

-:MONGODB:-

Assignment-1

ASHRAF NOMANI

Use mongo_practice

```
db.createCollection('movies')
```

Insert Documents

```
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. Tolkien',
```

```
    'year': 2012,
```

```
    'franchise': 'The Hobbit'
```

```
  },
```

```
  { 'title': 'The Hobbit: The Desolation of Smaug',
```

```
    'writer': 'J.R.R. Tolkien',
```

```
    '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'
}})

```

Query/Find Documents

1. db.movies.find()

```

> db.movies.find()
{ "_id" : ObjectId("617e84e9eb452ed295e878a7"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("617e850eb452ed295e878a8"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : 1994, "actors" : [ "John Travolta", "Uma Thurman" ] }
{ "_id" : ObjectId("617e8510eb452ed295e878a9"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : 2009, "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
{ "_id" : ObjectId("617e851eeb452ed295e878aa"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "year" : 2012, "franchise" : "The Hobbit" }
{ "_id" : ObjectId("617e8529eb452ed295e878ab"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "year" : 2012, "franchise" : "The Hobbit" }
{ "_id" : ObjectId("617e8538eb452ed295e878ac"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "year" : 2012, "franchise" : "The Hobbit" }
{ "_id" : ObjectId("617e854aeb452ed295e878ad"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "year" : 2012, "franchise" : "The Hobbit" }
{ "_id" : ObjectId("617e8597eb452ed295e878ae"), "title" : "Avatar", "year" : 2009, "actors" : [ "Sigourney Weaver", "Sam Worthington" ] }
{ "_id" : ObjectId("617e85a4eb452ed295e878af"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R. Tolkien", "year" : 2013, "franchise" : "The Hobbit" }
{ "_id" : ObjectId("617e85b0eb452ed295e878b0"), "title" : "The Hobbit: The Battle of Five Armies", "writer" : "J.R.R. Tolkien", "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("617e85bbeb452ed295e878b1"), "title" : "Pee Wee Herman's Big Adventure" }
>

```

2. db.movies.find({ writer: 'Quentin Taranito'})

3. db.movies.find({ actors: 'Brad Pitt'})

```

> db.movies.find({actors:"Brad Pitt"})
{ "_id" : ObjectId("617e84e9eb452ed295e878a7"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("617e8510eb452ed295e878a9"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : 2009, "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
>

```

4. db.movies.find({ franchise: "The Hobbit" })

```
> db.movies.find({actors:"Brad Pitt"})
{ "_id" : ObjectId("617e84e9eb452ed295e878a7"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("617e8510eb452ed295e878a8"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : 2009, "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }
>
```

5. db.movies.find({year:{\$lt: 1999, \$gt:1990}})

```
> db.movies.find({$and: [{year:{$gt:1990}}, {year:{$lt:1999}}])
{ "_id" : ObjectId("617e84e9eb452ed295e878a7"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("617e8500eb452ed295e878a8"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : 1994, "actors" : [ "John Travolta", "Uma Thurman" ] }
>
```

6. db.movies.find({year:{\$lt: 2010,\$gt: 2000}})

```
> db.movies.find({$and: [{year:{$gt:1990}}, {year:{$lt:2000}}])
{ "_id" : ObjectId("617e84e9eb452ed295e878a7"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("617e8500eb452ed295e878a8"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : 1994, "actors" : [ "John Travolta", "Uma Thurman" ] }
> db.movies.find({$and: [{year:{$gt:1990}}, {year:{$lt:2000}}])
{ "_id" : ObjectId("617e84e9eb452ed295e878a7"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : 1999, "actors" : [ "Brad Pitt", "Edward Norton" ] }
{ "_id" : ObjectId("617e8500eb452ed295e878a8"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : 1994, "actors" : [ "John Travolta", "Uma Thurman" ] }
>
```

Update Documents

1. 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 roup of dwarves to reclaim their mountain home = and the gold within it - from the dragon smaug.'} })

```
> 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 roup of dwarves to reclaim their mountain home = and the gold within it - from the dragon smaug.'} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

2. 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 magical ring.'} })

```
> 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 roup of dwarves to reclaim their mountain home = and the gold within it - from the dragon smaug.'} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
>
```

3. db.movies.update({title:'Pulp Fiction'},{\$push:{actors: 'Samuel L. Jackson'}})

```
> db.movies.update({title:'Pulp Fiction'},{$push:{actors: Samuel L. Jackson}})
uncaught exception: SyntaxError: missing } after property list :
@(:shell):1:63
>
```

Text Search

db.movies.createIndex({synopsis:'text'})

1. db.movies.find({\$text:{\$search:'Bilbo'}})

```
> db.movies.find({$text:{$search:'Bilbo'}})
{ "_id" : ObjectId("617e8508eb452ed295e878a0"), "title" : "The Hobbit: The Battle of Five Armies", "writer" : "J.R.R. Tolkien", "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("617e851eeb452ed295e878aa"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "year" : 2012, "franchise" : "The Hobbit", "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited roup of dwarves to reclaim their mountain home = and the gold within it - from the dragon smaug." }
>
```

2. db.movies.find({\$text:{\$search:'Gandalf'}})

```
> db.movies.find({$text:{$search:'Bilbo'}})
{ "_id" : ObjectId("617e85b0eb452ed295e878b0"), "title" : "The Hobbit: The Battle of Five Armies", "writer" : "J.R.R. Tolkien", "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("617e851eeb452ed295e878aa"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "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 the dragon smaug." }
> 2.db.movies.find({$text:{$search:'Gandalf'}})
uncaught exception: SyntaxError: Identifier starts immediately after numeric literal :
@shell>1:0
> db.movies.find({$text:{$search:'Gandalf'}})
> db.movies.find({$text:{$search:'Bilbo -Gandalf'}})
{ "_id" : ObjectId("617e85b0eb452ed295e878b0"), "title" : "The Hobbit: The Battle of Five Armies", "writer" : "J.R.R. Tolkien", "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("617e851eeb452ed295e878aa"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "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 the dragon smaug." }
>
```

3. db.movies.find({\$text:{\$search:'Bilbo -Gandalf'}})

```
> db.movies.find({$text:{$search:'Bilbo -Gandalf'}})
{ "_id" : ObjectId("617e85b0eb452ed295e878b0"), "title" : "The Hobbit: The Battle of Five Armies", "writer" : "J.R.R. Tolkien", "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("617e851eeb452ed295e878aa"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "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 the dragon smaug." }
>
```

4. db.movies.find({\$text:{\$search:'dwarves hobbit'}})

```
> db.movies.find({$text:{$search:'Bilbo -Gandalf'}})
{ "_id" : ObjectId("617e85b0eb452ed295e878b0"), "title" : "The Hobbit: The Battle of Five Armies", "writer" : "J.R.R. Tolkien", "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("617e851eeb452ed295e878aa"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "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 the dragon smaug." }
>
```

5. db.movies.find({\$text:{\$search:'"gold' 'dragon'"}})

```
> db.movies.find({$text:{$search:'"gold' 'dragon'"}})
{ "_id" : ObjectId("617e851eeb452ed295e878aa"), "title" : "The Hobbit: An Unexpected Journey", "writer" : "J.R.R. Tolkien", "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 the dragon smaug." }
>
```

Delete Documents

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

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

2. db.movies.remove({title:"Avtar"})

```
> db.movies.remove({title: "Avatar"})
WriteResult({ "nRemoved" : 0 })
>
```

Relationships

Insert documents of users collections

```
db.createCollection('users')
```

```

{ "ok" : 1 }
> db.createCollection('users')
{
  "ok" : 0,
  "errmsg" : "Collection already exists. NS: mongo_practice.users",
  "code" : 48,
  "codeName" : "NamespaceExists"
}
>

```

```

db.users.insertMany([
  { username: 'GoodGuyGreg',
    'first_name': "Good Guy",
    'last_name': "Greg",
  },
  { 'username': 'ScumbagSteve',
    'full_name':
    {'first_name': "Scumbag",
    'last_name': "Steve",
  }}])

```

```

> db.users.insertMany([
... { username: 'GoodGuyGreg',
...   'first_name': "Good Guy",
...   'last_name': "Greg",
... },
... { 'username': 'ScumbagSteve',
...   'full_name':
...   {'first_name': "Scumbag",
...   'last_name': "Steve",
...   } } ])
uncaught exception: SyntaxError: illegal character :
@(shell):2:13
> 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("617e9102eb452ed295e878b2"),
    ObjectId("617e9102eb452ed295e878b3"),
    ObjectId("617e9102eb452ed295e878b4"),
    ObjectId("617e9102eb452ed295e878b5"),
    ObjectId("617e9102eb452ed295e878b6"),
    ObjectId("617e9102eb452ed295e878b7")
  ]
}
>

```

Insert documents of posts collections

```
db.createCollection('posts')

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',
  })
])
```

```

> db.createCollection('posts')
uncaught exception: SyntaxError: illegal character :
@(shell):1:20
> 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("617e914beb452ed295e878b8"),
    ObjectId("617e914beb452ed295e878b9"),
    ObjectId("617e914beb452ed295e878ba"),
    ObjectId("617e914beb452ed295e878bb"),
    ObjectId("617e914beb452ed295e878bc"),
    ObjectId("617e914beb452ed295e878bd")
  ]
}
>

```

Insert documents of comments collections

```

db.createCollection('comments')

db.comments.insertMany([
  'username':'GoodGuyGreg',
  'comment':"Hope you got a good deal!",
  'post':['617aa9cc6b910c064c32c339'],
},
{ 'username':'GoodGuyGreg',
  'comment':"What's mine is yours!",
  'post':['617aa9cc6b910c064c32c33a'],
},

```

```
{ 'username': 'GoodGuyGreg',
  'comment': "Don't violate the licensing agreement",
  'post': ['617aa9cc6b910c064c32c33b'],
},
{ 'username': 'ScumbagSteve',
  'comment': "It still isn't clean",
  'post': ['617aa9cc6b910c064c32c336'],
},
{ 'username': 'ScumbagSteve',
  'comment': "Denied your PR cause I found a hack",
  'post': ['617aa9cc6b910c064c32c338'],
}]})
```

