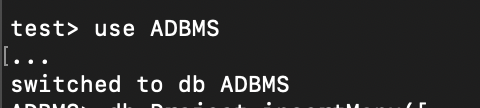
**Lab 8**

1. Create a database named ADBMS



1. Create two collections named Employee and Project

* It automatically creates the Employee and project .

1. Insert three documents in project

{ "\_id" : "p1", "pname" : "Care Nepal", "Location" : "Banepa" }

{ "\_id" : "p2", "pname" : "Agro Nepal", "Location" : "Butwal" }

{ "\_id" : "p3", "pname" : "Tech Nepal", "Location" : "Kathmandu", "ESTD" : 2001 }

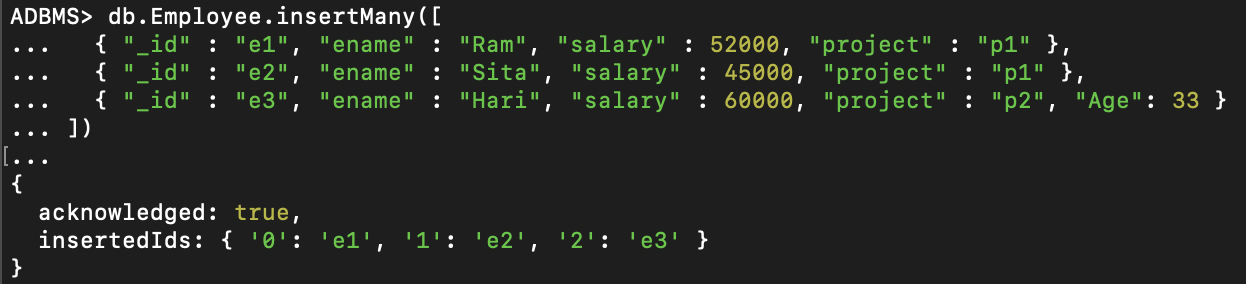


1. Insert three documents in Employee

{ "\_id" : "e1", "ename" : "Ram", "salary" : 52000, "project" : "p1" }

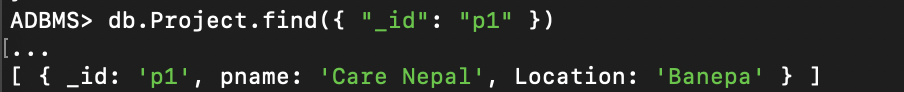
{ "\_id" : "e2", "ename" : "Sita", "salary" : 45000, "project" : "p1" }

{ "\_id" : "e3", "ename" : "Hari", "salary" : 60000, "project" : "p2", Age" : 33}



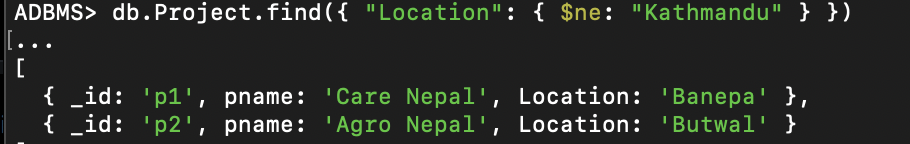
1. Find record of project p1

db.Project.find({"\_id":"p1"})



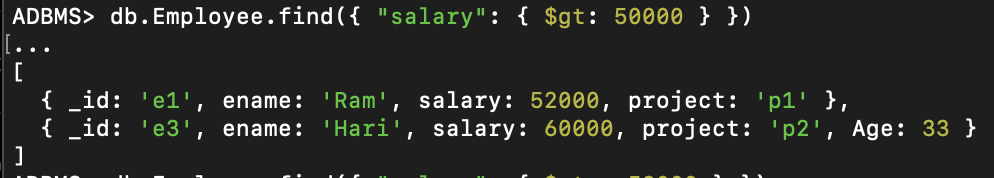
1. Find all projects outside of Kathmandu

db.Project.find({"Location":{$ne:"Kathmandu"}})



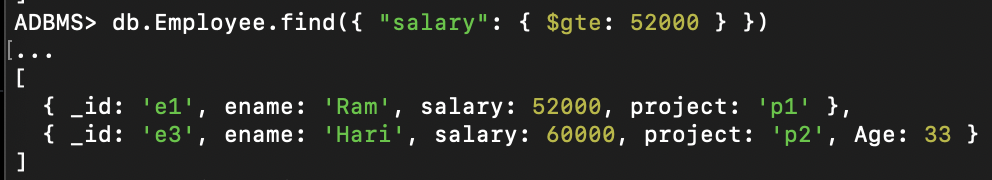
1. Find records of all employees having salary more than 50000.

