MongoDB-> it is capable of handling huge and complex data in a simple way.

It stores the data in the form of JSON.

JSON-Java Script Object Notation.-> it is derived from js object.

JS object- Syntex- { }

Everything is in the form of Key:value

NAME AGE GENDER

Sundari 25 Female

Sheela 26 Fe

Tinku 25 male

1.How to see the database present in it?

Ans. show dbs

2.Clear the screen

Ans. cls

SQL MongoDB

Database Database

Table Collection

Row Document

Column Field

Q. What is collection:-

Ans. In Mongodb, a collection is a grouping of Mongodb documents. Each document is a set of key-values pairs and represents a unit of data within the collection.

What is document:-

A documents is a basic unit of data stored as a set of key values pairs similar to a row in a traditional database table. Documents are stored within collections and can vary in structure, allowing for flexibility in schema design.

**MongoDB uses CamelCase**

Q.How to create a database

Ans. use <databasename>

Q.How to create a collection inside a database?

- db.createCollection("collectionName")

- db.createCollection("academics")

Q.How to see collections inside a database

Ans. show collections

Q.How to create a document

Ans. db.collectionName.insertOne({})

Q.How to see a document

Ans. db.collectionName.find()

It returns in the form of array format.

Q.How to rename a collection

Ans. **db.oldcollectionName.renameCollection(“newCollectionName”)**

Q. How to insert multiple documents in a collection?

Ans. db.collectionName.insertMany([{},{}])

Q. How to find a specific name document?

Ans. db.collectionname.find({key:value})

Q. How to find one document in a collection

Ans. db.collectionName.findOne()

how to find how many documents are in a collection

Ans. db.collectionName.find().count() // 4

Inserting all the datatypes

db.academic.insertOne({course:"MERN",duration:4,placement:true,canteen:null,lift:undefined,friends:["Laxmi","avilipsa",'avijeet','vivek','shanu','biswajit'],subjects:{sub1:'web',sub2:'sql',sub3:'mongodb'}})

{

acknowledged: true,

insertedId: ObjectId('66f12d7359791ff10cc73bf8')

}

sundari> db.academic.find()

[

{

\_id: ObjectId('66f12d7359791ff10cc73bf8'),

course: 'MERN',

duration: 4,

placement: true,

canteen: null,

lift: null, // undefined is treated as null in mongodb

friends: [ 'Laxmi', 'avilipsa', 'avijeet', 'vivek', 'shanu', 'biswajit' ],

subjects: { sub1: 'web', sub2: 'sql', sub3: 'mongodb' }

}

]

**insertOne**():

- It helps to insert a single document into a collection.

It returns a document.

It contains 2 values:

o A Boolean acknowledge as true if any error occur it will return false

o A field with an \_id with the inserted document.

**insertMany**():

- It helps to insert a multiple documents into a collection.

- Returns: It returns a document.

It contains 2 values:

o A Boolean acknowledge as true if any error occur it will return false

o A

**insert**():

- It helps to inserts a document or documents into a collection

NOTE- this method is deprecated

find(): - db.collection\_name.find(query,projection)

projection works for fields

- sundari> db.books.find({name:'maths'},{})

[

{

\_id: ObjectId('66ee74fbc893081f20c73bf8'),

name: 'maths',

price: 500

}

]

- sundari> db.books.find({name:'maths'},{\_id:0})

[ { name: 'maths', price: 500 } ]

- sundari> db.books.find({},{\_id:0})

[

{ name: 'maths', price: 500 },

{ name: 'physics', price: 400 },

{ name: 'english', price: 250 },

{ name: 'history', price: 300 }

]

- db.books.find({},{\_id:0,price:0})

[

{ name: 'maths' },

{ name: 'physics' },

{ name: 'english' },

{ name: 'history' }

]

- sundari> db.books.find({},{\_id:0,price:1,name:1})

[

{ name: 'maths', price: 500 },

{ name: 'physics', price: 400 },

{ name: 'english', price: 250 },

{ name: 'history', price: 300 }

]

MongoServerError[Location31254]: Cannot do exclusion on field name in inclusion projection

Id is exceptional , in remaining fields we can only use 1 or 0.