```
> db.createCollection('comments')
uncaught exception: SyntaxError: illegal character :
@(shell):1:20
> db.comments.insertMany([
...   { 'username': 'GoodGuyGreg',
...     'comment': "Hope you got a good deal!",
...     'post': ['617aa9cc6b910c064c32c339'],
...   },
...   { 'username': 'GoodGuyGreg',
...     'comment': "What's mine is yours!",
...     'post': ['617aa9cc6b910c064c32c33a'],
...   },
...   { 'username': 'GoodGuyGreg',
...     'comment': "Don't violate the licensing agreement",
...     'post': ['617aa9cc6b910c064c32c33b'],
...   },
...   { 'username': 'ScumbagSteve',
...     'comment': "It still isn't clean",
...     'post': ['617aa9cc6b910c064c32c336'],
...   },
...   { 'username': 'ScumbagSteve',
...     'comment': "Denied your PR cause I found a hack",
...     'post': ['617aa9cc6b910c064c32c338'],
...   }
... ])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("617e9179eb452ed295e878be"),
    ObjectId("617e9179eb452ed295e878bf"),
    ObjectId("617e9179eb452ed295e878c0"),
    ObjectId("617e9179eb452ed295e878c1"),
    ObjectId("617e9179eb452ed295e878c2")
  ]
}
```

Querying related collections

1. `db.users.find()`

2. db.posts.find()

```
> db.posts.find()
{ "_id" : ObjectId("617e9102eb452ed295e878b2"), "username" : "GoodGuyGreg", "title" : "Passes out at party", "body" : "Wakes up early and cleans house" }
{ "_id" : ObjectId("617e9102eb452ed295e878b3"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("617e9102eb452ed295e878b4"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
{ "_id" : ObjectId("617e9102eb452ed295e878b5"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("617e9102eb452ed295e878b6"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ "_id" : ObjectId("617e9102eb452ed295e878b7"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }
{ "_id" : ObjectId("617e914beb452ed295e878b8"), "username" : "GoodGuyGreg", "title" : "Passes out at party", "body" : "Wakes up early and cleans house" }
{ "_id" : ObjectId("617e914beb452ed295e878b9"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }
{ "_id" : ObjectId("617e914beb452ed295e878ba"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a Pull Request" }
{ "_id" : ObjectId("617e914beb452ed295e878bb"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }
{ "_id" : ObjectId("617e914beb452ed295e878bc"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }
{ "_id" : ObjectId("617e914beb452ed295e878bd"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }
>
```

3. db.posts.find({username:'GoodGuyGreg'})

4. db.posts.find({username:'ScumbagSteve'})

5. db.comments.find()

```
> db.comments.find()
{ "_id" : ObjectId("617e9179eb452ed295e878be"), "username" : "GoodGuyGreg", "comment" : "Hope you got a good deal!", "post" : [ "617aa9cc6b910c064c32c339" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878bf"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : [ "617aa9cc6b910c064c32c33a" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c0"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement", "post" : [ "617aa9cc6b910c064c32c33b" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c1"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : [ "617aa9cc6b910c064c32c336" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c2"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : [ "617aa9cc6b910c064c32c338" ] }
>
```

6. db.comments.find({username:'GoodGuyGreg'})

7. db.comments.find({username:'ScumbagSteve'})

```
> db.comments.find({username:'ScumbagSteve'})
{ "_id" : ObjectId("617e9179eb452ed295e878c1"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : [ "617aa9cc6b910c064c32c336" ] }
{ "_id" : ObjectId("617e9179eb452ed295e878c2"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : [ "617aa9cc6b910c064c32c338" ] }
>
```

8. db.posts.find({title:'Reports a bug in your code'})

```
> db.posts.find({title:'Reports a bug in your code'})
{ "_id" : ObjectId("617e9102eb452ed295e878b4"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }
{ "_id" : ObjectId("617e914beb452ed295e878ba"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a Pull Request" }
>
```

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({city:'ATLANTA',state:'GA'})
```

```

> show dbs
admin            0.000GB
config          0.000GB
local           0.000GB
mongoDB_practice 0.000GB
mongo_practice  0.002GB
population      0.002GB
studentmng      0.000GB
> use population
switched to db population
> db.zipcodes.find({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" }
Type "it" for more
>

```

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

```
db.zipcodes.aggregate([{$match:{city:'ATLANTA',state:'GA'}}])
```

```

> 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" }
Type "it" for more

```

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

```
db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{ $group:{_id:'$ _id'}},{ $count:'uni_zipcode'}}])
```

```

> db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{ $group:{_id:'$ _id'}},{ $count:'uni_zipcode'}}])
{ "uni_zipcode" : 41 }
>

```

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

```
db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{ $group:{_id:'city',totalpop:{ $sum:'$pop'}}}])
```

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

Populations By State

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

```
db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}}])
```

```
> db.zipcodes.aggregate([{$match:{city:'ATLANTA'}},{ $group:{_id:'city',totalpop:{$sum:'$pop'}}}])
{ "_id" : "city", "totalpop" : 630046 }
> db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}}])
{ "_id" : "NC", "totalpop" : 6628637 }
{ "_id" : "OH", "totalpop" : 10846517 }
{ "_id" : "MN", "totalpop" : 4372982 }
{ "_id" : "ME", "totalpop" : 1226648 }
{ "_id" : "IN", "totalpop" : 5544136 }
{ "_id" : "NY", "totalpop" : 17990402 }
{ "_id" : "CT", "totalpop" : 3287116 }
{ "_id" : "ND", "totalpop" : 638272 }
{ "_id" : "MT", "totalpop" : 798948 }
{ "_id" : "MD", "totalpop" : 4781379 }
{ "_id" : "AL", "totalpop" : 4040587 }
{ "_id" : "NJ", "totalpop" : 7730188 }
{ "_id" : "MO", "totalpop" : 5110648 }
{ "_id" : "KS", "totalpop" : 2475285 }
{ "_id" : "GA", "totalpop" : 6478216 }
{ "_id" : "NE", "totalpop" : 1578139 }
{ "_id" : "LA", "totalpop" : 4217595 }
{ "_id" : "AR", "totalpop" : 2350725 }
{ "_id" : "CO", "totalpop" : 3293755 }
{ "_id" : "WY", "totalpop" : 453528 }
Type "it" for more
>
```

2. sort the results by population, highest first

```
db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}},{ $sort:{totalpop:-1}}])
```

```
> db.zipcodes.aggregate([{$group:{_id:'$state',totalpop:{$sum:'$pop'}}},{ $sort:{totalpop:-1}}])
{ "_id" : "CA", "totalpop" : 29754890 }
{ "_id" : "NY", "totalpop" : 17990402 }
{ "_id" : "TX", "totalpop" : 16984601 }
{ "_id" : "FL", "totalpop" : 12686644 }
{ "_id" : "PA", "totalpop" : 11881643 }
{ "_id" : "IL", "totalpop" : 11427576 }
{ "_id" : "OH", "totalpop" : 10846517 }
{ "_id" : "MI", "totalpop" : 9295297 }
{ "_id" : "NJ", "totalpop" : 7730188 }
{ "_id" : "NC", "totalpop" : 6628637 }
{ "_id" : "GA", "totalpop" : 6478216 }
{ "_id" : "VA", "totalpop" : 6181479 }
{ "_id" : "MA", "totalpop" : 6016425 }
{ "_id" : "IN", "totalpop" : 5544136 }
{ "_id" : "MO", "totalpop" : 5110648 }
{ "_id" : "WI", "totalpop" : 4891769 }
{ "_id" : "TN", "totalpop" : 4876457 }
{ "_id" : "WA", "totalpop" : 4866692 }
{ "_id" : "MD", "totalpop" : 4781379 }
{ "_id" : "MN", "totalpop" : 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',totalpop:{$sum:'$pop'}}},{ $sort:{totalpop:-1}},{ $limit:3}])
```

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

```
> db.zipcodes.aggregate([{$group:{_id: { city: '$city', state: '$state' } ,totalpop:{$sum:'$pop'}}]])
{ "_id" : { "city" : "DUNCANNON", "state" : "PA" }, "totalpop" : 10021 }
{ "_id" : { "city" : "MILLS", "state" : "PA" }, "totalpop" : 653 }
{ "_id" : { "city" : "OTTO", "state" : "NC" }, "totalpop" : 2297 }
{ "_id" : { "city" : "BOLIVAR", "state" : "TN" }, "totalpop" : 10327 }
{ "_id" : { "city" : "SONDHEIMER", "state" : "LA" }, "totalpop" : 887 }
{ "_id" : { "city" : "DOGWOOD", "state" : "TX" }, "totalpop" : 5942 }
{ "_id" : { "city" : "MC LOUD", "state" : "OK" }, "totalpop" : 3334 }
{ "_id" : { "city" : "KONAWA", "state" : "OK" }, "totalpop" : 2800 }
{ "_id" : { "city" : "WINDYVILLE", "state" : "MO" }, "totalpop" : 769 }
{ "_id" : { "city" : "HURT", "state" : "VA" }, "totalpop" : 5283 }
{ "_id" : { "city" : "LORAIN", "state" : "ND" }, "totalpop" : 1455 }
{ "_id" : { "city" : "NEWPORT CENTER", "state" : "VT" }, "totalpop" : 1367 }
{ "_id" : { "city" : "RANGER", "state" : "WV" }, "totalpop" : 5703 }
{ "_id" : { "city" : "GATES", "state" : "TN" }, "totalpop" : 2508 }
{ "_id" : { "city" : "MALVERN", "state" : "AL" }, "totalpop" : 1686 }
{ "_id" : { "city" : "ROCKY GAP", "state" : "VA" }, "totalpop" : 510 }
{ "_id" : { "city" : "OBERON", "state" : "ND" }, "totalpop" : 556 }
{ "_id" : { "city" : "LODGE GRASS", "state" : "MT" }, "totalpop" : 2938 }
{ "_id" : { "city" : "MILLSAP", "state" : "TX" }, "totalpop" : 9110 }
{ "_id" : { "city" : "VERNAL", "state" : "UT" }, "totalpop" : 17641 }
type "it" for more
```

2. sort the results by population, highest first

```
db.zipcodes.aggregate([{$group:{_id: { city: '$city', state: '$state' } ,totalpop:{$sum:'$pop'}}},{$sort:{totalpop:-1}}])
```

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

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

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

```
db.zipcodes.aggregate([{$match:{state:'TX'}},{ $group:{_id:{state:'$state',city:'$city'},totalpop:{$sum:'$pop'}}},{ $sort:{totalpop:-1}},{ $limit:3}])
```

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

Bonus

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

```
db.zipcodes.aggregate([{$group:{_id:{state:'$state',city:'$city'},Avgpop:{$avg:'$pop'}}}])
```



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

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

```
db.zipcodes.aggregate([{$group: {_id: {state: '$state', city: '$city'}, Avgpop: {$avg: '$pop'}}}, {$sort: {Avgpop: -1}}, {$limit: 3}])
```

