**PRACTICAL-4**

**AIM: Indexing ,Aggregation and Map Reduce in NoSQL-DB.**

1. **PRACTICE QUESTIONS:**

**1. Indexing :- Query :- for( var iCounter=1;iCounter<= 1000000;iCounter++)**

**{ db.Asset.insert(**

**{**

**"Name":"Voting"+iCounter,**

**"Desc":"Story about a college student"+iCounter,**

**"Rank":iCounter,**

**"Language":["English","Hindi","Tamil"],**

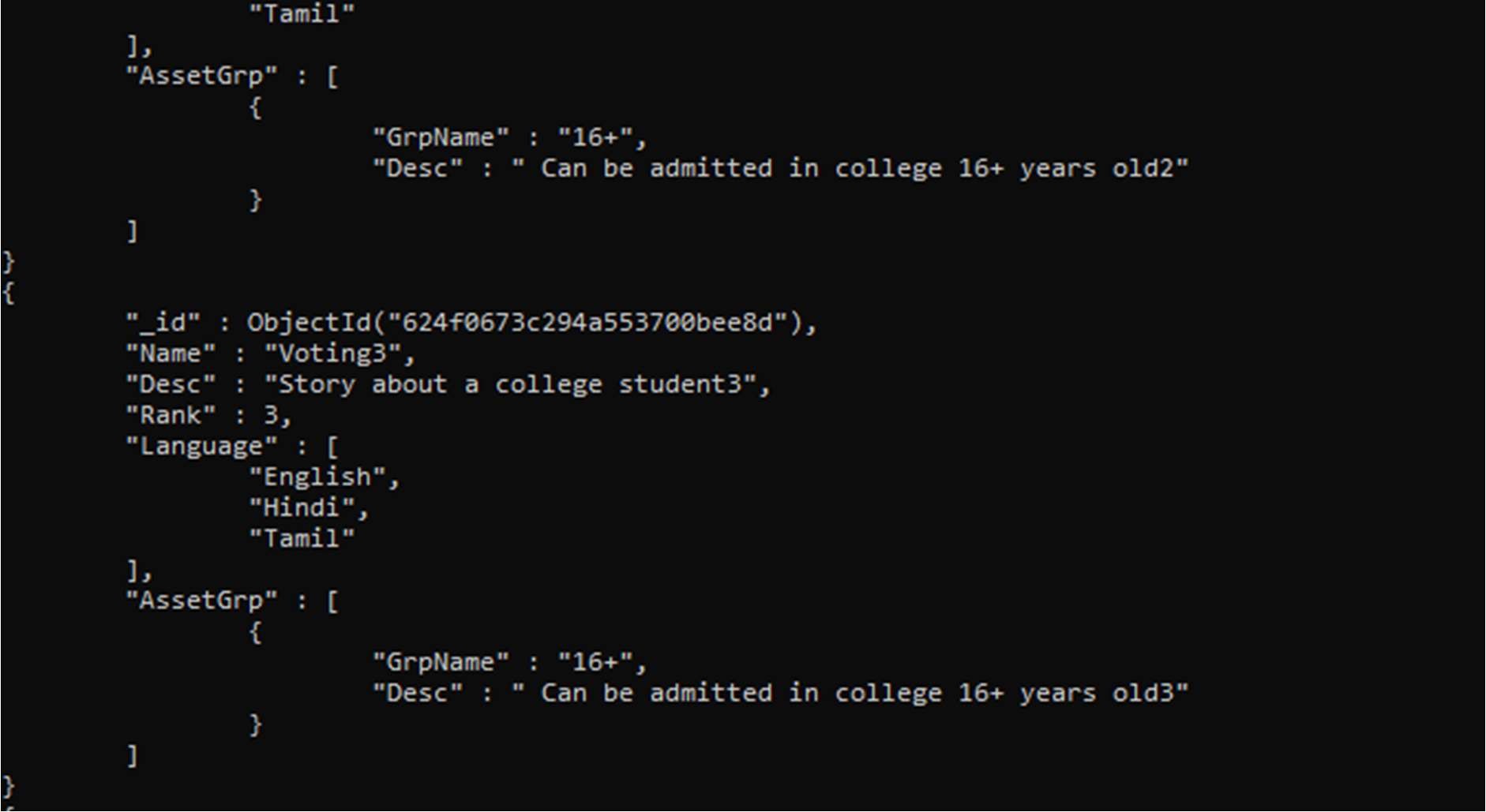
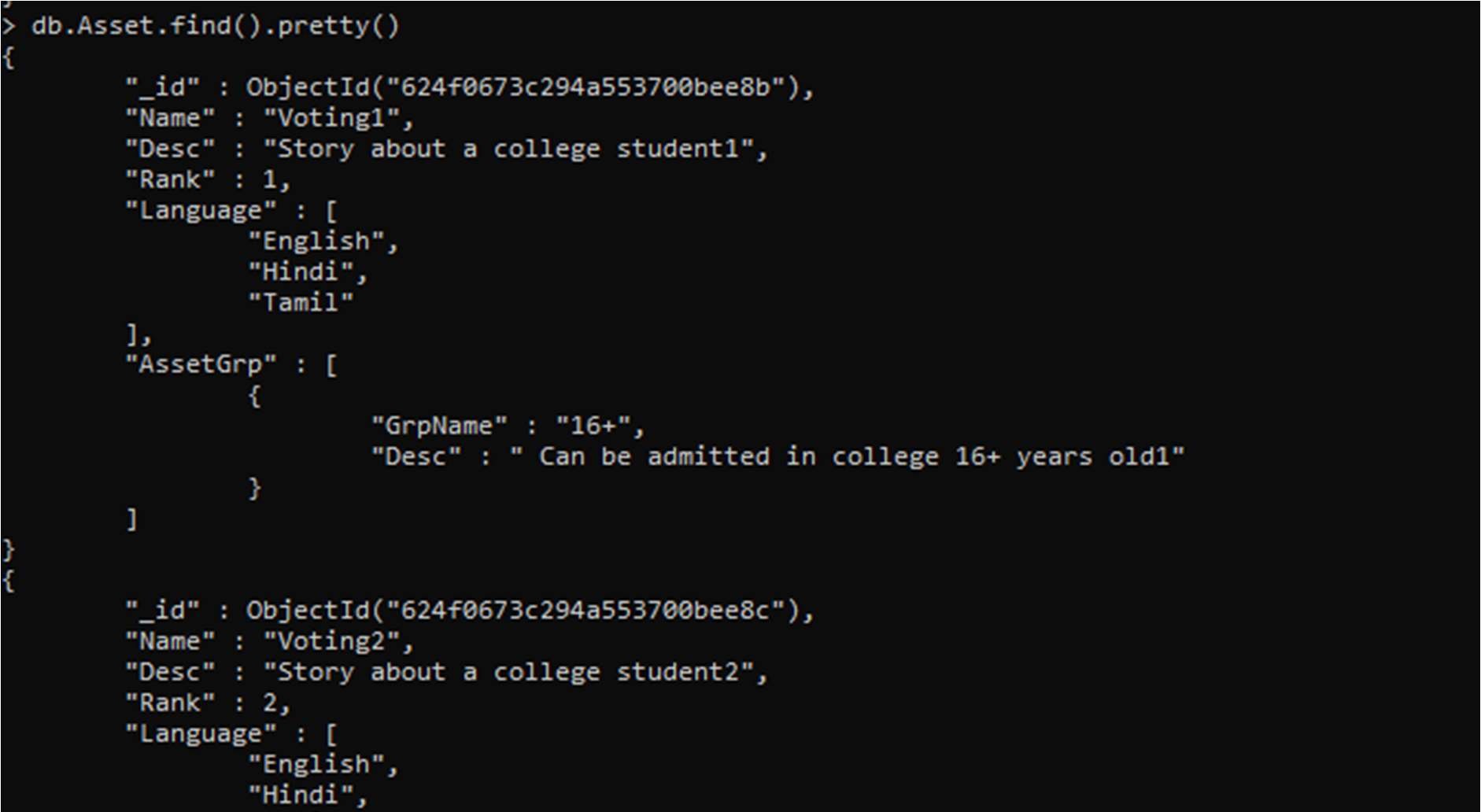
**"AssetGrp":[ { "GrpName":"16+", "Desc":" Can be admitted in college**

