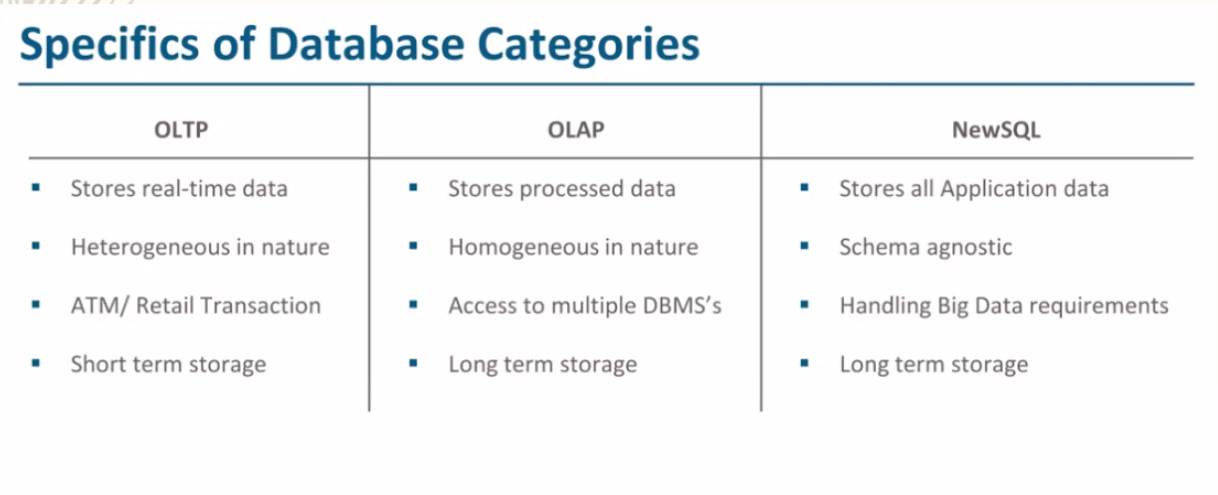
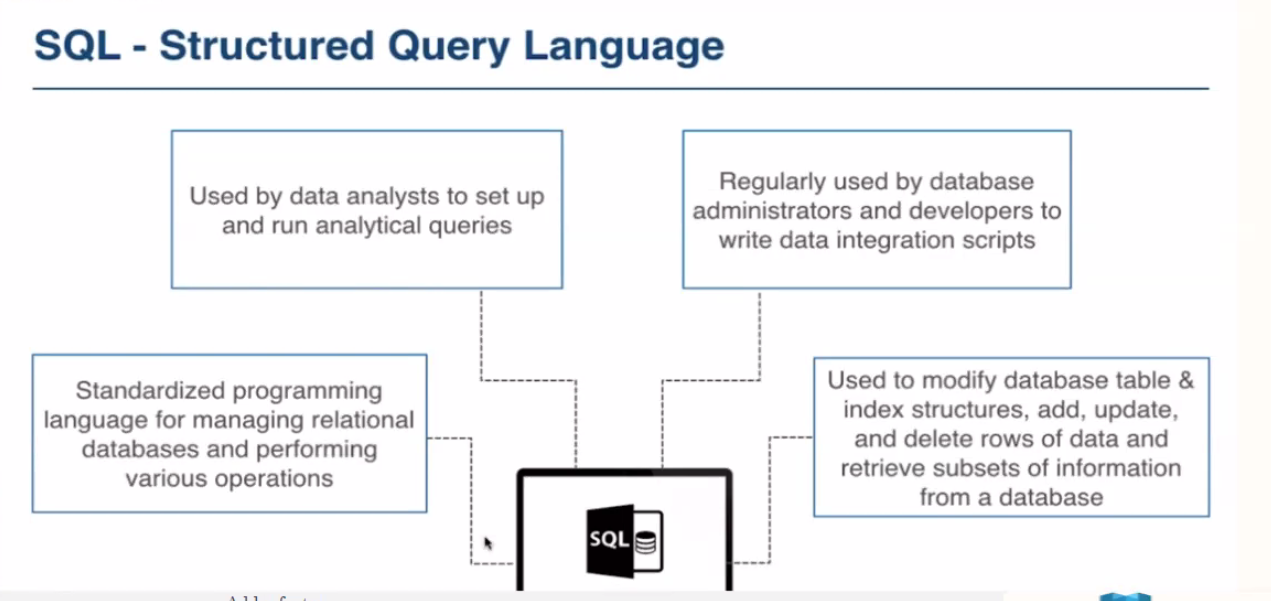
**MongoDB**



