**Shahuraj Bhoite 258077**

**DAY 9**

**Solve the following**

**Loops**

**example 1. Print the following patterns using loop :**

**a.**

**\***

**\*\***

**\*\*\***

**\*\*\*\***

**delimiter //**

**create procedure displayP1(in num int)**

**begin**

**declare str varchar(250) default '\*';**

**declare str1 varchar(250) default '\n';**

**declare cnt int default 1;**

**while cnt <= num do**

**set str1 = concat(str1, str, '\n');**

**set str = concat(str, '\*');**

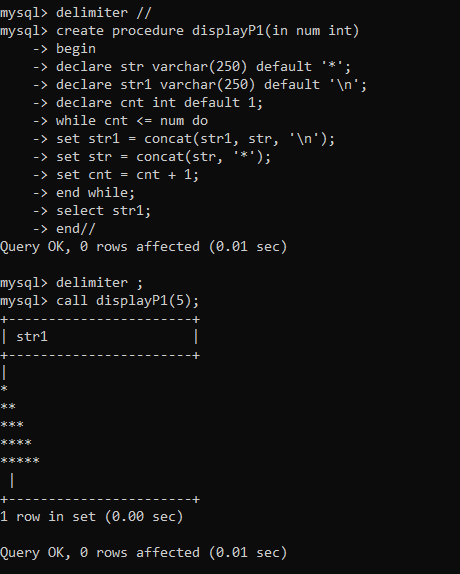
**set cnt = cnt + 1;**

**end while;**

**select str1;**

**end//**

**delimiter ;**

****

**b.**

**\***

**\*\*\***

**\*\*\*\*\***

**\*\*\***

**\***

**drop procedure displayP3;**

**delimiter //**

**create procedure displayP3(in num int)**

**begin**

**declare str varchar(16383) default '\*';**

**declare str1 varchar(250) default '\n';**

**declare cnt int default 1;**

**set cnt= ceil(num/2);**

**while cnt <= num do**

**set str1 = concat(str1, lpad(str,cnt,' '), '\n');**

**set cnt = cnt + 1;**

**set str = concat(str, '\*\*');**

**end while;**

**set cnt = num - 1;**

**set str = substr(str, 1, num-2);**

**while cnt >= ceil(num/2) do**

**set str1 = concat(str1, lpad(str,cnt,' '), '\n');**

**set str = substr(str, 1, length(str) - 2);**

**set cnt = cnt - 1;**

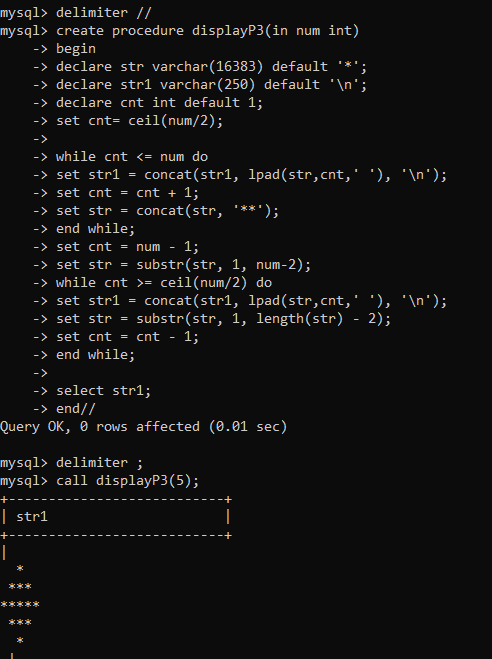
