Create collection called CUSTOMER with following fields in documentsCust\_No, First\_Name, Last\_Name, Address, City, State, Pincode, B\_Date, Status: the values for status must be in (‘Married’,’Unmarried’,’Divorcee’). Implement following queries • Display all the documents where state is KARNATAKA. • Delete the document where PIN CODE is 576201. • Change the ADDRESS as “PICT,Trimurti chowk, Dhankawadi” AND Pin cde as 411041 where CUST\_NO is 1003. • Display Total Number of Married, unmarried and Divorcee Customers • Sort and display the customer data, in the alphabetic order of city. • Retrieve records of Karnataka / Kerala customers who are Married (‘M’). • Perform Create Index, get Index and drop index operation on collection. • Write a mapreduce/aggregation function to calculate total customer per City.

Code:

db.CUSTOMER.insertMany([

{ Cust\_No: 1001, First\_Name: "Rajesh", Last\_Name: "Kumar", Address: "123 Main St", City: "Bangalore", State: "Karnataka", Pincode: 560001, B\_Date: ISODate("1985-01-15"), Status: "Married" },

{ Cust\_No: 1002, First\_Name: "Priya", Last\_Name: "Sharma", Address: "456 Oak St", City: "Mysuru", State: "Karnataka", Pincode: 570001, B\_Date: ISODate("1990-05-21"), Status: "Unmarried" },

{ Cust\_No: 1003, First\_Name: "Amit", Last\_Name: "Patil", Address: "789 Pine St", City: "Mangalore", State: "Karnataka", Pincode: 576201, B\_Date: ISODate("1982-11-30"), Status: "Divorcee" },

// Add more sample documents as needed

]);

// 3. Display all the documents where state is Karnataka

db.CUSTOMER.find({ State: "Karnataka" });

// 4. Delete the document where PIN CODE is 576201

db.CUSTOMER.deleteOne({ Pincode: 576201 });

// 5. Change the ADDRESS and Pin code where CUST\_NO is 1003

db.CUSTOMER.updateOne({ Cust\_No: 1003 }, { $set: { Address: "PICT, Trimurti chowk, Dhankawadi", Pincode: 411041 } });

// 6. Display Total Number of Married, unmarried and Divorcee Customers

db.CUSTOMER.aggregate([

{ $group: { \_id: "$Status", count: { $sum: 1 } } }

]);

// 7. Sort and display the customer data in alphabetic order of city

db.CUSTOMER.find().sort({ City: 1 });

// 8. Retrieve records of Karnataka customers who are Married ('Married')

db.CUSTOMER.find({ State: "Karnataka", Status: "Married" });

// 9. Create Index

db.CUSTOMER.createIndex({ City: 1 });

// 10. Get Index

db.CUSTOMER.getIndexes();

// 11. Drop Index

db.CUSTOMER.dropIndex("City\_1");

// 12. Write a map-reduce/aggregation function to calculate total customers per City

db.CUSTOMER.mapReduce(

function () {

emit(this.City, 1);

},

function (key, values) {

return Array.sum(values);

},

{ out: { inline: 1 } }

);

2. Design the Employee Management System(Institute have different departments like Administrative, Account, Library, CSE,IT,ET,FE etc) each department have different employees with different attribute like empid, ename, city, educational background, salary, post , join date, leaving date if any, Skills etc. using MongoDB • List out the employees who are earning salary between 30000 and 45000. • List out the department name having at least four employees. • Find out no. of employees working in “IT” department. • Display the name of employee who get the maximum salary. • Display Name of Department who have maximum of employees. • Update Name of Department from ‘IT’ to “Information Technology”. • Perform Create Index, get Index and drop index operation on collection. • Write a MapReduce/Aggregation function to display total number of employees per department.

