1. **Connect to a running mongo instance, use a database named mongo\_practice.**

Command- use mongo\_practice

1. **Insert the following documents into a movies collection.**

**title : Fight Club**

**writer : Chuck Palahniuk**

**year : 1999**

**actors : [**

**Brad Pitt**

**Edward Norton**

**]**

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

**title : Pulp Fiction**

**writer : Quentin Tarantino**

**year : 1994**

**actors : [**

**John Travolta**

**Uma Thurman**

**]**

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

**title : Inglorious Basterds**

**writer : Quentin Tarantino**

**year : 2009**

**actors : [**

**Brad Pitt**

**Diane Kruger**

**Eli Roth**

**]**

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

**title : The Hobbit: An Unexpected Journey**

**writer : J.R.R. Tolkein**

**year : 2012**

**franchise : The Hobbit**

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

title : The Hobbit: The Desolation of Smaug

writer : J.R.R. Tolkein

year : 2013

franchise : The Hobbit

**command-** db.movies.insert({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**.

**Command-** 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."})

**title : Pee Wee Herman's Big Adventure**

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

**title : Avatar**

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

**QUERY/FIND DOCUMENT**

**query the movies collection to**

1. **get all documents**

command- db.movies.find()

1. **get all documents with writer set to "Quentin Tarantino"**

command- db.movies.find({writer:”Quentin Tarantino”})

1. **get all documents where actors include “Brad Pitt”**

command- db.movies.find({actors:”Brad Pitt”})

1. **get all documents with franchise set to “The Hobbit”**

command- db.movies.find({franchise:”The Hobbit”})

1. **get all movies released in the 90s**

command- db.movies.find({year:{$gt:”1990”, $lt:”2000”}})

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

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

**UPDATE DOCUMENTS**

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

**COMMAND-** db.movies.update({\_id:ObjectId("5c9f98e5e5c2dfe9b3729bfe")}, {$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."}})

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

**Command-** db.movies.update({\_id:ObjectId("5c9fa42ae5c2dfe9b3729c03")}, {$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."}})

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

**command-** db.movies.update({\_id:ObjectId("5c9f983ce5c2dfe9b3729bfc")}, {$push:{actors:"Samuel L. Jackson"}})

**TEXT SEARCH**

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

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

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

command- db.movies.find({synopsis:{$regex:”Gandalf”}})

1. **find all movies that have a synopsis that contains the word “Bilbo” and not the word “Gandalf”**

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

1. **find all movies that have a synopsis that contains the word “dwarves” or “hobbit”**

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

1. **find all movies that have a synopsis that contains the word “gold” and “dragon”**

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

**DELETE DOCUMENT**

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

command- db.movies.remove({\_id:ObjectId(“5c9f992ae5c2dfe9b3729c00”)})

1. **delete the movie “Avatar”**

command- db.movies.remove({\_id:ObjectId("5c9f9936e5c2dfe9b3729c01")})

**RELATIONSHIPS**

**Insert the following documents into a users collection**

**username : GoodGuyGreg**

**first\_name : "Good Guy"**

**last\_name : "Greg"**

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

**username : ScumbagSteve**

**full\_name :**

**first : "Scumbag"**

**last : "Steve"**

command- db.users.insert({\_id:2, username:"ScumbagSteve", fullname:{first: "Scumbag", last:"Steve"}})

**INSERT THE FOLLOWING INTO POSTS COLLECTIONS**

**username : GoodGuyGreg**

**title : Passes out at party**

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

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

**username : GoodGuyGreg**

**title : Steals your identity**

**body : Raises your credit score**

command-db.posts.insert({ 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**

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

**username : ScumbagSteve**

**title : Borrows something**

**body : Sells it**

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

**username : ScumbagSteve**

**title : Borrows everything**

**body : The end**

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

**username : ScumbagSteve**

**title : Forks your repo on github**

**body : Sets to private**

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

**INSERT THE FOLLOWING DOCUMENTS INTO COMMENTS SECTION**

**username : GoodGuyGreg**

**comment : Hope you got a good deal!**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Borrows something"**

**command-** db.comments.insert({ username:"GoodGuyGreg", comment:"Hope you got a good deal!", post:ObjectId("5ca0b7e96435f98b5901f463")})

**username : GoodGuyGreg**

**comment : What's mine is yours!**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Borrows everything"**

**command-** db.comments.insert({username:"GoodGuyGreg", comment:"What's mine is yours!", post:ObjectId("5ca0b9706435f98b5901f46a")})

**username : GoodGuyGreg**

**comment : Don't violate the licensing agreement!**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Forks your repo on github**

**command-** db.comments.insert({username:"GoodGuyGreg", comment:"Don't violate the licensing agreement!", post:ObjectId("5ca0b8766435f98b5901f467")})

**username : ScumbagSteve**

**comment : It still isn't clean**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Passes out at party"**

**command-** db.comments.insert({username:"ScumbagSteve", comment:"It still isn't clean", post:ObjectId("5ca0b8546435f98b5901f466")})

**username : ScumbagSteve**

**comment : Denied your PR cause I found a hack**

**post : [post\_obj\_id]**

**where [post\_obj\_id] is the ObjectId of the posts document: "Reports a bug in your code**

**command-** db.comments.insert({username:"ScumbagSteve", comment:"Denied your PR cause I found a hack", post:ObjectId("5ca0b9256435f98b5901f469")})

