Students

StudRollNo
StudName
Grade
Hobbies
DOJ

Show dbs;
Use ch6mng;

	RDBMS	MONGODB
Create	Create table Students (StudRollNo varchar2(20), StudName varchar2(50), Grade varchar2(5), Hobbies varchar2(50), DOJ date)	db.createCollection("Students")

INSERT	Insert into Students (StudRollNo,
	StudName,
	Grade,
	Hobbies,
	DOJ) values ('S101','','VII',...)
		db.Students.insert(
		{
		_id:1,
		StudRollNo:'S101',
		StudName:,
		Grade:'VII',
		Hobbies:'Net Surfing',
		DOJ:'10-OCT-2012',
		}
		)

Update	Update Students
	Set Hobbies='Ice Hockey'
	Where StudRollNo='S101'
	
	Update Students
	Set Hobbies='Ice Hockey'
		db.Students.update(
		{StudRollNo:'S101'},
		{$set:
		{
		Hobbies:'Ice Hockey'
		}
		}
		)
		
		db.Students.update(
		{ },
		{$set:
		{
		Hobbies:'Ice Hockey'
		}

		}, {multi:true},)

Delete	Delete from Students where StudRollNo='S101'
	
	Delete from Students
		db.Students.remove(
		{
		StudRollNo:'S101'
		}
		)
		db.Students.remove({}
		)

Select	Select * from Students Select * from Students Where StudRollNo='S101' Select StudRollNo, StudName, Hobbies from Students Select StudRollNo, StudName, Hobbies from Students Where Grade='VII' and Hobbies='Ice Hockey' Select StudRollNo, StudName, Hobbies from Students Where Grade='VII' or Hobbies='Ice Hockey'	db.Students.find({}).pretty() db.Students.find({StudRollNo:'S101'}) db.Students.find({}, {StudRollNo:1, StudName:1, Hobbies:1, _id:0}) db.Students.find({Grade:'VII', Hobbies:'Ice Hockey'}, {StudRollNo:1, StudName:1, Hobbies:1, _id:0}) db.Students.find({ $or :[{Grade:'VII'},{Hobbies:'Ice Hockey'}] })

	Select * from Students Where StudName like 'S%'	db.Students.find({$or:[{Grade:'VII'}, {Hobbies:'Ice Hockey'}]}, {StudRollNo:1, StudName:1, Hobbies:1, _id:0}) https://docs.mongodb.com/manual/reference/operator/query/or/#mongodb-query-op.-or db.Students.find({StudName:/^S/})

	Objective/Aim	MongoDB Query
1	Show all collections	Show collections
2	Drop a collection named food	db.food.drop()
3	Update else insert	db.Students.update(..., ..., {upsert:true})

4	save() method
	_id exist then replace else insert
		db.Students.save(
		{
		_id:1,
		StudRollNo:'S101',
		StudName:'James',
		Grade:'VII',
		Hobbies:'Net Surfing',
		DOJ:'10-OCT-2012',
		}
		)

5	Adding a new field to existing document	db.Students.update({},{$set:{Status:"A"}} ,{multi:true})
6	Removing an existing field from an existing document	db.Students.update({"_id" : 1},{$unset:{Status:"A"}})
7	Relational operators	db.Students.find({Grade:{$ne:'VII'}}).prett

	$eq $ne $gte $lte $gt $lt To find those documents where the grade is not set to 'VII'	y()
8	Regular expression	db.Students.find({StudName:{$regex:"^J"}}).pretty()
9	Dealing with NULL values	db.Students.update({},{$set:{Location:null}}) db.Students.find({Location:{$ne:null}}).pretty() https://docs.mongodb.com/manual/tutorial/query-for-null-fields/
10	Count, Limit, Sort and skip	db.Students.count() db.Students.count({}) db.Students.find({}).limit(3) db.Studetns.find({}).sort({StudName:1}) 1 Ascending -1 Descending db.Studetns.find({}).sort({StudName:1,Hobbies:-1}) db.Students.find({}).skip(2)

	In vs nin
	Select * from
	Students
		db.Students.find({
		Hobbies:{$in:['Chess','Skating']
		}

	Where hobbies in ('Chess','Skating');	})
		
		db.Students.find({
		Hobbies:{$nin:['Chess','Skating']
		}
		})

		db.Students.count()
Arrays	Food	db.food.insert({_id:1,fruits:['banana','apple','cherry']}) db.food.insert({_id:2,fruits:['orange','butterfruit','mango']}) db.food.insert({_id:3,fruits:['pineapple','strawberry','grapes']}) db.food.insert({_id:4,fruits:['banana','strawberry','grapes']}) db.food.insert({_id:5,fruits:['orange','grapes']})
	To find those documents from the food have fruits array having grapes in the first index position. The index position begins at 0.	db.food.find({'fruits.1':'grapes'})
	To find those documents from the food have fruits array having grapes in the 2nd index position. The index position begins at 0.	db.food.find({'fruits.2':'grapes'})
	To find those documents from the “food” where the size of the array fruits is 2. The size implies that the array holds only 2 values.	db.food.find({fruits:{$size:2}}) db.food.find({"fruits":{$size:2}})
	To find those documents from the “food” where the size of the array fruits is 3. The size implies that the array holds only 3 values.	db.food.find({fruits:{$size:3}}) db.food.find({"fruits":{$size:3}})
	Show only first two elements of array	db.food.find({_id:1},{"fruits":{$slice:2}})
	Start from 0 index location, show total two	db.food.find({_id:1},{"fruits":{$slice:[0,2]}})
	Start from 1 index location, show total two	db.food.find({_id:1},{"fruits":{$slice:[1,2]}})
		db.food.find({_id:1},{"fruits":{$slice:[2,3]}}) May show 3 or less if not available