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

ASSIGNMENT 3

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

```
> db.addresses.find()
{ "_id" : ObjectId("617bf91aa9a1cd505ba79aa"), "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", "score" : 9 }, { "date" : ISODate("2011-03-10T00:00:00Z"), "grade" : "B", "score" : 14 } ], "name" : "Morris Park Bake Shop", "restaurant_id" : "33005405" }
{ "_id" : ObjectId("617bf91aa9a1cd505ba79ab"), "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", "score" : 12 } ], "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "_id" : ObjectId("617bf91aa9a1cd505ba79ac"), "address" : { "building" : "351", "coord" : [ -73.98513599999999, 40.767919 ], "street" : "West 57 Street", "zipcode" : "10019", "borough" : "Manhattan", "cuisine" : "Irish", "grades" : [ { "date" : ISODate("2014-09-06T00:00:00Z"), "grade" : "A", "score" : 2 }, { "date" : ISODate("2013-07-22T00:00:00Z"), "grade" : "A", "score" : 11 }, { "date" : ISODate("2012-07-31T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2011-12-29T00:00:00Z"), "grade" : "A", "score" : 12 } ], "name" : "DJ Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{ "_id" : ObjectId("617bf91aa9a1cd505ba79ad"), "address" : { "building" : "2780", "coord" : [ -73.98241999999999, 40.579505 ], "street" : "Stillwell Avenue", "zipcode" : "11224", "borough" : "Brooklyn", "cuisine" : "American", "grades" : [ { "date" : ISODate("2014-06-10T00:00:00Z"), "grade" : "A", "score" : 5 }, { "date" : ISODate("2013-06-05T00:00:00Z"), "grade" : "A", "score" : 7 }, { "date" : ISODate("2012-04-13T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2011-10-12T00:00:00Z"), "grade" : "A", "score" : 12 } ], "name" : "Riviera Caterer", "restaurant_id" : "40356018" }
{ "_id" : ObjectId("617bf91aa9a1cd505ba79ae"), "address" : { "building" : "97-22", "coord" : [ -73.8601152, 40.7311739 ], "street" : "63 Road", "zipcode" : "11374", "borough" : "Queens", "cuisine" : "Jewish/Kosher", "grades" : [ { "date" : ISODate("2014-11-24T00:00:00Z"), "grade" : "Z", "score" : 20 }, { "date" : ISODate("2013-01-17T00:00:00Z"), "grade" : "A", "score" : 13 }, { "date" : ISODate("2012-08-02T00:00:00Z"), "grade" : "A", "score" : 13 }, { "date" : ISODate("2011-12-15T00:00:00Z"), "grade" : "B", "score" : 25 } ], "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{ "_id" : ObjectId("617bf91aa9a1cd505ba79af"), "address" : { "building" : "8825", "coord" : [ -73.8803827, 40.7643124 ], "street" : "Astoria Boulevard", "zipcode" : "11369", "borough" : "Queens", "cuisine" : "American", "grades" : [ { "date" : ISODate("2014-11-15T00:00:00Z"), "grade" : "Z", "score" : 38 }, { "date" : ISODate("2014-05-02T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2013-03-02T00:00:00Z"), "grade" : "A", "score" : 7 }, { "date" : ISODate("2012-02-10T00:00:00Z"), "grade" : "A", "score" : 13 } ], "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151" }
{ "_id" : ObjectId("617bf91aa9a1cd505ba79b0"), "address" : { "building" : "2286", "coord" : [ -74.1372286, 40.6119572 ], "street" : "Victory Boulevard", "zipcode" : "10314", "borough" : "Staten Island", "cuisine" : "Jewish/Kosher", "grades" : [ { "date" : ISODate("2014-10-06T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2014-05-20T00:00:00Z"), "grade" : "A", "score" : 13 } ] }
```

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})
{"_id": "ObjectID('617bf91aaa91cd505ba79aa')", "borough": "Bronx", "cuisine": "Bakery", "name": "Morris Park Bake Shop", "restaurant_id": "30075445" }
{"_id": "ObjectID('617bf91aaa91cd505ba79ab')", "borough": "Brooklyn", "cuisine": "Hamburgers", "name": "Wendy'S", "restaurant_id": "30112340" }
{"_id": "ObjectID('617bf91aaa91cd505ba79ac')", "borough": "Manhattan", "cuisine": "Irish", "name": "Dj Reynolds Pub And Restaurant", "restaurant_id": "30191841" }
{"_id": "ObjectID('617bf91aaa91cd505ba79ad')", "borough": "Brooklyn", "cuisine": "American ", "name": "Riviera Caterer", "restaurant_id": "40356018" }
{"_id": "ObjectID('617bf91aaa91cd505ba79ae')", "borough": "Queens", "cuisine": "Jewish/Kosher", "name": "Tov Kosher Kitchen", "restaurant_id": "40356068" }
{"_id": "ObjectID('617bf91aaa91cd505ba79af')", "borough": "Queens", "cuisine": "American ", "name": "Brunos On The Boulevard", "restaurant_id": "40356151" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b0')", "borough": "Staten Island", "cuisine": "Jewish/Kosher", "name": "Kosher Island", "restaurant_id": "40356442" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b1')", "borough": "Brooklyn", "cuisine": "Delicatessen", "name": "Wilken'S Fine Food", "restaurant_id": "40356483" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b2')", "borough": "Brooklyn", "cuisine": "American ", "name": "Regina Caterers", "restaurant_id": "40356649" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b3')", "borough": "Brooklyn", "cuisine": "Ice Cream, Gelato, Yogurt, Ices", "name": "Taste The Tropics Ice Cream", "restaurant_id": "40356731" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b4')", "borough": "Bronx", "cuisine": "American ", "name": "Wild Asia", "restaurant_id": "40357217" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b5')", "borough": "Brooklyn", "cuisine": "American ", "name": "C & C Catering Service", "restaurant_id": "40357437" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b6')", "borough": "Brooklyn", "cuisine": "Chinese", "name": "May May Kitchen", "restaurant_id": "40358429" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b7')", "borough": "Manhattan", "cuisine": "American ", "name": "1 East 66Th Street Kitchen", "restaurant_id": "40359480" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b8')", "borough": "Brooklyn", "cuisine": "Jewish/Kosher", "name": "Seuda Foods", "restaurant_id": "40360045" }
{"_id": "ObjectID('617bf91aaa91cd505ba79b9')", "borough": "Brooklyn", "cuisine": "Ice Cream, Gelato, Yogurt, Ices", "name": "Carvel Ice Cream", "restaurant_id": "40360076" }
{"_id": "ObjectID('617bf91aaa91cd505ba79ba')", "borough": "Queens", "cuisine": "Ice Cream, Gelato, Yogurt, Ices", "name": "Carvel Ice Cream", "restaurant_id": "40361322" }
{"_id": "ObjectID('617bf91aaa91cd505ba79bb')", "borough": "Brooklyn", "cuisine": "Delicatessen", "name": "Nordic Delicacies", "restaurant_id": "40361390" }
{"_id": "ObjectID('617bf91aaa91cd505ba79bc')", "borough": "Manhattan", "cuisine": "American ", "name": "Glorious Food", "restaurant_id": "40361521" }
{"_id": "ObjectID('617bf91aaa91cd505ba79bd')", "borough": "Brooklyn", "cuisine": "American ", "name": "The Movable Feast", "restaurant_id": "40361606" }
type "it" for more
>
```

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 })
{"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" }
type "it" for more
>
```

3. 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, _id:0, 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" }
type "it" for more
>
```

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

```

> db.addresses.find(borough:"Bronx").limit(5)
{"_id": ObjectId("617bf91aa9a1cd585ba79aa"), "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", "score": 9 }, { "date": ISODate("2011-03-10T00:00:00Z"), "grade": "B", "score": 14 } ], "name": "Morris Park Bake Shop", "restaurant_id": "30075445" }
{"_id": ObjectId("617bf91aa9a1cd585ba79ba"), "address": {"building": "2300", "coord": [ -73.8786113, 40.8502883 ], "street": "Southern Boulevard", "zipcode": "10460"}, "borough": "Bronx", "cuisine": "American", "grades": [ { "date": ISODate("2014-05-28T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2013-06-19T00:00:00Z"), "grade": "A", "score": 4 }, { "date": ISODate("2012-06-15T00:00:00Z"), "grade": "A", "score": 3 } ], "name": "Wild Asia", "restaurant_id": "40357217" }
{"_id": ObjectId("617bf91aa9a1cd585ba79c9d"), "address": {"building": "1006", "coord": [ -73.84856870000002, 40.8903781 ], "street": "East 233 Street", "zipcode": "10466"}, "borough": "Bronx", "cuisine": "Ice Cream, Gelato, Yogurt, Ices", "grades": [ { "date": ISODate("2014-04-24T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-09-05T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-02-21T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-07-03T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-07-11T00:00:00Z"), "grade": "A", "score": 5 } ], "name": "Carvel Ice Cream", "restaurant_id": "40363093" }
{"_id": ObjectId("617bf91aa9a1cd585ba79cd"), "address": {"building": "1236", "coord": [ -73.8893654, 40.81376179999999 ], "street": "238 Spofford Ave", "zipcode": "10474"}, "borough": "Bronx", "cuisine": "Chinese", "grades": [ { "date": ISODate("2013-12-30T00:00:00Z"), "grade": "A", "score": 8 }, { "date": ISODate("2013-01-08T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-06-12T00:00:00Z"), "grade": "B", "score": 13 } ], "name": "Happy Garden", "restaurant_id": "40364296" }
{"_id": ObjectId("617bf91aa9a1cd585ba79d4f"), "address": {"building": "2777", "coord": [ -73.8941893, 40.8634684 ], "street": "East Kingsbridge Road", "zipcode": "10458"}, "borough": "Bronx", "cuisine": "Chinese", "grades": [ { "date": ISODate("2014-03-03T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-09-26T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-03-19T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-08-29T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-08-17T00:00:00Z"), "grade": "A", "score": 9 } ], "name": "Happy Garden", "restaurant_id": "40364296" }
> db.addresses.find(borough:"Bronx").limit(5)
{"_id": ObjectId("617bf91aa9a1cd585ba79aa"), "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", "score": 9 }, { "date": ISODate("2011-03-10T00:00:00Z"), "grade": "B", "score": 14 } ], "name": "Morris Park Bake Shop", "restaurant_id": "30075445" }
{"_id": ObjectId("617bf91aa9a1cd585ba79ba"), "address": {"building": "2300", "coord": [ -73.8786113, 40.8502883 ], "street": "Southern Boulevard", "zipcode": "10460"}, "borough": "Bronx", "cuisine": "American", "grades": [ { "date": ISODate("2014-05-28T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2013-06-19T00:00:00Z"), "grade": "A", "score": 4 }, { "date": ISODate("2012-06-15T00:00:00Z"), "grade": "A", "score": 3 } ], "name": "Wild Asia", "restaurant_id": "40357217" }
{"_id": ObjectId("617bf91aa9a1cd585ba79c9d"), "address": {"building": "1006", "coord": [ -73.84856870000002, 40.8903781 ], "street": "East 233 Street", "zipcode": "10466"}, "borough": "Bronx", "cuisine": "Ice Cream, Gelato, Yogurt, Ices", "grades": [ { "date": ISODate("2014-04-24T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-09-05T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-02-21T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-07-03T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-07-11T00:00:00Z"), "grade": "A", "score": 5 } ], "name": "Carvel Ice Cream", "restaurant_id": "40363093" }
{"_id": ObjectId("617bf91aa9a1cd585ba79cd"), "address": {"building": "1236", "coord": [ -73.8893654, 40.81376179999999 ], "street": "238 Spofford Ave", "zipcode": "10474"}, "borough": "Bronx", "cuisine": "Chinese", "grades": [ { "date": ISODate("2013-12-30T00:00:00Z"), "grade": "A", "score": 8 }, { "date": ISODate("2013-01-08T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-06-12T00:00:00Z"), "grade": "B", "score": 13 } ], "name": "Happy Garden", "restaurant_id": "40364296" }

```

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