**end while;**

**select str1;**

**end//**

**delimiter ;**

**call displayP3(5);**

****

**c.**

**1010101**

**10101**

**101**

**1**

**d.**

**1**

**1 2**

**1 2 3**

**1 2 3 4**

**1 2 3 4 5**

**delimiter //**

**create procedure displayP2(in num int)**

**begin**

**declare str varchar(250) default '1';**

**declare str1 varchar(250) default '\n';**

**declare cnt int default 1;**

**while cnt <= num do**

**set str1 = concat(str1, str, '\n');**

**set cnt = cnt + 1;**

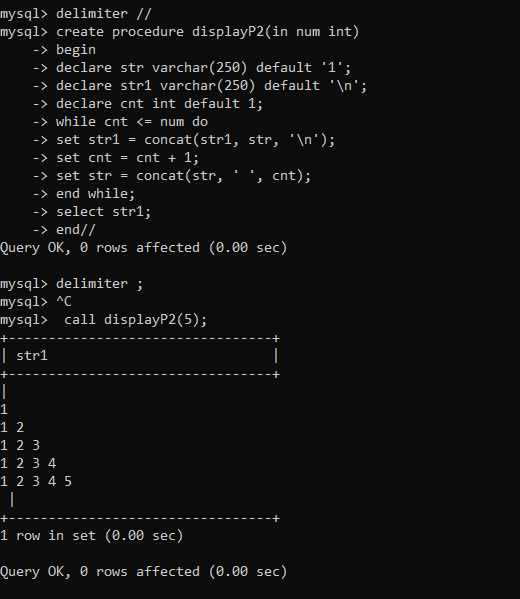
**set str = concat(str, ' ', cnt);**

**end while;**

**select str1;**

**end//**

**delimiter ;**

****

**2. write a procedure to insert record into employee table.**

**the procedure should accept empno, ename, sal, job, hiredate as input parameter**

**write insert statement inside procedure insert\_rec to add one record into table**

**delimiter //**

**create procedure insert\_rec(**

**peno int,**

**pname varchar(20),**

**psal double(9,2),**

**pjob varchar(20),**

**phiredate date**

**)**

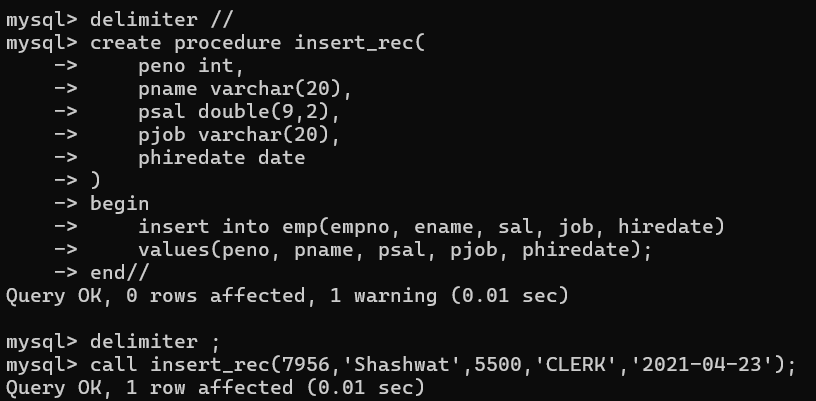
**begin**

**insert into emp(empno, ename, sal, job, hiredate)**

**values(peno, pname, psal, pjob, phiredate);**

**end//**

**delimiter ;**

****

**3. write a procedure to delete record from employee table.the procedure should accept empno as input parameter.write delete statement inside procedure delete\_emp to delete one record from emptable**

