10) Implement CRUD operations. SAVE method. Use following Collection. Perform Map Reduce to count quantity of each item.

Item: Item ID, Item quantity, price, brand, discount

db.createCollection("Item");

db.Item.insertMany([

{ ItemID: 1, ItemQuantity: 10, Price: 20, Brand: "Brand1", Discount: 5 },

{ ItemID: 2, ItemQuantity: 5, Price: 15, Brand: "Brand2", Discount: 10 },

{ ItemID: 3, ItemQuantity: 8, Price: 25, Brand: "Brand1", Discount: 7 },

{ ItemID: 4, ItemQuantity: 12, Price: 30, Brand: "Brand2", Discount: 12 },

{ ItemID: 5, ItemQuantity: 15, Price: 22, Brand: "Brand1", Discount: 8 }

]);

var mapfunction=function(){emit(this.Brand,this.ItemQuantity)};

dbms-prac> var reducefunction=function(key,values){return Array.sum(values)};

db.Item.mapReduce( mapfunction,reducefunction,{out:"Brand\_quantity"});

db.Brand\_quantity.find();

1. Display the count of item brand wise.

db.Item.aggregate([{$group:{\_id:"$Brand",count:{$sum:1}}}]);

2. Dsiplay item with minimum price.

db.Item.find().sort({Price:1}).limit(1);

3. Display maximum discount given for item.

db.Item.find().sort({Discount:-1}).limit(1);

11) Implement CRUD operations. SAVE method. Use following Collection.

Item: Item ID, Item quantity, price, brand, discount

1. Display the count of item brand wise.

2. Dsiplay item with minimum price.

3. Display maximum discount given for item.

.

12) Implement CRUD operations. SAVE method. Use following Collection.

Item: Item ID, Item quantity, price, brand, discount

1. Display the count of item brand wise.

2. Dsiplay item with minimum price.

3. Display maximum discount given for item.

13) Implement Map reduces operation for counting the marks of students.

Use: student (roll\_no, name marks, class)

Expected output: student name or roll no and total marks.

db.createCollection('students');

db.student.insertMany([

{ roll\_no: 1, name: "Alice", marks: [85, 90, 78], class: "10th" },

{ roll\_no: 2, name: "Bob", marks: [75, 80, 92], class: "10th" },

{ roll\_no: 3, name: "Charlie", marks: [90, 88, 86], class: "10th" },

{ roll\_no: 4, name: "David", marks: [92, 95, 89], class: "10th" },

{ roll\_no: 5, name: "Eve", marks: [78, 85, 88], class: "10th" }

]);

var mapFunction=function(){

emit(this.name,Array.sum(this.marks));

};

var reduceFunction=function(key,values){

return Array.sum(values);

};

db.student.mapReduce(mapFunction,reduceFunction,{out:"total\_marks"});

db.total\_marks.find();

14) Implement Map reduces operation for displaying persons with same profession.

Use: person (person\_id, name, addr, profession)

db.createCollection("person")

db.person.insertMany([

{ person\_id: 1, name: "Alice", addr: "123 Main St", profession: "Engineer" },

{ person\_id: 2, name: "Bob", addr: "456 Elm St", profession: "Doctor" },

{ person\_id: 3, name: "Charlie", addr: "789 Oak St", profession: "Engineer" },

{ person\_id: 4, name: "David", addr: "101 Pine St", profession: "Lawyer" },

{ person\_id: 5, name: "Eve", addr: "202 Maple St", profession: "Doctor" }

]);

var mapFunction = function () {

emit(this.profession, this.name);

};

db.person.mapReduce(mapFunction, reduceFunction, {

out: "persons\_by\_profession"

});

db.persons\_by\_profession.find();

15) Perform CRUD operation in mongo db –

Use : person( person\_id, name, addr, profession )

1.Create Collection.

db.createCollection("person");

2.Inserting data in collection.

db.person.insertOne({

person\_id: 1,

name: "Alice",

addr: "123 Main St",

profession: "Engineer"

})

db.person.insertMany([

{

person\_id: 2,

name: "Bob",

addr: "456 Elm St",

profession: "Doctor"

},

{

person\_id: 3,

name: "Charlie",

addr: "789 Oak St",

profession: "Engineer"

}

]);

3.Reading data of collection.

db.person.find();

db.person.findOne({ person\_id: 1 });

4.Updating data of collection.

db.person.updateOne(

{ person\_id: 1 },

{ $set: { profession: "Software Developer" } }

);

5.Deleting data from collection.

db.person.deleteOne({ person\_id: 2 });

16) Perform CRUD operation and Aggregation in mongo db

employee(emp\_id,e\_name,salary ,Date of Joining,Dapt\_no,Designation)

db.createCollection("employee");

db.employee.insertMany([

{ emp\_id: 1, e\_name: "Alice", salary: 60000, Date\_of\_Joining: "2016-05-15", Dept\_no: 101, Designation: "Manager" },

{ emp\_id: 2, e\_name: "Bob", salary: 55000, Date\_of\_Joining: "2017-03-20", Dept\_no: 102, Designation: "Sales Associate" },

{ emp\_id: 3, e\_name: "Charlie", salary: 75000, Date\_of\_Joining: "2016-06-10", Dept\_no: 101, Designation: "Engineer" },

{ emp\_id: 4, e\_name: "David", salary: 80000, Date\_of\_Joining: "2018-01-10", Dept\_no: 103, Designation: "Analyst" },