```

> db.addresses.aggregate([
...   { $match: { 'borough': 'Bronx' } },
...   { $limit: 5 }
... ])
{"_id": ObjectId("617bf91aa9a1cd585ba79aa"), "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", "score": 9 }, { "date": ISODate("2011-03-10T00:00:00Z"), "grade": "B", "score": 14 } ], "name": "Morris Park Bake Shop", "restaurant_id": "30075445" }
{"_id": ObjectId("617bf91aa9a1cd585ba79ba"), "address": {"building": "2300", "coord": [ -73.8786113, 40.8502883 ], "street": "Southern Boulevard", "zipcode": "10460"}, "borough": "Bronx", "cuisine": "American", "grades": [ { "date": ISODate("2014-05-28T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2013-06-19T00:00:00Z"), "grade": "A", "score": 4 }, { "date": ISODate("2012-06-15T00:00:00Z"), "grade": "A", "score": 3 } ], "name": "Wild Asia", "restaurant_id": "40357217" }
{"_id": ObjectId("617bf91aa9a1cd585ba79c9d"), "address": {"building": "1006", "coord": [ -73.84856870000002, 40.8903781 ], "street": "East 233 Street", "zipcode": "10466"}, "borough": "Bronx", "cuisine": "Ice Cream, Gelato, Yogurt, Ices", "grades": [ { "date": ISODate("2014-04-24T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-09-05T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-02-21T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-07-03T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-07-11T00:00:00Z"), "grade": "A", "score": 5 } ], "name": "Carvel Ice Cream", "restaurant_id": "40363093" }
{"_id": ObjectId("617bf91aa9a1cd585ba79cd"), "address": {"building": "1236", "coord": [ -73.8893654, 40.81376179999999 ], "street": "238 Spofford Ave", "zipcode": "10474"}, "borough": "Bronx", "cuisine": "Chinese", "grades": [ { "date": ISODate("2013-12-30T00:00:00Z"), "grade": "A", "score": 8 }, { "date": ISODate("2013-01-08T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-06-12T00:00:00Z"), "grade": "B", "score": 13 } ], "name": "Happy Garden", "restaurant_id": "40364296" }

```

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

db.addresses.find({"borough": "Bronx"}).skip(5).limit(5)

```

> db.addresses.find({"borough": "Bronx"}).skip(5).limit(5)
{"_id": ObjectId("617bf91aa9a1cd585ba79e7"), "address": {"building": "658", "coord": [ -73.81363999999999, 40.82941100000001 ], "street": "Clarence Ave", "zipcode": "10465"}, "borough": "Bronx", "cuisine": "American", "grades": [ { "date": ISODate("2014-06-21T00:00:00Z"), "grade": "A", "score": 5 }, { "date": ISODate("2012-07-11T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2011-03-10T00:00:00Z"), "grade": "A", "score": 10 } ], "name": "Manhenn Club", "restaurant_id": "40364363" }
{"_id": ObjectId("617bf91aa9a1cd585ba79ff"), "address": {"building": "2222", "coord": [ -73.84971599999999, 40.8304811 ], "street": "Haviland Avenue", "zipcode": "10462"}, "borough": "Bronx", "cuisine": "American", "grades": [ { "date": ISODate("2014-12-18T00:00:00Z"), "grade": "A", "score": 7 }, { "date": ISODate("2014-05-01T00:00:00Z"), "grade": "B", "score": 17 }, { "date": ISODate("2013-03-14T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2012-09-20T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-02-06T00:00:00Z"), "grade": "B", "score": 19 } ], "name": "The New Starling Athletic Club Of The Bronx", "restaurant_id": "40364956" }
{"_id": ObjectId("617bf91aa9a1cd585ba7a17"), "address": {"building": "72", "coord": [ -73.92586, 40.8275556 ], "street": "East 161 Street", "zipcode": "10451"}, "borough": "Bronx", "cuisine": "American", "grades": [ { "date": ISODate("2014-04-15T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2013-11-14T00:00:00Z"), "grade": "A", "score": 4 }, { "date": ISODate("2013-07-29T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-12-31T00:00:00Z"), "grade": "B", "score": 15 }, { "date": ISODate("2012-05-30T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2012-01-09T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2011-08-15T00:00:00Z"), "grade": "C", "score": 37 } ], "name": "Waikkee Tavern", "restaurant_id": "40365499" }
{"_id": ObjectId("617bf91aa9a1cd585ba7a2a"), "address": {"building": "331", "coord": [ -73.87765399999999, 40.8724377 ], "street": "East 204 Street", "zipcode": "10467"}, "borough": "Bronx", "cuisine": "Irish", "grades": [ { "date": ISODate("2014-06-26T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2014-03-26T00:00:00Z"), "grade": "B", "score": 23 }, { "date": ISODate("2013-09-11T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2012-12-18T00:00:00Z"), "grade": "B", "score": 27 }, { "date": ISODate("2011-10-20T00:00:00Z"), "grade": "A", "score": 13 } ], "name": "McDwyers Pub", "restaurant_id": "40365893" }
{"_id": ObjectId("617bf91aa9a1cd585ba7a43"), "address": {"building": "5828", "coord": [ -73.9002615, 40.885186 ], "street": "Broadway", "zipcode": "10463"}, "borough": "Bronx", "cuisine": "American", "grades": [ { "date": ISODate("2014-02-26T00:00:00Z"), "grade": "A", "score": 5 }, { "date": ISODate("2013-10-09T00:00:00Z"), "grade": "B", "score": 19 }, { "date": ISODate("2013-05-15T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-11-28T00:00:00Z"), "grade": "B", "score": 18 }, { "date": ISODate("2011-10-17T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2011-06-22T00:00:00Z"), "grade": "C", "score": 35 }, { "date": ISODate("2011-03-10T00:00:00Z"), "grade": "C", "score": 41 } ], "name": "The Punch Bowl", "restaurant_id": "40366497" }

```

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

db.addresses.find({grades: { \$elemMatch:{"score":{\$gt :90}}}});

```

> db.addresses.find({grades: { $elemMatch:{"score":{$gt :90}}}});
{"_id": ObjectId("617bf91aa9a1cd585ba7b08"), "address": {"building": "65", "coord": [ -73.9782725, 40.7624022 ], "street": "West 54 Street", "zipcode": "10019"}, "borough": "Manhattan", "cuisine": "American", "grades": [ { "date": ISODate("2014-08-22T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2014-03-26T00:00:00Z"), "grade": "C", "score": 131 }, { "date": ISODate("2013-09-25T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2013-04-08T00:00:00Z"), "grade": "B", "score": 25 }, { "date": ISODate("2012-10-15T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-18-19T00:00:00Z"), "grade": "A", "score": 13 } ], "name": "Murali On 54/Randolphs", "restaurant_id": "40372466" }
{"_id": ObjectId("617bf91aa9a1cd585ba7ba9"), "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 }, { "date": ISODate("2013-05-30T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2013-08-24T00:00:00Z"), "grade": "B", "score": 2 }, { "date": ISODate("2012-10-01T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-04-08T00:00:00Z"), "grade": "C", "score": 92 }, { "date": ISODate("2011-11-03T00:00:00Z"), "grade": "C", "score": 41 } ], "name": "Gandhi", "restaurant_id": "40381295" }
{"_id": ObjectId("617bf91aa9a1cd585ba7b0c"), "address": {"building": "130", "coord": [ -73.984758, 40.7457939 ], "street": "Madison Avenue", "zipcode": "10016"}, "borough": "Manhattan", "cuisine": "Pizza/Italian", "grades": [ { "date": ISODate("2014-12-24T00:00:00Z"), "grade": "Z", "score": 31 }, { "date": ISODate("2014-06-17T00:00:00Z"), "grade": "C", "score": 9 }, { "date": ISODate("2013-12-12T00:00:00Z"), "grade": "C", "score": 22 }, { "date": ISODate("2013-05-22T00:00:00Z"), "grade": "B", "score": 21 }, { "date": ISODate("2012-05-02T00:00:00Z"), "grade": "A", "score": 11 } ], "name": "Bella Napoli", "restaurant_id": "40393488" }

```

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.