Code:

// 2. Insert some sample documents into the EMPLOYEE collection

db.EMPLOYEE.insertMany([

{ empid: 1, ename: "Rahul Sharma", city: "Delhi", education: "MBA", salary: 45000, post: "Manager", join\_date: ISODate("2021-12-15"), department: "Administrative", skills: ["Management", "Communication"] },

{ empid: 2, ename: "Pooja Patel", city: "Ahmedabad", education: "B.Com", salary: 35000, post: "Accountant", join\_date: ISODate("2022-01-20"), department: "Account", skills: ["Accounting", "Excel"] },

{ empid: 3, ename: "Suresh Kumar", city: "Bangalore", education: "B.Tech", salary: 40000, post: "Software Engineer", join\_date: ISODate("2022-02-10"), department: "CSE", skills: ["Java", "Python"] },

{ empid: 4, ename: "Priya Singh", city: "Mumbai", education: "MCA", salary: 38000, post: "Librarian", join\_date: ISODate("2022-03-05"), department: "Library", skills: ["Library Science"] },

{ empid: 5, ename: "Vikas Yadav", city: "Chennai", education: "B.E", salary: 42000, post: "IT Manager", join\_date: ISODate("2022-04-18"), department: "IT", skills: ["IT Management", "Networking"] },

{ empid: 6, ename: "Anita Verma", city: "Kolkata", education: "Ph.D.", salary: 48000, post: "Professor", join\_date: ISODate("2022-05-22"), department: "ET", skills: ["Electronics", "Teaching"] },

{ empid: 7, ename: "Rajat Mishra", city: "Hyderabad", education: "B.Tech", salary: 41000, post: "Field Engineer", join\_date: ISODate("2022-06-30"), department: "FE", skills: ["Field Work", "Engineering"] },

// Add more sample documents as needed

]);

// 3. List out the employees who are earning a salary between 30000 and 45000

db.EMPLOYEE.find({ salary: { $gte: 30000, $lte: 45000 } });

// 4. List out the department name having at least four employees

db.EMPLOYEE.aggregate([

{ $group: { \_id: "$department", count: { $sum: 1 } } },

{ $match: { count: { $gte: 4 } } }

]);

// 5. Find out the number of employees working in "IT" department

db.EMPLOYEE.count({ department: "IT" });

// 6. Display the name of the employee who gets the maximum salary

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

// 7. Display the name of the department with the maximum number of employees

db.EMPLOYEE.aggregate([

{ $group: { \_id: "$department", count: { $sum: 1 } } },

{ $sort: { count: -1 } },

{ $limit: 1 }

]);

// 8. Update the name of the department from 'IT' to "Information Technology"

db.EMPLOYEE.updateMany({ department: "IT" }, { $set: { department: "Information Technology" } });

// 9. Create Index

db.EMPLOYEE.createIndex({ department: 1 });

// 10. Get Index

db.EMPLOYEE.getIndexes();

// 11. Drop Index

db.EMPLOYEE.dropIndex("department\_1");

// 12. Write a MapReduce/Aggregation function to display the total number of employees per department

db.EMPLOYEE.mapReduce(

function () {

emit(this.department, 1);

},

function (key, values) {

return Array.sum(values);

},

{ out: { inline: 1 } }

);

3. Create Order Management System using MongoDB and Implement Following Statements • Retrieve all the documents fromcollection. • List name of Customer who purchased product “Mobile”. • Change the product quantity from 1 to 3 of product ” Laptop” of any order. • Using $exists, tell me how many customers belongs from Pune city. • Find the customer who purchased shoes and cloth product. • Find the top 3 buyers. • Display all the orders where total amount is >20000. • Perform Create Index, get Index and drop index operation on collection. • write a MapReduce or aggregation function which will return the Total Price per order

Code:

// 2. Insert some sample documents into the ORDER collection

db.ORDER.insertMany([

{ order\_id: 1, customer\_name: "Amit Kumar", product: "Mobile", quantity: 2, price: 15000, city: "Delhi" },

{ order\_id: 2, customer\_name: "Pooja Patel", product: "Laptop", quantity: 1, price: 45000, city: "Ahmedabad" },

{ order\_id: 3, customer\_name: "Rajesh Sharma", product: "Shoes", quantity: 3, price: 2000, city: "Pune" },

{ order\_id: 4, customer\_name: "Neha Singh", product: "Cloth", quantity: 2, price: 5000, city: "Pune" },

// Add more sample documents as needed

]);

// 3. Retrieve all the documents from the collection

db.ORDER.find();

// 4. List the name of the customer who purchased the product "Mobile"

db.ORDER.find({ product: "Mobile" }, { customer\_name: 1, \_id: 0 });

// 5. Change the product quantity from 1 to 3 of the product "Laptop" of any order

db.ORDER.updateOne({ product: "Laptop" }, { $set: { quantity: 3 } });

// 6. Using $exists, tell me how many customers belong to Pune city

db.ORDER.find({ city: { $exists: true, $eq: "Pune" } }).count();

