stammtisch',
    restaurant_id: '40367377' }
  { _id: ObjectId("6176ef75f3f5cb33f9cab56f"),
    address:
     { building: '107109',
       coord: [ -73.9744668, 40.731155 ],
```

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.

```
> db.addresses.find().sort(
                            {"cuisine":1,"borough" : -1,}
                           );
< { _id: ObjectId("6176ef74f3f5cb33f9cab061"),
    address:
     { building: '1345',
       coord: [ -73.959249, 40.768076 ],
       street: '2 Avenue',
       zipcode: '10021' },
    borough: 'Manhattan',
    cuisine: 'Afghan',
    grades:
     [ { date: 2014-10-07T00:00:00.000Z, grade: 'A', score: 9 },
       { date: 2013-10-23T00:00:00.000Z, grade: 'A', score: 8 },
       { date: 2012-10-26T00:00:00.000Z, grade: 'A', score: 13 },
       { date: 2012-04-26T00:00:00.000Z, grade: 'A', score: 7 },
       { date: 2012-01-12T00:00:00.000Z, grade: 'P', score: 10 } ],
    name: 'Afghan Kebab House',
    restaurant_id: '40552806' }
  { _id: ObjectId("6176ef74f3f5cb33f9cab18c"),
    address:
     { building: '34',
       coord: [ -73.9883612, 40.7286391 ],
       street: 'St Marks Place',
       zipcode: '10003' },
    borough: 'Manhattan',
    cuisine: 'Afghan',
    grades:
     [ { date: 2014-02-20T00:00:00.000Z, grade: 'A', score: 12 },
```

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

```
> db.addresses.find(
                      {"address.street" :
                          { $exists : true }
                      }
                     );
< { _id: ObjectId("6176ef74f3f5cb33f9caa976"),
    address:
     { building: '1007',
       coord: [ -73.856077, 40.848447 ],
       street: 'Morris Park Ave',
       zipcode: '10462' },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades:
     [ { date: 2014-03-03T00:00:00.000Z, grade: 'A', score: 2 },
       { date: 2013-09-11T00:00:00.000Z, grade: 'A', score: 6 },
       { date: 2013-01-24T00:00:00.000Z, grade: 'A', score: 10 },
       { date: 2011-11-23T00:00:00.000Z, grade: 'A', score: 9 },
       { date: 2011-03-10T00:00:00.000Z, grade: 'B', score: 14 } ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445' }
  { _id: ObjectId("6176ef74f3f5cb33f9caa977"),
    address:
     { building: '469',
       coord: [ -73.961704, 40.662942 ],
       street: 'Flatbush Avenue',
```

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

```
Type "it" for more
db.addresses.find(
                    {"address.coord" :
                       {$type : 1}
                    }
                   );
{ _id: ObjectId("6176ef74f3f5cb33f9caa976"),
  address:
   { building: '1007',
     coord: [ -73.856077, 40.848447 ],
     street: 'Morris Park Ave',
     zipcode: '10462' },
  borough: 'Bronx',
  cuisine: 'Bakery',
  grades:
   [ { date: 2014-03-03T00:00:00.000Z, grade: 'A', score: 2 },
     { date: 2013-09-11T00:00:00.000Z, grade: 'A', score: 6 },
     { date: 2013-01-24T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2011-11-23T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2011-03-10T00:00:00.000Z, grade: 'B', score: 14 } ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa977"),
  address:
```

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

```
db.addresses.find(
                    {"grades.score" :
                       {$mod : [7,0]}
                    },
                       {"restaurant_id" : 1,"name":1,"grades":1}
                  );
{ _id: ObjectId("6176ef74f3f5cb33f9caa976"),
  grades:
   [ { date: 2014-03-03T00:00:00.000Z, grade: 'A', score: 2 },
     { date: 2013-09-11T00:00:00.000Z, grade: 'A', score: 6 },
     { date: 2013-01-24T00:00:00.000Z, grade: 'A', score: 10 },
     { date: 2011-11-23T00:00:00.000Z, grade: 'A', score: 9 },
     { date: 2011-03-10T00:00:00.000Z, grade: 'B', score: 14 } ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa979"),
  grades:
   [ { date: 2014-06-10T00:00:00.000Z, grade: 'A', score: 5 },
     { date: 2013-06-05T00:00:00.000Z, grade: 'A', score: 7 },
     { date: 2012-04-13T00:00:00.000Z, grade: 'A', score: 12 },
     { date: 2011-10-12T00:00:00.000Z, grade: 'A', score: 12 } ],
  name: 'Riviera Caterer',
  restaurant_id: '40356018' }
{ _id: ObjectId("6176ef74f3f5cb33f9caa97b"),
  grades:
   [ { date: 2014-11-15T00:00:00.000Z, grade: 'Z', score: 38 },
     { date: 2014-05-02T00:00:00.000Z, grade: 'A', score: 10 },
```

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.

```
> db.addresses.find(
                    { name :
                      { $regex : "mon.*", $options: "i" }
                    },
                        {
                          "name":1,
                          "borough":1,
                          "address.coord":1,
                          "cuisine" :1
                         }
                    );
< { _id: ObjectId("6176ef74f3f5cb33f9caaa0a"),
    address: { coord: [ -73.98306099999999, 40.7441419 ] },
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Desmond\'S Tavern' }
  { _id: ObjectId("6176ef74f3f5cb33f9caaa13"),
    address: { coord: [ -73.8221418, 40.7272376 ] },
    borough: 'Queens',
    cuisine: 'Jewish/Kosher',
    name: 'Shimons Kosher Pizza' }
  { _id: ObjectId("6176ef74f3f5cb33f9caaa1f"),
    address: { coord: [ -74.10465599999999, 40.58834 ] },
    borough: 'Staten Island',
    cuisine: 'American ',
```

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/i, }
                    },
                        {
                          "name":1,
                          "borough":1,
                          "address.coord":1,
                          "cuisine" :1
                         }
                    );
{ _id: ObjectId("6176ef74f3f5cb33f9caaeb2"),
  address: { coord: [ -73.9860597, 40.7431194 ] },
  borough: 'Manhattan',
  cuisine: 'American ',
  name: 'Madison Square' }
{ _id: ObjectId("6176ef74f3f5cb33f9caaf80"),
  address: { coord: [ -73.98302199999999, 40.742313 ] },
  borough: 'Manhattan',
  cuisine: 'Indian',
  name: 'Madras Mahal' }
{ _id: ObjectId("6176ef74f3f5cb33f9cab22e"),
  address: { coord: [ -74.000002, 40.72735 ] },
  borough: 'Manhattan',
  cuisine: 'American ',
  name: 'Madame X' }
```