```
db.addresses.find({'address.coord': {'$lt': -95.46183}})
{"_id": "ObjectID('617bf91fa9a1dc50ba7f722')", "address": "The \"building\": \"3787\", \"coord\": [ -181.8945214, 33.5197474 ], \"street\": \"82 Street\", \"zipcode\": \"11372\", \"borough\": \"Queens\", \"cuisine\": \"American\", \"grades\": [ { \"date\": ISODate(\"2014-06-04T00:00:00Z\"), \"grade\": \"A\", \"score\": 12 }, { \"date\": ISODate(\"2013-11-07T00:00:00Z\"), \"grade\": \"B\", \"score\": 19 }, { \"date\": ISODate(\"2013-05-17T00:00:00Z\"), \"grade\": \"A\", \"score\": 10 }, { \"date\": ISODate(\"2012-08-29T00:00:00Z\"), \"grade\": \"A\", \"score\": 11 }, { \"date\": ISODate(\"2012-04-03T00:00:00Z\"), \"grade\": \"C\", \"score\": 13 } ], \"name\": \"Burger King\", \"restaurant_id\": \"40534667\", \"type\": \"restaurant\"}
{"_id": "ObjectID('617bf91fa9a1dc50ba83bd9')", "address": "The \"building\": \"15259\", \"coord\": [ -119.6368672, 36.2584996 ], \"street\": \"14 Avenue\", \"zipcode\": \"11357\", \"borough\": \"Queens\", \"cuisine\": \"Italian\", \"grades\": [ { \"date\": ISODate(\"2014-09-04T00:00:00Z\"), \"grade\": \"A\", \"score\": 11 }, { \"date\": ISODate(\"2014-03-26T00:00:00Z\"), \"grade\": \"A\", \"score\": 8 }, { \"date\": ISODate(\"2013-03-04T00:00:00Z\"), \"grade\": \"A\", \"score\": 10 }, { \"date\": ISODate(\"2012-09-29T00:00:00Z\"), \"grade\": \"A\", \"score\": 10 }, { \"date\": ISODate(\"2012-04-20T00:00:00Z\"), \"grade\": \"C\", \"score\": 13 } ], \"name\": \"Pizzeria\", \"restaurant_id\": \"40534681\", \"type\": \"restaurant\"}
{"_id": "ObjectID('617bf921aa91dc50ba8868')", "address": "The \"building\": \"56\", \"coord\": [ -119.9752059, 42.8970272 ], \"street\": \"West Side Highway\", \"zipcode\": \"10006\", \"borough\": \"Manhattan\", \"cuisine\": \"Japanese\", \"grades\": [ { \"date\": ISODate(\"2014-03-20T00:00:00Z\"), \"grade\": \"A\", \"score\": 9 }, { \"date\": ISODate(\"2014-05-28T00:00:00Z\"), \"grade\": \"A\", \"score\": 11 }, { \"date\": ISODate(\"2012-07-05T00:00:00Z\"), \"grade\": \"A\", \"score\": 13 }, { \"date\": ISODate(\"2011-07-27T00:00:00Z\"), \"grade\": \"A\", \"score\": 2 }, { \"name\": \"Sports Center At Chelsea Place (Sushi Bar)\", \"restaurant_id\": \"46882356\" } ]
```

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.

```

{"grades.score": "{$gt: 70}},

{"address.coord": "{$lt: -65.754168}}

});

db.addresses.find( { $and: ( { cuisine : {$ne: 'American' } },
...
    { "grades.coord" : {$gt: 70} },
    { "address.coord" : {$lt: -65.754168} }
    )
});

[ { "id" : ObjectId("617b791aa9a1cd95b97ba9"), "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 }, { "date" : ISODate("2013-05-30T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2013-04-24T00:00:00Z"), "grade" : "P", "score" : 2 }, { "date" : ISODate("2012-10-01T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2012-04-06T00:00:00Z"), "grade" : "C", "score" : 92 }, { "date" : ISODate("2011-11-03T00:00:00Z"), "grade" : "C", "score" : 41 } ], "name" : "Gan dui", "restaurant_id" : "40381295" },
{ "id" : ObjectId("617b791aa9a1cd95b97d0c"), "address" : { "building" : "130", "coord" : [ -73.984758, 40.7457939 ], "street" : "Madison Avenue", "zipcode" : "10016" }, "borough" : "Manhattan", "cuisine" : "Pizza/Italian", "grades" : { { "date" : ISODate("2014-12-24T00:00:00Z"), "grade" : "Z", "score" : 31 }, { "date" : ISODate("2014-06-17T00:00:00Z"), "grade" : "C", "score" : 98 }, { "date" : ISODate("2013-12-12T00:00:00Z"), "grade" : "C", "score" : 32 }, { "date" : ISODate("2013-05-22T00:00:00Z"), "grade" : "B", "score" : 21 }, { "date" : ISODate("2012-05-02T00:00:00Z"), "grade" : "A", "score" : 11 }, "name" : "Bella Napoli", "restaurant_id" : "40393488" },
{ "id" : ObjectId("617b791aa9a1cd95b97d17"), "address" : { "building" : "101", "coord" : [ -73.9243061, 40.8276297 ], "street" : "East 161 Street", "zipcode" : "10451" }, "borough" : "Bronx", "cuisine" : "Latin (Cuban, Dominican, Puerto Rican, South & Central American)", "grades" : [ { "date" : ISODate("2014-04-10T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-01-01T00:00:00Z"), "grade" : "A", "score" : 6 }, { "date" : ISODate("2013-04-11T00:00:00Z"), "grade" : "B", "score" : 25 }, { "date" : ISODate("2012-10-25T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2012-10-10T00:00:00Z"), "grade" : "P", "score" : 12 }, { "date" : ISODate("2012-05-25T00:00:00Z"), "grade" : "B", "score" : 14 }, { "date" : ISODate("2011-09-14T00:00:00Z"), "grade" : "B", "score" : 26 }, { "date" : ISODate("2011-05-25T00:00:00Z"), "grade" : "A", "score" : 76 }, { "date" : ISODate("2011-01-01T00:00:00Z"), "grade" : "A", "score" : 11 }, "name" : "El Molino Rojo Restaurant", "restaurant_id" : "40393688" },
{ "id" : ObjectId("617b791aa9a1cd95b97e77"), "address" : { "building" : "289", "coord" : [ -73.94610279999999, 40.7137557 ], "street" : "Manhattan Avenue", "zipcode" : "11211" }, "borough" : "Brooklyn", "cuisine" : "Bakery", "grades" : [ { "date" : ISODate("2014-03-19T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2013-10-10T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2013-05-07T00:00:00Z"), "grade" : "A", "score" : 13 }, { "date" : ISODate("2012-09-11T00:00:00Z"), "grade" : "B", "score" : 18 }, { "date" : ISODate("2012-04-10T00:00:00Z"), "grade" : "A", "score" : 9 }, { "date" : ISODate("2011-09-27T00:00:00Z"), "grade" : "A", "score" : 8 }, { "date" : ISODate("2011-05-03T00:00:00Z"), "grade" : "C", "score" : 77 } ], "name" : "Fortunato Bros Cafe & Bakery", "restaurant_id" : "40080551" },
{ "id" : ObjectId("617b791aa9a1cd95b98491"), "address" : { "building" : "231", "coord" : [ -73.9772294, 40.7527262 ], "street" : "Grand Central Station", "zipcode" : "10017" }, "borough" : "Manhattan", "cuisine" : "Italian", "grades" : [ { "date" : ISODate("2015-01-23T00:00:00Z"), "grade" : "Z", "score" : 20 }, { "date" : ISODate("2014-07-03T00:00:00Z"), "grade" : "B", "score" : 14 }, { "date" : ISODate("2013-12-21T00:00:00Z"), "grade" : "A", "score" : 13 }, { "date" : ISODate("2013-05-17T00:00:00Z"), "grade" : "C", "score" : 76 }, { "date" : ISODate("2012-04-20T00:00:00Z"), "grade" : "A", "score" : 12 }, { "date" : ISODate("2012-01-01T00:00:00Z"), "grade" : "A", "score" : 13 }, "name" : "Two Boots Grand Central", "restaurant_id" : "40725591" } ]

```

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}
});
```

```
db.addresses.find(
  ... "cuisine": { $ne: "American" },
  ... "grades.score": { $gt: 70 },
  ... "address.coord": { $not: [-65.754168]
  ... })
  { "_id": ObjectId("617b9fda9a9a1c05b5b7b99"), "address": { "building": "945", "coord": [ -73.9864626, 40.7266739 ], "street": "East 6 Street", "zipcode": "10003", "borough": "Manhattan", "cuisine": "Indian", "grades": { { "date": ISODate("2014-08-15T00:00:00Z"), "grade": "A", "score": 5 }, { "date": ISODate("2014-01-14T00:00:00Z"), "grade": "A", "score": 8 }, { "date": ISODate("2013-03-30T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2013-04-24T00:00:00Z"), "grade": "B", "score": 2 }, { "date": ISODate("2012-10-01T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-04-06T00:00:00Z"), "grade": "C", "score": 92 }, { "date": ISODate("2011-01-03T00:00:00Z"), "grade": "C", "score": 41 }, { "name": "Gan dhi", "restaurant_id": "40831295" } },
  { "_id": ObjectId("617b9fda9a9a1c05b5b7b9c"), "address": { "building": "130", "coord": [ -73.984759, 40.7457939 ], "street": "Madison Avenue", "zipcode": "10016", "borough": "Manhattan", "cuisine": "Pizza/Italian", "grades": { { "date": ISODate("2014-12-24T00:00:00Z"), "grade": "Z", "score": 31 }, { "date": ISODate("2014-06-18T00:00:00Z"), "grade": "C", "score": 9 }, { "date": ISODate("2013-12-12T00:00:00Z"), "grade": "A", "score": 32 }, { "date": ISODate("2013-05-22T00:00:00Z"), "grade": "B", "score": 21 }, { "date": ISODate("2012-05-02T00:00:00Z"), "grade": "A", "score": 11 }, { "name": "Bella Napoli", "restaurant_id": "4093488" } },
  { "_id": ObjectId("617b9fda9a9a1c05b5b7b9d"), "address": { "building": "1", "coord": [ -73.9243961, 40.8726797 ], "street": "East 161 Street", "zipcode": "10451", "borough": "Bronx", "cuisine": "Latin (Cuban, Dominican, Puerto Rican, South & Central American)", "grades": { { "date": ISODate("2014-04-10T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-01-30T00:00:00Z"), "grade": "A", "score": 6 }, { "date": ISODate("2013-04-10T00:00:00Z"), "grade": "B", "score": 25 }, { "date": ISODate("2012-10-25T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2012-10-10T00:00:00Z"), "grade": "P", "score": 12 }, { "date": ISODate("2012-05-25T00:00:00Z"), "grade": "B", "score": 14 }, { "date": ISODate("2011-09-14T00:00:00Z"), "grade": "C", "score": 20 }, { "date": ISODate("2010-04-25T00:00:00Z"), "grade": "C", "score": 76 }, { "name": "El Molino Rojo Restaurant", "restaurant_id": "4093503" } },
  { "_id": ObjectId("617b9fda9a9a1c05b5b7b9e"), "address": { "building": "289", "coord": [ -73.94610275999999, 40.7137587 ], "street": "Manhattan Avenue", "zipcode": "11211", "borough": "Brooklyn", "cuisine": "Bakery", "grades": { { "date": ISODate("2014-03-19T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-10-10T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-05-07T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2012-09-11T00:00:00Z"), "grade": "B", "score": 18 }, { "date": ISODate("2012-04-06T00:00:00Z"), "grade": "A", "score": 8 }, { "date": ISODate("2011-05-03T00:00:00Z"), "grade": "C", "score": 77 }, { "name": "Fortunato Bros. Cakes & Bakes", "restaurant_id": "40400561" } },
  { "_id": ObjectId("617b9fda9a9a1c05b5b8491"), "address": { "building": "731", "coord": [ -73.9772294, 40.7527262 ], "street": "Grand Central Station", "zipcode": "10017", "borough": "Manhattan", "cuisine": "Italian", "grades": { { "date": ISODate("2015-01-07T00:00:00Z"), "grade": "Z", "score": 120 }, { "date": ISODate("2014-07-03T00:00:00Z"), "grade": "B", "score": 13 }, { "date": ISODate("2013-12-21T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2013-05-17T00:00:00Z"), "grade": "C", "score": 76 }, { "date": ISODate("2012-04-20T00:00:00Z"), "grade": "A", "score": 12 }, { "name": "Two Boots Grand Central", "restaurant_id": "40725591" } }
```