**delimiter //**

**create procedure delete\_emp(peno int)**

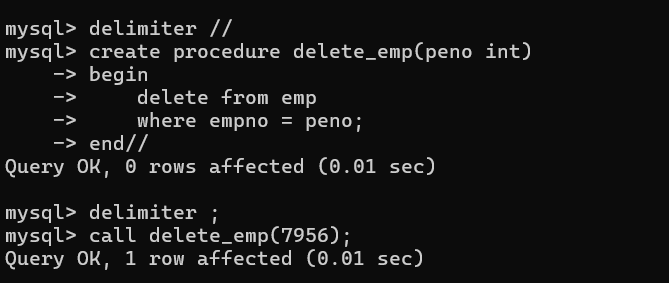
**begin**

**delete from emp**

**where empno = peno;**

**end//**

**delimiter ;**

****

**4. write a procedure to display empno,ename,deptno,dname for all employees with sal**

**> given salary. pass salary as a parameter to procedure**

**delimiter //**

**create procedure display\_emp\_by\_sal(psal double(9,2))**

**begin**

**select e.empno, e.ename, e.deptno, d.dname**

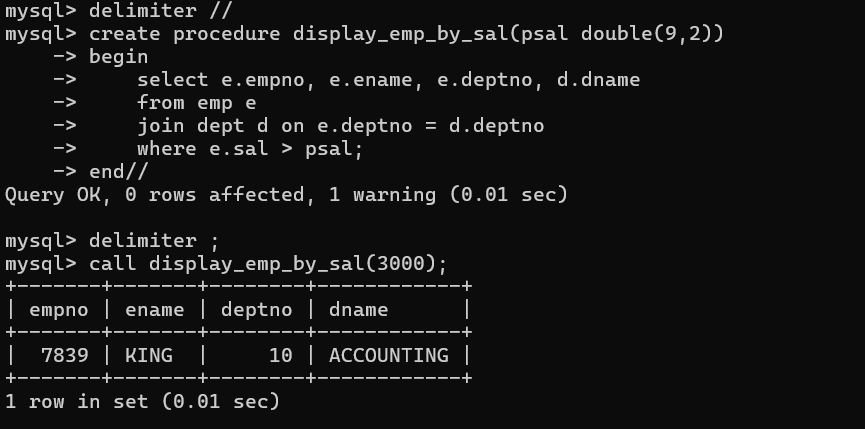
**from emp e**

**join dept d on e.deptno = d.deptno**

**where e.sal > psal;**

**end//**

**delimiter;**

****

**5. write a procedure to find min,max,avg of salary and number of employees in the**

**given deptno.**

**deptno --→ in parameter**

**min,max,avg and count ---→ out type parameter**

**execute procedure and then display values min,max,avg and count**

**delimiter //**

**create procedure display\_mix\_max\_avg(pdptno int)**

**begin**

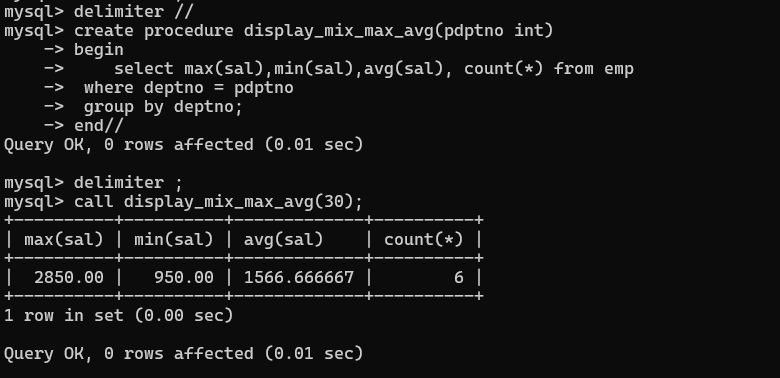
**select max(sal),min(sal),avg(sal), count(\*) from emp**