A screenshot of a computer

Description automatically generated

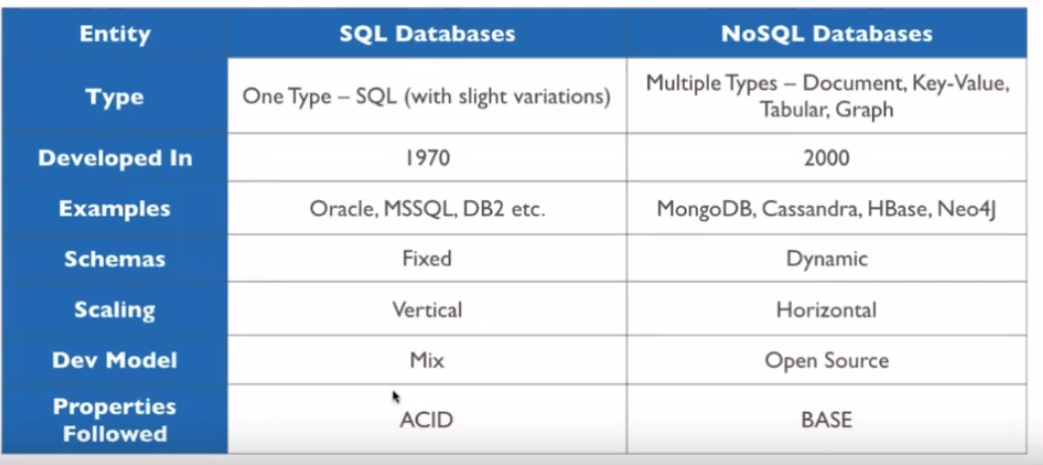


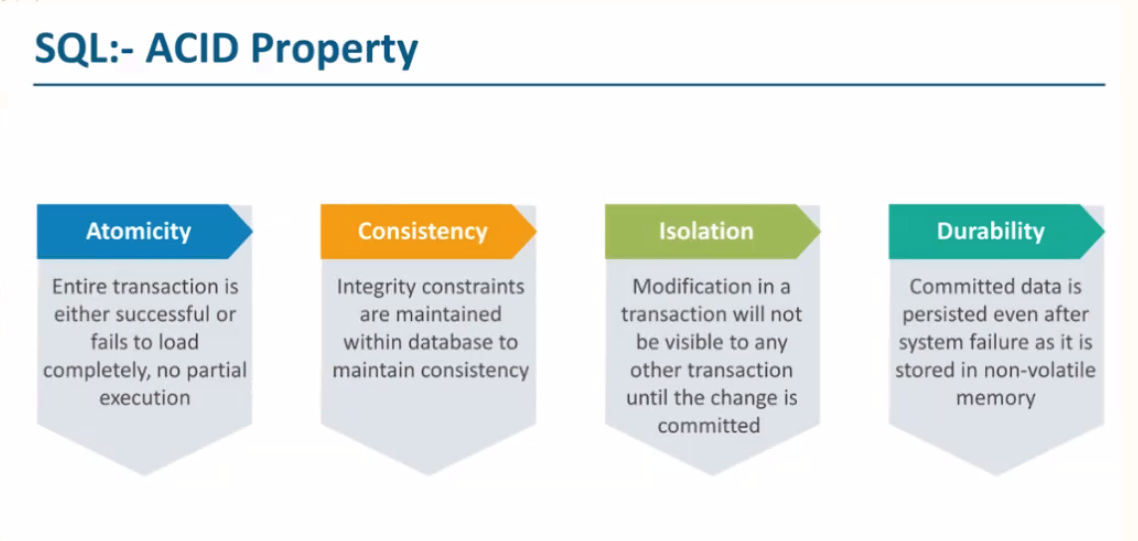


A diagram of a software company

Description automatically generated with medium confidence

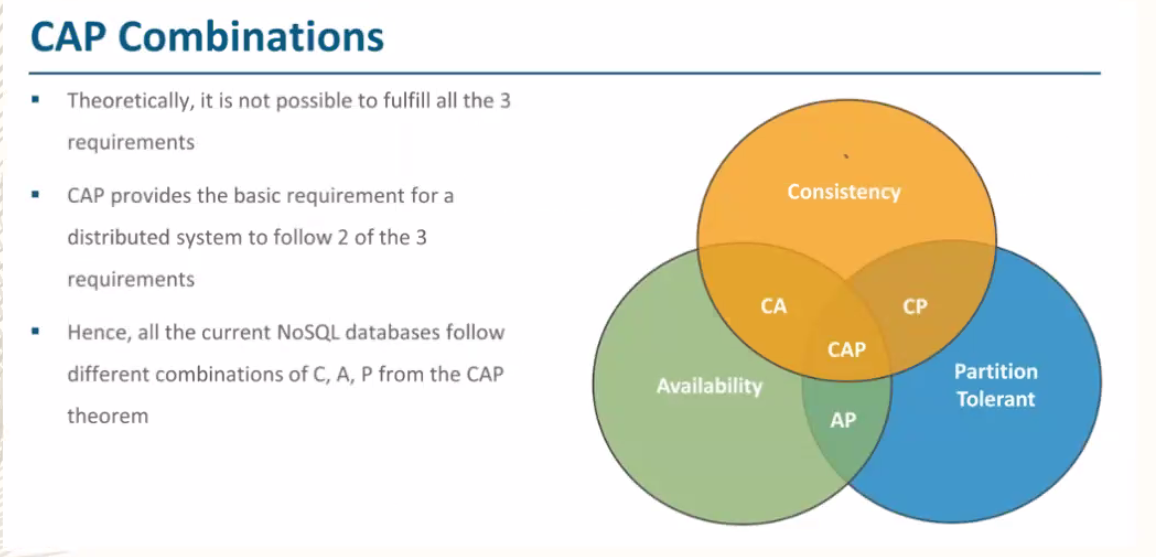
Increasing the storage or ram and cpu in same system 🡪verstical scalabale