// 7. Find the customer who purchased both "Shoes" and "Cloth" products

db.ORDER.find({ $and: [{ product: "Shoes" }, { product: "Cloth" }] });

// 8. Find the top 3 buyers

db.ORDER.aggregate([

{ $group: { \_id: "$customer\_name", totalAmount: { $sum: "$price" } } },

{ $sort: { totalAmount: -1 } },

{ $limit: 3 }

]);

// 9. Display all the orders where the total amount is >20000

db.ORDER.find({ price: { $gt: 20000 } });

// 10. Create Index

db.ORDER.createIndex({ customer\_name: 1 });

// 11. Get Index

db.ORDER.getIndexes();

// 12. Drop Index

db.ORDER.dropIndex("customer\_name\_1");

// 13. Write a MapReduce or aggregation function which will return the Total Price per order

db.ORDER.mapReduce(

function () {

emit(this.order\_id, this.price);

},

function (key, values) {

return Array.sum(values);

},

{ out: { inline: 1 } }

);

4. Design the Employee Management System(Institute have different departments like Administarative, Account, Library, CSE,IT,ET, FE etc) each department have different employees with different attribute like empid, ename, city, educational background, salary, post , join\_date, leaving date if any, Skills etc. using MongoDB Implement following statements. • List all the employee frominstitute. • List the employee details that are from Baroda or Ahmedabad and working in CSE department. • List of the empid, ename, department number and skill of employee whose join date is 20th of anymonth. • Calculate total experience of employee. Consider the today’s date. • List the name of employee whose name staring with ‘s’ or ‘m’ character who are working in FE department and having “Programming” skill. • Count the no of employee working in ETC department of Pune Location. • Calculate department wise total salary and display only those departments which pay maximumsalary. • Perform Create Index, get Index and drop index operation on collection. • Using Mapreduce/aggregation Display total no of employees from each department.

// 2. Insert some sample documents into the EMPLOYEE collection

db.EMPLOYEE.insertMany([

{ empid: 1, ename: "Rahul Sharma", city: "Delhi", education: "MBA", salary: 50000, post: "Manager", join\_date: ISODate("2022-01-15"), department: "Administrative", skills: ["Management", "Communication"] },

{ empid: 2, ename: "Pooja Patel", city: "Ahmedabad", education: "B.Com", salary: 35000, post: "Accountant", join\_date: ISODate("2022-02-20"), department: "Account", skills: ["Accounting", "Excel"] },

{ empid: 3, ename: "Suresh Kumar", city: "Baroda", education: "B.Tech", salary: 40000, post: "Software Engineer", join\_date: ISODate("2022-03-10"), department: "CSE", skills: ["Java", "Python"] },

{ empid: 4, ename: "Manisha Singh", city: "Ahmedabad", education: "MCA", salary: 38000, post: "Librarian", join\_date: ISODate("2022-04-05"), department: "Library", skills: ["Library Science"] },

{ empid: 5, ename: "Mohan Verma", city: "Pune", education: "B.E", salary: 42000, post: "IT Manager", join\_date: ISODate("2022-05-18"), department: "IT", skills: ["IT Management", "Networking"] },

{ empid: 6, ename: "Sonia Verma", city: "Baroda", education: "Ph.D.", salary: 48000, post: "Professor", join\_date: ISODate("2022-06-22"), department: "ET", skills: ["Electronics", "Teaching"] },

{ empid: 7, ename: "Nitin Sharma", city: "Pune", education: "B.Tech", salary: 41000, post: "Field Engineer", join\_date: ISODate("2022-07-30"), department: "FE", skills: ["Field Work", "Engineering"] },

// Add more sample documents as needed

]);

// 3. List all the employees from the institute

db.EMPLOYEE.find();

// 4. List the employee details that are from Baroda or Ahmedabad and working in CSE department

db.EMPLOYEE.find({ $and: [{ $or: [{ city: "Baroda" }, { city: "Ahmedabad" }] }, { department: "CSE" }] });

// 5. List the empid, ename, department number, and skill of employees whose join date is 20th of any month

db.EMPLOYEE.find({ join\_date: { $gte: ISODate("2022-01-20"), $lt: ISODate("2023-01-01") } }, { empid: 1, ename: 1, department: 1, skills: 1, \_id: 0 });

