MONGODB\_VARSHINI.S

-----------------------------------------------------ASSN1--------------------------------------------------------------------------

> show dbs;

admin 0.000GB

config 0.000GB

local 0.000GB

movies 0.000GB

users 0.000GB

> use users;

switched to db users

> show dbs;

admin 0.000GB

config 0.000GB

local 0.000GB

movies 0.000GB

users 0.000GB

> show collections

movies

> db.movies.insert({title:"Fight Club", writer: "Chuck Palahniuk", year: "1999", actors:["Brad Pitt", "Edward Norton"]})

WriteResult({ "nInserted" : 1 })

> db.movies.find()

{ "\_id" : ObjectId("6104eaba4f2b23e6df471cfe"), "title" : "fight club", "writer" : "chuck palahniuko", "year" : 1999, "actors" : [ "brad pitt", "edward norton" ] }

{ "\_id" : ObjectId("6104ed914f2b23e6df471cff"), "title" : "pulp fiction", "writer" : "quentin tarantino", "year" : 1994, "actors" : [ "john travolta", "uma thurman" ] }

{ "\_id" : ObjectId("6104ed914f2b23e6df471d00"), "title" : "inglorious basterds", "writer" : "quentin tarantino", "year" : 2009, "actors" : [ "brad pitt", "diane kruger", "eli roth" ] }

{ "\_id" : ObjectId("6104eeccf5aef474cca3c9c7"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

> db.movies.drop()

true

> db.movies.insert()

uncaught exception: Error: no object passed to insert! :

DBCollection.prototype.insert@src/mongo/shell/collection.js:275:15

@(shell):1:1

> db.createCollection("movies")

{ "ok" : 1 }

> show collections

movies

> db.movies.insert({title:"Fight Club", writer: "Chuck Palahniuk", year: "1999", actors:["Brad Pitt", "Edward Norton"]})

WriteResult({ "nInserted" : 1 })

> db.movies.find()

{ "\_id" : ObjectId("6104ef47f5aef474cca3c9c8"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

> db.movies.insert({title:"Pulp Fiction", writer:"Quentin Tarantino", year:"2009", actors:["John Travolta", "Uma Thurman"]})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"Inglorious Basterds", writer:"Quentin Tarantino", year:"2009", actors:["Brad Pitt", "Diane Kruger", "Eli Roth"]})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"The Hobbit: An unexpected Journey", writer:"J.R.R. Tolkein", year:"2012",franchise:"The Hobbit"})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"The Hobbit: The Desolation of Smaug", writer:"J.R.R Tolkien", year:"2013", franchise:"The Hobbit"})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"The Hobbit: The Battle of the Five Armies", writer:"J.R.R Tolkien", year:"2002", 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."})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"Pee Wee Herman's Big Adventures"})

WriteResult({ "nInserted" : 1 })

> db.movies.insert({title:"Avatar"})

WriteResult({ "nInserted" : 1 })

>

> db.movies.find()

{ "\_id" : ObjectId("6104ef47f5aef474cca3c9c8"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("6104ef62f5aef474cca3c9c9"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : ObjectId("6104ef70f5aef474cca3c9ca"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

{ "\_id" : ObjectId("6104ef7af5aef474cca3c9cb"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("6104ef87f5aef474cca3c9cc"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("6104ef96f5aef474cca3c9cd"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "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("6104efa0f5aef474cca3c9ce"), "title" : "Pee Wee Herman's Big Adventures" }

{ "\_id" : ObjectId("6104efadf5aef474cca3c9cf"), "title" : "Avatar" }

> db.movies.insert({title:"Avatar"})

WriteResult({ "nInserted" : 1 })

>

> db.movies.find()

{ "\_id" : ObjectId("6104ef47f5aef474cca3c9c8"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("6104ef62f5aef474cca3c9c9"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : ObjectId("6104ef70f5aef474cca3c9ca"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

{ "\_id" : ObjectId("6104ef7af5aef474cca3c9cb"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("6104ef87f5aef474cca3c9cc"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("6104ef96f5aef474cca3c9cd"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "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("6104efa0f5aef474cca3c9ce"), "title" : "Pee Wee Herman's Big Adventures" }

{ "\_id" : ObjectId("6104efadf5aef474cca3c9cf"), "title" : "Avatar" }

{ "\_id" : ObjectId("6104efc5f5aef474cca3c9d0"), "title" : "Avatar" }

> db.movies.find({writer:"Quentin Tarantino"})

{ "\_id" : ObjectId("6104ef62f5aef474cca3c9c9"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : ObjectId("6104ef70f5aef474cca3c9ca"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

> db.movies.find({actors:"Brad Pitt"})

{ "\_id" : ObjectId("6104ef47f5aef474cca3c9c8"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("6104ef70f5aef474cca3c9ca"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

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

{ "\_id" : ObjectId("6104ef7af5aef474cca3c9cb"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("6104ef87f5aef474cca3c9cc"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("6104ef96f5aef474cca3c9cd"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "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." }

> db.movies.find({year:{$gt:"1990", $lt:"2000"})

...

...