**where deptno = pdptno**

**group by deptno;**

**end//**

**delimiter ;**

****

**6. write a procedure to display all pid,pname,cid,cname and salesman name(use**

**product,category and salesman table)**

**delimiter //**

**create procedure display\_vehicle\_product\_details()**

**begin**

**select**

**v.Vid ,**

**v.Vname ,**

**c.Custid,**

**c.Cname,**

**s.Sname as salesman\_name**

**from cust\_vehicle cv**

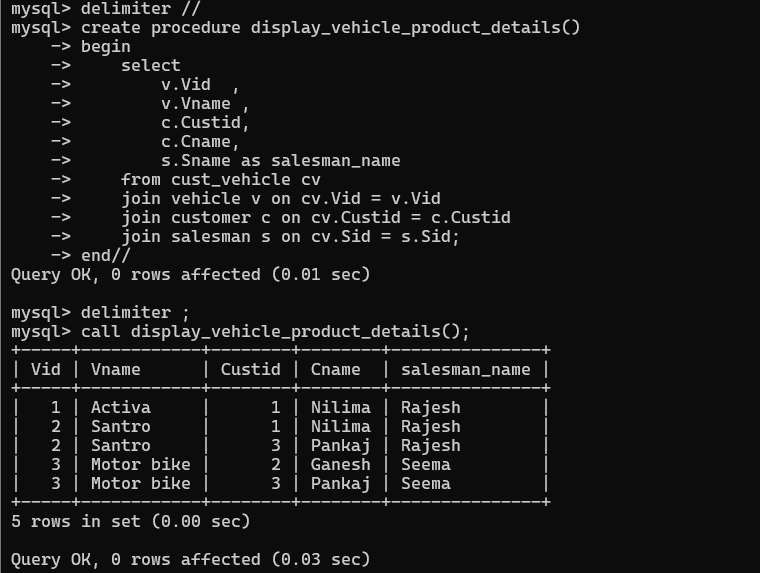
**join vehicle v on cv.Vid = v.Vid**

**join customer c on cv.Custid = c.Custid**

**join salesman s on cv.Sid = s.Sid;**

**end//**

**delimiter ;**

****

**7. write a procedure to display all vehicles bought by a customer. pass customer name**

**as a parameter.(use vehicle,salesman,custome and relation table)**

**delimiter //**

**create procedure display\_vehicles\_by\_customer(p\_custname varchar(50))**

**begin**

**select**

**c.Cname,**

**v.Vid,**

**v.Vname,**

**v.Price,**