**16+ years old"+iCounter } ]**

**})**

**}**

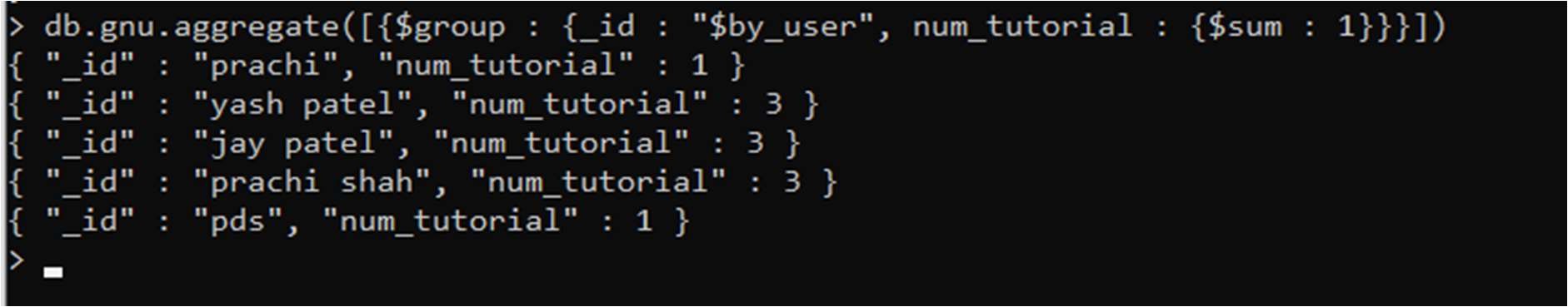
**Output :-**



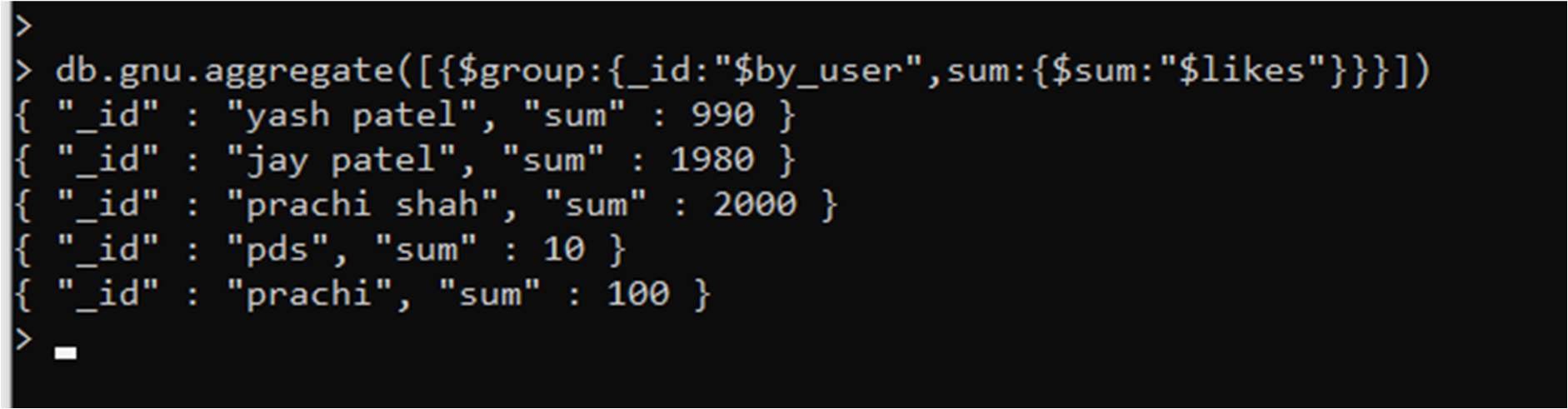
2. Aggregation :-

 Ex-1 : Create collection name as “gnu”

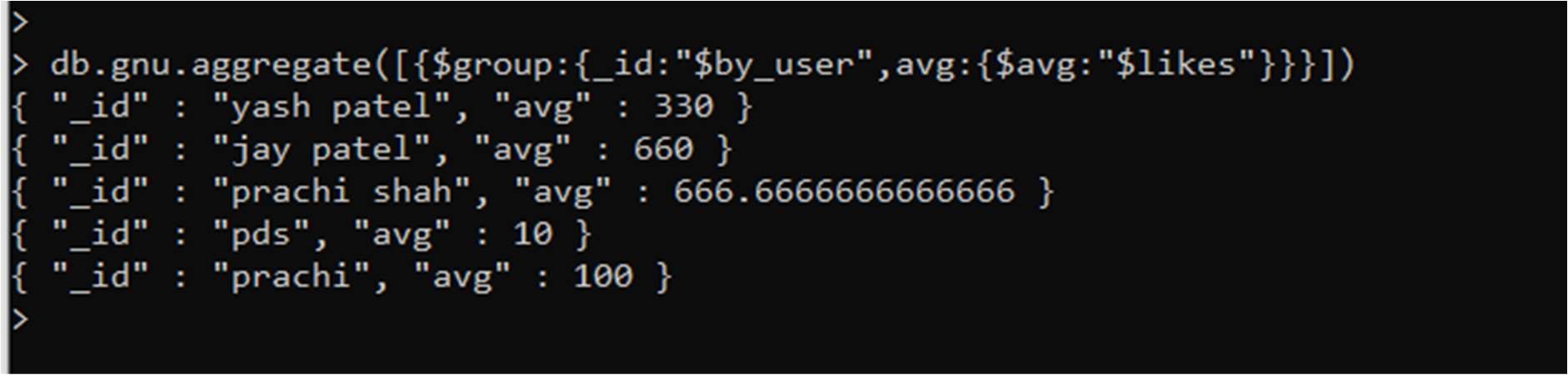
Add 10 relevant documents in same collection. Query :- db.gnu.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}]) Output :-



Query:- db.gnu.aggregate([{$group:{\_id:"$by\_user",sum:{$sum:"$likes"}}}]) Output :-