MongoDB stores the date using the ISO date format, which is a BSON datatype.

Specifically the date are stored as a 64-bits integer representing the no of milliseconds.

1. WAQTD ALL THE EMP DETAILS.

- Db.emp.find()

2.wqtd details where jobe is manager.

- db.emp.find({job:"manager"},{})

db.emp.find({job:"clerk"},{})

db.emp.find({},{\_id:0,ename:1})

db.emp.find({job:'salesman'},{\_id:0,ename:1})

db.emp.find({job:'salesman'},{\_id:0,ename:2,job:1})

db.emp.find({deptno:20})

deleteOne({condition})- only 1st document whoever satisfied the condition

deleteOne({})-1st document

deleteOne()- error

deleteMany({condition})- all documents whoever satisfied the condition

deleteMany({})-all document

deleteMany()-error

Delete a collection – db.collectionname.drop()

db.stu.drop() //true

Delete a Database – db.dropDatabase() // { ok: 1, dropped: 'abc' }

UPDATION:

db.collectionname.updateOne({condition}{ $set:{updatation}})- only one whoever satisfied the condition

db.collectionname.updateOne({}{ $set:{updated\_value}})- only 1st document

db.collectionname.updateMany({condition}{ $set:{updation }})- all documents whoever satisfied the condn.

db.collectionname.updateMany({}{ $set:{updated\_value}})- all documents

how to count number of collections in mongodb-

how to create a collection without using create collections in mongodb without inserting a document-

question:35

- db.emp.find({$and:[{deptno:{$ne:10}},{job:{$ne:'analyst'}}]},{ename:1,job:1,deptno:1,\_id:0})

- db.emp.find({ deptno: { $nin: [10] }, job: { $nin: ['analyst'] } }, { ename: 1, job: 1, deptno: 1, \_id: 0 });

- db.emp.find({ deptno: { $not: { $in: [10] } }, job: { $not: { $in: ['analyst'] } } }, { ename: 1, job: 1, deptno: 1, \_id: 0 });

- db.emp.find({$nor: [{ deptno: 10 },{ job: 'analyst' }]}, { ename: 1, job: 1, deptno: 1, \_id: 0 });

- db.emp.find({deptno: { $not: { $eq: 10 } }, job: { $not: { $eq: 'analyst' } }}, { ename: 1, job: 1, deptno: 1, \_id: 0 });

question:

$pop

It removes the first or last element from an array field within a document. You can specify wherher to remove the 1st element or last element by providing value of 1 or -1.

<value>: 1 remove the 1st element or -1 last element

Syntax: db.collection.updateOne({query},{$pop:{<arrayfield>:<value>}})

db.stores.insertMany([

{

id: 1,

name:"anup",

fruits: [ "apples", "pears", "oranges", "grapes", "bananas" ],

vegetables: ["carrots", "celery", "squash", "carrots"],

bills:[100,20,40,5,8,12,50,5]

},

{

\_id: 2,

name: "vicky",

fruits: [ "plums", "kiwis", "oranges", "bananas", "apples" ],

vegetables: [ "broccoli", "zucchini", "carrots", "onions"],

bills:[50,66,90,2,19,4,8,7,40]

}

])

Waqt remove apples and oranges where name is anup

db.stores.updateOne({name:'anup'},{$pull:{fruits:{$in:['apples','oranges']}}})

$pull

It is used to remove specific elements from an array field within a document. Unlike $pop, which removes elements based on their positions(1st ,last), $pull allows you to specify a condition to match the elements you want to remove .