**s.Sname as Salesman,**

**cv.Buy\_price**

**from cust\_vehicle cv**

**join customer c on cv.Custid = c.Custid**

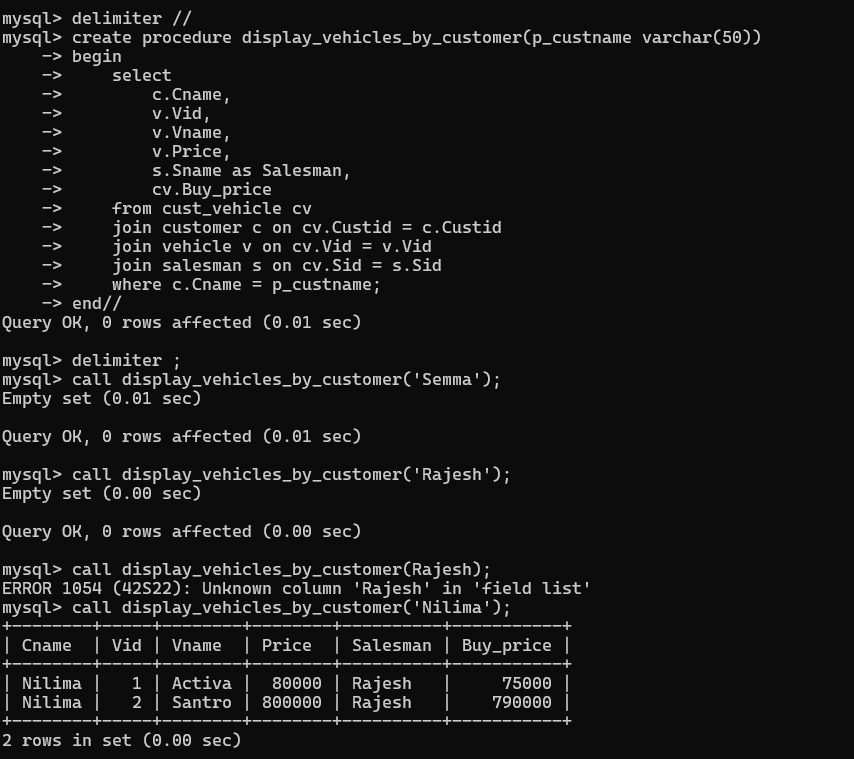
**join vehicle v on cv.Vid = v.Vid**

**join salesman s on cv.Sid = s.Sid**

**where c.Cname = p\_custname;**

**end//**

**delimiter ;**

****

**8. Write a procedure that displays the following information of all emp**

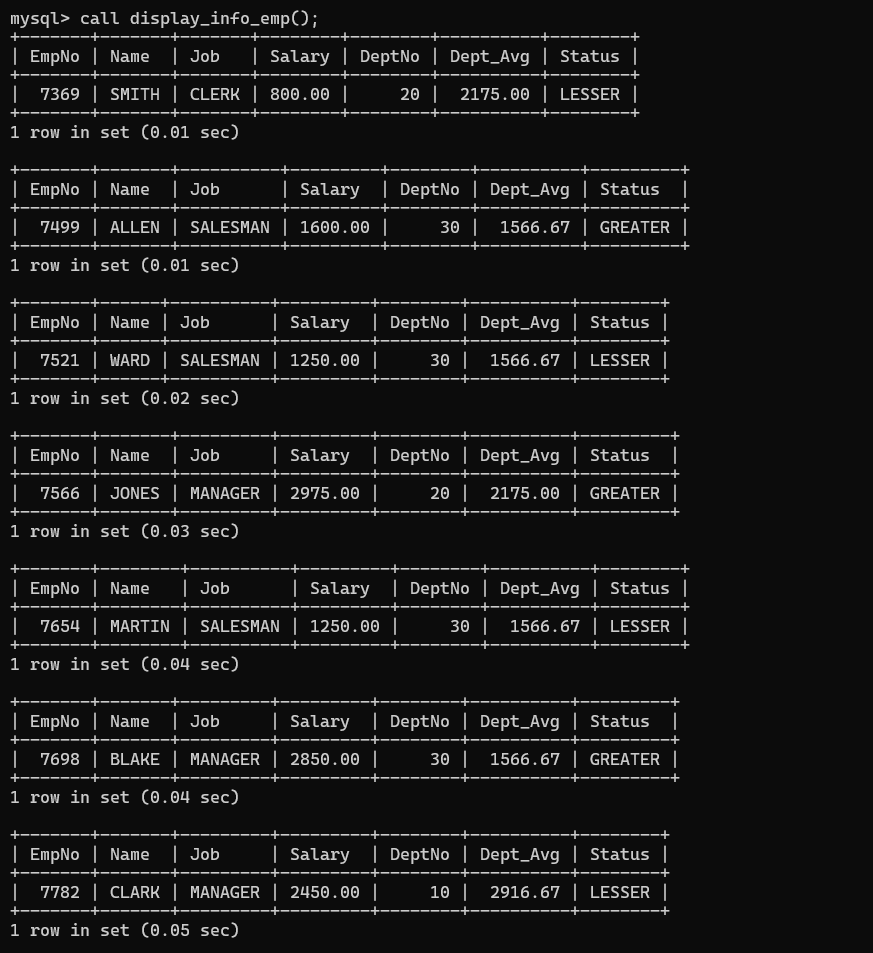
**Empno,Name,job,Salary,Status,deptno**

**Note: - Status will be (Greater, Lesser or Equal) respective to average salary of their own**

**department. Display an error message Emp table is empty if there is no matching**

**record.**

****

****

**9. Write a procedure to update salary in emp table based on following rules.**

**Exp< =35 then no Update**

**Exp> 35 and <=38 then 20% of salary**

**Exp> 38 then 25% of salary**

**DELIMITER //**

**CREATE PROCEDURE updatesal()**

**BEGIN**

**DECLARE vempno INT;**

**DECLARE