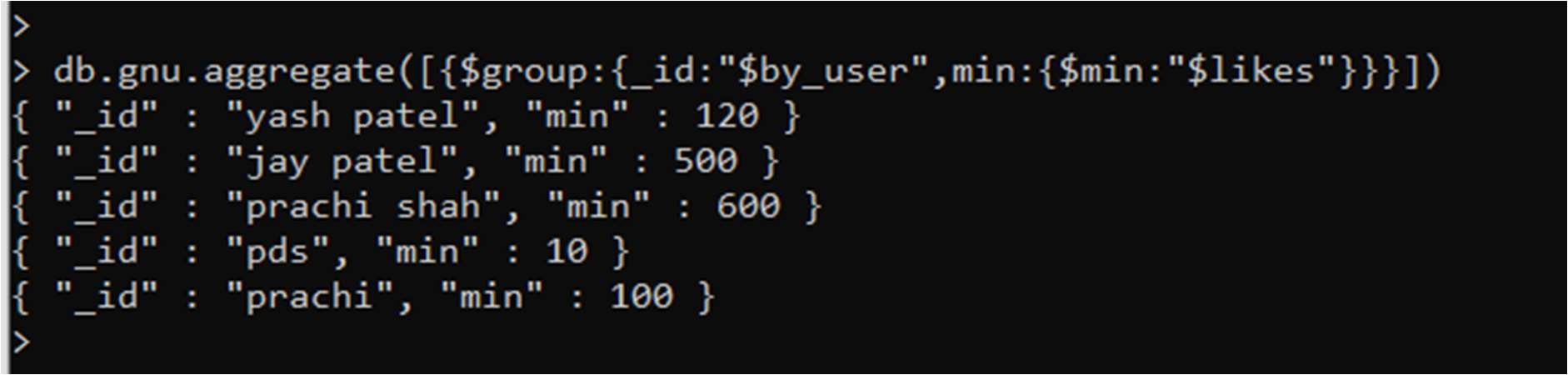


Query :- db.gnu.aggregate([{$group:{\_id:"$by\_user",avg:{$avg:"$likes"}}}]) Output :-

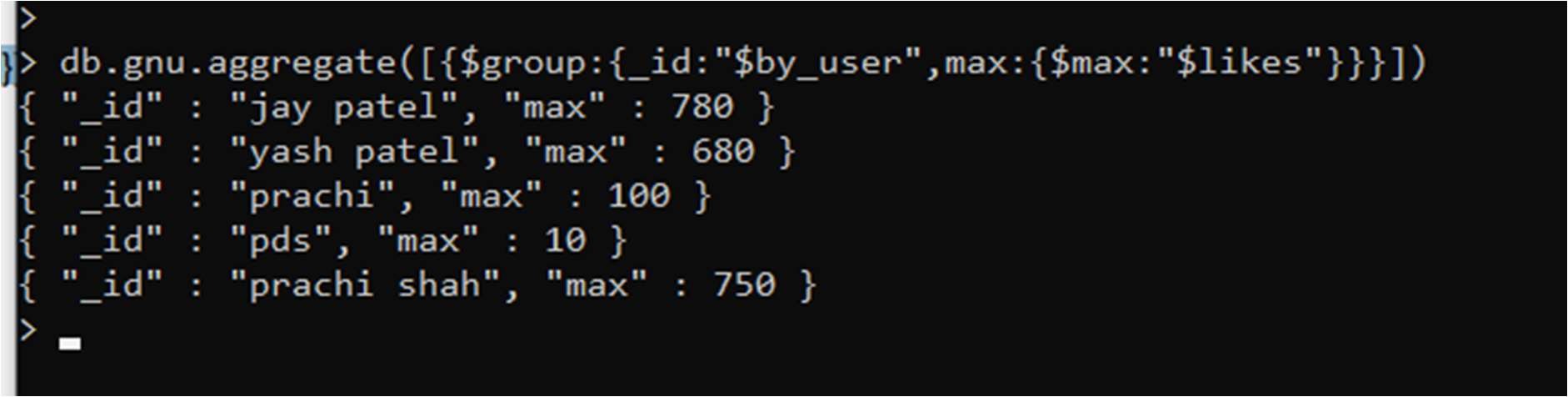


Query :- db.gnu.aggregate([{$group:{\_id:"$by\_user",min:{$min:"$likes"}}}])

Output :-



Query :- db.gnu.aggregate([{$group:{\_id:"$by\_user",max:{$max:"$likes"}}}]) Output :-



Query :- db.gnu. aggregate([{$group:{\_id:"$by\_user",first\_url:{$first:"$url"}}}]) Output :-



Query :- db.gnu.aggregate([{$group:{\_id:"$by\_user",last\_url:{$last:"$url"}}}])

Output :-

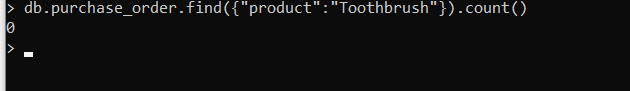


 EX-2 :

Create a collection called purchase\_orders having fildes product (toothbrush , guitar , milk , pizza ) , price , customer\_name insert 10 records into collections.

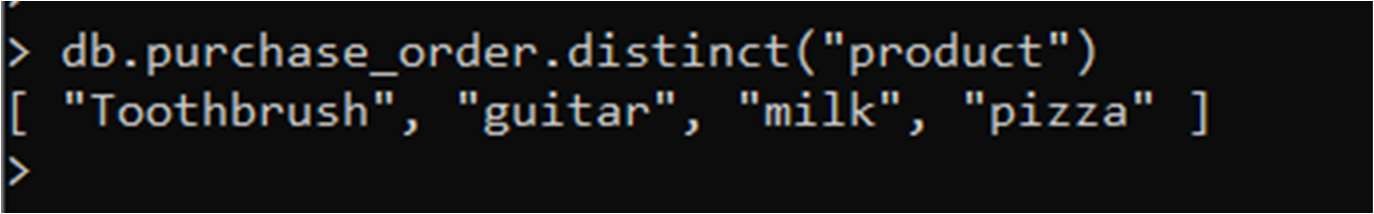
 Query: -

1. find out how many toothbrushes were sold. Query: - db.purchase\_order.find({"product":"Toothbrush"}).count() Output :-