Syntax: db.collection.updateOne({query},{$pull:{<arrayField>:<condition>}})

Waqt remove carrot from vegetyables

db.stores.updateMany({},{$pull:{vegetables:{$in:['carrots']}}})

waqt remove bills less then 10 from bills array

db.stores.updateMany({},{$pull:{bills:{$lt:10}}})

wqt remove bills more then 50

db.stores.updateMany({},{$pull:{bills:{$gt:50}}})

$pullAll

It is used to remove multiple specified values from an array field within a document. Unlike $pull,which allows for more complex queries to define which elements to remove ,$pullAll specifically targets an arrays of values to be removed

Syntax: db.collection.update({query},{$pullAll:{<arrayField>:<value1>,<value2>}})

wqt remove 50 from bills array.

db.stores.updateMany({},{$pullAll:{bills:[50]}})

$each

db.collectionName.updateOne($push:{<field>:{$each:[<value1>,<value2>]}})

$position

db.collectionName.updateOne({},{$push:{<field>:{$each:[<value1>,<value2>],$position:number}}})

Aggregation Operations:

//! Aggregation

Aggregation in MongoDB is a powerful framework for processing

and transforming data within collections. It allows you to

perform operations on documents to retrieve aggregated

results, making it ideal for tasks such as data analysis,

eporting, and complex queries.

//! syntax:

db.collection.aggregate([

{ $stage1: { /\* stage1 parameters \*/ } },

{ $stage2: { /\* stage2 parameters \*/ } },

// More stages...

])

// Key Concepts of Aggregation

// 1) Aggregation Pipeline: The core of the aggregation framework is the aggregation pipeline, which consists of multiple stages that process documents in sequence. Each stage transforms the data and passes the output to the next stage.

// Common Stages: Here are some of the most commonly used stages in the aggregation pipeline:

sort():

wqtd ename in ascending order

sql> db.emp.find({},{\_id:0,ename:1}).sort({ename:1})

sql> db.emp.find({},{\_id:0,ename:1}).sort({ename:'ascending'})

wqtd ename in descending

sql> db.emp.find({},{\_id:0,ename:1}).sort({ename:-1})

sql> db.emp.find({},{\_id:0,ename:1}).sort({ename:'descending'})

wqtd ename,sal in ascending order

-db.emp.find({},{\_id:0,ename:1,sal:1}).sort({sal:1})

wqtd ename,sal in descending order

-db.emp.find({},{\_id:0,ename:1,sal:1}).sort({sal:-1})

waqtd who is getting comm more then sal

- db.emp.aggregate([{$match:{$expr:{$gt:['$comm','$sal']}}}])

$match:

Filters documents to pass only those that meet the specified criteria.

Syntex: {$match:{<conditions>}}

Waqtd all the details who is earning more then 2000

- db.emp.find({sal:{$gt:2000}})

-db.emp.aggregate([{$match:{sal:{$gt:2000}}}])

waqtd only ename sal of the employees

- db.emp.aggregate([{$project:{\_id:0,ename:1,sal:1}}])

waqtd empname sal comm job of the emp who is salesman or analyst

-db.emp.aggregate([{$match:{job:{$in:['analyst','salesman']}}},{$project:{\_id:0,ename:1,sal:1,comm:1,job:1}}])

waqtd details of emps whose name contains character i & earning more then 3000 & job is analyst or prcident

- db.emp.aggregate([{$match:{ename:{$regex:/i/}}},{$match:{sal:{$gt:3000}}},{$match:{job:{$in:['analyst','president']}}}])

waqtd name sal comm who is earning more then 2500 comm is not null

- db.emp.aggregate([{$match:{$and:[{sal:{$gt:2500}},{comm:{$ne:null}}]}},{$project:{\_id:0,ename:1,sal:1,comm:1}}])

waqtd name sal comm who is earning more then 1500 comm is not null

- db.emp.aggregate([{$match:{$and:[{sal:{$gt:1500}},{comm:{$ne:null}}]}},{$project:{\_id:0,ename:1,sal:1,comm:1}}])

waqtd annual sal of all emps

- db.emp.aggregate([{$addFields:{annual\_sal:{$multiply:["$sal",12]}}},{$project:{\_id:0,ename:1, sal:1,annual\_sal:1}])

` db.emp.aggregate([{$set:{annual\_sal:{$multiply:["$sal",12]}}},{$project:{\_id:0,ename:1, sal:1,annual\_sal:1}])