	all	db.food.find({"fruits":{$all:["orange","grapes"]}})
		db.food.find({'fruits.0':'orange'}) db.food.find({"fruits.0":'orange'})
	Array update Replace the element in the 1st index position	db.food.update({_id:4},{$set:{'fruits.1':'apple'}}
		db.food.update({_id:1,'fruits':'apple'},{$set:{'fruits.$':'An apple'}}
		db.food.update({_id:2},{$push:{price:{orange:60,butterfruit:200,mango:120}}} '
		db.food.update({_id:4},{$addToSet:{fruits:'orange'}})
	Pop The element popped is the one from the end of the array.	db.food.update({_id:4},{$pop:{fruits:1}})
	from the beginning of the array.	db.food.update({_id:4},{$pop:{fruits:-1}})
		db.food.update({_id:3},{$pullAll:{fruits:['pineapple','grapes']}})
		db.food.update({fruits:'banana'},{$pull:{fruits:'banana'}})
	To pull out an array element based on index position	db.food.update({_id:4},{$unset:{'fruits.1':null}}) db.food.update({_id:4},{$pull:{'fruits':null}})
	Aggregate Function	https://docs.mongodb.com/manual/aggregation/
		db.Customers.count()

	Single Purpose Aggregation Operations	db.Customers.distinct("CustID") ["C123", "C111"] db.Customers.find() db.Customers.aggregate({$group:{_id:"$CustID",TotalAccBal:{$sum:"$AccBal"}}}) db.Customers.aggregate({$group:{_id:"$CustID",MaxAccBal:{$max:"$AccBal"}}}) db.Customers.aggregate({$group:{_id:"$CustID",MinAccBal:{$min:"$AccBal"}}}) db.Customers.aggregate({$group:{_id:"$CustID",AvgAccBal:{$avg:"$AccBal"}}}) db.Customers.aggregate({$match:{AccBal:{$gte:1200}}},{$group:{_id:"$CustID",AvgAccBal:{$avg:"$AccBal"}}}) db.Customers.aggregate({$group:{_id:"$CustID",AvgAccBal:{$avg:"$AccBal"}}},{$match:{AvgAccBal:{$gt:1000}}}) db.Customers.aggregate({$match:{AccBal:{$gte:1200}}},{$group:{_id:"$CustID",AvgAccBal:{$avg:"$AccBal"}}},{$match:{AvgAccBal:{$gt:1200}}})

		db.Customers.aggregate({$match:{AccBal:{$gte:1200}}},{$group:{_id:"$CustID",AvgAccBal:{$avg:"$AccBal"}}},{ $sort: { AvgAccBal:1} }) db.Customers.aggregate({$match:{AccBal:{$gte:1200}}},{$group:{_id:"$CustID",AvgAccBal:{$avg:"$AccBal"}}},{ $sort: { AvgAccBal:-1} })
	Cursor and more	db.alphabets.insert({_id:1,alphabet:"a"}) db.alphabets.insert({_id:2,alphabet:"b"}) db.alphabets.insert({_id:3,alphabet:"c"}) db.alphabets.insert({_id:4,alphabet:"d"}) db.alphabets.insert({_id:5,alphabet:"e"}) db.alphabets.insert({_id:6,alphabet:"f"}) db.alphabets.insert({_id:7,alphabet:"g"}) db.alphabets.insert({_id:8,alphabet:"h"}) db.alphabets.insert({_id:9,alphabet:"i"}) db.alphabets.insert({_id:10,alphabet:"j"}) db.alphabets.insert({_id:11,alphabet:"k"}) db.alphabets.insert({_id:12,alphabet:"l"}) db.alphabets.insert({_id:13,alphabet:"m"}) db.alphabets.insert({_id:14,alphabet:"n"}) db.alphabets.insert({_id:15,alphabet:"o"}) db.alphabets.insert({_id:16,alphabet:"p"}) db.alphabets.insert({_id:17,alphabet:"q"}) db.alphabets.insert({_id:18,alphabet:"r"}) db.alphabets.insert({_id:19,alphabet:"s"}) db.alphabets.insert({_id:20,alphabet:"t"}) db.alphabets.insert({_id:21,alphabet:"u"}) db.alphabets.insert({_id:22,alphabet:"v"}) db.alphabets.insert({_id:23,alphabet:"w"}) db.alphabets.insert({_id:24,alphabet:"x"}) db.alphabets.insert({_id:25,alphabet:"y"})

		db.alphabets.insert({_id:26,alphabet:"z"})
		
		var myCur= db.alphabets.find({})
		while(myCur.hasNext()){
		var myRec=myCur.next()
		print("The alphabet is :
		"+myRec.alphabet);
		}
		
		db.alphabets.ensureIndex({"alphabet":1}
		);
		
		db.alphabets.getIndexs()
		
		sequence
		
		db.usercounters.insert(
		{
		_id:"empid",
		seq:0
		}
		)
		function getnextseq(name)
		{
		var
		ret=db.usercounters.findAndModify({
		query={_id:name},
		update:{$inc:{seq:1}},
		new:true
		}
		
		});
		
		}