> db.movies.find({year:{$gt:"1990",$lt:"2000"}})

{ "\_id" : ObjectId("6104ef47f5aef474cca3c9c8"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

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

{ "\_id" : ObjectId("6104ef47f5aef474cca3c9c8"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("6104ef7af5aef474cca3c9cb"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("6104ef87f5aef474cca3c9cc"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

>

UPDATE

>db.movies.update({\_id:ObjectId("6104ef7af5aef474cca3c9cb")},{$set:{synopsis:"A reluctant hobbit,Bilbo baggins,srts out to the Lonely Mountain with a sprited group of dwarves to reclaim their mountain home- the gold within it-from the dragon Smaug."}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>db.movies.update({\_id:ObjectId("6104ef87f5aef474cca3c9cc")}, {$set:{synopsis:"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>db.movies.update({\_id:ObjectId("6104ef62f5aef474cca3c9c9")}, {$push:{actors:"Samuel L. Jackson"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

db.movies.find()

{ "\_id" : ObjectId("6104ef47f5aef474cca3c9c8"), "title" : "Fight Club", "writer" : "Chuck Palahniuk", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("6104ef62f5aef474cca3c9c9"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "John Travolta", "Uma Thurman", "Samuel L. Jackson" ] }

{ "\_id" : ObjectId("6104ef70f5aef474cca3c9ca"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

{ "\_id" : ObjectId("6104ef7af5aef474cca3c9cb"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit", "synopsis" : "A reluctant hobbit,Bilbo baggins,srts out to the Lonely Mountain with a sprited group of dwarves to reclaim their mountain home- the gold within it-from the dragon Smaug." }

{ "\_id" : ObjectId("6104ef87f5aef474cca3c9cc"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit", "synopsis" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring." }

{ "\_id" : ObjectId("6104ef96f5aef474cca3c9cd"), "title" : "The Hobbit: The Battle of the Five Armies", "writer" : "J.R.R Tolkien", "year" : "2002", "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("6104efa0f5aef474cca3c9ce"), "title" : "Pee Wee Herman's Big Adventures" }

{ "\_id" : ObjectId("6104efadf5aef474cca3c9cf"), "title" : "Avatar" }

{ "\_id" : ObjectId("6104efc5f5aef474cca3c9d0"), "title" : "Avatar" }

DELETE

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

WriteResult({ "nRemoved" : 0 })

> db.movies.remove({title:"Avatar"})

WriteResult({ "nRemoved" : 2 })

--

------------------------------------------RELATIONSHIP-------------------------------------------------------------------------------

> show dbs;

admin 0.000GB

config 0.000GB

local 0.000GB

movies 0.000GB

users 0.000GB

> use users;

switched to db users

> show dbs;

admin 0.000GB

config 0.000GB

local 0.000GB

movies 0.000GB

users 0.000GB

> show collections

movies

> db.movies.drop()

true

> db.createCollection("users")

{ "ok" : 1 }

> db.users.insert({\_id:1,username:"GoodGuyGreg", first\_name:"Good Guy", last\_name:"Greg"})

WriteResult({ "nInserted" : 1 })

> db.users.insert({username:"GoogGuyGreg", first\_name:"Good Guy", last\_name:"Greg", username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"}})

WriteResult({ "nInserted" : 1 })

> db.createCollection("posts")

{ "ok" : 1 }

> db.posts.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Raises your credit score"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({ username:"GoodGuyGreg", title:"Steals your identity", body:"Raises your credit score"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({username:"GoodGuyGreg", title:"Reports a bug in your code", body:"Sends you a pull request"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({ username:"ScumbagSteve", title:"Borrows something", body:"Sells it"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({ username:"ScumbagSteve", title:"Borrows everything", body:"The end"})

WriteResult({ "nInserted" : 1 })

> db.posts.insert({username:"ScumbagSteve", title:"Forks your repo on github", body:"Sets to private"})

WriteResult({ "nInserted" : 1 })

> db.users.find()

{ "\_id" : 1, "username" : "GoodGuyGreg", "first\_name" : "Good Guy", "last\_name" : "Greg" }

{ "\_id" : ObjectId("6105038e44b81ba1da1f1f1c"), "username" : "ScumbagSteve", "first\_name" : "Good Guy", "last\_name" : "Greg", "fullname" : { "first" : "Scumbag", "last" : "Steve" } }

> db.posts.find()

{ "\_id" : ObjectId("610503e344b81ba1da1f1f1d"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("610503f144b81ba1da1f1f1e"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("610503fb44b81ba1da1f1f1f"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("6105040744b81ba1da1f1f20"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }

{ "\_id" : ObjectId("6105041344b81ba1da1f1f21"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }

{ "\_id" : ObjectId("6105041e44b81ba1da1f1f22"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }

{ "\_id" : ObjectId("6105042844b81ba1da1f1f23"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }

> db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("6105041344b81ba1da1f1f21"")})

uncaught exception: SyntaxError: "" literal not terminated before end of script :

@(shell):1:126

> db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("6105041344b81ba1da1f1f21")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("6105041e44b81ba1da1f1f22")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("6105042844b81ba1da1f1f23")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("610503e344b81ba1da1f1f1d")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("6105040744b81ba1da1f1f20")})

