**Mongo Db Database – Mentor Session**

**Program**

**Input**

**Process**

**Output**

**File base system**

**Emp.txt**

**Id,name,salary**

**1,ravi,12000**

**Database system : table format**

**DBMS : Excel sheet is type of DBMS**

**Employee**

**Id name salary**

**1 Ravi 12000**

**1 Ramesh 14000**

**RDMBS**

**Relational Database Management System**

**Trainer**

**PK**

**TId TName tech**

**1 Raj Java**

**2 Ravi Python**

**Student**

**PK FK**

**Sid Sname Age TSId**

**100 Seeta 21 1**

**101 Reeta 22 1**

**102 Veeta 23 2**

**Limitation of RDBMS**

**Json format, xml format etc.**

**Non table format**

**No SQL Database**

**Key-value format Redis**

**Graph format Neo4j**

**Document Format mongo db**

**Column family Cassandra**

**Mongo DB : is an open source which help to store the data using document in the form of json.**

**{“id”:100,”name”:”Ravi”,”salary”:12000,”result”:true}**

**Mysql :int, float, char, varchar(100), date, json**

**In MySQL**

**Employee ---Schema (logical entity)**

**Id 🡪int Name 🡪varchar(10) Salary-🡪 float age city**

**1 Raj 12000 null null**

**2 Ramesh 14000 null Bangalore**

**3 Raju 16000 null null**

**4 Ajay 17000 21 null**

**5 Mahesh 19000 null Bangalore**

**Mongo DB allows use to create table without schema. Because mongo db help use to**

**Store data document format. Every document can contains one or more than attribute**

**MySQL Mongo DB**

**Database database**

**Table Collection**

**Records document (json)**

**show dbs;**

**or**

**show databases**

**create database mydb; In MySQL**

**use mydb;**

**Mongo DB**

**use mydb; this command is use to create the database as well as it will switch to that database. If database already present it will switch to that database.**

**show tables;**

**or**

**show collections**

**mongo db provided pre-defined object ie db which contains lot of pre-defined function which help to create, delete, update, retrieve all document from a collection.**

**db.createCollection(“Sample”) : it is use to create the collection.**

**In mongo db collection is like a table. In collection we can store more than one document in the form of json.**

**Insert document in Collection**

**db.Sample.insert({"Name":"Ravi"});**

**db.Sample.insert({"FName":"Raj",Lname:"Deep",Age:34,City:"Bangalore"});**

**View the document from a collection**

**db.Sample.find();**

**by default mongo db internally created \_id field to store unique value by default unique value generated using ObjectId object. if we want we can pass the unique value but field name must be \_id.**