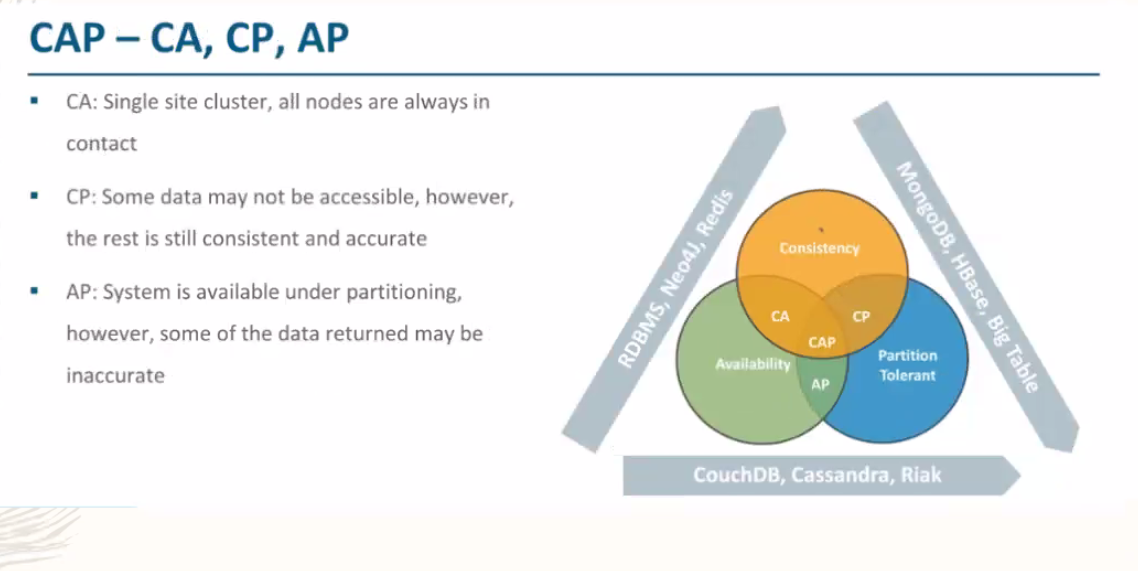
Difference between sql and nosql

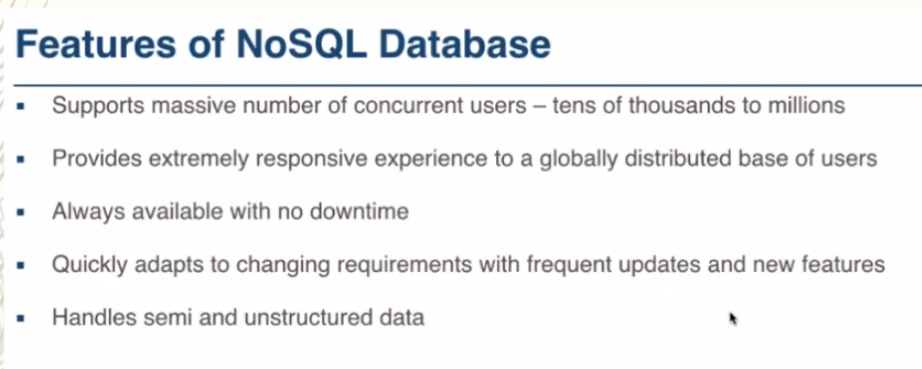




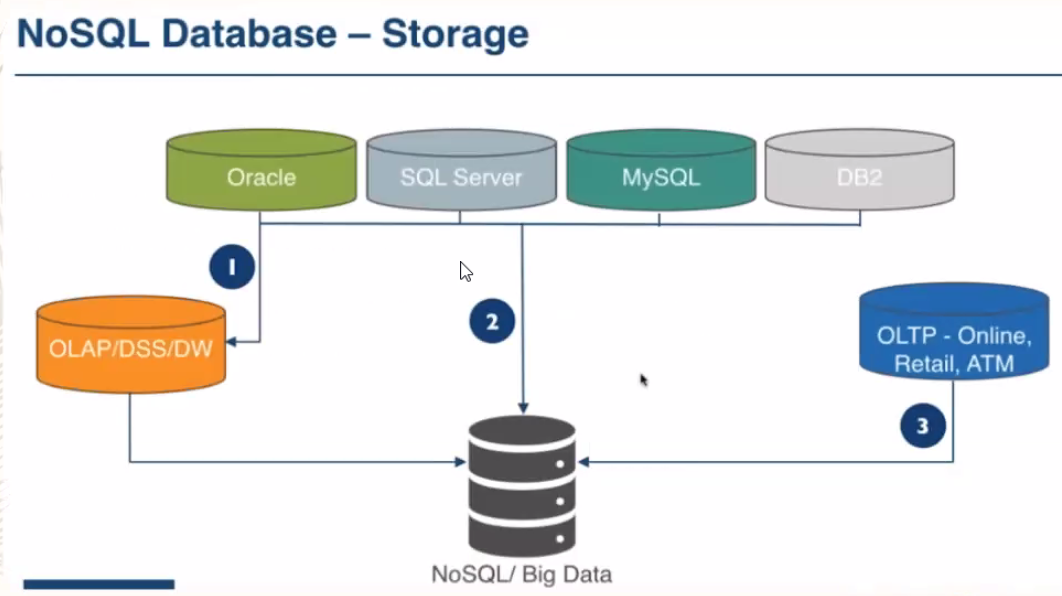




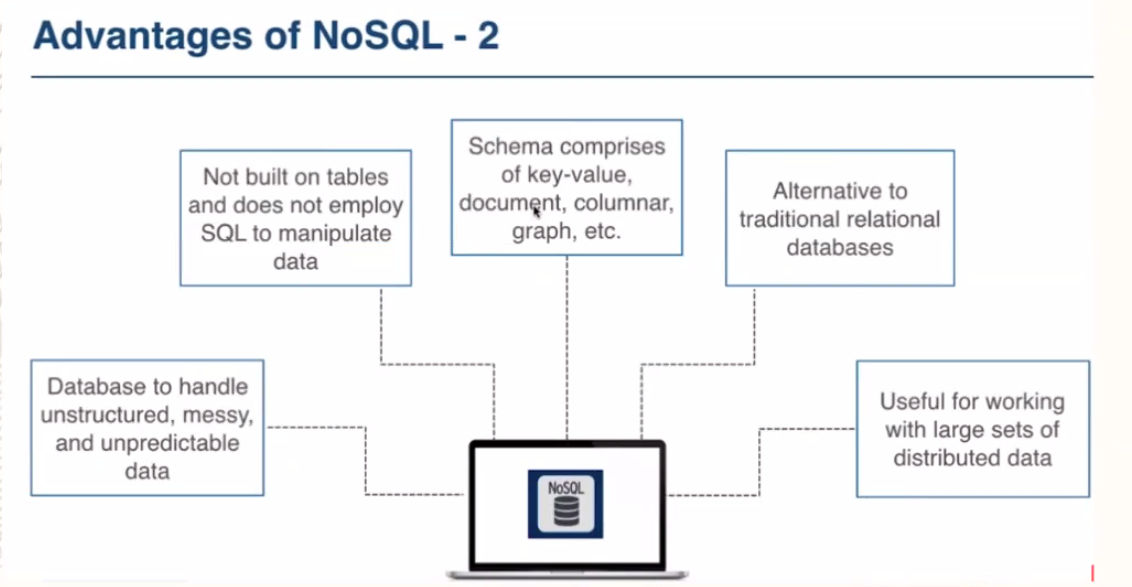