WriteResult({ "nInserted" : 1 })

>

> db.users.find()

{ "\_id" : 1, "username" : "GoodGuyGreg", "first\_name" : "Good Guy", "last\_name" : "Greg" }

{ "\_id" : ObjectId("6105038e44b81ba1da1f1f1c"), "username" : "ScumbagSteve", "first\_name" : "Good Guy", "last\_name" : "Greg", "fullname" : { "first" : "Scumbag", "last" : "Steve" } }

> db.posts.find()

{ "\_id" : ObjectId("610503e344b81ba1da1f1f1d"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("610503f144b81ba1da1f1f1e"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("610503fb44b81ba1da1f1f1f"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("6105040744b81ba1da1f1f20"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }

{ "\_id" : ObjectId("6105041344b81ba1da1f1f21"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }

{ "\_id" : ObjectId("6105041e44b81ba1da1f1f22"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }

{ "\_id" : ObjectId("6105042844b81ba1da1f1f23"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }

> db.posts.find().pretty()

{

"\_id" : ObjectId("610503e344b81ba1da1f1f1d"),

"username" : "GoodGuyGreg",

"title" : "Passes out at Party",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("610503f144b81ba1da1f1f1e"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("610503fb44b81ba1da1f1f1f"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("6105040744b81ba1da1f1f20"),

"username" : "GoodGuyGreg",

"title" : "Reports a bug in your code",

"body" : "Sends you a pull request"

}

{

"\_id" : ObjectId("6105041344b81ba1da1f1f21"),

"username" : "ScumbagSteve",

"title" : "Borrows something",

"body" : "Sells it"

}

{

"\_id" : ObjectId("6105041e44b81ba1da1f1f22"),

"username" : "ScumbagSteve",

"title" : "Borrows everything",

"body" : "The end"

}

{

"\_id" : ObjectId("6105042844b81ba1da1f1f23"),

"username" : "ScumbagSteve",

"title" : "Forks your repo on github",

"body" : "Sets to private"

}

> db.posts.find({username:"GoodGuyGreg"})

{ "\_id" : ObjectId("610503e344b81ba1da1f1f1d"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("610503f144b81ba1da1f1f1e"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("610503fb44b81ba1da1f1f1f"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("6105040744b81ba1da1f1f20"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull request" }

> db.posts.find({username:"GoodGuyGreg"}).pretty()

{

"\_id" : ObjectId("610503e344b81ba1da1f1f1d"),

"username" : "GoodGuyGreg",

"title" : "Passes out at Party",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("610503f144b81ba1da1f1f1e"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("610503fb44b81ba1da1f1f1f"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("6105040744b81ba1da1f1f20"),

"username" : "GoodGuyGreg",

"title" : "Reports a bug in your code",

"body" : "Sends you a pull request"

}

> db.posts.find({username:"ScumbagSteve"})

{ "\_id" : ObjectId("6105041344b81ba1da1f1f21"), "username" : "ScumbagSteve", "title" : "Borrows something", "body" : "Sells it" }

{ "\_id" : ObjectId("6105041e44b81ba1da1f1f22"), "username" : "ScumbagSteve", "title" : "Borrows everything", "body" : "The end" }

{ "\_id" : ObjectId("6105042844b81ba1da1f1f23"), "username" : "ScumbagSteve", "title" : "Forks your repo on github", "body" : "Sets to private" }

> db.comment.find()

> db.comments.find()

{ "\_id" : ObjectId("6105052044b81ba1da1f1f24"), "username" : "GoodGuyGreg", "comment" : "Hope you got a good deal!", "post" : ObjectId("6105041344b81ba1da1f1f21") }

{ "\_id" : ObjectId("6105066e44b81ba1da1f1f25"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : ObjectId("6105041e44b81ba1da1f1f22") }

{ "\_id" : ObjectId("6105067b44b81ba1da1f1f26"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement!", "post" : ObjectId("6105042844b81ba1da1f1f23") }

{ "\_id" : ObjectId("6105068444b81ba1da1f1f27"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : ObjectId("610503e344b81ba1da1f1f1d") }

{ "\_id" : ObjectId("6105068d44b81ba1da1f1f28"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : ObjectId("6105040744b81ba1da1f1f20") }

> db.commentsfind({username:"GoodGuyGreg"})

uncaught exception: TypeError: db.commentsfind is not a function :

@(shell):1:1

> db.comments.find({username:"GoodGuyGreg"})

{ "\_id" : ObjectId("6105052044b81ba1da1f1f24"), "username" : "GoodGuyGreg", "comment" : "Hope you got a good deal!", "post" : ObjectId("6105041344b81ba1da1f1f21") }

{ "\_id" : ObjectId("6105066e44b81ba1da1f1f25"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : ObjectId("6105041e44b81ba1da1f1f22") }

{ "\_id" : ObjectId("6105067b44b81ba1da1f1f26"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement!", "post" : ObjectId("6105042844b81ba1da1f1f23") }

> db.comments.find({username:"ScumbagSteve"})