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

[illegible]

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}
);
```

```
db.addresses.find(
...   {name: /^Wil/},
...   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... );
{ "_id" : ObjectId("617bf91aa9a1cd505ba79b1"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "_id" : ObjectId("617bf91aa9a1cd505ba79b4"), "borough" : "Bronx", "cuisine" : "American", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "_id" : ObjectId("617bf921a9a1cd505ba87b9"), "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}
);
```

```
> db.addresses.find(
...   {name: /ces$/},
...   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... );
{"_id" : ObjectId("617bf91daa9a1cd505ba7e3d"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Pieces", "restaurant_id" : "40399910" }
{"_id" : ObjectId("617bf91daa9a1cd505ba7efc"), "borough" : "Queens", "cuisine" : "American ", "name" : "S.M.R Restaurant Services", "restaurant_id" : "40403857" }
{"_id" : ObjectId("617bf91daa9a1cd505ba7f92"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Good Shepherd Services", "restaurant_id" : "40403989" }
{"_id" : ObjectId("617bf91faa9a1cd505ba83b5"), "borough" : "Queens", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "The Ice Box-Ralph'S Famous Italian Ices", "restaurant_id" : "40698899" }
{"_id" : ObjectId("617bf921aa9a1cd505ba85b7"), "borough" : "Brooklyn", "cuisine" : "Jewish/Kosher", "name" : "Alices", "restaurant_id" : "40702042" }
{"_id" : ObjectId("617bf921aa9a1cd505ba87d3"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Re: Sources", "restaurant_id" : "40876068" }
```

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}
);
```

```
> db.addresses.find(
...   {name: /. *Reg.*/},
...   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... );
{"_id" : ObjectId("617bf91aa9a1cd505ba79b2"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7aaf"), "borough" : "Manhattan", "cuisine" : "Café/Coffee/Tea", "name" : "Caffe Reggio", "restaurant_id" : "40369418" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7bbe"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Regency Hotel", "restaurant_id" : "40382579" }
{"_id" : ObjectId("617bf91daa9a1cd505ba7e0b"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Regency Unit Club", "restaurant_id" : "40402377" }
{"_id" : ObjectId("617bf91daa9a1cd505ba7fbc"), "borough" : "Queens", "cuisine" : "American ", "name" : "Rego Park Cafe", "restaurant_id" : "40523342" }
{"_id" : ObjectId("617bf921aa9a1cd505ba862c"), "borough" : "Queens", "cuisine" : "Pizza", "name" : "Regina Pizza", "restaurant_id" : "40801325" }
{"_id" : ObjectId("617bf921aa9a1cd505ba8843"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Regal Entertainment Group", "restaurant_id" : "40891782" }
```

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" } ]
});
```

```
> db.addresses.find({ "borough": "Bronx",
...   $or : [ { "cuisine" : "American " }, { "cuisine" : "Chinese" } ]
... });
{"_id" : ObjectId("617bf91aa9a1cd505ba79b4"), "address" : { "building" : "2300", "coord" : [ -73.8786113, 40.8501883 ], "street" : "Southern Boulevard", "zipcode" : "10460" }, "borough" : "Bronx", "cuisine" : "American ", "grades" : [ { "date" : ISODate("2014-05-28T00:00:00Z"), "grade" : "A", "score" : 11 }, { "date" : ISODate("2013-06-19T00:00:00Z"), "grade" : "A", "score" : 4 } ], { "date" : ISODate("2012-06-15T00:00:00Z"), "grade" : "A", "score" : 3 } ], "name" : "Wild Asia", "restaurant_id" : "40357217" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79cd"), "address" : { "building" : "1236", "coord" : [ -73.8893654, 40.81376179999999 ], "street" : "238 Spofford Ave", "zipcode" : "10474" }, "borough" : "Bronx", "cuisine" : "Chinese", "grades" : [ { "date" : ISODate("2013-12-30T00:00:00Z"), "grade" : "A", "score" : 8 }, { "date" : ISODate("2013-01-08T00:00:00Z"), "grade" : "A", "score" : 18 }, { "date" : ISODate("2012-06-12T00:00:00Z"), "grade" : "B", "score" : 15 } ], "name" : "Happy Garden", "restaurant_id" : "40363289" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79df"), "address" : { "building" : "277", "coord" : [ -73.8941893, 40.8634684 ], "street" : "East Kingsbridge Road", "zipcode" : "10458" }, "borough" : "Bronx", "cuisine" : "Chinese", "grades" : [ { "date" : ISODate("2014-03-03T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2013-09-26T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2013-03-19T00:00:00Z"), "grade" : "A", "score" : 10 }, { "date" : ISODate("2012-08-29T00:00:00Z"), "grade" : "A", "score" : 11 }, { "date" : ISODate("2011-08-17T00:00:00Z"), "grade" : "A", "score" : 13 } ], "name" : "Happy Garden", "restaurant_id" : "40364296" }
```

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.

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

```

> db.addresses.find(
...   {"borough" : {$in : ["Staten Island","Queens","Bronx","Brooklyn"]}},
...   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... );
{"_id" : ObjectId("617bf91aa9a1cd505ba79a9"), "borough" : "Bronx", "cuisine" : "Bakery", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79ab"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79ac"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Riviera Caterer", "restaurant_id" : "40356018" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79ad"), "borough" : "Queens", "cuisine" : "Jewish/Kosher", "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79af"), "borough" : "Queens", "cuisine" : "American ", "name" : "Brunos On The Boulevard", "restaurant_id" : "40356151" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b0"), "borough" : "Staten Island", "cuisine" : "Jewish/Kosher", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b1"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b2"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Regina Caterers", "restaurant_id" : "40356649" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b3"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b4"), "borough" : "Bronx", "cuisine" : "American ", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b5"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b6"), "borough" : "Brooklyn", "cuisine" : "Chinese", "name" : "May Hay Kitchen", "restaurant_id" : "40358429" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b7"), "borough" : "Brooklyn", "cuisine" : "Jewish/Kosher", "name" : "Suda Foods", "restaurant_id" : "40360045" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b8"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79ba"), "borough" : "Queens", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79bb"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79bc"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "The Movable Feast", "restaurant_id" : "40361600" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79bd"), "borough" : "Queens", "cuisine" : "Delicatessen", "name" : "Sal'S Deli", "restaurant_id" : "40361618" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79c0"), "borough" : "Queens", "cuisine" : "Delicatessen", "name" : "Steve Chu'S Deli & Grocery", "restaurant_id" : "40361998" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79c4"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "White Castle", "restaurant_id" : "40362344" }
type "it" for more

```

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 Bronx or Brooklyn.

```

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

```

```

> db.addresses.find(
...   {"borough" :{$nin :["Staten Island","Queens","Bronx","Brooklyn"]}},
...   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... );
{"_id" : ObjectId("617bf91aa9a1cd505ba79ac"), "borough" : "Manhattan", "cuisine" : "Irish", "name" : "Dj Reynolds Pub And Restaurant", "restaurant_id" : "30191841" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b7"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79bc"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Glorious Food", "restaurant_id" : "40361511" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79bf"), "borough" : "Manhattan", "cuisine" : "Delicatessen", "name" : "Bully'S Deli", "restaurant_id" : "40361708" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79c1"), "borough" : "Manhattan", "cuisine" : "Chicken", "name" : "Harriet'S Kitchen", "restaurant_id" : "40362098" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79c2"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "P & S Deli Grocery", "restaurant_id" : "40362264" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79c3"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Angelika Film Center", "restaurant_id" : "40362274" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79c6"), "borough" : "Manhattan", "cuisine" : "Turkish", "name" : "The Country Cafe", "restaurant_id" : "40362715" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79c8"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Downtown Deli", "restaurant_id" : "40363021" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79cc"), "borough" : "Manhattan", "cuisine" : "Bakery", "name" : "Olive'S", "restaurant_id" : "40363151" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79ce"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Cafe Metro", "restaurant_id" : "40363298" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79d0"), "borough" : "Manhattan", "cuisine" : "Sandwiches/Salads/Mixed Buffet", "name" : "Lexler Deli", "restaurant_id" : "40363426" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79d4"), "borough" : "Manhattan", "cuisine" : "Continental", "name" : "Lorenzo & Maria'S", "restaurant_id" : "40363630" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79d5"), "borough" : "Manhattan", "cuisine" : "Pizza", "name" : "Domino'S Pizza", "restaurant_id" : "40363644" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79d6"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Berkely", "restaurant_id" : "40363685" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79d7"), "borough" : "Manhattan", "cuisine" : "Pizza", "name" : "Domino'S Pizza", "restaurant_id" : "40363945" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79db"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Spoon Bread Catering", "restaurant_id" : "40364179" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79e1"), "borough" : "Manhattan", "cuisine" : "Chicken", "name" : "Texas Rotisserie", "restaurant_id" : "40364304" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79e4"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Metropolitan Club", "restaurant_id" : "40364347" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79e5"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Palm Restaurant", "restaurant_id" : "40364355" }
type "it" for more

```

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}
);