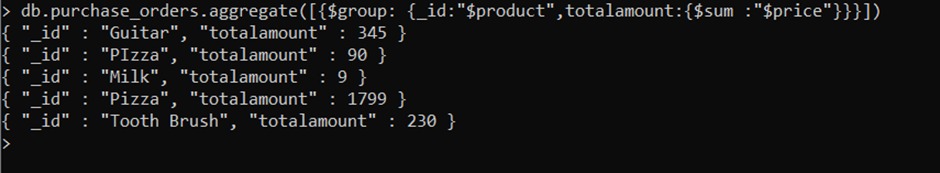
1. find the list of all products sold. Query: - db.purchase\_order.distinct("product")

Output :-

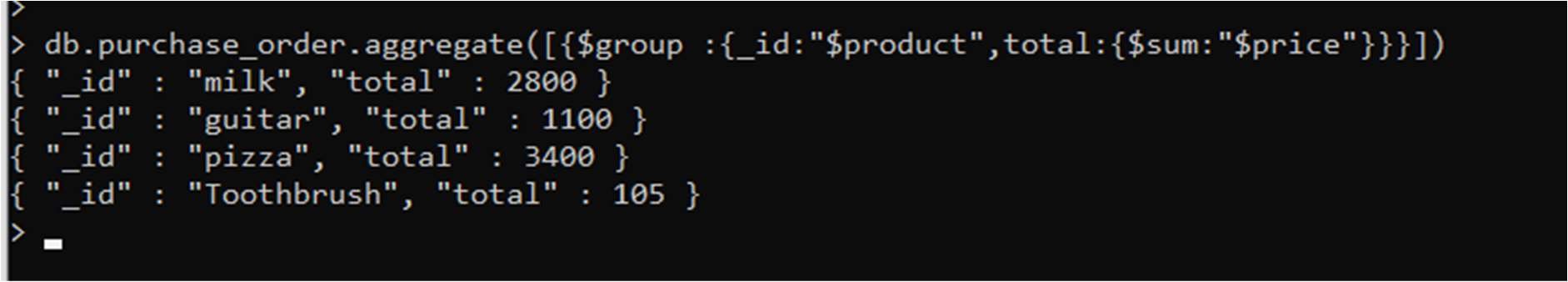


1. find the total amount of money spent by each customer. Query: - db.purchase\_order.aggregate([{$group :{\_id:"$customername",total:{$sum:"$price"}}}])

Output :-



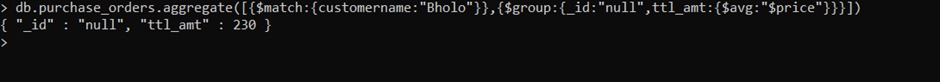
1. find the total amount of money spent on each product. Query: - db.purchase\_order.aggregate([{$group :{\_id:"$product",total:{$sum:"$price"}}}]) Output :-



1. find how much money each customer has spent on toothbrushes or pizza. Query: -

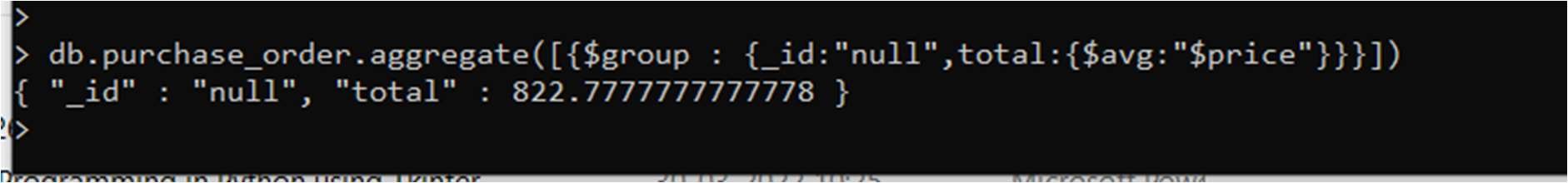
db.purchase\_order.aggregate([{$match:{$or:[{"product":"pizza"},{"product":"Toothbrush"}] }},{$group:{\_id:"$customername",spent:{$sum:"$price"}}}])

Output :-



1. calc. the avg purchase price of ABC. Query: -

db.purchase\_order.aggregate([{$group : {\_id:"null",total:{$avg:"$price"}}}]) Output :-

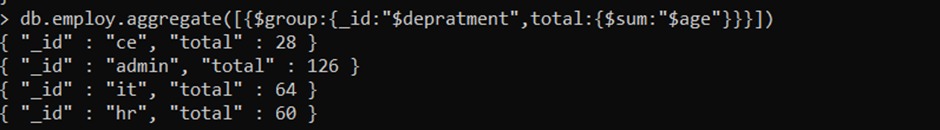


 EX-3:

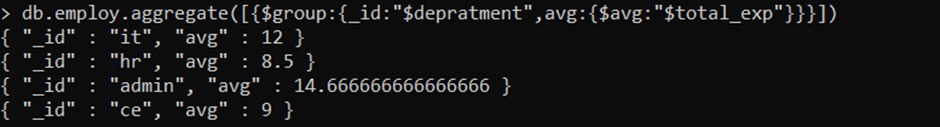
create a collection called employee having fields name, department(Admin,ce,it,hr) , age , total\_exp, languages(diff languages) insert 8 records into collections

 Queries:-

1. find the total age of employees for each department.