{ "\_id" : ObjectId("6105068444b81ba1da1f1f27"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : ObjectId("610503e344b81ba1da1f1f1d") }

{ "\_id" : ObjectId("6105068d44b81ba1da1f1f28"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : ObjectId("6105040744b81ba1da1f1f20") }

> db.comments.find({title:"Reports a bug in your code"})

> db.comments.find({title:"Reports a bug in your code"})

>

---------------------------------assn2------------------------------------------------------------------------------------------

> show dbs;

admin 0.000GB

config 0.000GB

local 0.000GB

movies 0.000GB

population 0.002GB

users 0.000GB

> show collections;

> use population;

switched to db population

> show collections;

zipcodes

> db.zipcodes.aggregate([])

{ "\_id" : "01007", "city" : "BELCHERTOWN", "loc" : [ -72.410953, 42.275103 ], "pop" : 10579, "state" : "MA" }

{ "\_id" : "01005", "city" : "BARRE", "loc" : [ -72.108354, 42.409698 ], "pop" : 4546, "state" : "MA" }

{ "\_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338, "state" : "MA" }

{ "\_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state" : "MA" }

{ "\_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240, "state" : "MA" }

{ "\_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" : "MA" }

{ "\_id" : "01010", "city" : "BRIMFIELD", "loc" : [ -72.188455, 42.116543 ], "pop" : 3706, "state" : "MA" }

{ "\_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177, "state" : "MA" }

{ "\_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396, "state" : "MA" }

{ "\_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495, "state" : "MA" }

{ "\_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864, "state" : "MA" }

{ "\_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" : 13367, "state" : "MA" }

{ "\_id" : "01030", "city" : "FEEDING HILLS", "loc" : [ -72.675077, 42.07182 ], "pop" : 11985, "state" : "MA" }

{ "\_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" : "MA" }

{ "\_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385, "state" : "MA" }

{ "\_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" : "MA" }

{ "\_id" : "01022", "city" : "WESTOVER AFB", "loc" : [ -72.558657, 42.196672 ], "pop" : 1764, "state" : "MA" }

{ "\_id" : "01035", "city" : "HADLEY", "loc" : [ -72.571499, 42.36062 ], "pop" : 4231, "state" : "MA" }

{ "\_id" : "01036", "city" : "HAMPDEN", "loc" : [ -72.431823, 42.064756 ], "pop" : 4709, "state" : "MA" }

{ "\_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" : "MA" }

Type "it" for more

> db.zipcodes.aggregate({$match:{$and:[{city:"ATLANTA"},{state:"GA"}]}})

{ "\_id" : "30303", "city" : "ATLANTA", "loc" : [ -84.388846, 33.752504 ], "pop" : 1845, "state" : "GA" }

{ "\_id" : "30305", "city" : "ATLANTA", "loc" : [ -84.385145, 33.831963 ], "pop" : 19122, "state" : "GA" }

{ "\_id" : "30307", "city" : "ATLANTA", "loc" : [ -84.335957, 33.769138 ], "pop" : 16330, "state" : "GA" }

{ "\_id" : "30306", "city" : "ATLANTA", "loc" : [ -84.351418, 33.786027 ], "pop" : 20081, "state" : "GA" }

{ "\_id" : "30309", "city" : "ATLANTA", "loc" : [ -84.388338, 33.798407 ], "pop" : 14766, "state" : "GA" }

{ "\_id" : "30308", "city" : "ATLANTA", "loc" : [ -84.375744, 33.771839 ], "pop" : 8549, "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" : "30310", "city" : "ATLANTA", "loc" : [ -84.423173, 33.727849 ], "pop" : 34017, "state" : "GA" }

{ "\_id" : "30324", "city" : "ATLANTA", "loc" : [ -84.354867, 33.820609 ], "pop" : 15044, "state" : "GA" }

{ "\_id" : "30317", "city" : "ATLANTA", "loc" : [ -84.31685, 33.749788 ], "pop" : 16395, "state" : "GA" }

{ "\_id" : "30319", "city" : "ATLANTA", "loc" : [ -84.335091, 33.868728 ], "pop" : 32138, "state" : "GA" }

{ "\_id" : "30327", "city" : "ATLANTA", "loc" : [ -84.419966, 33.862723 ], "pop" : 18467, "state" : "GA" }

{ "\_id" : "30326", "city" : "ATLANTA", "loc" : [ -84.358232, 33.848168 ], "pop" : 125, "state" : "GA" }

{ "\_id" : "30329", "city" : "ATLANTA", "loc" : [ -84.321402, 33.823555 ], "pop" : 17013, "state" : "GA" }

{ "\_id" : "30330", "city" : "ATLANTA", "loc" : [ -84.434735, 33.70645 ], "pop" : 643, "state" : "GA" }

Type "it" for more

> db.zipcodes.find({$and:[{city:"ATLANTA"},{state:"GA"}]})

{ "\_id" : "30303", "city" : "ATLANTA", "loc" : [ -84.388846, 33.752504 ], "pop" : 1845, "state" : "GA" }