Concurrent – at same time we can add delete or update data



A screenshot of a graph

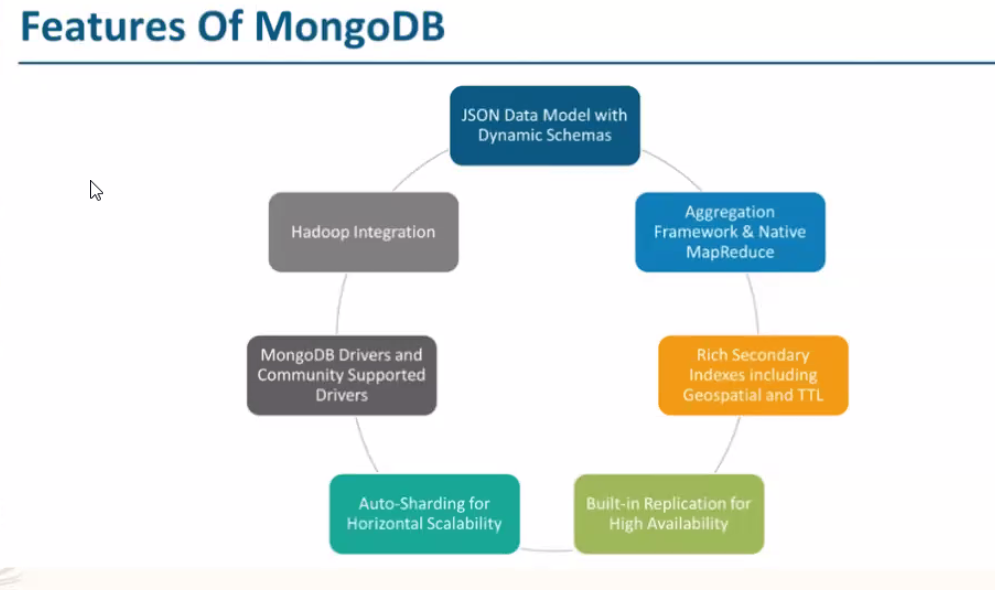
Description automatically generatedA diagram of software development