db.Employee.find({"salary":{$gt:50000}})



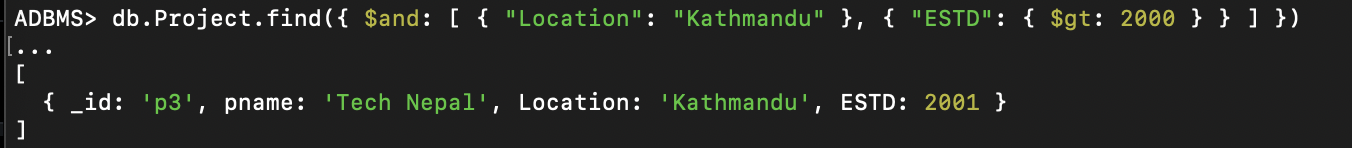
1. Find records of employees having salary at least 52000

db.Employee.find({"salary":{$gte:52000}})



1. Find projects located in Kathmandu and established after 2000.

db.Project.find({$and:[{"Location":"Kathmandu"},{"ESTD":{$gt:2000}}]})



1. Find records of all employees along with their project details.

db.Employee.aggregate([{$lookup:{from:"Project",localField:"project",foreignField:"\_id",as:"Project"}}])



**RESULT AND CONCLUSION:**

Therefore, we learned how to create and manage MongoDB collections and databases. We practiced inserting documents into collections and performing queries to retrieve specific data, such as filtering by location or salary. Additionally, we explored MongoDB's aggregation capabilities, particularly the $lookup operator for joining data from different collections, which helped us combine employee details with project information. This lab enhanced our understanding of basic MongoDB operations and how to efficiently retrieve and organize data using aggregation.

**Lab 9**

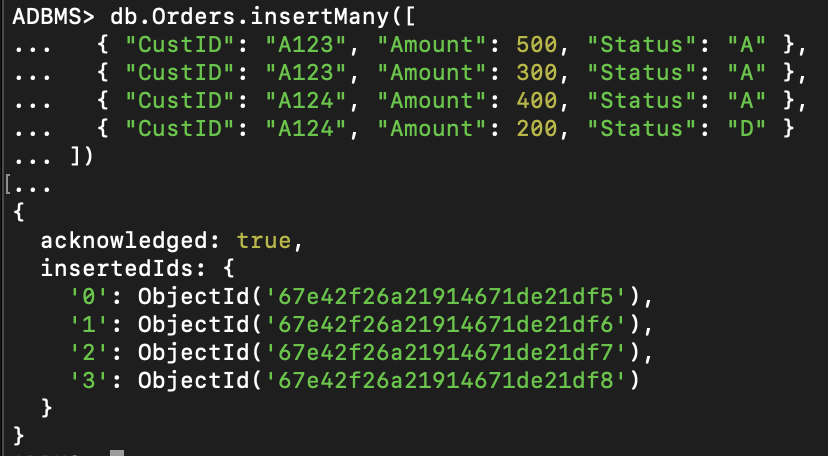
1. Create a collection named Orders and insert following documents in it.

{"CustID" : "A123", "Amount" : 500, "Status" : "A" }

{"CustID" : "A123", "Amount" : 300, "Status" : "A" }

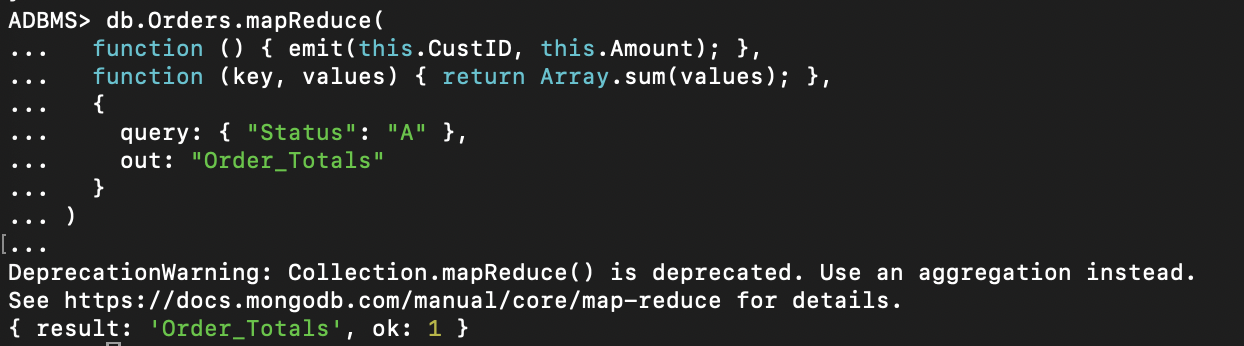
{"CustID" : "A124", "Amount" : 400, "Status" : "A" }