vdeptno INT;**

**DECLARE vexp INT;**

**DECLARE vename VARCHAR(30);**

**DECLARE vsal DOUBLE(9,2);**

**DECLARE vhdt DATE;**

**DECLARE vset INT DEFAULT 0;**

**DECLARE empsalcur CURSOR FOR**

**SELECT empno, ename, sal, deptno, hiredate FROM emp;**

**DECLARE CONTINUE HANDLER FOR NOT FOUND SET vset = 1;**

**OPEN empsalcur;**

**emp\_loop: LOOP**

**FETCH empsalcur INTO vempno, vename, vsal, vdeptno, vhdt;**

**IF vset = 1 THEN**

**LEAVE emp\_loop;**

**END IF;**

**SET vexp = calexp(vhdt);**

**IF vexp > 35 AND vexp <= 38 THEN**

**UPDATE emp SET sal = sal \* 1.20 WHERE empno = vempno;**

**ELSEIF vexp > 38 THEN**

**UPDATE emp SET sal = sal \* 1.25 WHERE empno = vempno;**

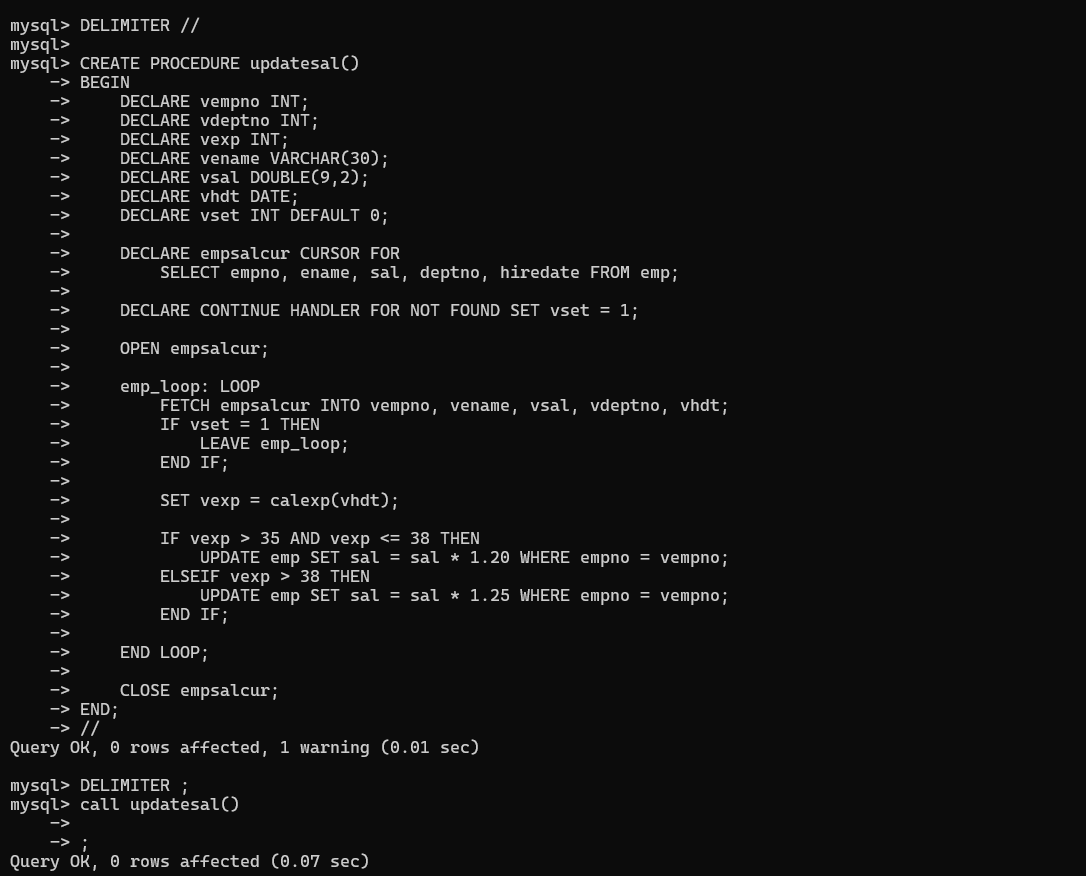
**END IF;**

**END LOOP;**

**CLOSE empsalcur;**

**END //**

**DELIMITER ;**

****

**10. Write a procedure and a function.**

**Function: write a function to calculate number of years of experience of employee.