Description automatically generatedA screenshot of a graph store

Description automatically generatedA diagram of a database

Description automatically generatedA diagram of a step-by-step process

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Mongo db starts from here

A close-up of a computer

Description automatically generated

Collection == table in sql

Document == rows in sql

A diagram of a collection

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Write in json formate and get stored in BSON format

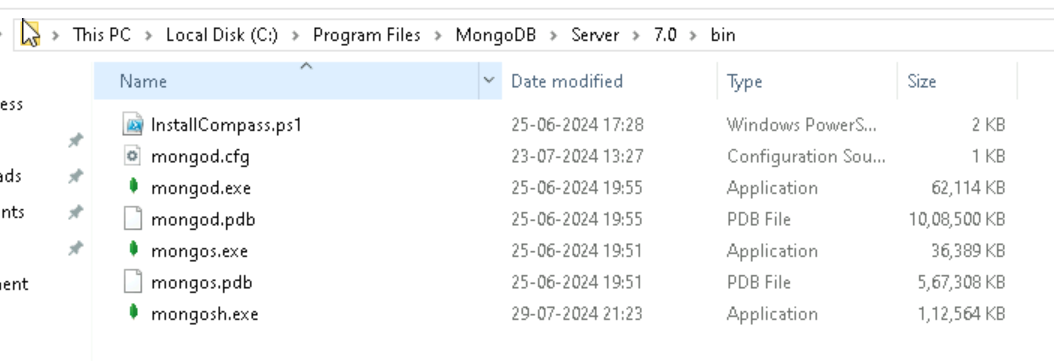
A white paper with text and numbers

Description automatically generated

<https://www.mongodb.com/try/download/tools>

download mongo shell to get mongosh

Connect:

Goto cmd and type mongosh

* Show dbs 🡪 it will show list of db
* Mongo - -host Akshay.com –port 27001
* Use ustdb 🡪 it will create db but wont show until we have no collection
* db.createCollection(“Employee”) 🡪 it will create collection and it will show in dbs
* db.dropDatabase() 🡪 it will delete the db which is using
* use dbname 🡪 it will switch to db which we gave
* show collections 🡪 it will show the list of collections
* db.collectionname.drop() 🡪 it will delete the collection
* db.collectionname.insertOne(

{

“name” : “Akshay”,

“age”: 29,

“married” : false

}

) 🡪 to insert one set

* db.collectionname.insertMany(

[

{

“\_id” : 1,

“name” : “Akshay”

},

{

“\_id”: 2,

“name”:”Sudhir”

}

]

) 🡪to insert many columns

**Note:**

While adding many entries or document we need to do it in list format

* db.collectionname.find() 🡪 will get all the values
* db.collectionname.findOne() 🡪 will get first value
* db.collectionname.find({“name”:”Akshay”}) 🡪 will get if name is Akshay
* db.coll ectionname.findOne({“name”:”Akshay”}) will get first occurrence of Akshay in case if there are many Akshay
* db.collectionname.find({“name”:”Akshay”},{\_id:0,name:1}) 🡪 0 means wont display and 1 means it will display
* db.collectionname.find({},{\_id:0, name:1}).sort({name:1}) 🡪 sort 1 means asending order, sort -1 means descending order
* db.emp.find(  
  {age:{$gt:25,$lt:45}}, 🡪 between  
  {\_id:0, firstName:1,age:1}
* db.emp.find({designation:{$in:["ID","GD"]}}) 🡪 in
* db.inventory.find({"tags" :{$size:2}}) 🡪 size will be given when we have list
* db.inventory.find({ $or: [{ qty: 100 }, { item: "notebook" }] }) 🡪 or and condition
* db.inventory.find({ $or: [{ qty: 100 }, { item: "notebook" }] }) 🡪 all tag will check everythong should present
* db.persons.find({email:{$regex:/@gmail/}}) 🡪 / works as like%
* db.persons.find({dob:{$gt:new Date('2000-01-01')}}) 🡪 when we want to validate based on date
* db.persons.find(  
  {"address.state":'Goa',  
  "address.city":'Goa'}) 🡪 validate both the condition
* Employees which has either skills java or react or both  
  db.employees.find({skills: {$in : ["Java","React"]}}) 🡪 in condition
* Employees which has both skills Java and React  
  db.employees.find({skills: {$all : ["Java","React"]}}) 🡪 if all present then display
* db.employees.find({skills: {$size : 3}}) 🡪 all employees who has 3 skills