-db.emp.aggregate([{$project:{ \_id: 0,ename: 1,sal:1,annual\_sal:{$multiply:["$sal", 12]}}}])

waqtd who is getting annual sal > 3000

- db.emp.aggregate([{$addFields:{annual\_sal:{$multiply:["$sal",12]}}},{$match:{annual\_sal:{$gt:30000}}},{$project:{\_id: 0,ename:1,sal:1,annual\_sal:1}}])

waqtd no. of empys present in each dept

- db.emp.aggregate([{$group:{\_id:'$deptno',no\_of\_emps:{$sum:1}}}])

waqtd no of emps working in each job

- db.emp.aggregate([{$group:{\_id:'$job',no\_of\_emps:{$sum:1}}}])

waqtd max sal , min sal avg sal

- db.emp.aggregate([{$group:{\_id:'$deptno',max\_sal:{$max:'$sal'},min\_sal:{$min:'$sal'},avg\_sal:{$avg:{$round:'$sal'}}}}

waqtd no of emps in each dept expect precident

- db.emp.aggregate([{$match:{job:{$ne:'president'}}},{$group:{\_id:'$deptno',no\_of\_emps:{$sum:1}}}])

waqtd no of emps max sal of all the emps in each job if the max sal is more then 2000

- db.emp.aggregate([{$group:{\_id:'$job',max\_sal:{$max:'$sal'},no\_of\_emp:{$sum:1}}},{$match:{max\_sal:{$gt:2000}}}])

waqtd details of all emps & their dept details

- db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'d\_details'}}])

Here's a breakdown of the query step by step:

db.emp.aggregate: This indicates that you are starting an aggregation operation on the "emp" collection.

$lookup: This is the aggregation stage that performs the join operation.

from: "dept": Specifies the collection you're joining with, which is the "dept" collection'.

foreignField: "deptno": Specifies the field in the "dept" collection to match against.

localField: "deptno": Specifies the field in the "emp" collection to match against.

as: "dept": Specifies the name of the new field that will hold the matching "dept" documents.

waqtd location of king

db.emp.aggregate([{$match:{ename:'king'}},{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$project:{\_id:0,ename:1,'dd.loc':1}}])

waqtd nam sal & loc of all emps

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$project:{\_id:0,ename:1,sal:1,'dd.loc':1}}])

waqtd nam sal & loc of all emps who is earning more then 2000 and less then 3000

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$match:{sal:{$gt:2000}}},{$match:{sal:{$lt:3000}}},{$project:{\_id:0,ename:1,sal:1,'dd.loc':1}}])

waqtd the name sal & loc of all emps who is earning more then 2000 and less then 3000 & location starts withn n

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$match:{sal:{$gt:2000}}},{$match:{sal:{$lt:3000}}},{$match:{'dd.loc':{$regex:/^n/}}},{$project:{\_id:0,ename:1,sal:1,'dd.loc':1}}])

waqtd ename and dname of emps having character a in their dname

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$match:{'dd.dname':{$regex:/a/}}},{$project:{\_id:0,ename:1,'dd.dname':1}}])

waqtd dname job of all emps whose job and dname starts with s

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$match:{'dd.dname':{$regex:/^s/}}},{$match:{job:{$regex:/^s/}}},{$project:{\_id:0,job:1,'dd.dname':1}}])

waqtd dname and empno of all emps whos empno are 7839 or 7902 & workingh in new York

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$match:{empno:{$in:[7839,7902]}}},{$match:{'dd.loc':{$eq:'new york'}}},{$project:{\_id:0,empno:1,'dd.dname':1}}])

waqtd name & hiredate who join 1st

db.emp.find({},{\_id:0,ename:1,hiredate:1}).sort({hiredate:1}).limit(1)

db.emp.aggregate([{$sort:{hiredate:1}},{$limit:1},{$project:{\_id:0,ename:1,hiredate:1}}])

waqtd deptno, avg sal for each dept excluding dept no 20

db.emp.aggregate([ { $match: {deptno: { $ne: 20 } } }, { $group: { \_id: "$deptno",avgSal: { $avg: "$sal" }} }])

waqtd who is getting 8th min sal

db.emp.find({},{\_id:0,ename:1,sal:1}).sort({sal:1}).skip(7).limit(1)