{ "CustID" : "A124", "Amount" : 200, "Status" : "D" }



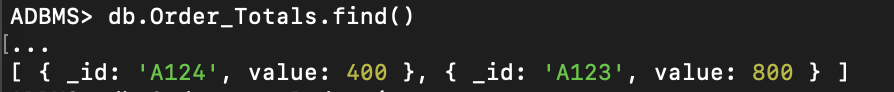
1. Write down MapReduce functions to find total amount of each active customer.

db.Orders.mapReduce(function(){emit(this.CustID,this.Amount);},function(key,values){return Array.sum(values)},{query:{"Status":"A"}, out:"Order\_Totals"})



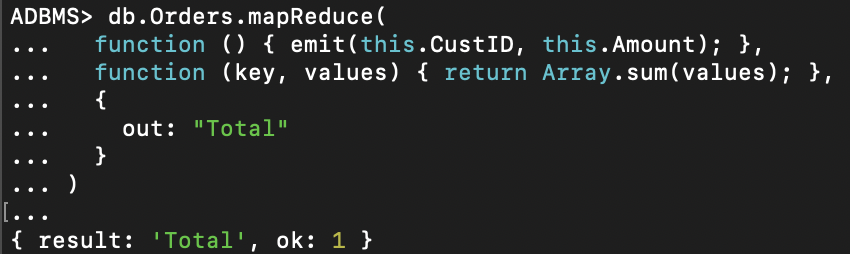
1. Find documents in Order\_Totals

db.Order\_Totals.find()



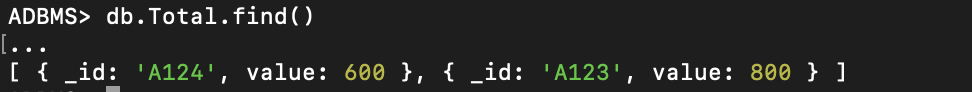
1. Write down MapReduce functions to find total amount of each customer.

db.Orders.mapReduce(function(){emit(this.CustID,this.Amount);},function(key,values){return Array.sum(values)},{out:"Total"})



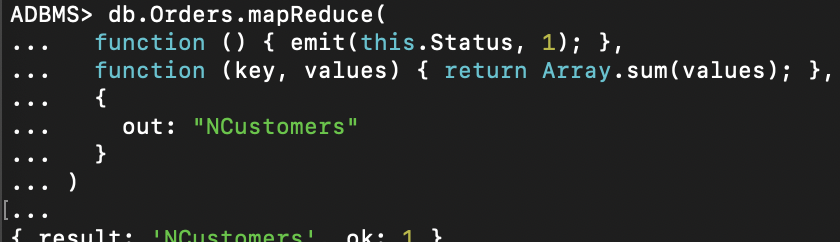
1. Find documents in Total

Db.Total.find()



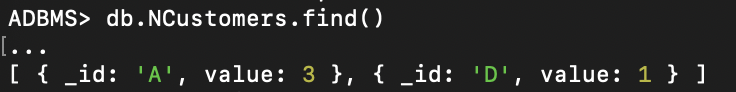
1. Write down MapReduce functions to find total number of active and inactive customer orders.

db.Orders.mapReduce(function(){emit(this.Status,1);},function(key,values){return Array.sum(values)},{out:"NCustomers"})



1. Find documents in NCustomers

db.NCustomers.find()



