**Assignment lab-1**

**1.**

Microsoft Windows [Version 10.0.19042.985]

(c) Microsoft Corporation. All rights reserved.

C:\Users\sanke>mongo

MongoDB shell version v4.4.6

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("5d180afb-aa67-46db-bdd9-dfb8b708ebf7") }

MongoDB server version: 4.4.6

---

The server generated these startup warnings when booting:

2021-05-19T01:37:00.588+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

mongo\_practice 0.000GB

mydata 0.000GB

> use mongo\_practice

switched to db mongo\_practice

> show collections

movies

> db.movies.find().pretty()

> db.movies.find()

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

WriteResult({ "nInserted" : 1 })

> movies.insert({title:"Pulp Fiction", writer:"Quentin Tarantino", year:"1994", actors:["John Travolta", "Uma Thurman"]})

uncaught exception: ReferenceError: movies is not defined :

@(shell):1:1

> db.movies.insert({title:"Pulp Fiction", writer:"Quentin Tarantino", year:"1994", 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:"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."})

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().pretty()

{

"\_id" : ObjectId("60a5533d326b901242b2791e"),

"title" : "Fight Club",

"writer" : "Chuck Palahniuko",

"year" : "1999",

"actors" : [

"Brad Pitt",

"Edward Norton"

]

}

{

"\_id" : ObjectId("60a5538b326b901242b2791f"),

"title" : "Pulp Fiction",

"writer" : "Quentin Tarantino",

"year" : "1994",

"actors" : [

"John Travolta",

"Uma Thurman"

]

}

{

"\_id" : ObjectId("60a553ce326b901242b27920"),

"title" : "Inglorious Basterds",

"writer" : "Quentin Tarantino",

"year" : "2009",

"actors" : [

"Brad Pitt",

"Diane Kruger",

"Eli Roth"

]

}

{

"\_id" : ObjectId("60a55427326b901242b27921"),

"title" : "The Hobbit: An unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : "2012",

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("60a5546d326b901242b27922"),

"title" : "The Hobbit: The Desolation of Smaug",

"writer" : "J.R.R Tolkien",

"year" : "2013",

"franchise" : "The Hobbit"

}

{

"\_id" : ObjectId("60a554d8326b901242b27923"),

"title" : "The Hobbit: The Battle of the 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("60a554f4326b901242b27924"),

"title" : "Pee Wee Herman's Big Adventures"

}

{ "\_id" : ObjectId("60a5551e326b901242b27925"), "title" : "Avatar" }

**2.**

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

{ "\_id" : ObjectId("60a5538b326b901242b2791f"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "1994", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : ObjectId("60a553ce326b901242b27920"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

**3.**

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

{ "\_id" : ObjectId("60a5533d326b901242b2791e"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("60a553ce326b901242b27920"), "title" : "Inglorious Basterds", "writer" : "Quentin Tarantino", "year" : "2009", "actors" : [ "Brad Pitt", "Diane Kruger", "Eli Roth" ] }

**4.**

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

{ "\_id" : ObjectId("60a55427326b901242b27921"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("60a5546d326b901242b27922"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("60a554d8326b901242b27923"), "title" : "The Hobbit: The Battle of the 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." }

**5.**

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

{ "\_id" : ObjectId("60a5533d326b901242b2791e"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("60a5538b326b901242b2791f"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "1994", "actors" : [ "John Travolta", "Uma Thurman" ] }

**6.**

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

{ "\_id" : ObjectId("60a5533d326b901242b2791e"), "title" : "Fight Club", "writer" : "Chuck Palahniuko", "year" : "1999", "actors" : [ "Brad Pitt", "Edward Norton" ] }

{ "\_id" : ObjectId("60a5538b326b901242b2791f"), "title" : "Pulp Fiction", "writer" : "Quentin Tarantino", "year" : "1994", "actors" : [ "John Travolta", "Uma Thurman" ] }

{ "\_id" : ObjectId("60a55427326b901242b27921"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("60a5546d326b901242b27922"), "title" : "The Hobbit: The Desolation of Smaug", "writer" : "J.R.R Tolkien", "year" : "2013", "franchise" : "The Hobbit" }

{ "\_id" : ObjectId("60a554d8326b901242b27923"), "title" : "The Hobbit: The Battle of the 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." }

**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 group 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 and magical ring."}})

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

**3.**

> db.movies.update({"title" : "Pulp Fiction"}, {$push:{actors:"Samuel L. Jackson"}})

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

**Text Search**

**1.**

> db.movies.find({synopsis:{$regex:"Bilbo"}})

{ "\_id" : ObjectId("60a55427326b901242b27921"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit", "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." }