db.emp.aggregate([{$sort:{sal:1}},{$skip:7},{$limit:1},{$project:{\_id: 0,ename:1,sal: 1}}])

waqtd avg sal of each dept, deptno, dname expect deptno 20

db.emp.aggregate([{$match:{deptno:{$ne:20}}},{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$unwind:'$dd'},{$group:{\_id:'$deptno',avgSal:{$avg:'$sal'},dname:{$first:'$dd.dname'}}}])

db.emp.aggregate([{$match:{deptno:{$ne:20}}},{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$unwind:'$dd'},{$group:{\_id:'$deptno',avgSal:{$avg:'$sal'},empname:{$push:'$dd.dname'}}}])

waqtd avg sal of each dept, deptno, ename expect deptno 20

db.emp.aggregate([{$match:{deptno:{$ne:20}}},{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$unwind:'$dd'},{$group:{\_id:'$deptno',avgSal:{$avg:'$sal'},empname:{$push:'$ename'}}}])

waqtd empnames in each job

db.emp.aggregate([{$group:{\_id:'$job',empname:{$push:'$ename'}}}])

waqtd dname and sal of all the emps who is working in accounting

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$match:{'dd.dname':'accounting'}},{$project:{\_id:0,'dd.dname':1,sal:1,ename:1,deptno:1}}])

waqtd ename and dname of the emps who are getting comm in dept 10 or 30

db.emp.aggregate([{$lookup:{from:'dept',localField:'deptno',foreignField:'deptno',as:'dd'}},{$unwind:'$dd'},{$match:{comm:{$ne:null},deptno:{$in:[10,30]}}},{$project:{\_id:0,ename:1,'dd.dname':1}}])

waqtd details of all the emps and their manager

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}}])

waqtd empno ename mgr of all the emps and their manager

db.emp.aggregate([{$project:{\_id:0,empno:1,ename:1,mgr:1}}])

waqtd ename and their manager name

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$project:{\_id:0,ename:1,mgr\_name:'$m.ename'}}])

waqtd ename whose manager sal>2000

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$match:{'m.sal':{$gt:2000}}},{$project:{\_id:0,ename:1}}])

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$match:{'m.sal':{$gt:2000}}},{$project:{\_id:0,ename:1,mgrname:'$m.ename',mgrsal:'$m.sal'}}])

waqtd ename,mgr if mgr sal = emp sal

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$match:{'m.sal':'$sal'}},{$project:{\_id:0,ename:1,mgrname:'$m.ename',mgrsal:'$m.sal'}}])

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$unwind:'$m'},{$match:{$expr:{$eq:['$m.sal','$sal']}}},{$project:{\_id:0,ename:1,mgrname:'$m.ename',mgrsal:'$m.sal'}}])

waqtd ename,mgr if emp sal > mgr sal

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$unwind:'$m'},{$match:{$expr:{$gt:['$sal','$m.sal']}}},{$project:{\_id:0,ename:1,sal:1,mgrname:'$m.ename',mgrsal:'$m.sal'}}])

waqtd mgr\_name and how many emps are reporting to that manager

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$unwind:'$m'},{$group:{\_id:'$m.ename',no\_of\_emps\_reporting:{$sum:1}}}])

waqtd mgr\_name and how many emps are reporting to that manager and also show empnames

db.emp.aggregate([{$lookup:{from:'emp',localField:'mgr',foreignField:'empno',as:'m'}},{$unwind:'$m'},{$group:{\_id:'$m.ename',no\_of\_emps\_reporting:{$sum:1},empnames:{$push:'$ename'}}}])

waqtd empname and hiredate, managername and manager hierdate if manager was highred before employees

db.emp.aggregate([ { $lookup: { from: "emp", localField: "mgr", foreignField: "empno", as: "m" } }, { $unwind: "$m" }, { $match: { $expr: { $lt: ["$m.hiredate", "$hiredate"] } } }, { $project: { \_id:

0, ename: 1, hiredate: 1, "m.ename": 1, "m.hiredate": 1 } }] )