// 6. Calculate total experience of employees. Consider today’s date.

db.EMPLOYEE.aggregate([

{

$addFields: {

totalExperience: {

$divide: [

{ $subtract: [new Date(), "$join\_date"] },

1000 \* 60 \* 60 \* 24 \* 365

]

}

}

}

]);

// 7. List the name of employees whose name starts with 's' or 'm' character who are working in FE department and have "Programming" skill

db.EMPLOYEE.find({

$and: [

{ $or: [{ ename: /^S/ }, { ename: /^M/ }] },

{ department: "FE" },

{ skills: "Programming" }

]

});

// 8. Count the number of employees working in ETC department of Pune Location

db.EMPLOYEE.count({ $and: [{ department: "ETC" }, { city: "Pune" }] });

// 9. Calculate department-wise total salary and display only those departments which pay the maximum salary

db.EMPLOYEE.aggregate([

{

$group: {

\_id: "$department",

totalSalary: { $sum: "$salary" }

}

},

{

$sort: { totalSalary: -1 }

},

{

$limit: 1

}

]);

// 10. Create Index

db.EMPLOYEE.createIndex({ department: 1 });

// 11. Get Index

db.EMPLOYEE.getIndexes();

// 12. Drop Index

db.EMPLOYEE.dropIndex("department\_1");

// 13. Using MapReduce/aggregation, display the total number of employees from each department

db.EMPLOYEE.mapReduce(

function () {

emit(this.department, 1);

},

function (key, values) {

return Array.sum(values);

},

{ out: { inline: 1 } }

);

5. Create Order Management System using MongoDB and Implement Following Statements • Display all documents in a collection • List the customer in ascending order of theirnames. • Display all the orders which placed before April2022 • Display Name of Customer who purchased order whose price is more than 25000. • Display all orders that contain product "PenDrive" • Update Order\_date of Any order Purchased by Customer “ABC”. • List all documents with orders that contain products whose quantity is less than 10. • Display the Mob No of customers who have highest Buying Total. • Perform Create Index, get Index and drop index operation on collection. • Using MapReduce/Aggregation display total order per customer.

Code:

// 2. Insert some sample documents into the ORDER collection

db.ORDER.insertMany([

{ order\_id: 1, customer\_name: "Rahul Sharma", product: "Mobile", quantity: 2, price: 15000, order\_date: ISODate("2022-01-15"), mobile\_number: "9876543210" },

{ order\_id: 2, customer\_name: "Pooja Patel", product: "Laptop", quantity: 1, price: 45000, order\_date: ISODate("2022-02-20"), mobile\_number: "8765432109" },

{ order\_id: 3, customer\_name: "Suresh Kumar", product: "PenDrive", quantity: 5, price: 2000, order\_date: ISODate("2022-03-10"), mobile\_number: "7654321098" },

{ order\_id: 4, customer\_name: "Manisha Singh", product: "Laptop", quantity: 2, price: 50000, order\_date: ISODate("2022-03-15"), mobile\_number: "6543210987" },

{ order\_id: 5, customer\_name: "Mohan Verma", product: "PenDrive", quantity: 10, price: 10000, order\_date: ISODate("2022-04-18"), mobile\_number: "5432109876" },

{ order\_id: 6, customer\_name: "Sonia Verma", product: "Mobile", quantity: 3, price: 30000, order\_date: ISODate("2022-05-22"), mobile\_number: "4321098765" },

{ order\_id: 7, customer\_name: "Nitin Sharma", product: "Laptop", quantity: 1, price: 41000, order\_date: ISODate("2022-06-30"), mobile\_number: "3210987654" },

// Add more sample documents as needed

]);

// 3. Display all documents in the collection

db.ORDER.find();

// 4. List the customer in ascending order of their names

db.ORDER.find().sort({ customer\_name: 1 });

// 5. Display all the orders which placed before April 2022

db.ORDER.find({ order\_date: { $lt: ISODate("2022-04-01") } });

// 6. Display Name of Customer who purchased order whose price is more than 25000

db.ORDER.find({ price: { $gt: 25000 } }, { customer\_name: 1, \_id: 0 });

// 7. Display all orders that contain the product "PenDrive"

db.ORDER.find({ product: "PenDrive" });

// 8. Update Order\_date of Any order Purchased by Customer "ABC"

db.ORDER.updateOne({ customer\_name: "ABC" }, { $set: { order\_date: ISODate("2022-07-01") } });