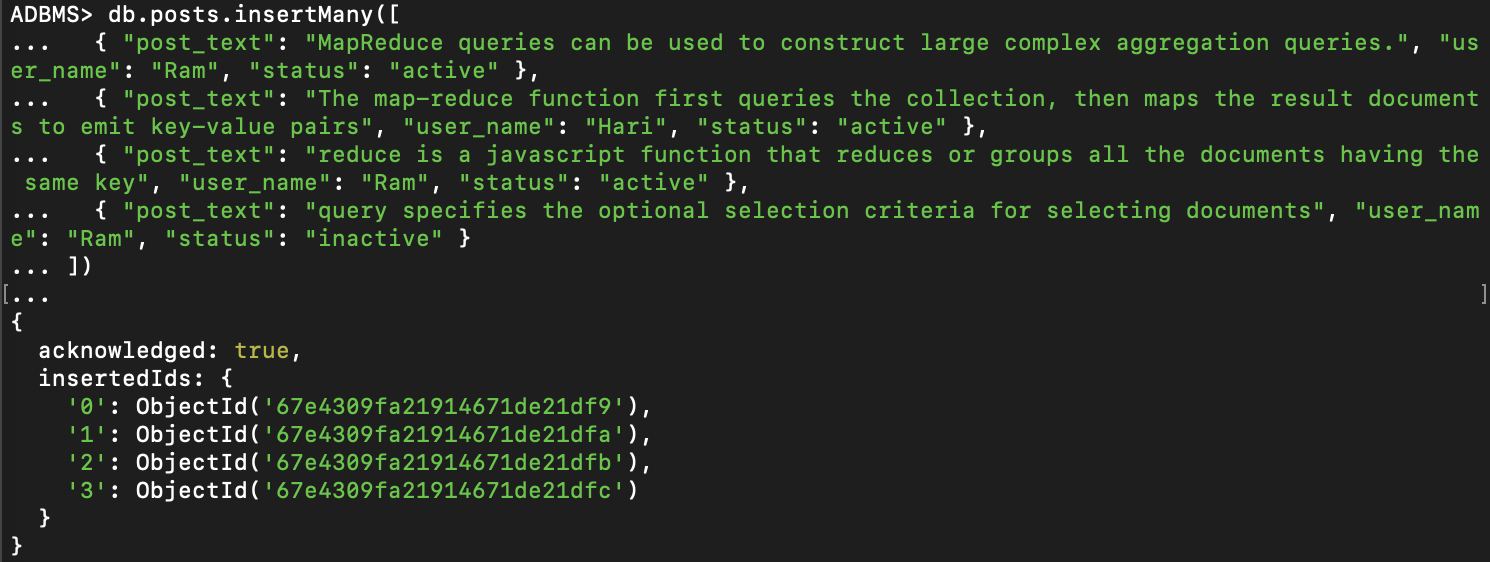
1. Insert following documents in collection named posts.

{ “post\_text”: “MapReduce queries can be used to construct large complex aggregation queries.”, “user\_name”: “Ram”, “status”:”active”}

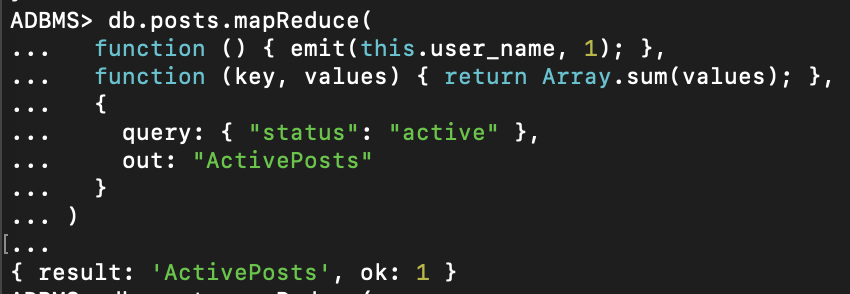
{ “post\_text”: “The map-reduce function first queries the collection, then maps the result documents to emit key-value pairs”, “user\_name”: “Hari”, “status”:”active”}

{ “post\_text”: “reduce is a javascript function that reduces or groups all the documents having the same key”, “user\_name”: “Ram”, “status”:”active”}

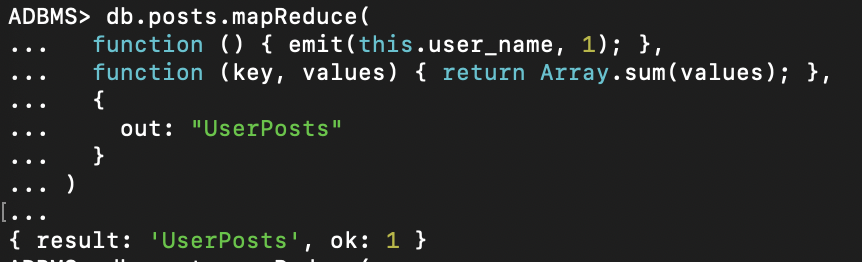
{ “post\_text”: “query specifies the optional selection criteria for selecting documents”, “user\_name”: “Ram”, “status”:”inactive”}



1. Write down MapReduce function to find total number of posts by currently active users.



1. Write down MapReduce function to find total number of posts by each user.



1. Write down MapReduce function to find total number of active posts by each user.



**RESULT AND CONCLUSION:**

Thus, By implementing MapReduce functions, we were able to calculate totals, counts, and other summary statistics based on specific criteria, such as customer activity and post status. This lab deepened our understanding of how MapReduce can be applied in real-world scenarios for large-scale data processing and aggregation, making it a valuable tool for data analysis in MongoDB.