```

```

> db.addresses.find(
...   {"grades.score" : { $not: { $gt : 10}}},
...   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... );
{"_id" : ObjectId("617bf91aa9a1cd505ba79b5"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "C & C Catering Service", "restaurant_id" : "40357437" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79b7"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "1 East 66Th Street Kitchen", "restaurant_id" : "40359480" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79bb"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79bd"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "White Castle", "restaurant_id" : "40362344" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79d7"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "Sonny'S Heros", "restaurant_id" : "40363744" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79e7"), "borough" : "Bronx", "cuisine" : "American ", "name" : "Manhenn Club", "restaurant_id" : "40364363" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79f5"), "borough" : "Staten Island", "cuisine" : "Italian", "name" : "Great Kills Yacht Club", "restaurant_id" : "40364610" }
{"_id" : ObjectId("617bf91aa9a1cd505ba79fc"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "Serendipity 3", "restaurant_id" : "40364863" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a00"), "borough" : "Manhattan", "cuisine" : "American ", "name" : "White Horse Tavern", "restaurant_id" : "40364958" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a09"), "borough" : "Manhattan", "cuisine" : "Irish", "name" : "Dorrian'S Red Hand Restaurant", "restaurant_id" : "40365239" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a5a"), "borough" : "Manhattan", "cuisine" : "Mexican", "name" : "Mexico Lindo Restaurant", "restaurant_id" : "40367038" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a7a"), "borough" : "Brooklyn", "cuisine" : "Greek", "name" : "El Greco Diner", "restaurant_id" : "40367795" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a80"), "borough" : "Bronx", "cuisine" : "Not Listed/Not Applicable", "name" : "The Lark'S Nest", "restaurant_id" : "40367946" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a81"), "borough" : "Bronx", "cuisine" : "Café/Coffee/Tea", "name" : "Terrace Cafe", "restaurant_id" : "40368018" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a82"), "borough" : "Bronx", "cuisine" : "African", "name" : "African Terrace", "restaurant_id" : "40368021" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a85"), "borough" : "Bronx", "cuisine" : "American ", "name" : "African Market (Baboon Cafe)", "restaurant_id" : "40368026" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a86"), "borough" : "Staten Island", "cuisine" : "Italian", "name" : "Roadhouse Restaurant", "restaurant_id" : "40368034" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a88"), "borough" : "Manhattan", "cuisine" : "French", "name" : "Pergola Des Artistes", "restaurant_id" : "40369139" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a8d"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Mcdonald'S", "restaurant_id" : "40369535" }
{"_id" : ObjectId("617bf91aa9a1cd505ba7a8f"), "borough" : "Brooklyn", "cuisine" : "American ", "name" : "The River Cafe", "restaurant_id" : "40369608" }
type "it" for more

```


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 : "Chinese"}}
    ]}
  ]},
  {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
);
```

```
type "it" for more
> db.addresses.find(
...   {$or: [{name: /^Wil/},
...     {"$and": [
...       {"cuisine": {$ne : "American"}},
...       {"cuisine": {$ne : "Chinese"}}
...     ]}
...   ]},
...   {"restaurant_id" : 1,"name":1,"borough":1,"cuisine" :1}
... );
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79aa"), "borough" : "Bronx", "cuisine" : "Bakery", "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ab"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "Wendy'S", "restaurant_id" : "30112340" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ac"), "borough" : "Manhattan", "cuisine" : "Irish", "name" : "DJ Reynolds Pub And Restaurant", "restaurant_id" : "30101841" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ad"), "borough" : "Queens", "cuisine" : "Jewish/Kosher", "name" : "Tov Kosher Kitchen", "restaurant_id" : "40356068" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ae"), "borough" : "Staten Island", "cuisine" : "Jewish/Kosher", "name" : "Kosher Island", "restaurant_id" : "40356442" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79af"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Wilken'S Fine Food", "restaurant_id" : "40356483" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b0"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Taste The Tropics Ice Cream", "restaurant_id" : "40356731" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b1"), "borough" : "Bronx", "cuisine" : "American", "name" : "Wild Asia", "restaurant_id" : "40357217" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b2"), "borough" : "Brooklyn", "cuisine" : "Jewish/Kosher", "name" : "Seuda Foods", "restaurant_id" : "40360045" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b3"), "borough" : "Brooklyn", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40360076" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b4"), "borough" : "Queens", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40361322" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b5"), "borough" : "Brooklyn", "cuisine" : "Delicatessen", "name" : "Nordic Delicacies", "restaurant_id" : "40361390" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b6"), "borough" : "Queens", "cuisine" : "Delicatessen", "name" : "Sal'S Deli", "restaurant_id" : "40361618" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b7"), "borough" : "Manhattan", "cuisine" : "Delicatessen", "name" : "Bully'S Deli", "restaurant_id" : "40361708" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b8"), "borough" : "Queens", "cuisine" : "Delicatessen", "name" : "Steve Chu'S Deli & Grocery", "restaurant_id" : "40361998" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79b9"), "borough" : "Manhattan", "cuisine" : "Chicken", "name" : "Harriet'S Kitchen", "restaurant_id" : "40362098" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79ba"), "borough" : "Brooklyn", "cuisine" : "Hamburgers", "name" : "White Castle", "restaurant_id" : "40362344" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bb"), "borough" : "Manhattan", "cuisine" : "Turkish", "name" : "The Country Cafe", "restaurant_id" : "40362715" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bc"), "borough" : "Brooklyn", "cuisine" : "Caribbean", "name" : "Shashemene Int'L Restaura", "restaurant_id" : "40362869" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba79bd"), "borough" : "Bronx", "cuisine" : "Ice Cream, Gelato, Yogurt, Ices", "name" : "Carvel Ice Cream", "restaurant_id" : "40363093" }
type "it" for more
```

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}
);
```

```

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("617bf91aa9a1cd505ba7a28"), "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", "score": 10 }, { "date": ISODate("2011-08-24T00:00:00Z"), "grade": "A", "score": 11 } ], "name": "Neary'S Pub", "restaurant_id": "40365871" }
{ "_id": ObjectId("617bf91aa9a1cd505ba70b3"), "grades": [ { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2013-12-10T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2013-06-10T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2012-06-08T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2012-01-25T00:00:00Z"), "grade": "A", "score": 6 }, { "date": ISODate("2011-09-13T00:00:00Z"), "grade": "A", "score": 12 }, ], "name": "Don Filippo Restaurant", "restaurant_id": "40372417" }
{ "_id": ObjectId("617bf91daa9a1cd505ba7dcb"), "grades": [ { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 6 }, { "date": ISODate("2013-08-29T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2013-04-25T00:00:00Z"), "grade": "C", "score": 2 }, { "date": ISODate("2012-10-23T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2012-04-16T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-11-15T00:00:00Z"), "grade": "A", "score": 12 }, ], "name": "Mustang Sally'S Restaurant", "restaurant_id": "40397374" }
{ "_id": ObjectId("617bf91daa9a1cd505ba7fd5"), "grades": [ { "date": ISODate("2015-01-12T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2014-01-14T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2013-02-07T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-04-30T00:00:00Z"), "grade": "A", "score": 11 }, ], "name": "Club Macanudo (Cigar Bar)", "restaurant_id": "40526406" }
{ "_id": ObjectId("617bf91daa9a1cd505ba80ca"), "grades": [ { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 8 }, { "date": ISODate("2014-01-09T00:00:00Z"), "grade": "B", "score": 27 }, { "date": ISODate("2013-06-11T00:00:00Z"), "grade": "A", "score": 4 }, { "date": ISODate("2012-10-09T00:00:00Z"), "grade": "B", "score": 19 }, { "date": ISODate("2012-05-10T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-11-23T00:00:00Z"), "grade": "A", "score": 9 }, ], "name": "Marino'S Pizza & Restaurant", "restaurant_id": "40560917" }
{ "_id": ObjectId("617bf91faa9a1cd505ba8272"), "grades": [ { "date": ISODate("2015-01-05T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2014-03-04T00:00:00Z"), "grade": "B", "score": 22 }, { "date": ISODate("2013-09-13T00:00:00Z"), "grade": "A", "score": 5 }, { "date": ISODate("2013-01-16T00:00:00Z"), "grade": "A", "score": 11 }, ], "name": "Gene'S Coffee Shop", "restaurant_id": "40614916" }
{ "_id": ObjectId("617bf91faa9a1cd505ba83e0"), "grades": [ { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2014-02-20T00:00:00Z"), "grade": "A", "score": 7 }, { "date": ISODate("2013-08-29T00:00:00Z"), "grade": "A", "score": 7 }, { "date": ISODate("2013-03-29T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2012-03-07T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2011-11-16T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-10-12T00:00:00Z"), "grade": "P", "score": 4 }, ], "name": "Union Cafe Restaurant", "restaurant_id": "40698823" }
{ "_id": ObjectId("617bf921aa9a1cd505ba84d"), "grades": [ { "date": ISODate("2014-08-11T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-08-28T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-09-05T00:00:00Z"), "grade": "A", "score": 11 }, { "date": ISODate("2011-08-25T00:00:00Z"), "grade": "A", "score": 7 }, ], "name": "Jojo'S Pizza", "restaurant_id": "40892913" }

```

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"

```

b.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})

);

```

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})

);

```

```

> db.addresses.find(
...
{
  "address.coord.1": {$gt : 42, $lte : 52}
},
{"restaurant_id": 1,"name":1,"address":1,"coord":1})
...
};
{ "_id": ObjectId("617bf91aa9a1cd505ba7c4c"), "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("617bf91aa9a1cd505ba7c78"), "address": { "building": "11", "coord": [ -0.7119979, 51.6514664 ], "street": "Pennplaza E, Penn Sta", "zipcode": "10001" }, "name": "T.G.I. Fridays", "restaurant_id": "40388936" }
{ "_id": ObjectId("617bf91daa9a1cd505ba7ed1"), "address": { "building": "3000", "coord": [ -87.86567699999999, 42.61150920000001 ], "street": "47 Avenue", "zipcode": "11101" }, "name": "Ol' Luvido'S Deli", "restaurant_id": "40402104" }
{ "_id": ObjectId("617bf91daa9a1cd505ba106c"), "address": { "building": "21972199", "coord": [ -78.589606, 42.8912372 ], "street": "Broadway", "zipcode": "10024" }, "name": "La Caridad 78", "restaurant_id": "40568285" }
{ "_id": ObjectId("617bf921aa9a1cd505ba87d8"), "address": { "building": "7981", "coord": [ -84.9751215, 45.4713351 ], "street": "Hoyt Street", "zipcode": "11201" }, "name": "Bijan'S", "restaurant_id": "40876010" }
{ "_id": ObjectId("617bf91aa9a1cd505ba87ed"), "address": { "building": "0", "coord": [ -88.0778799, 42.4154769 ], "street": "& Grand Central", "zipcode": "10017" }, "name": "Hyatt, Ny Central/Room Service", "restaurant_id": "40879243" }
{ "_id": ObjectId("617bf921aa9a1cd505ba8803"), "address": { "building": "60", "coord": [ -111.9975205, 42.0970258 ], "street": "West Side Highway", "zipcode": "10006" }, "name": "Sports Center At Chelsea Piers (Sushi Bar)", "restaurant_id": "40882356" }

```