**\_id is like primary key.**

**We will create collection with name Employee like field as \_id, name, salary, age, city**

**Store minimum 5 to 8 documents.**

**We can retrieve the document from collection using index position.**

**db.Employee.find()[0]; it is use to retrieve 0 position document**

**db.Employee.find()[3].\_id it is use to retrieve 3 index position id value.**

**db.Employee.find()[2].name it is use to retrieve 2 index position name value.**

**Select \* from employee; it will retrieve all columns from table**

**Select id,name from employee it will retrieve id and name column from table**

**db.Employee.find(); : it will display all document fields from a collection**

**db.Employee.find({condition},{fieldsName});**

**db.Employee.find({},{name:1}) : 1 is true consider.**

**db.Employee.find({},{name:1}); it retrieve name and \_id fields**

**db.Employee.find({},{name:1,age:1}) it retrieve \_id,name and age fields**

**db.Employee.find({},{name:1,age:1,\_id:0}) it retrieve name and age**

**limit() and skip() these function help control viewing documents from collection.**

**db.Employee.find().limit(2); : top 2 document display**

**db.Employee.find().skip(2); : top 2 document skip**

**db.Employee.find().skip(2).limit(2); : skip 2 and display after only 2**

**if we want to apply condition while retrieving document from collection like where clause**

**db.CollectionName.find({condition});**

**db.Employee.find({\_id:1});**

**db.Employee.find({name:"Raju"});**

**db.Employee.find({city:"Bangalore"});**

**db.Employee.find({age:25});**

**db.Employee.find({salary:{$gt:25000}});**

**db.Employee.find({salary:{$gte:25000}});**

**db.Employee.find({salary:{$lt:25000}});**

**db.Employee.find({salary:{$lte:25000}});**

**db.Employee.find({salary:{$eq:25000}});**

**db.Employee.find({salary:{$ne:25000}});**

**write more than one condition wit**

**$and : both the condition must be true**

**$or : any one of the condition must be true**

**db.Employee.find({$and:[{\_id:2},{salary:32000}]});**

**db.Employee.find({$or:[{\_id:3},{salary:32000}]});**

**sort() : this function is use to display the document in ascending or descending order.**

**db.Employee.find().sort({salary:1}); sort by salary 1 means asc and -1 desc**

**db.Employee.find().sort({salary:-1});**

**update the document**

**db.Employee.update({\_id:1},{$set:{salary:35000}});**

**db.Employee.updateMany({city:"Bangalore"},{$set:{city:"Bengaluru"}});**

**remove the document**

**db.Employee.remove({}); all document removed from database but empty collection present.**

**db.Employee.remove({\_id:6});**

**db.Employee.remove({city:'New Delhi'});**

**to remove collection or drop collection.**

**db.Employee.drop(); : this command is use to remove all document as well as collection from database.**

**Student -🡪 Table**

**Sid SName Age**

**1 Raj 21**

**2 Lokesh 25**

**Subject 🡪 Table**

**SubId SubName**

**100 Phy**

**101 Che**

**102 Bio to retrieve records from more than one table we need to use**

**103 Math join it may inner or equi join.**

**StudentSubject**

**Id Sid SubId**

**1111 1 100**

**2222 1 101**

**3333 1 102**

**4444 2 100**

**5555 2 101**

**Storing array values in collection**

**db.Student.insert({\_id:1,sname:"Ravi",age:21,sub:["Phy","Math","Che","Bio"]});**

**db.Student.find({sub:'Math'});**

**db.Student.find({'sub.1':'Math'});**

**relationship**

**MySQL Relationship**

**One to Many -🡪 Trainer to Student one means PK and many means FK**

**One to One 🡪 Person to Passport**

**Many to one 🡪 Employees to project or departments**

**Many to many -🡪 Employees to SkillSet**

**In mongo DB we can achieve same type of relationship using two ways without FK**

1. **Mongo DB embedded relationship**
2. **Mongo DB linking relationship**

**Embedded type relationship ( we are storing add data in only one collection)**

**One employee have only one address -🡪 one to one relationship**

**One employee working more than one project 🡪 one to many relationship**

**Employees**

**\_id**

**name**

**salary**

**add**

**city**

**state**

**projects**

**db.Employees.find().pretty();**

**Linking style we use more than one collection.**

**One to many relationship**

**Trainer**

**db.Trainer.insert({\_id:1,tname:"Raj",tech:"Java"});**

**db.Trainer.insert({\_id:2,tname:"Ravi",tech:"Python"});**

**Student1 : we store only trainer Id with student document**

**db.Student1.insert({\_id:100,sname:"Seeta",age:21,tsid:db.Trainer.find()[0].\_id});**

**db.Student1.insert({\_id:101,sname:"Reeta",age:22,tsid:db.Trainer.find()[0].\_id});**

**db.Student1.insert({\_id:103,sname:"Leeta",age:24,tsid:db.Trainer.find()[1].\_id});**

**db.Student1.insert({\_id:104,sname:"Heeta",age:25,tsid:[db.Trainer.find()[0].\_id,db.Trainer.find()[0].\_id]});**

**Student2 :we store complete trainer details or document with student document.**

**Aggregate function**

**It is use to group multiple document from same collection or different collection and then perform aggregate operation on those document and it return single result or multiple result base upon group.**

**db.Employee.insertMany([**

**{\_id:1,name:"Raj",salary:24000,deptId:100,city:"Bangalore"},**

**{\_id:2,name:"Ravi",salary:26000,deptId:101,city:"Delhi"},**

**{\_id:3,name:"Ramesh",salary:23000,deptId:101,city:"Bangalore"},**

**{\_id:4,name:"Rajesh",salary:22000,deptId:102,city:"Bangalore"},**

**{\_id:5,name:"Reeta",salary:24000,deptId:102,city:"Mumbai"},**

**{\_id:6,name:"Lokesh",salary:28000,deptId:100,city:"Bangalore"},**

**{\_id:7,name:"Mahesh",salary:29000,deptId:101,city:"Delhi"},**

**{\_id:8,name:"Ram",salary:21000,deptId:101,city:"Delhi"},**

**{\_id:9,name:"Rajev",salary:22000,deptId:102,city:"Bangalore"},**

**{\_id:10,name:"Ramu",salary:23000,deptId:102,city:"Mumbai"}**

**]);**

**Total salary of all employee in particular collection**

**db.Employee.aggregate([{$group:{\_id:"",totalSalary:{$sum:"$salary"}}}]);**

**db.Employee.aggregate([{$group:{\_id:"",maxSalary:{$max:"$salary"}}}]);**

**db.Employee.aggregate([{$group:{\_id:"",minSalary:{$min:"$salary"}}}]);**

**db.Employee.aggregate([{$group:{\_id:"",avgSalary:{$avg:"$salary"}}}]);**