use ustdb  
db.createCollection(<collectionname>)  
db.createCollection("Employee")  
db.dropDatabase() 🡪 make sure you are in db which you want to delete  
show collections()  
db.<collectionname>.drop()  
db.Employee.drop()  
  
db.emp.insertOne(  
{  
firstName : "Parag",  
lastName : "Joshi",  
age: 40,  
married : true  
}  
)  
  
  
db.emp.insertMany([  
{  
\_id:1,  
firstName : "Prachi",  
lastName : "Joshi",  
age: 40,  
designation: "ID",  
married : true  
},  
{  
\_id:2,  
firstName : "Manas",  
lastName : "Joshi",  
age: 20,  
designation: "ST",  
married : false  
}  
])

db.emp.find();

db.emp.find() -> It will show all the documents for the collection emp  
  
db.emp.find().pretty() - It will show all the documents for the collection emp in formatted way  
  
db.emp.findOne() - first document of the emp collection  
`  
db.emp.find({firstName:"Daesha"})  
  
db.emp.find({lastName:"Joshi"}).pretty()  
db.emp.findOne({lastName:"Joshi"})

db.emp.find({age:{$gt:20}})

$gt, $lt, $gte, $lte, $ne

db.emp.find(  
{age:{$ne:20}},  
{\_id:0,firstName:1,lastName:1}  
) 🡪 greate then age 20  
  
db.emp.find(  
{age:{$ne:20}},  
{\_id:0,married:0}  
) 🡪 not equal to 20   
  
db.emp.find(  
{},{\_id:0,firstName:1,lastName:1}).sort({firstName:1}) 🡪 projection and sort 1 – ascending sort-1 means decending  
  
db.emp.find(  
{},{\_id:0,firstName:1,lastName:1}).sort({firstName:-1}) 🡪 descending order  
  
  
db.emp.find(  
{},{\_id:0,firstName:1,lastName:1}).sort({lastName:-1, firstName:1})  
  
db.emp.find(  
{},{\_id:0,firstName:1,lastName:1,age:1}).sort({lastName:1, firstName:-1})  
  
db.emp.find(  
{},{firstName:1,lastName:1,age:1}).sort({age:1, firstName:-1})  
  
  
db.emp.find(  
{age:{$gt:25,$lt:45}}, 🡪 between  
{\_id:0, firstName:1,age:1}  
)  
  
db.emp.find({designation:{$in:["ID","GD"]}}) 🡪 in

Assessment:

db.inventory.insertMany([  
   { item: "journal", qty: 25, tags: ["blank", "red"], dim\_cm: [ 14, 21 ] },  
   { item: "notebook", qty: 50, tags: ["red", "blank"], dim\_cm: [ 14, 21 ] },  
   { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim\_cm: [ 14, 21 ] },  
   { item: "planner", qty: 75, tags: ["blank", "red"], dim\_cm: [ 22.85, 30 ] },  
   { item: "postcard", qty: 45, tags: ["blue"], dim\_cm: [ 10, 15.25 ] }  
]);  
  
db.inventory.find({"tags" :{$size:2}})  
db.emp.find({$and:[{lastName:"Joshi",designation:"ST"}]})  
db.emp.find({$or:[{age:32},{designation:"HR"}]})

question to solve:

- Find all the documents which has qty >= 50 🡪 **db.inventory.find({“qty”:{$gte: 50}})**  
- Find all the documents whose qty is either 100 or item is notebook 🡪 **db.inventory.find({ $or: [{ qty: 100 }, { item: "notebook" }] })**  
- Find first document whose tags contains "red" and "blank" 🡪 **db.inventory.findOne({ tags: { $all: ["red", "blank"] } })**