{ "\_id" : ObjectId("60a5546d326b901242b27922"), "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("60a554d8326b901242b27923"), "title" : "The Hobbit: The Battle of the 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." }

**2.**

> db.movies.find({synopsis:{$regex:"Gandalf"}})

{ "\_id" : ObjectId("60a5546d326b901242b27922"), "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." }

3.

db.movies.find({$and:[{synopsis:{$regex:/Bilbo/}}, {synopsis:{$not:/Gandalf/}}]}).pretty()

{

"\_id" : ObjectId("60a55427326b901242b27921"),

"title" : "The Hobbit: An unexpected Journey",

"writer" : "J.R.R. Tolkein",

"year" : "2012",

"franchise" : "The Hobbit",

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

}

{

"\_id" : ObjectId("60a554d8326b901242b27923"),

"title" : "The Hobbit: The Battle of the 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."

}

4.

db.movies.find({$or:[{synopsis:{$regex:"dwarves"}}, {synopsis:{$regex:"hobbit"}}]})

{ "\_id" : ObjectId("60a55427326b901242b27921"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit", "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." }

{ "\_id" : ObjectId("60a5546d326b901242b27922"), "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."

5.

> db.movies.find({$and:[{synopsis:{$regex:"gold"}}, {synopsis:{$regex:"dragon"}}]})

{ "\_id" : ObjectId("60a55427326b901242b27921"), "title" : "The Hobbit: An unexpected Journey", "writer" : "J.R.R. Tolkein", "year" : "2012", "franchise" : "The Hobbit", "synopsis" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug." }

**Delete Documents**

**1.**

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

WriteResult({ "nRemoved" : 1 })

**2.**

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

WriteResult({ "nRemoved" : 1 })

**Relationships**

**Insert docs.**

Microsoft Windows [Version 10.0.19042.985]

(c) Microsoft Corporation. All rights reserved.

C:\Users\sanke>mongo

MongoDB shell version v4.4.6

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("12953178-ff53-4023-a3d3-3471a1961092") }

MongoDB server version: 4.4.6

---

The server generated these startup warnings when booting:

2021-05-19T01:37:00.588+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted

---

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

mongo\_practice 0.000GB

mydata 0.000GB

> db

test

> use mongo\_practice

switched to db mongo\_practice

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

db.users.insert({username:"ScumbagSteve", full\_name:{first: "Scumbag", last:"Steve"}})

> db.users.find().pretty()

{

"\_id" : ObjectId("60a5cd38326b901242b27927"),

"username" : "GoodGuyGreg",

"first\_name" : "Good Guy",

"last\_name" : "Greg"

}

{

"\_id" : ObjectId("60a5cde5326b901242b27928"),

"username" : "ScumbagSteve",

"full\_name" : {

"first" : "Scumbag",

"last" : "Steve"

}

}

> db.createCollection('posts')

{ "ok" : 1 }

> db.posts.insert({username:"GoodGuyGreg", title:"Passes out at Party", body:"Wakes up early and cleans house"})

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 })

**Inset docs in comment collection**

> db.createCollection('comments')

{ "ok" : 1 }

> db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("60a5d198b24e44727128c7b7")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({ username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("60a5d1c4b24e44727128c7b8")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({ username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("60a5d1f7b24e44727128c7b9")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({ username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("60a5d0ccb24e44727128c7b5")})

WriteResult({ "nInserted" : 1 })

> db.comments.insert({ username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("60a5d319b24e44727128c7ba")})

WriteResult({ "nInserted" : 1 })

**1.**

> db.users.find().pretty()

{

"\_id" : ObjectId("60a5cd38326b901242b27927"),

"username" : "GoodGuyGreg",

"first\_name" : "Good Guy",

"last\_name" : "Greg"

}

{

"\_id" : ObjectId("60a5cde5326b901242b27928"),

"username" : "ScumbagSteve",

"full\_name" : {

"first" : "Scumbag",

"last" : "Steve"

}

}

**2.**

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

{

"\_id" : ObjectId("60a5d0ccb24e44727128c7b5"),

"username" : "GoodGuyGreg",

"title" : "Passes out at Party",

"body" : "Wakes up early and cleans house"

}

{

"\_id" : ObjectId("60a5d13ab24e44727128c7b6"),

"username" : "GoodGuyGreg",

"title" : "Steals your identity",

"body" : "Raises your credit score"

}

{

"\_id" : ObjectId("60a5d198b24e44727128c7b7"),

"username" : "Scumbagsteve",

"title" : "Borrows something",

"body" : "Sells it"