Query:- db.emp.aggregate([{$group : {\_id:"$department",age\_sum:{$sum:"$age"}}}]) Output :- 

1. calc. the avg experience of each department. Query:- db.emp.aggregate([{$group : {\_id:"$department",avgexp:{$avg:"$exp"}}}]) Output :-



1. find the youngest and oldest employee. Query:-

db.emp.aggregate([{$group:{\_id:null,youngest\_emp: {$min:"$age"},oldest\_emp:{$max:"$age"}}}])

Output :-



1. find the minimum and maximum experienced employee from department admin.

Query:- db.emp.aggregate([{$match:{"department":"admin"}},{$group:{\_id:null,min\_total\_exp:{$mi n:"$total\_exp"},max\_total\_exp:{$max:"$total\_exp"}}}]) Output :-



3. Map-Reduce :-

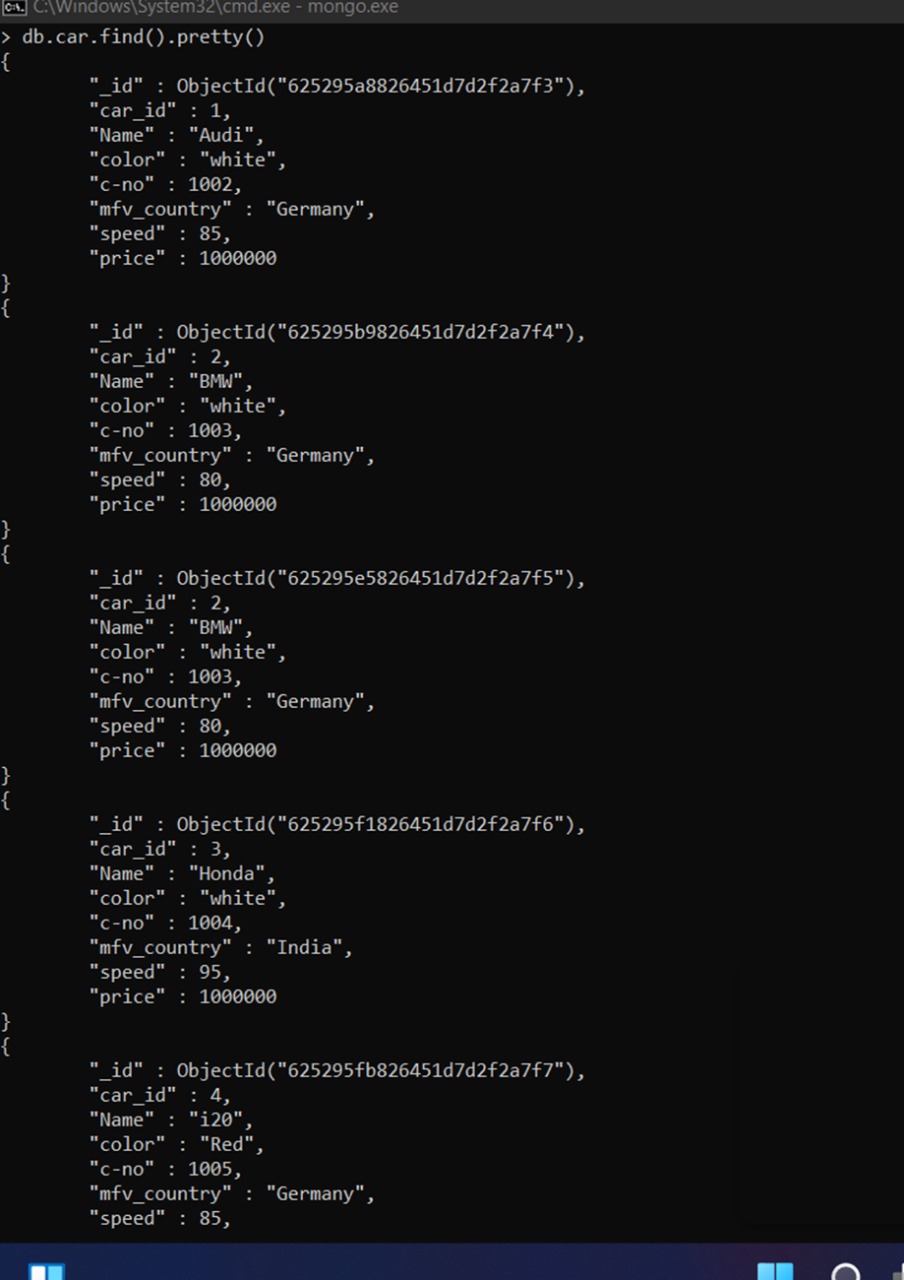
 Ex-1:

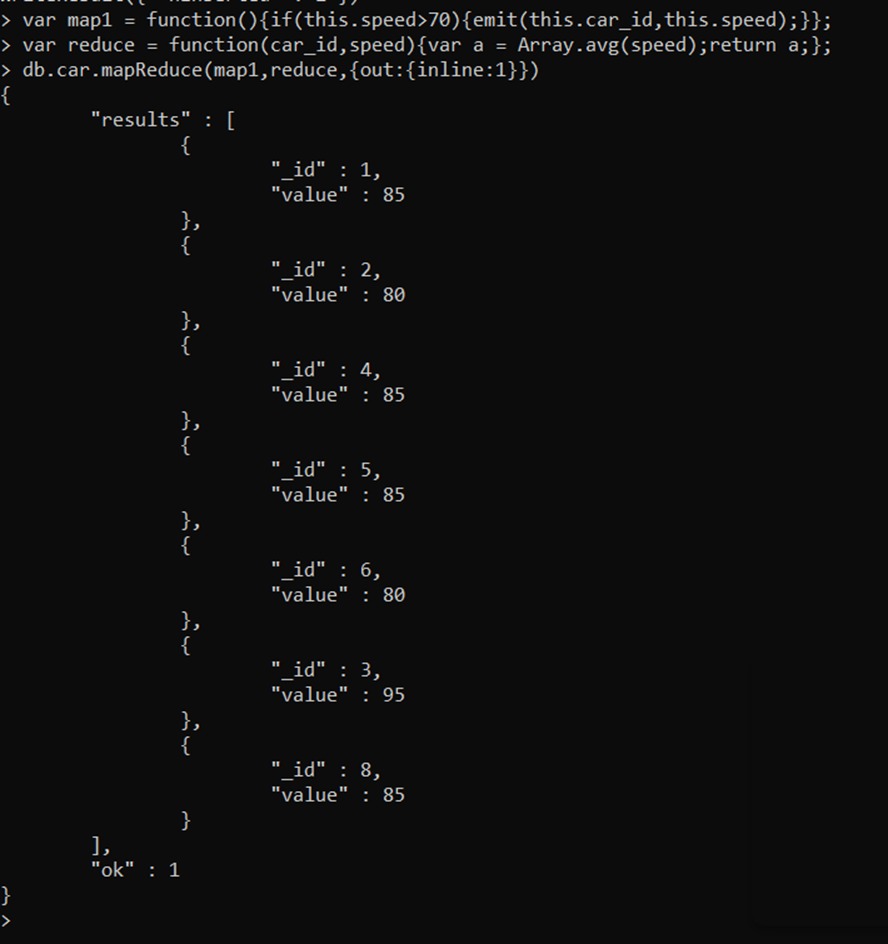
create a collection called car having fields car\_id, name, color, car\_number, mfd\_country, speed and price insert 8 records

create a map function that will get data of cars having speeds greater than 70 create a reduce function that will find the average speed code :- var map1=function(){ if(this.speed>70){ emit(this.car\_id,this.speed);

} }; var reduce=function(car\_id,speed){ var a=Array.avg(speed); return a;

};

Query :- db.car.mapReduce(map,reduce1,{out:{inline:1}}) Output :- 



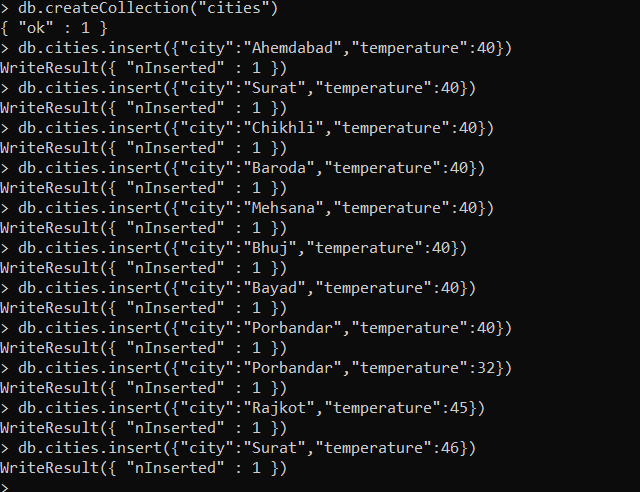
 Ex-2:

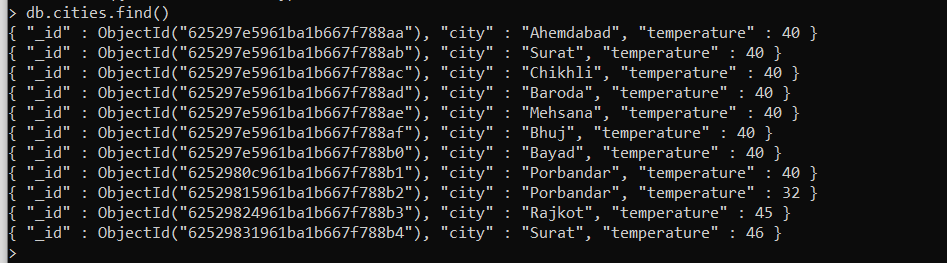
create a collection called city having two fields city(Ahemdabad, Mehsana,

Baroda) and Temperature.

insert 8 records

Output :-





create a map and reduce function to find maximum temperature for each city.

var map=function(){ emit(this.city,this.tempereture)

}; var maxt=function(city, tempereture){ var max= tempereture [0]; for(var i=0;i< tempereture.length;i++){ if(tempereture [i]>max){ max= tempereture [i];