**QUERYING RELATED COLLECTIONS**

1. **find all users**

command- db.users.find().pretty()

1. **find all posts**

command- db.posts.find().pretty()

1. **find all posts that was authored by “GoodGuyGreg”**

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

1. **find all posts that was authored by "ScumbagSteve**"

command- db.posts.find({username:”ScumbagSteve”})

1. **find all comments**

command- db.comments.find().pretty()

1. **find all comments that was authored by “GoodGuyGreg”**

command- db.comments.find({username:”GoodGuyGreg”})

1. **find all comments that was authored by “ScumbagSteve”**

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

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

**MongoDB -Aggregation Exercises**

**Atlanta Population**

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

**and state is GA.**

Command- db.zipcodes.find({$and:[{city:”ATLANTA”,state:”GA”}]})

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

Command- db.zipcodes.aggregate({$match:{city:”ATLANTA”,state:”GA”}})

**Populations By State**

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

Command-db.zipcodes.aggregate({$group:{\_id:{state:’$state’},population:{$sum:’$pop’}}})

1. **sort the results by population, highest first**

command- db.zipcodes.aggregate({$group:{\_id:{state:’$state’},population:{$sum:’$pop’}}},{$sort:{population:-1}})

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

**population?**

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

**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' }**

command- db.zipcodes.aggregate({$group:{\_id:{city:‘$city’,state:’$state’},population:{$sum:’$pop’}}})

**2.sort the results by population, highest first**

Command- db.zipcodes.aggregate({$group:{\_id:{city:‘$city’,state:’$state’},population:{$sum:’$pop’}}},{$sort:{population:-1}})

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

**population?**

Command- db.zipcodes.aggregate({$group:{\_id:{city:‘$city’,state:’$state’},population:{$sum:’$pop’}}},{$sort:{population:-1}},{$limit:3})

**Bonus**

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

Command- db.zipcodes.aggregate({$group:{\_id:{city:‘$city’,state:’$state’},average:{$avg:’$pop’}}})

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

Command- db.zipcodes.aggregate({$group:{\_id:{city:‘$city’,state:’$state’},average:{$avg:’$pop’}}},{$limit:3})

**MONGODB COMPLEX QUERIES**

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

Command- db.restaurants.find()

1. **Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant**

Command- db.restaurants.find({}, {restaurant\_id:1, name:1,borough:1, cuisine:1 })

1. **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.**

Command- db.restaurants.find({}, {restaurant\_id:1, \_id:0, name:1,borough:1, cuisine:1 })

1. **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.**

Command- db.restaurants.find({}, {restaurant\_id:1, \_id:0, name:1,borough:1, "address.zipcode":1 })

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

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

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

Command- db.restaurants.find({"borough":"Bronx"})

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

Command- db.restaurants.find({"borough":"Bronx"}).limit(5).skip(5)

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

Command- db.restaurants.find({"grades.score":{$gt:90}})

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

Command- db.restaurants.find({"grades.score":{$gt:80, $lt:100}})

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

Command- db.restaurants.find({"address.coord":{$lt : -95.754168}})

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

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

**12**.**Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168.**

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

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

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

**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.**

Command- db.restaurants.find({name: /^Wil/},{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1});

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

Command- db.restaurants.find({name: /ces$/},{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1});

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

Command- db.restaurants.find({"name": /.\*Reg.\*/},{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1});

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

Command- db.restaurants.find({"borough": "Bronx" ,$or : [{ "cuisine" : "American " },{ "cuisine" : "Chinese" }]});

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

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

**19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.**

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

**20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.**

Command- db.restaurants.find({"grades.score" : { $not: {$gt : 10}}},{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1});

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

Command- db.restaurants.find({$or:[{name: /^Wil/},{"$and": [{"cuisine" : {$ne :"American "}},{"cuisine" : {$ne :"Chinees"}}]}]},{"restaurant\_id" : 1,"name":1,"borough":1,"cuisine" :1});

**22. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates..**

Command- db.restaurants.find({"grades.date": ISODate("2014-08-11T00:00:00Z"),"grades.grade":”A”,grades.score" : 11},{"restaurant\_id" : 1,"name":1,"grades":1});

**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".**

Command- db.restaurants.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.**

Command- db.restaurants.find({"address.coord.1": {$gt : 42, $lte : 52}},{"restaurant\_id" :1,"name":1,"address":1,"coord":1});

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

Command- db.restaurants.find().sort({"name":1});

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

Command- db.restaurants.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.**

Command- db.restaurants.find().sort({"cuisine":1,"borough" : -1,});

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

Command- db.restaurants.find({"address.street" :{ $exists : true }});

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

Command- db.restaurants.find({"address.coord" : {$type : 1}});

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

Command- db.restaurants.find({"grades.score" :{$mod : [7,0]}},{"restaurant\_id" : 1,"name":1,"grades":1});

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

Command- db.restaurants.find({name :{ $regex : "mon.\*", $options: "i" }},{"name":1,"borough":1,"address.coord":1,"cuisine" :1});

**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.**

Command- db.restaurants.find({name:{$regex:/^Mad/i,}},{"name":1,"borough":1,"address.coord":1,"cuisine" :1});