Q. Design database for Zen class programme users codekata attendance topics tasks company\_drives mentors

answer-->

Ans:

* create database

**use zen\_class;**

* create collection users and insert data db.createCollection("users");

db.users.insertMany([ { userid:1, name:"Suman", email:"[suman11@gmail.com](mailto:suman11@gmail.com)",

},

{

userid:2,

name:"ganesh",

email:"ganesh@gmail.com",

},

{

userid:3,

name:"chanchal",

email:"chanchal@gmail.com",

},

{

userid:4,

name:"ekta",

email:"ekta@gmail.com",

},

{

userid:5,

name:"Rani",

email:"Rani@gmail.com",

}

])

\* create collection codekata and insert data

db.createCollection("codekata");

db.codekata.insertMany([ { userid:1, problems:80 }, { userid:2, problems:110 }, { userid:3, problems:95 }, { userid:4, problems:55 }, { userid:5, problems:75 } ])

* create collection topics and insert data db.createCollection("topics");

db.topics.insertMany([ {  
topicid:1, topic:"HTML", topic\_date:new Date("4-oct-2020") }, { topicid:2, topic:"CSS", topic\_date:new Date("11-oct-2020") }, { topicid:3, topic:"Bootstrap", topic\_date:new Date("18-oct-2020") }, { topicid:4, topic:"JavaScript", topic\_date:new Date("25-oct-2020") }, { topicid:5, topic:"React JS", topic\_date:new Date("10-oct-2020") } ])

\* create collection tasks and insert data

db.createCollection("tasks");

db.tasks.insertMany([

{

taskid:1,

topicid:1,

userid:1,

task:"HTML task",

due\_date:new Date("4-oct-2020"),

submitted:true

},

{

taskid:2,

topicid:2,

userid:2,

task:"CSS task",

due\_date:new Date("11-oct-2020"),

submitted:true

},

{

taskid:3,

topicid:3,

userid:3,

task:"Bootstrap task",

due\_date:new Date("18-oct-2020"),

submitted:false

},

{

taskid:4,

topicid:4,

userid:4,

task:"JavaScript task",

due\_date:new Date("25-oct-2020"),

submitted:true

},

{

taskid:5,

topicid:5,

userid:5,

task:"React JS task",

due\_date:new Date("10-oct-2020"),

submitted:false

}

])

---------------

\* create collection attendance and insert data

db.createCollection("attendance");

db.attendance.insertMany([

{

userid:1,

topicid:2,

attended:true

},

{

userid:2,

topicid:1,

attended:true

},

{

userid:3,

topicid:5,

attended:false

},

{

userid:4,

topicid:3,

attended:true

},

{

userid:5,

topicid:4,

attended:false

}

])

-----------

\* create collection mentors and insert data

db.createCollection("mentors");

db.mentors.insertMany([

{

mentorid:1,

mentorname:"Ravi",

mentor\_email:"ravi@gmail.com",

mentee\_count: 15

},

{

mentorid:2,

mentorname:"Praveen",

mentor\_email:"praveen@gmail.com",

mentee\_count:20

},

{

mentorid:3,

mentorname:"Suman",

mentor\_email:"suman@gmail.com",

mentee\_count:10

},

{

mentorid:4,

mentorname:"Kishor",

mentor\_email:"kishor@gmail.com",

mentee\_count:18

},

{

mentorid:5,

mentorname:"Riyaz",

mentor\_email:"riyaz@gmail.com",

mentee\_count:25

}

])

---------

\* create collection company drives and insert data

db.createCollection("companydrives");

db.comapnydrives.insertMany([

{

userid:1,

drive\_date:new Date("5-oct-2020"),

company:"TCS"

},

{

userid:1,

drive\_date:new Date("18-oct-2020"),

company:"HCL"

},

{

userid:2,

drive\_date:new Date("23-oct-2020"),

company:"Amazon"

},

{

userid:3,

drive\_date:new Date("28-oct-2020"),

company:"Wipro"

},

{

userid:4,

drive\_date:new Date("2-nov-2020"),

company:"ITC"

}

])

|  |
| --- |
|  |
| Find all the information about each products |
|  | Find the product price which are between 400 to 800 |
|  | Find the product price which are not between 400 to 600 |
|  | List the four product which are grater than 500 in price |
|  | Find the product with a row id of 10 |
|  | Find only the product name and product material |
|  | Find all products which contain the value of soft in product material |
|  | Find products which contain product color indigo and product price 492.00 |
|  | Delete the products which product price value are same |
|  |  |
|  |  |
|  | 1>Find all the information about each products |
|  | answer-> |
|  | db().collection().find({}) |
|  |  |
|  |  |
|  | 2>Find the product price which are between 400 to 800 |
|  | answer--> |
|  | db.collection().find({product\_price:{$gt:400 , $lt:800} }) |
|  |  |
|  |  |
|  | 3>Find the product price which are not between 400 to 600 |
|  | answer--> |
|  | db.collection().find({product\_price:{$not:{$gt:400 , $lt:800}}}).pretty() |
|  |  |
|  | 4>List the four product which are grater than 500 in price |
|  | answer--> |
|  | db.collection().find({product\_price:{$gt:500}}).limit(4).pretty() |
|  |  |
|  |  |
|  | 5>Find the product with a row id of 10 |
|  | answer--> |
|  | db.collection().find({"id":"10"}) |
|  |  |
|  |  |
|  | 6>Find only the product name and product material |
|  | answer--> |
|  | db.people.find({},{product\_name:1,product\_material:1}).pretty() |
|  |  |
|  |  |
|  | 7>Find all products which contain the value of soft in product material |
|  | answer--> |
|  | db.people.find({product\_material:"Soft"}).pretty() |
|  |  |
|  |  |
|  |  |
|  | 8>Find products which contain product color indigo and product price 492.00 |
|  | answer--> |
|  | db.people.find({product\_color:"indigo",product\_price:"492.00"}).pretty() |
|  |  |
|  | 9>Delete the products which product price value are same |
|  | answer--> |
|  | Ans: db.people.aggregate([{$group:{\_id:"$product\_price",count: {$sum: 1}}},{$match:{count:{"$gt":1}}}]) |
|  |  |
|  | db.people.remove({"product\_price":36}) |
|  |  |
|  | db.people.remove({"product\_price":47}) |
|  |  |
|  | 10>Find the product name and product material of each products |
|  | Ans: db.people.find({},{product\_name:1,product\_material:1,\_id:0}) |
|  |  |