**Question :2**

db.persons.insertMany(  
[{  
firstName: "Vinay",  
lastName: "Rao",  
email: "vr@hotmail.com",  
salary: 56000,  
dob: new Date('Mar 2,1999'),  
skills: ["Java","React","Mysql"],  
address: {  
street: "Kumar Sublime",  
house: "A304",  
city : "Pune",  
state : "MH",  
country : "India",  
pincode: 411048  
}  
},  
{  
firstName: "Keshav",  
lastName: "Ingle",  
email: "ik@gmail.com",  
salary: 100000,  
dob: new Date('Oct 14,1995'),  
skills: ["Angular","React","Mysql"],  
address: {  
street: "Five Gardens Society",  
house: "A456",  
city : "Mumbai",  
state : "MH",  
country : "India",  
pincode: 400001  
}  
},  
{  
firstName: "Kirti",  
lastName: "Moghe",  
email: "kirti@yahoo.com",  
salary: 50000,  
dob: new Date('Apr 15,1996'),  
skills: ["Oracle","React",".Net"],  
address: {  
street: "Seashore Villas",  
city : "Goa",  
state : "Goa",  
country : "India",  
pincode: 600001  
}  
}  
])

db.persons.find({skills:"Java"})  
  
  
db.persons.find({email:{$regex:/@gmail/}})  
  
db.persons.find({dob:{$gt:new Date('2000-01-01')}})  
  
  
db.persons.find(  
{"address.state":'Goa',  
"address.city":'Goa'})  
  
  
-Employees which has either skills java or react or both  
db.employees.find({skills: {$in : ["Java","React"]}})  
  
-Employees which has both skills Java and React  
db.employees.find({skills: {$all : ["Java","React"]}})  
  
All the employees which has the 3 skills  
db.employees.find({skills: {$size : 3}})

**assessment1**:

use mongo\_practice  
  
## Insert Documents  
  
Insert the following documents into a `movies` collection.  
  
  
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  
]  
  
title : Inglorious Basterds  
writer : Quentin Tarantino  
year : 2009  
actors : [  
  Brad Pitt  
  Diane Kruger  
  Eli Roth  
]  
  
  
title : The Hobbit: An Unexpected Journey  
writer : J.R.R. Tolkein  
year : 2012  
franchise : The Hobbit  
  
title : The Hobbit: The Desolation of Smaug  
writer : J.R.R. Tolkein  
year : 2013  
franchise : The Hobbit  
  
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  
  
query the `movies` collection to  
  
1. get all documents 🡪 db.movies.find()  
  
2. get all documents with `writer` set to "Quentin Tarantino" 🡪 db.movies.find({“writer”:” Quentin Tarantino”})  
  
3. get all documents where `actors` include "Brad Pitt" 🡪 db.movies.find({“actors”:”Brad Pitt”})  
  
4. get all documents with `franchise` set to "The Hobbit" 🡪 db.movies.find({“franchise“:”The Hobbit”})  
  
5. get all movies released in the 90s 🡪 db.movies.find({“year”:{$gte:1990,$lte:1999}})  
  
6. get all movies released before the year 2000 or after 2010 🡪 db.movies.find({$or:[{“year”:{$lt:2000}},{“year”:{$gt:2010}}]})  
  
  
## Text Search  
  
1. find all movies that whose title start with Inglorious 🡪 db.movies.find({“title”:{$regex:/^ Inglorious /}})  
  
2. Project only film name and actor names in descending order of title 🡪 db.movies.find({},{\_id:0,title:1}).sort({“title”:-1})  
  
3. Sort the movies by year in descending manner 🡪 db.movies.find().sort({year:-1})  
  
4. find the last record of the doucment  
  
5. find the 3rd record from the document

**update and delete**

db.emp.updateOne({firstName:"Manas"},{$set : {age:25}})  
  
db.emp.updateMany({firstName:"Manas"},{$set : {designation:"CTO"}})  
  
db.emp.updateOne({lastName:"Joshi"}, {$set:{salary:15000, email:"parag.joshi@hematitecorp.com",  
age:42}})  
  
db.emp.updateMany({lastName:"Joshi"},{$set : {salary:10000}})  
  
db.emp.updateMany({married:true},{$inc : {salary:1000}})  
  
  
db.emp.updateMany({},{$currentDate:{updatedDate:true}},false)

db.emp.updateOne({firstName:"Parag"},{$unset: {email:""}})  
  
db.inventory.updateOne(  
{item:"journal"},{$set:{"tags":["blue","red"]}})  
  
db.inventory.deleteOne({item:"journal"})  
db.inventory.deleteMany({})

Group by Aggregate method:

db.sales.insertMany([  
  { "\_id" : 1, "item" : "abc", "price" : Decimal128("10"), "quantity" : Int32("2"), "date" : ISODate("2014-03-01T08:00:00Z") },  
  { "\_id" : 2, "item" : "jkl", "price" : Decimal128("20"), "quantity" : Int32("1"), "date" : ISODate("2014-03-01T09:00:00Z") },  
  { "\_id" : 3, "item" : "xyz", "price" : Decimal128("5"), "quantity" : Int32( "10"), "date" : ISODate("2014-03-15T09:00:00Z") },  
  { "\_id" : 4, "item" : "xyz", "price" : Decimal128("5"), "quantity" :  Int32("20") , "date" : ISODate("2014-04-04T11:21:39.736Z") },  
  { "\_id" : 5, "item" : "abc", "price" : Decimal128("10"), "quantity" : Int32("10") , "date" : ISODate("2014-04-04T21:23:13.331Z") },  
  { "\_id" : 6, "item" : "def", "price" : Decimal128("7.5"), "quantity": Int32("5" ) , "date" : ISODate("2015-06-04T05:08:13Z") },  
  { "\_id" : 7, "item" : "def", "price" : Decimal128("7.5"), "quantity": Int32("10") , "date" : ISODate("2015-09-10T08:43:00Z") },  
  { "\_id" : 8, "item" : "abc", "price" : Decimal128("10"), "quantity" : Int32("5" ) , "date" : ISODate("2016-02-06T20:20:13Z") },  
])  
  