(note:**

**pass hiredate as a parameter)**

**Procedure: Capture the value returned by the above function to calculate the additional**

**allowance for the emp based on the experience.**

**Additional Allowance = Year of experience x 3000**

**Calculate the additional allowance**

**and store Empno, ename,Date of Joining, and Experience in**

**years and additional allowance in Emp\_Allowance table.**

**create table emp\_allowance(**

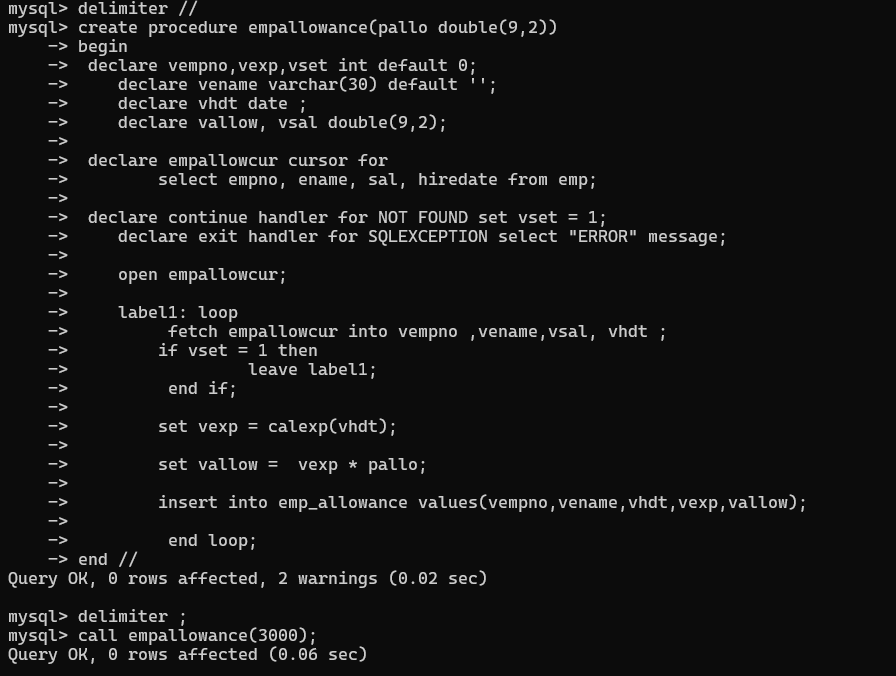
**empno int,**

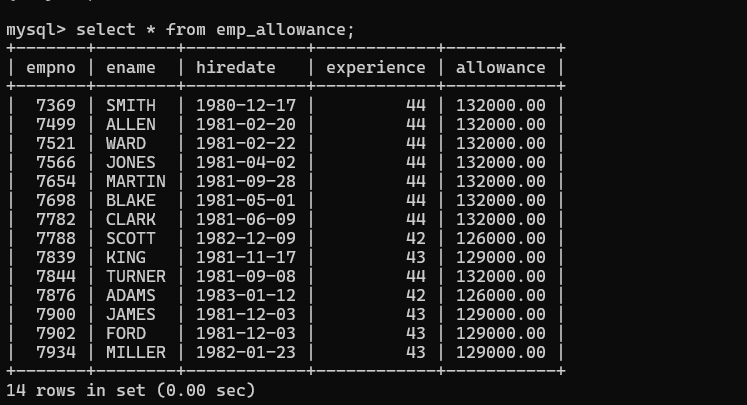
**ename varchar(20),**

**hiredate date,**

**experience int,**

**allowance decimal(9,2));**

****

****

**11. Write a function to compute the following. Function should take sal and hiredate**

**as i/p and return the cost to company.**

**DA = 15% Salary, HRA= 20% of Salary, TA= 8% of Salary.**

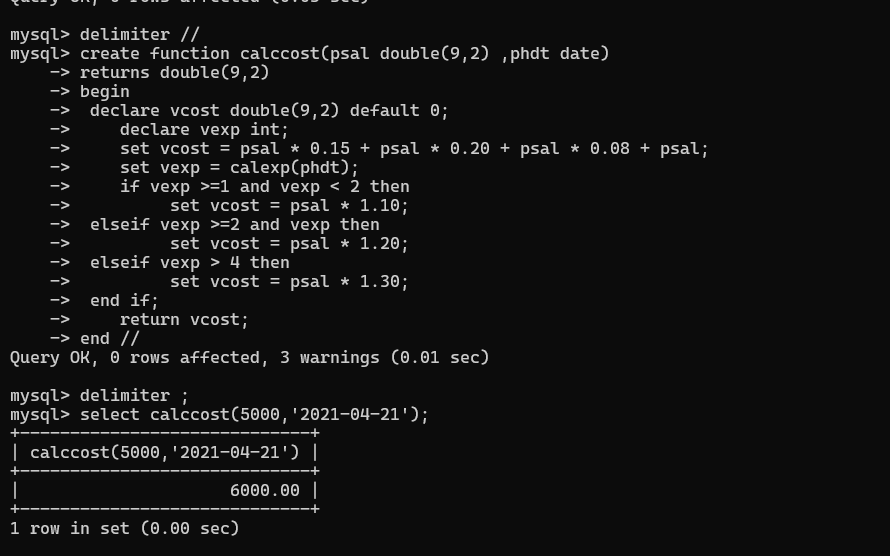
**Special Allowance will be decided based on the service in the company.**