{ "\_id" : "30305", "city" : "ATLANTA", "loc" : [ -84.385145, 33.831963 ], "pop" : 19122, "state" : "GA" }

{ "\_id" : "30307", "city" : "ATLANTA", "loc" : [ -84.335957, 33.769138 ], "pop" : 16330, "state" : "GA" }

{ "\_id" : "30306", "city" : "ATLANTA", "loc" : [ -84.351418, 33.786027 ], "pop" : 20081, "state" : "GA" }

{ "\_id" : "30309", "city" : "ATLANTA", "loc" : [ -84.388338, 33.798407 ], "pop" : 14766, "state" : "GA" }

{ "\_id" : "30308", "city" : "ATLANTA", "loc" : [ -84.375744, 33.771839 ], "pop" : 8549, "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" : "30310", "city" : "ATLANTA", "loc" : [ -84.423173, 33.727849 ], "pop" : 34017, "state" : "GA" }

{ "\_id" : "30324", "city" : "ATLANTA", "loc" : [ -84.354867, 33.820609 ], "pop" : 15044, "state" : "GA" }

{ "\_id" : "30317", "city" : "ATLANTA", "loc" : [ -84.31685, 33.749788 ], "pop" : 16395, "state" : "GA" }

{ "\_id" : "30319", "city" : "ATLANTA", "loc" : [ -84.335091, 33.868728 ], "pop" : 32138, "state" : "GA" }

{ "\_id" : "30327", "city" : "ATLANTA", "loc" : [ -84.419966, 33.862723 ], "pop" : 18467, "state" : "GA" }

{ "\_id" : "30326", "city" : "ATLANTA", "loc" : [ -84.358232, 33.848168 ], "pop" : 125, "state" : "GA" }

{ "\_id" : "30329", "city" : "ATLANTA", "loc" : [ -84.321402, 33.823555 ], "pop" : 17013, "state" : "GA" }

{ "\_id" : "30330", "city" : "ATLANTA", "loc" : [ -84.434735, 33.70645 ], "pop" : 643, "state" : "GA" }

Type "it" for more

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

{ "ATLANTA" : 16584 }

POPULATIONS BY STATE----------------------------------------------------------------------------------------

tot pop in each state

> db.zipcodes.aggregate([{$group: {\_id: "$state",population: {$sum: "$pop"}}}])

{ "\_id" : "TX", "population" : 16984601 }

{ "\_id" : "MS", "population" : 2573216 }

{ "\_id" : "NV", "population" : 1201833 }

{ "\_id" : "GA", "population" : 6478216 }

{ "\_id" : "AL", "population" : 4040587 }

{ "\_id" : "CO", "population" : 3293755 }

{ "\_id" : "OK", "population" : 3145585 }

{ "\_id" : "HI", "population" : 1108229 }

{ "\_id" : "SD", "population" : 695397 }

{ "\_id" : "MT", "population" : 798948 }

{ "\_id" : "UT", "population" : 1722850 }

{ "\_id" : "AR", "population" : 2350725 }

{ "\_id" : "CT", "population" : 3287116 }

{ "\_id" : "KS", "population" : 2475285 }

{ "\_id" : "CA", "population" : 29754890 }

{ "\_id" : "ME", "population" : 1226648 }

{ "\_id" : "NJ", "population" : 7730188 }

{ "\_id" : "FL", "population" : 12686644 }

{ "\_id" : "IN", "population" : 5544136 }

{ "\_id" : "VT", "population" : 562758 }

Type "it" for more

sort by pop count asc

> db.zipcodes.aggregate([{$group: {\_id:{state: "$state",population: {$sum: "$pop"}}}}, {$sort: {\_id: 1}}])

{ "\_id" : { "state" : "AK", "population" : 0 } }

{ "\_id" : { "state" : "AK", "population" : 1 } }

{ "\_id" : { "state" : "AK", "population" : 3 } }

{ "\_id" : { "state" : "AK", "population" : 4 } }

{ "\_id" : { "state" : "AK", "population" : 7 } }

{ "\_id" : { "state" : "AK", "population" : 8 } }

{ "\_id" : { "state" : "AK", "population" : 20 } }

{ "\_id" : { "state" : "AK", "population" : 27 } }

{ "\_id" : { "state" : "AK", "population" : 32 } }

{ "\_id" : { "state" : "AK", "population" : 42 } }

{ "\_id" : { "state" : "AK", "population" : 59 } }

{ "\_id" : { "state" : "AK", "population" : 63 } }

{ "\_id" : { "state" : "AK", "population" : 64 } }

{ "\_id" : { "state" : "AK", "population" : 68 } }

{ "\_id" : { "state" : "AK", "population" : 73 } }

{ "\_id" : { "state" : "AK", "population" : 77 } }

{ "\_id" : { "state" : "AK", "population" : 96 } }

{ "\_id" : { "state" : "AK", "population" : 99 } }

{ "\_id" : { "state" : "AK", "population" : 103 } }

{ "\_id" : { "state" : "AK", "population" : 107 } }

Type "it" for more

highest first

> db.zipcodes.aggregate([{$group: {\_id:{state: "$state",population: {$sum: "$pop"}}}}, {$sort: {\_id: -1}}])