```
db.addresses.find().sort({"name":1});
```

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}
);
```

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},
);
```

```
db.addresses.find(
```



```
db.addresses.find(
  {'grades.score':
    {$mod : [7,0]}
  }, {
    "restaurant_id": 1,"name":"Grades:11"}
);
{"_id": ObjectId("617b791aa9a1cd95ba79aa"), "grades": [ { "date": ISODate("2014-03-03T00:00:00Z"), "grade": "A", "score": 2 }, { "date": ISODate("2013-09-11T00:00:00Z"), "grade": "A", "score": 5 }, { "date": ISODate("2013-08-21T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2014-01-11T23:00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2011-03-10T00:00:00Z"), "grade": "B", "score": 14 } ], "name": "Morris Park Bake Shop", "restaurant_id": "430875445"}
{"_id": ObjectId("617b791aa9a1cd95ba79ad"), "grades": [ { "date": ISODate("2014-06-10T00:00:00Z"), "grade": "A", "score": 5 }, { "date": ISODate("2013-06-05T00:00:00Z"), "grade": "A", "score": 7 }, { "date": ISODate("2012-04-13T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2011-10-12T00:00:00Z"), "grade": "A", "score": 12 } ], "name": "Riviera Catering & Restaurant", "restaurant_id": "40355615"}
{"_id": ObjectId("617b791aa9a1cd95ba79af"), "grades": [ { "date": ISODate("2014-11-15T00:00:00Z"), "grade": "Z", "score": 38 }, { "date": ISODate("2014-05-02T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2013-03-02T00:00:00Z"), "grade": "A", "score": 7 }, { "date": ISODate("2012-02-10T00:00:00Z"), "grade": "A", "score": 13 } ], "name": "Brunos On The Boulevard", "restaurant_id": "40356151"}
{"_id": ObjectId("617b791aa9a1cd95ba79b6"), "grades": [ { "date": ISODate("2014-09-16T00:00:00Z"), "grade": "B", "score": 21 }, { "date": ISODate("2013-08-26T00:00:00Z"), "grade": "A", "score": 56 }, { "date": ISODate("2012-08-15T00:00:00Z"), "grade": "B", "score": 27 }, { "date": ISODate("2012-03-28T00:00:00Z"), "grade": "B", "score": 27 } ], "name": "May May Kitchen", "restaurant_id": "40358429"}
{"_id": ObjectId("617b791aa9a1cd95ba79b7"), "grades": [ { "date": ISODate("2014-05-07T00:00:00Z"), "grade": "A", "score": 3 }, { "date": ISODate("2013-05-03T00:00:00Z"), "grade": "A", "score": 4 }, { "date": ISODate("2012-04-13T00:00:00Z"), "grade": "A", "score": 6 }, { "date": ISODate("2011-12-27T00:00:00Z"), "grade": "A", "score": 0 }, { "date": ISODate("2011-01-06T00:00:00Z"), "grade": "A", "score": 13 }, { "date": ISODate("2012-10-04T00:00:00Z"), "grade": "A", "score": 7 }, { "date": ISODate("2012-05-21T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2011-12-30T00:00:00Z"), "grade": "B", "score": 19 } ], "name": "Seuda Foods", "restaurant_id": "40360045"}
{"_id": ObjectId("617b791aa9a1cd95ba79bc"), "grades": [ { "date": ISODate("2014-08-16T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2013-08-27T00:00:00Z"), "grade": "A", "score": 9 }, { "date": ISODate("2012-09-20T00:00:00Z"), "grade": "A", "score": 7 }, { "date": ISODate("2011-09-29T00:00:00Z"), "grade": "A", "score": 10 } ], "name": "Sal S Deli", "restaurant_id": "40361618"}
{"_id": ObjectId("617b791aa9a1cd95ba79c0"), "grades": [ { "date": ISODate("2014-03-17T00:00:00Z"), "grade": "A", "score": 3 }, { "date": ISODate("2013-03-13T00:00:00Z"), "grade": "A", "score": 12 }, { "date": ISODate("2012-03-27T00:00:00Z"), "grade": "A", "score": 8 }, { "date": ISODate("2011-04-05T00:00:00Z"), "grade": "A", "score": 7 } ], "name": "Steve Chu'S De Grando", "restaurant_id": "40351989"}
{"_id": ObjectId("617b791aa9a1cd95ba79c1"), "grades": [ { "date": ISODate("2014-09-15T00:00:00Z"), "grade": "A", "score": 10 }, { "date": ISODate("2014-03-04T00:00:00Z"), "grade": "A", "score": 12 }
```

```
db.addresses.find(
```

```

> db.addresses.find(
...   { name :
...     { $regex : "mon.*", $options: "i" }
...   },
...   {
...     "name":1,
...     "borough":1,
...     "address.coord":1,
...     "cuisine" :1
...   }
... );
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a3e"), "address" : { "coord" : [ -73.98306099999999, 40.7441419 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Desmond'S Tavern" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a47"), "address" : { "coord" : [ -73.8221418, 40.7272376 ] }, "borough" : "Queens", "cuisine" : "Jewish/Kosher", "name" : "Shimons Kosher Pizza" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a53"), "address" : { "coord" : [ -74.10465599999999, 40.58834 ] }, "borough" : "Staten Island", "cuisine" : "American ", "name" : "Richmond County Coun
try Club" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7a7e"), "address" : { "coord" : [ -73.9812843, 40.5947365 ] }, "borough" : "Brooklyn", "cuisine" : "Pizza/Italian", "name" : "Lb Spumoni Gardens" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7ad0"), "address" : { "coord" : [ -73.951199, 40.7166026 ] }, "borough" : "Brooklyn", "cuisine" : "Italian", "name" : "Bamonte'S Restaurant" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7b07"), "address" : { "coord" : [ -73.924072, 40.76108900000001 ] }, "borough" : "Queens", "cuisine" : "Greek", "name" : "Omonia Cafe" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7b57"), "address" : { "coord" : [ -74.0023553, 40.7333573 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Manhattan Monster" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7b84"), "address" : { "coord" : [ -74.001094, 40.729583 ] }, "borough" : "Manhattan", "cuisine" : "Italian", "name" : "Monte'S" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7b8a"), "address" : { "coord" : [ -73.9901605, 40.7526176 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Delmonico'S Kitchen" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7ba0"), "address" : { "coord" : [ -73.9979536, 40.6914024 ] }, "borough" : "Brooklyn", "cuisine" : "Bottled beverages, including water, sodas, juices, e
tc.", "name" : "Montero Bar & Grill" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7d2b"), "address" : { "coord" : [ -73.9707505, 40.7635651 ] }, "borough" : "Manhattan", "cuisine" : "Delicatessen", "name" : "Delmonico Gourmet" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7d98"), "address" : { "coord" : [ -73.9706637, 40.7508686 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Delmonico Restaurant" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7dac"), "address" : { "coord" : [ -73.9705633, 40.7605759 ] }, "borough" : "Manhattan", "cuisine" : "Italian", "name" : "Montebello Restaurant" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7e44"), "address" : { "coord" : [ -74.1914658, 40.55274360000001 ] }, "borough" : "Staten Island", "cuisine" : "Italian", "name" : "Villa Monte Pizzeria
" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7eef"), "address" : { "coord" : [ -73.97198209999999, 40.764464 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Harmonie Club" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7fb3"), "address" : { "coord" : [ -73.79571990000001, 40.7095637 ] }, "borough" : "Queens", "cuisine" : "Bakery", "name" : "Ramona'S Bakery" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba7fe7"), "address" : { "coord" : [ -73.97852100000001, 40.729192 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Mona'S" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba800c"), "address" : { "coord" : [ -73.98463480000001, 40.7630755 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Neil Simon Theatre" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba8047"), "address" : { "coord" : [ -73.9938361, 40.6091317 ] }, "borough" : "Brooklyn", "cuisine" : "Bakery", "name" : "Mondial Bakery" }
{ "_id" : ObjectId("617bf91aaa9a1cd505ba806e"), "address" : { "coord" : [ -74.028486, 40.630438 ] }, "borough" : "Brooklyn", "cuisine" : "Mediterranean", "name" : "Omonia Cafe" }
type "it" for more

```

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

  }

);

```

```

> db.addresses.find(
...   { name :
...     { $regex : /^Mad/i, }
...   },
...   {
...     "name":1,
...     "borough":1,
...     "address.coord":1,
...     "cuisine" :1
...   }
... );
{ "_id" : ObjectId("617bf91daa9a1cd505ba7ee6"), "address" : { "coord" : [ -73.9860597, 40.7431194 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Madison Square" }
{ "_id" : ObjectId("617bf91daa9a1cd505ba7fb4"), "address" : { "coord" : [ -73.98302199999999, 40.742313 ] }, "borough" : "Manhattan", "cuisine" : "Indian", "name" : "Madras Mahal" }
{ "_id" : ObjectId("617bf91faa9a1cd505ba8252"), "address" : { "coord" : [ -74.000002, 40.72735 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Madame X" }
{ "_id" : ObjectId("617bf91faa9a1cd505ba8312"), "address" : { "coord" : [ -73.98171959999999, 40.749406 ] }, "borough" : "Manhattan", "cuisine" : "French", "name" : "Madison Bistro" }
{ "_id" : ObjectId("617bf91faa9a1cd505ba839b"), "address" : { "coord" : [ -73.9717845, 40.6897199 ] }, "borough" : "Brooklyn", "cuisine" : "African", "name" : "Madiba" }
{ "_id" : ObjectId("617bf921aa9a1cd505ba86a0"), "address" : { "coord" : [ -73.9040753, 40.9069011 ] }, "borough" : "Bronx", "cuisine" : "Italian", "name" : "Madison'S" }
{ "_id" : ObjectId("617bf921aa9a1cd505ba871e"), "address" : { "coord" : [ -73.9886598, 40.7565811 ] }, "borough" : "Manhattan", "cuisine" : "Hotdogs", "name" : "Madame Tussaud'S" }
{ "_id" : ObjectId("617bf921aa9a1cd505ba8756"), "address" : { "coord" : [ -73.95623719999999, 40.7761697 ] }, "borough" : "Manhattan", "cuisine" : "American ", "name" : "Mad River Bar & Gri

```