{ emp\_id: 5, e\_name: "Eve", salary: 90000, Date\_of\_Joining: "2016-06-01", Dept\_no: 102, Designation: "Sales Manager" }

])

1. Display the count of employee department wise.

db.employee.aggregate([

{

$group: {

\_id: "$Dept\_no",

count: { $sum: 1 }

}

}

]);

2. Dsiplay the average salary of employee in sales department.

db.employee.aggregate([

{

$match: { Dept\_no: 102 }

},

{

$group: {

\_id: null,

averageSalary: { $avg: "$salary" }

}

}

]);

3. Dsiplay minimum salary to employees joins in June 2016

db.employee.find({

Date\_of\_Joining: {

$gte: "2016-06-01",

$lt: "2016-07-01"

}

}).sort({ salary: 1 }).limit(1);

4. Display maximum salary given to employee in production department.

db.employee.find({

Dept\_no: 103

}).sort({ salary: -1 }).limit(1);

5. Display record of first and last employee department wise.

db.employee.aggregate([

{

$sort: { Dept\_no: 1, emp\_id: 1 }

},

{

$group: {

\_id: "$Dept\_no",

firstEmployee: { $first: "$$ROOT" },

lastEmployee: { $last: "$$ROOT" }

}

}

]);

23) Implement Basic SQL queries.

1. Create table employee.

Create table employee(emp\_id int , e\_name varchar(20),salary int, Date\_of\_joining date ,Dept\_no int ,Designation varchar(20), Department varchar(20));

2. Insert 10 records in table.

3. Create a view emp\_vl of table employee which has emp\_id , name and dept-attributes.

Create view emp\_v1 as select emp\_id, e\_name, Dept\_no from employee;

4. Display name and department of employee working in Manager or Marketing department

select e\_name,Dept\_no from employee where Designation in ('Manager','Finance Manager');

5. Display employees who were hired earliest or latest.

SELECT \*

-> FROM emp1

-> WHERE Date\_of\_joining = (SELECT MIN(Date\_of\_joining) FROM emp1) OR Date\_of\_joining = (SELECT MAX(Date\_of\_joining) FROM emp1);

6. Display name and department no of employees who are manager, market analysts. Use

Predicates

SELECT e\_name, Dept\_no

-> FROM emp1

-> WHERE Designation IN ('Manager', 'Finance Analyst');

List employees hired in August.

SELECT \*

-> FROM emp1

-> WHERE EXTRACT(MONTH FROM Date\_of\_joining) = 8;

List employees who are hired after 31/12/2006.

SELECT \*

-> FROM emp1

-> WHERE Date\_of\_joining > '2006-12-31';

24) ) Indexing and join: Consider the relation

employee (emp\_id,e\_name,salary ,Date of Joining,Dapt\_no,Designation)

Customer(c\_id, c\_name , email , city , pincode)

Order(order\_id , date , amount , cust\_id.

a. create empid as primary key and indices on table employee.

Alter table employee add primary key (emp\_id);

b. create user defined index on any column

Create index in1 on employee (Dept\_no , Designation);

c. create sequence using auo-increment.

ALTER TABLE Customer

MODIFY c\_id INT AUTO\_INCREMENT;

d. truncate table.

TRUNCATE TABLE employee;

e. find list of customers who placed order and details of their orders.

SELECT c.c\_name, o.order\_id, o.date, o.amount

FROM Customer c

INNER JOIN Order o ON c.c\_id = o.cust\_id;

f. find info of customers and append order details to the table/

SELECT c.c\_id, c.c\_name, c.email, c.city, c.pincode, o.order\_id, o.date, o.amount

FROM Customer c

LEFT JOIN Order o ON c.c\_id = o.cust\_id;

g. list down customers who haven’t placed order.

SELECT c.c\_name

FROM Customer c

LEFT JOIN Order o ON c.c\_id = o.cust\_id

WHERE o.cust\_id IS NULL;

25) Implement aggregation and indexing with suitable example in mongodb.

db.createCollection("sales")

db.sales.insertMany([

{ product: "A", category: "Electronics", amount: 100 },

{ product: "B", category: "Books", amount: 50 },

{ product: "C", category: "Electronics", amount: 75 },

{ product: "D", category: "Books", amount: 40 },

{ product: "E", category: "Clothing", amount: 120 },

]);

db.sales.aggregate([{$group:{\_id:"$category",totalSales:{$sum:"$amount"}}}]) ;

db.sales.aggregate([{$group:{\_id:"$category",totalSales:{$avg:"$amount"}}}]) ;

db.sales.aggregate([{$group:{\_id:"$category",totalSales:{$min:"$amount"}}}]) ;

db.sales.aggregate([{$group:{\_id:"$category",totalSales:{$max:"$amount"}}}]) ;

db.sales.aggregate([{$group:{\_id:"$category",totalSales:{$first:"$amount"}}}]) ;

db.sales.aggregate([{$group:{\_id:"$category",totalSales:{$last:"$amount"}}}]) ;

db.sales.aggregate([{$group:{\_id:"$category",totalSales:{$push:"$amount"}}}]) ;

db.sales.aggregate([{$group:{\_id:"$category",count:{$sum:1}}}]) ;

db.sales.createIndex({ product: 1 });

db.sales.createIndex({ amount: -1 });

db.sales.getIndexes();

db.sales.dropIndex({amount:-1});