{ "\_id" : { "state" : "WY", "population" : 33107 } }

{ "\_id" : { "state" : "WY", "population" : 29327 } }

{ "\_id" : { "state" : "WY", "population" : 25968 } }

{ "\_id" : { "state" : "WY", "population" : 24812 } }

{ "\_id" : { "state" : "WY", "population" : 23927 } }

{ "\_id" : { "state" : "WY", "population" : 22028 } }

{ "\_id" : { "state" : "WY", "population" : 21224 } }

{ "\_id" : { "state" : "WY", "population" : 20025 } }

{ "\_id" : { "state" : "WY", "population" : 15471 } }

{ "\_id" : { "state" : "WY", "population" : 15050 } }

{ "\_id" : { "state" : "WY", "population" : 13956 } }

{ "\_id" : { "state" : "WY", "population" : 12789 } }

{ "\_id" : { "state" : "WY", "population" : 12577 } }

{ "\_id" : { "state" : "WY", "population" : 11985 } }

{ "\_id" : { "state" : "WY", "population" : 11770 } }

{ "\_id" : { "state" : "WY", "population" : 9914 } }

{ "\_id" : { "state" : "WY", "population" : 9608 } }

{ "\_id" : { "state" : "WY", "population" : 9575 } }

{ "\_id" : { "state" : "WY", "population" : 9078 } }

{ "\_id" : { "state" : "WY", "population" : 7693 } }

Type "it" for more

limit by 3

> db.zipcodes.aggregate([{$limit: 3}, {$group: {\_id:{state: "$state",population: {$sum: "$pop"}}}}, {$sort: {\_id: -1}}])

{ "\_id" : { "state" : "MA", "population" : 15338 } }

{ "\_id" : { "state" : "MA", "population" : 10579 } }

{ "\_id" : { "state" : "MA", "population" : 4546 } }

-------------------------pop by city---------------------------------------------------------------------------------------------------

> db.zipcodes.aggregate([{$group: {\_id: {state: "$state",city: "$city",population: {$sum: "$pop"}}}}])

{ "\_id" : { "state" : "AR", "city" : "SOLGOHACHIA", "population" : 133 } }

{ "\_id" : { "state" : "MO", "city" : "BIXBY", "population" : 464 } }

{ "\_id" : { "state" : "UT", "city" : "MORGAN", "population" : 5411 } }

{ "\_id" : { "state" : "KS", "city" : "MAPLETON", "population" : 351 } }

{ "\_id" : { "state" : "MI", "city" : "ZEELAND", "population" : 15649 } }

{ "\_id" : { "state" : "NY", "city" : "NEW LEBANON", "population" : 557 } }

{ "\_id" : { "state" : "MO", "city" : "GOBLER", "population" : 144 } }

{ "\_id" : { "state" : "NY", "city" : "ROCHESTER", "population" : 15167 } }

{ "\_id" : { "state" : "MI", "city" : "ONSTED", "population" : 4601 } }

{ "\_id" : { "state" : "MN", "city" : "VICTORIA", "population" : 1367 } }

{ "\_id" : { "state" : "KS", "city" : "WELLS", "population" : 242 } }

{ "\_id" : { "state" : "VA", "city" : "WEYERS CAVE", "population" : 2190 } }

{ "\_id" : { "state" : "ND", "city" : "UNDERWOOD", "population" : 1629 } }

{ "\_id" : { "state" : "WI", "city" : "ELTON", "population" : 135 } }

{ "\_id" : { "state" : "KY", "city" : "SMILAX", "population" : 634 } }

{ "\_id" : { "state" : "WV", "city" : "BLUEWELL", "population" : 22561 } }

{ "\_id" : { "state" : "LA", "city" : "LONGVILLE", "population" : 1324 } }

{ "\_id" : { "state" : "NE", "city" : "NORFOLK", "population" : 25728 } }

{ "\_id" : { "state" : "NE", "city" : "OCONTO", "population" : 539 } }

{ "\_id" : { "state" : "NJ", "city" : "PEMBERTON", "population" : 11000 } }

Type "it" for more

> db.zipcodes.aggregate([{$limit: 3}, {$group: {\_id:{state: "$state",city: "$city", population: {$sum: "$pop"}}}}, {$sort: {\_id: -1}}])

{ "\_id" : { "state" : "MA", "city" : "BELCHERTOWN", "population" : 10579 } }

{ "\_id" : { "state" : "MA", "city" : "BARRE", "population" : 4546 } }

{ "\_id" : { "state" : "MA", "city" : "AGAWAM", "population" : 15338 } }

> db.zipcodes.aggregate([{$group: {\_id:{state: "$state",city: "TEXAS", population: {$sum: "$pop"}}}}, {$sort: {\_id: -1}}])

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 33107 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 29327 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 25968 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 24812 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 23927 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 22028 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 21224 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 20025 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 15471 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 15050 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 13956 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 12789 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 12577 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 11985 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 11770 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 9914 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 9608 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 9575 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 9078 } }