// 9. List all documents with orders that contain products whose quantity is less than 10

db.ORDER.find({ quantity: { $lt: 10 } });

// 10. Display the Mobile No of customers who have the highest Buying Total

db.ORDER.aggregate([

{

$group: {

\_id: "$mobile\_number",

totalPurchase: { $sum: "$price" }

}

},

{

$sort: { totalPurchase: -1 }

},

{

$limit: 1

},

{

$project: { \_id: 0, mobile\_number: "$\_id" }

}

]);

// 11. Create Index

db.ORDER.createIndex({ customer\_name: 1 });

// 12. Get Index

db.ORDER.getIndexes();

// 13. Drop Index

db.ORDER.dropIndex("customer\_name\_1");

// 14. Using MapReduce/Aggregation, display the total order per customer

db.ORDER.mapReduce(

function () {

emit(this.customer\_name, 1);

},

function (key, values) {

return Array.sum(values);

},

{ out: { inline: 1 } }

);

6. Design the Student Management System(Institute have different departments like CSE,IT,ET,FE etc) each department have different employees with different attribute like student\_id, student\_name, address, birthdate, CGPA, fee, current\_year(FE/SE/TE/BE) , join\_date, Skills etc. using MongoDB Implement following statements. • Display the count of total no students frominstitute. • Display all the Students in seniority level (based on CGPA) • List the student details that are from Baroda or Ahmedabad and in CSE department. • List of the studentid, studentname, department number and skill of student whose birth date is 20th of any month. • Calculate age of each student. Consider the today’s date. • List the name of student whose name staring with ‘s’ or ‘m’ character who are in computer department and having typing skill. • Count the no of student in IT department of Pune. • Arrange the student name in alphabetic order whose age between 18 to 20 and in ETC department. • Perform Create Index, get Index and drop index operation on collection. • Write mapreduce or aggregation function to Display total no of students from eachdepartment

Code;

db.STUDENT.insertMany([

{ student\_id: 1, student\_name: "Rahul Sharma", address: "Delhi", birthdate: ISODate("2000-05-15"), CGPA: 8.5, fee: 50000, current\_year: "BE", join\_date: ISODate("2018-07-01"), skills: ["Programming"], department: "CSE" },

{ student\_id: 2, student\_name: "Pooja Patel", address: "Ahmedabad", birthdate: ISODate("1999-08-20"), CGPA: 7.8, fee: 45000, current\_year: "TE", join\_date: ISODate("2019-06-15"), skills: ["Database"], department: "IT" },

{ student\_id: 3, student\_name: "Suresh Kumar", address: "Baroda", birthdate: ISODate("2001-03-10"), CGPA: 9.2, fee: 55000, current\_year: "SE", join\_date: ISODate("2020-05-10"), skills: ["Networking"], department: "CSE" },

{ student\_id: 4, student\_name: "Manisha Singh", address: "Baroda", birthdate: ISODate("2002-01-05"), CGPA: 7.0, fee: 48000, current\_year: "FE", join\_date: ISODate("2021-02-15"), skills: ["Communication"], department: "ETC" },

{ student\_id: 5, student\_name: "Mohan Verma", address: "Pune", birthdate: ISODate("2000-12-18"), CGPA: 8.8, fee: 52000, current\_year: "BE", join\_date: ISODate("2018-07-01"), skills: ["Programming"], department: "IT" },

{ student\_id: 6, student\_name: "Sonia Verma", address: "Baroda", birthdate: ISODate("1998-06-22"), CGPA: 9.5, fee: 55000, current\_year: "TE", join\_date: ISODate("2019-06-15"), skills: ["Database"], department: "ET" },

{ student\_id: 7, student\_name: "Nitin Sharma", address: "Pune", birthdate: ISODate("1999-07-30"), CGPA: 8.2, fee: 50000, current\_year: "SE", join\_date: ISODate("2020-05-10"), skills: ["Networking"], department: "FE" },

// Add more sample documents as needed

]);

// 3. Display the count of total no students from the institute

db.STUDENT.count();

// 4. Display all students in seniority level (based on CGPA)

db.STUDENT.find().sort({ CGPA: -1 });

// 5. List the student details that are from Baroda or Ahmedabad and in CSE department

db.STUDENT.find({ $and: [{ $or: [{ address: "Baroda" }, { address: "Ahmedabad" }] }, { department: "CSE" }] });