}

{

"\_id" : ObjectId("60a5d1c4b24e44727128c7b8"),

"username" : "Scumbagsteve",

"title" : "Borrows everything",

"body" : "The end"

}

{

"\_id" : ObjectId("60a5d1f7b24e44727128c7b9"),

"username" : "Scumbagsteve",

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

"body" : "Sets to private"

}

{

"\_id" : ObjectId("60a5d319b24e44727128c7ba"),

"username" : "GoodGuyGreg",

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

"body" : "Sends you a pull Request"

}

**3.**

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

{ "\_id" : ObjectId("60a5d0ccb24e44727128c7b5"), "username" : "GoodGuyGreg", "title" : "Passes out at Party", "body" : "Wakes up early and cleans house" }

{ "\_id" : ObjectId("60a5d13ab24e44727128c7b6"), "username" : "GoodGuyGreg", "title" : "Steals your identity", "body" : "Raises your credit score" }

{ "\_id" : ObjectId("60a5d319b24e44727128c7ba"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull Request" }

**4.**

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

{ "\_id" : ObjectId("60a5d198b24e44727128c7b7"), "username" : "Scumbagsteve", "title" : "Borrows something", "body" : "Sells it" }

{ "\_id" : ObjectId("60a5d1c4b24e44727128c7b8"), "username" : "Scumbagsteve", "title" : "Borrows everything", "body" : "The end" }

{ "\_id" : ObjectId("60a5d1f7b24e44727128c7b9"), "username" : "Scumbagsteve", "title" : "Forks your repo on github", "body" : "Sets to private" }

**5.**

> db.comments.find().pretty()

{

"\_id" : ObjectId("60a5d946b24e44727128c7be"),

"username" : "GoodGuyGreg",

"comment" : "Hope you got a good deal!",

"post" : ObjectId("60a5d198b24e44727128c7b7")

}

{

"\_id" : ObjectId("60a5d99cb24e44727128c7bf"),

"username" : "GoodGuyGreg",

"comment" : "What's mine is yours!",

"post" : ObjectId("60a5d1c4b24e44727128c7b8")

}

{

"\_id" : ObjectId("60a5da29b24e44727128c7c0"),

"username" : "GoodGuyGreg",

"comment" : "Don't violate the licensing agreement!",

"post" : ObjectId("60a5d1f7b24e44727128c7b9")

}

{

"\_id" : ObjectId("60a5dad1b24e44727128c7c1"),

"username" : "ScumbagSteve",

"comment" : "It still isn't clean",

"post" : ObjectId("60a5d0ccb24e44727128c7b5")

}

{

"\_id" : ObjectId("60a5db5fb24e44727128c7c2"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("60a5d319b24e44727128c7ba")

}

**6.**

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

{ "\_id" : ObjectId("60a5d946b24e44727128c7be"), "username" : "GoodGuyGreg", "comment" : "Hope you got a good deal!", "post" : ObjectId("60a5d198b24e44727128c7b7") }

{ "\_id" : ObjectId("60a5d99cb24e44727128c7bf"), "username" : "GoodGuyGreg", "comment" : "What's mine is yours!", "post" : ObjectId("60a5d1c4b24e44727128c7b8") }

{ "\_id" : ObjectId("60a5da29b24e44727128c7c0"), "username" : "GoodGuyGreg", "comment" : "Don't violate the licensing agreement!", "post" : ObjectId("60a5d1f7b24e44727128c7b9") }

**7.**

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

{ "\_id" : ObjectId("60a5dad1b24e44727128c7c1"), "username" : "ScumbagSteve", "comment" : "It still isn't clean", "post" : ObjectId("60a5d0ccb24e44727128c7b5") }

{ "\_id" : ObjectId("60a5db5fb24e44727128c7c2"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : ObjectId("60a5d319b24e44727128c7ba") }

**8.**

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

{ "\_id" : ObjectId("60a5d319b24e44727128c7ba"), "username" : "GoodGuyGreg", "title" : "Reports a bug in your code", "body" : "Sends you a pull Request" }

> db.comments.find({"post" : ObjectId("60a5d319b24e44727128c7ba")})

{ "\_id" : ObjectId("60a5db5fb24e44727128c7c2"), "username" : "ScumbagSteve", "comment" : "Denied your PR cause I found a hack", "post" : ObjectId("60a5d319b24e44727128c7ba") }

> db.comments.find({"post" : ObjectId("60a5d319b24e44727128c7ba")}).pretty()

{

"\_id" : ObjectId("60a5db5fb24e44727128c7c2"),

"username" : "ScumbagSteve",

"comment" : "Denied your PR cause I found a hack",

"post" : ObjectId("60a5d319b24e44727128c7ba")

}