} return max;

} }; var mint=function(city, tempereture){

var min= tempereture [0]; for(var i=0;i< tempereture.length;i++){ if(tempereture [i]<min){ min= tempereture [i];

} return min;

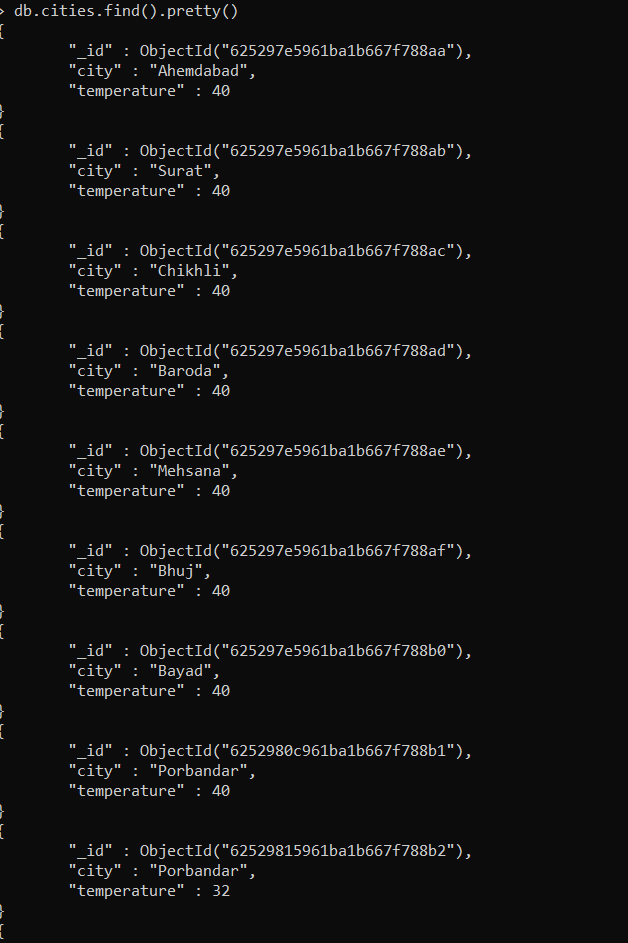
}

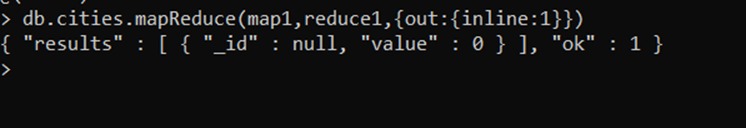
};

Query:

db.city.mapReduce(map,mint,{out:{inline:1}})

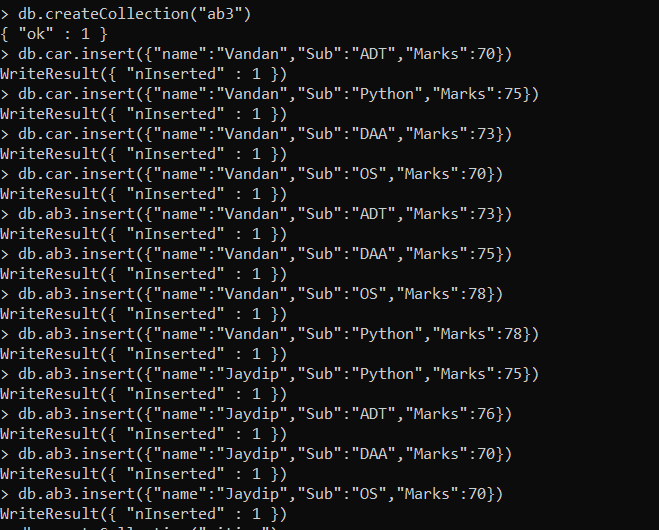
Output :-



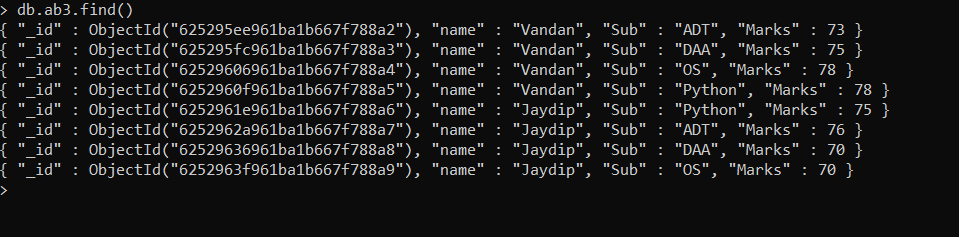


 Ex-3 : create a collection called AB3 having fields student\_names,subject and marks.

insert 8 records



Output :



create map and reduce function to get total marks for each student and output should be written in collection name total.

db.ab3.find() db.createCollection("Hello")

var map=function(){ emit(this.name,this.marks)

};

var red=function(name,marks){

var sum=0; for(var i=0;i<marks.length;i++){ sum=sum+marks[i];

} return sum;

}

Query:

db.ab3.mapReduce(map,red,{out:"Hello"}) db.Hello.find()

Output :-