// 6. List of the studentid, studentname, department number, and skill of the student whose birth date is 20th of any month

db.STUDENT.find({ birthdate: { $gte: ISODate("2000-01-20"), $lt: ISODate("2000-12-21") } }, { student\_id: 1, student\_name: 1, department: 1, skills: 1, \_id: 0 });

// 7. Calculate age of each student. Consider today’s date

db.STUDENT.aggregate([

{

$addFields: {

age: {

$floor: {

$divide: [

{ $subtract: [new Date(), "$birthdate"] },

1000 \* 60 \* 60 \* 24 \* 365

]

}

}

}

}

]);

// 8. List the name of student whose name starts with 's' or 'm' character who are in computer department and having typing skill

db.STUDENT.find({

$and: [

{ $or: [{ student\_name: /^S/ }, { student\_name: /^M/ }] },

{ department: "CSE" },

{ skills: "Typing" }

]

});

// 9. Count the number of students in IT department of Pune

db.STUDENT.count({ $and: [{ department: "IT" }, { address: "Pune" }] });

// 10. Arrange the student name in alphabetic order whose age between 18 to 20 and in ETC department

db.STUDENT.find({

$and: [

{ age: { $gte: 18, $lte: 20 } },

{ department: "ETC" }

]

}).sort({ student\_name: 1 });

// 11. Create Index

db.STUDENT.createIndex({ department: 1 });

// 12. Get Index

db.STUDENT.getIndexes();

// 13. Drop Index

db.STUDENT.dropIndex("department\_1");

// 14. Using MapReduce/Aggregation, display the total number of students from each department

db.STUDENT.mapReduce(

function () {

emit(this.department, 1);

},

function (key, values) {

return Array.sum(values);

},

{ out: { inline: 1 } }

);

7. Create Order Management System using MongoDB and Implement Following Statements • Retrieve all the documents fromcollection. • List the customer in ascendingorder of their age. • Display total No of Orders. • Display the Mob No of customers who have purchased product “Shoes”. • Display how many customers are there in customer collection. • Display Total No product purchased in order\_id:2. • Add Another product with quantity 2 in order\_id:3 of customer “ABC”. • Perform Create Index, get Index and drop index operation on collection. • write a MapReduce/aggregate function which will return the Total order per Customer.

Code:

// 2. Insert some sample documents into the ORDER collection

db.ORDER.insertMany([

{ order\_id: 1, customer\_name: "Rahul Sharma", product: "Mobile", quantity: 2, price: 15000, mobile\_number: "9876543210" },

{ order\_id: 2, customer\_name: "Pooja Patel", product: "Shoes", quantity: 1, price: 2000, mobile\_number: "8765432109" },

{ order\_id: 3, customer\_name: "ABC", product: "Laptop", quantity: 1, price: 45000, mobile\_number: "7654321098" },

// Add more sample documents as needed

]);

// 3. Retrieve all the documents from the collection

db.ORDER.find();

// 4. List the customer in ascending order of their age

// Note: Age is not explicitly provided in the sample data, so this query assumes you have an "age" field in your document.

db.ORDER.find().sort({ age: 1 });

// 5. Display total No of Orders

db.ORDER.find().count();

// 6. Display the Mob No of customers who have purchased the product "Shoes"

db.ORDER.find({ product: "Shoes" }, { mobile\_number: 1, \_id: 0 });

// 7. Display how many customers are there in the customer collection

db.ORDER.distinct("customer\_name").length;

// 8. Display Total No product purchased in order\_id:2

db.ORDER.findOne({ order\_id: 2 }).quantity;

// 9. Add another product with quantity 2 in order\_id:3 of customer "ABC"

db.ORDER.update(

{ order\_id: 3, customer\_name: "ABC" },

{ $push: { products: { product: "PenDrive", quantity: 2 } } }

);

// 10. Create Index

db.ORDER.createIndex({ customer\_name: 1 });

// 11. Get Index

db.ORDER.getIndexes();

// 12. Drop Index

db.ORDER.dropIndex("customer\_name\_1");

// 13. Write a MapReduce/aggregate function which will return the Total order per Customer

db.ORDER.mapReduce(

function () {

emit(this.customer\_name, 1);

},

function (key, values) {

return Array.sum(values);

},

{ out: { inline: 1 } }

);