{ "\_id" : { "state" : "WY", "city" : "TEXAS", "population" : 7693 } }

Type "it" for more

---------------------------------------------BONUS--------------------------------------------------------------------------------

> db.zipcodes.aggregate([{$group:{"\_id":"\_id",pop\_avg: { $avg: "$pop" }}}])

{ "\_id" : "\_id", "pop\_avg" : 8462.794262937348 }

---------------------------------------------------------ASSN3--------------------------------------------------------------------

1)display whole document

> db.addresses.find()

2) display all fields

> db.addresses.find({},{"resturant\_is": 1, "name": 1, "borough": 1, "cusine": 1});

{ "\_id" : ObjectId("6107a30867eb3876f5352e58"), "borough" : "Brooklyn", "name" : "Wendy'S" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e59"), "borough" : "Manhattan", "name" : "Dj Reynolds Pub And Restaurant" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e5a"), "borough" : "Brooklyn", "name" : "Riviera Caterer" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e5b"), "borough" : "Bronx", "name" : "Morris Park Bake Shop" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e5c"), "borough" : "Staten Island", "name" : "Kosher Island" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e5d"), "borough" : "Queens", "name" : "Tov Kosher Kitchen" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e5e"), "borough" : "Queens", "name" : "Brunos On The Boulevard" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e5f"), "borough" : "Brooklyn", "name" : "Wilken'S Fine Food" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e60"), "borough" : "Brooklyn", "name" : "Taste The Tropics Ice Cream" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e61"), "borough" : "Brooklyn", "name" : "Regina Caterers" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e62"), "borough" : "Brooklyn", "name" : "C & C Catering Service" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e63"), "borough" : "Manhattan", "name" : "1 East 66Th Street Kitchen" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e64"), "borough" : "Brooklyn", "name" : "May May Kitchen" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e65"), "borough" : "Queens", "name" : "Carvel Ice Cream" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e66"), "borough" : "Brooklyn", "name" : "Seuda Foods" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e67"), "borough" : "Brooklyn", "name" : "Carvel Ice Cream" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e68"), "borough" : "Bronx", "name" : "Wild Asia" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e69"), "borough" : "Brooklyn", "name" : "Nordic Delicacies" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e6a"), "borough" : "Brooklyn", "name" : "The Movable Feast" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e6b"), "borough" : "Queens", "name" : "Sal'S Deli" }

Type "it" for more

3)exclude \_id

> db.addresses.find({},{"resturant\_is": 1, "name": 1, "borough": 1, "cusine": 1, "\_id": 0});

{ "borough" : "Brooklyn", "name" : "Wendy'S" }

{ "borough" : "Manhattan", "name" : "Dj Reynolds Pub And Restaurant" }

{ "borough" : "Brooklyn", "name" : "Riviera Caterer" }

{ "borough" : "Bronx", "name" : "Morris Park Bake Shop" }

{ "borough" : "Staten Island", "name" : "Kosher Island" }

{ "borough" : "Queens", "name" : "Tov Kosher Kitchen" }

{ "borough" : "Queens", "name" : "Brunos On The Boulevard" }

{ "borough" : "Brooklyn", "name" : "Wilken'S Fine Food" }

{ "borough" : "Brooklyn", "name" : "Taste The Tropics Ice Cream" }

{ "borough" : "Brooklyn", "name" : "Regina Caterers" }

{ "borough" : "Brooklyn", "name" : "C & C Catering Service" }

{ "borough" : "Manhattan", "name" : "1 East 66Th Street Kitchen" }

{ "borough" : "Brooklyn", "name" : "May May Kitchen" }

{ "borough" : "Queens", "name" : "Carvel Ice Cream" }

{ "borough" : "Brooklyn", "name" : "Seuda Foods" }

{ "borough" : "Brooklyn", "name" : "Carvel Ice Cream" }

{ "borough" : "Bronx", "name" : "Wild Asia" }

{ "borough" : "Brooklyn", "name" : "Nordic Delicacies" }

{ "borough" : "Brooklyn", "name" : "The Movable Feast" }

{ "borough" : "Queens", "name" : "Sal'S Deli" }

Type "it" for more

4)display zipcode under address field(address.zipcode)

> db.addresses.find({},{"resturant\_is": 1, "name": 1, "borough": 1, "address.zipcode": 1, "\_id": 0});

{ "address" : { "zipcode" : "11225" }, "borough" : "Brooklyn", "name" : "Wendy'S" }

{ "address" : { "zipcode" : "10019" }, "borough" : "Manhattan", "name" : "Dj Reynolds Pub And Restaurant" }

{ "address" : { "zipcode" : "11224" }, "borough" : "Brooklyn", "name" : "Riviera Caterer" }

{ "address" : { "zipcode" : "10462" }, "borough" : "Bronx", "name" : "Morris Park Bake Shop" }

{ "address" : { "zipcode" : "10314" }, "borough" : "Staten Island", "name" : "Kosher Island" }

{ "address" : { "zipcode" : "11374" }, "borough" : "Queens", "name" : "Tov Kosher Kitchen" }

{ "address" : { "zipcode" : "11369" }, "borough" : "Queens", "name" : "Brunos On The Boulevard" }