db.sales.aggregate( [  
  {  
    $group: {  
       \_id: null,  
       count: { $count: { } }  
    }  
  }  
] ) 🡪 count total rows or documents  
  
  
db.sales.aggregate( [ { $group :  
{ \_id : "$item" }  
} ] )  
  
  
  
db.sales.aggregate(  
  [  
    // First Stage  
    {  
      $group :  
        {  
          \_id : "$item",  
          totalSaleAmount: { $sum: { $multiply: [ "$price", "$quantity" ] } }  
        }  
     }  
    ]  
 )  
   
   
 db.sales.aggregate(  
  [  
    // First Stage  
    {  
      $group :  
        {  
          \_id : "$item",  
          totalSaleAmount: { $sum: { $multiply: [ "$price", "$quantity" ] } }  
        }  
     },  
     {  
        $match : { "totalSaleAmount": { $gte: 100 } }  
     }  
    ]  
 )  
   
 db.sales.aggregate([  
  // First Stage  
  {  
    $match : { "date": { $gte: new ISODate("2014-01-01"), $lt: new ISODate("2015-01-01") } }  
  },  
  {  
    $group : {  
       \_id : { $dateToString: { format: "%Y-%m-%d", date: "$date" } },  
       totalSaleAmount: { $sum: { $multiply: [ "$price", "$quantity" ] } },  
       averageQuantity: { $avg: "$quantity" },  
       count: { $sum: 1 }  
    }  
  },  
    {  
        $match : { "count": { $gt: -1 } }  
     },  
  // Third Stage  
  {  
    $sort : { totalSaleAmount: 1 }  
  }  
  ])

**$push, $$ROOT, $addfields**

db.books.insertMany([  
  { "\_id" : 8751, "title" : "The Banquet", "author" : "Dante", "copies" : 2 },  
  { "\_id" : 8752, "title" : "Divine Comedy", "author" : "Dante", "copies" : 1 },  
  { "\_id" : 8645, "title" : "Eclogues", "author" : "Dante", "copies" : 2 },  
  { "\_id" : 7000, "title" : "The Odyssey", "author" : "Homer", "copies" : 10 },  
  { "\_id" : 7020, "title" : "Iliad", "author" : "Homer", "copies" : 10 }  
])   
  
  
db.books.aggregate([  
   { $group : { \_id : "$author", books: { $push: "$title" } } }  
 ])  
   
 db.books.aggregate([  
   // First Stage  
   {  
     $group : { \_id : "$author", books: { $push: "$$ROOT" } }  
   },  
   {  
     $addFields:  
       {  
         totalCopies : { $sum: "$books.copies" }  
       }  
   }  
   ])

**Creating index:**

db.articles.createIndex( { subject: "text" } )

db.articles.insertMany( [

{ \_id: 1, subject: "coffee", author: "xyz", views: 50 },

{ \_id: 2, subject: "Coffee Shopping", author: "efg", views: 5 },

{ \_id: 3, subject: "Baking a cake", author: "abc", views: 90 },

{ \_id: 4, subject: "baking", author: "xyz", views: 100 },

{ \_id: 5, subject: "Café Con Leche", author: "abc", views: 200 },

{ \_id: 6, subject: "Сырники", author: "jkl", views: 80 },

{ \_id: 7, subject: "coffee and cream", author: "efg", views: 10 },

{ \_id: 8, subject: "Cafe con Leche", author: "xyz", views: 10 }

] )

db.articles.find( { $text: { $search: "coffee" } } )

db.articles.find( { $text: { $search: "bake coffee cake" } } )

db.articles.find( { $text: { $search: "\"coffee shop\"" } } )

db.articles.find( { $text: { $search: "\'coffee shop\' \'Cafe con Leche\'" } } )

db.articles.find( { $text: { $search: "coffee -shop" } } )

db.articles.find(

{ $text: { $search: "leche", $language: "es" } }

)

db.articles.find( { $text: { $search: "сы́рники CAFÉS" } } )

db.articles.find( { $text: { $search: "Coffee", $caseSensitive: true } } )

db.articles.find( {

$text: { $search: "\"Café Con Leche\"", $caseSensitive: true }

} )