{ "address" : { "zipcode" : "11234" }, "borough" : "Brooklyn", "name" : "Wilken'S Fine Food" }

{ "address" : { "zipcode" : "11226" }, "borough" : "Brooklyn", "name" : "Taste The Tropics Ice Cream" }

{ "address" : { "zipcode" : "11219" }, "borough" : "Brooklyn", "name" : "Regina Caterers" }

{ "address" : { "zipcode" : "11214" }, "borough" : "Brooklyn", "name" : "C & C Catering Service" }

{ "address" : { "zipcode" : "10065" }, "borough" : "Manhattan", "name" : "1 East 66Th Street Kitchen" }

{ "address" : { "zipcode" : "11208" }, "borough" : "Brooklyn", "name" : "May May Kitchen" }

{ "address" : { "zipcode" : "11004" }, "borough" : "Queens", "name" : "Carvel Ice Cream" }

{ "address" : { "zipcode" : "11223" }, "borough" : "Brooklyn", "name" : "Seuda Foods" }

{ "address" : { "zipcode" : "11218" }, "borough" : "Brooklyn", "name" : "Carvel Ice Cream" }

{ "address" : { "zipcode" : "10460" }, "borough" : "Bronx", "name" : "Wild Asia" }

{ "address" : { "zipcode" : "11209" }, "borough" : "Brooklyn", "name" : "Nordic Delicacies" }

{ "address" : { "zipcode" : "11215" }, "borough" : "Brooklyn", "name" : "The Movable Feast" }

{ "address" : { "zipcode" : "11356" }, "borough" : "Queens", "name" : "Sal'S Deli" }

Type "it" for more

5)display borough bronx

>db.addresses.find({borough: "Bronx"})

6)q5 with display only top 5(limit to 5)

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

7)skip first 5, limit to 5

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

8)grades greater than 90($gt)

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

9)greater than 80 , lesser than 100

> db.addresses.find({grades : { $elemMatch:{"score":{$gt : 80, $lt: 100}}}}).pretty()

10)latitude less than -95.75

>db.addresses.find({"address.coord" : {$lt : -95.754168}});

11)cuisine not american,grades(under score field greater than 70),address coordinates- latitude les than -65.75

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

12)q11 without $and

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

13)cuisine nt american,grades A, borough not brooklyn , sort desc

> db.addresses.find({"cuisine": {$ne: "American"}},{"grades.grade": "A"},{"borough": {$ne: "Brooklyn"}}).sort({"cuisine": -1})

14)find name with first letters wil(text search $regex)

> db.addresses.find({"name":{$regex: /^Wil/}},{"resturant\_is": 1, "name": 1, "borough": 1, "cusine": 1})

{ "\_id" : ObjectId("6107a30867eb3876f5352e5f"), "borough" : "Brooklyn", "name" : "Wilken'S Fine Food" }

{ "\_id" : ObjectId("6107a30867eb3876f5352e68"), "borough" : "Bronx", "name" : "Wild Asia" }

{ "\_id" : ObjectId("6107a30967eb3876f5353c68"), "borough" : "Bronx", "name" : "Wilbel Pizza" }

15)end ces

> db.addresses.find({"name":{$regex: /ces$/}},{"resturant\_is": 1, "name": 1, "borough": 1, "cusine": 1})

16) somewhere reg

> db.addresses.find({"name":{$regex: /.\*Reg.\*/}},{"resturant\_is": 1, "name": 1, "borough": 1, "cusine": 1})

17)bronx, chinese or american

> db.addresses.find({borough: "Bronx", $or: [{"cuisine": "American"},{"cuisine": "Chinese"}]})

18)borough is or q or b or b

> db.addresses.find( {"borough" :{$in :["Staten Island","Queens","Bronx","Brooklyn"]}}, { "restaurant\_id" : 1, "name":1,"borough":1, "cuisine" :1 } )

19)q18 not in borough

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

20)value not greater than 10

> db.addresses.find( {"grades.score": {$not: {$gt: 10}}}, { "restaurant\_id" : 1, "name":1,"borough":1, "cuisine" :1 } )

21)

db.addresses.find({$or: [{name: /^Wil/},{"$and": [{"cuisine": {$ne :"American "}},{"cuisine" :{$ne :"Chinese"}}]}]} ,{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine":1})

22)

> db.addresses.find({"grades.date": ISODate("2014-08-11T00:00:00Z"), "grades.grade": "A", "grades.score": 11},{"restuarant\_id": 1, "name": 1, "grades":1})

25)resturant name asc order

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

26)resturant name in desc order

>db.addresses.find().sort({"name": -1}).pretty()

27)cuisine asc, borough desc

>db.addresses.find().sort({"cuisine": 1,"borough": -1})

28)address contains street or not

db.addresses.find({"address.street": {$exists: true}})

31)

> db.addresses.find({ "name" :{ $regex : /.\*mon.\*/}},{"name":1,"borough":1,"address.coord":1,"cuisine" :1})

32)

> db.addresses.find({ "name" :{ $regex : /^Mad/}},{"name":1,"borough":1,"address.coord":1,"cuisine" :1})