**< 1 Year Nil**

**>=1 Year< 2 Year 10% of Salary**

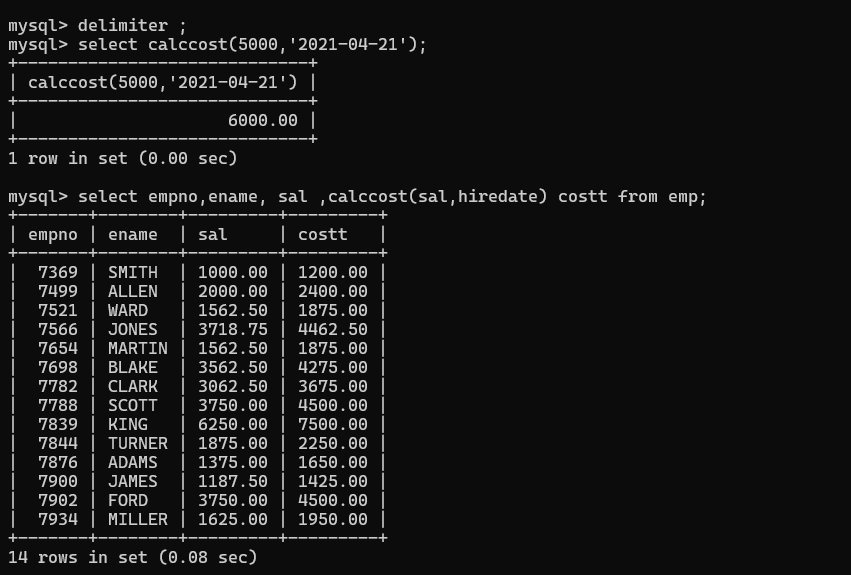
**>=2 Year< 4 Year 20% of Salary**

**>4 Year 30% of Salary**

****

**12. Write query to display empno,ename,sal,cost to company for all employees(note:**

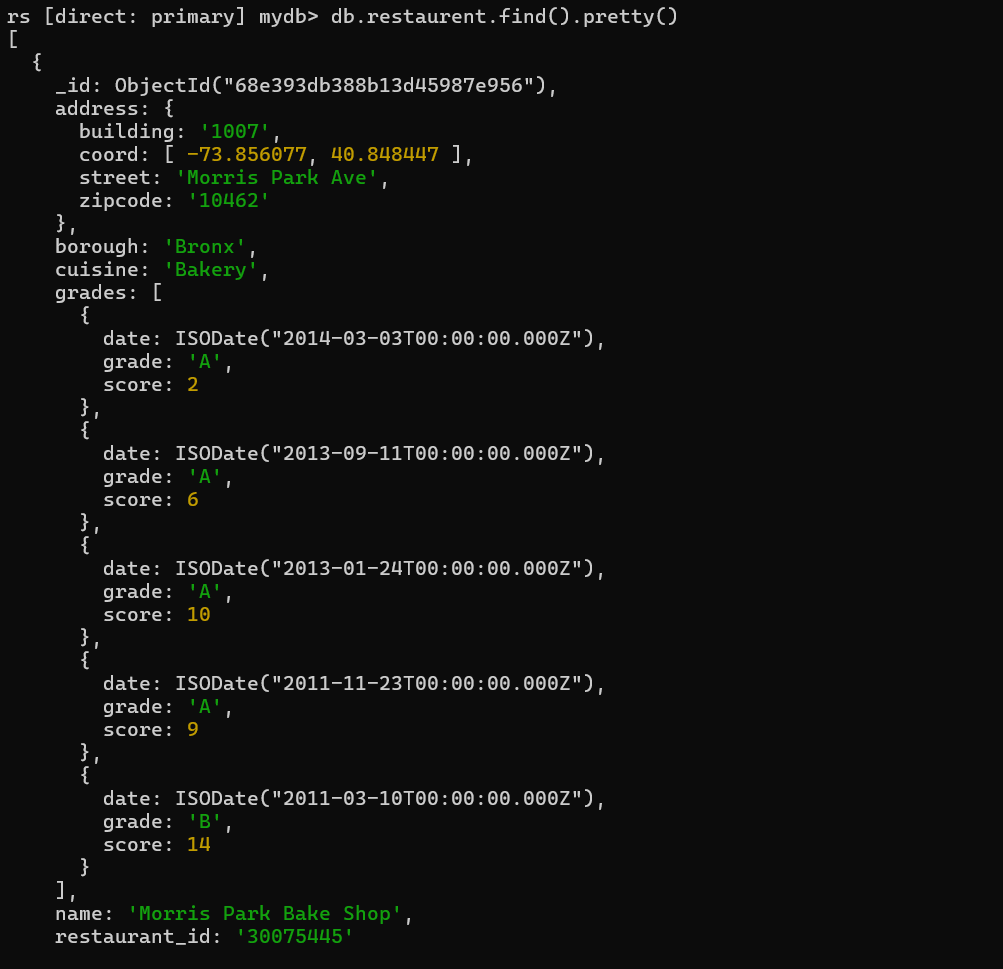
**use function written in question 10)**

****

**Lab 12**

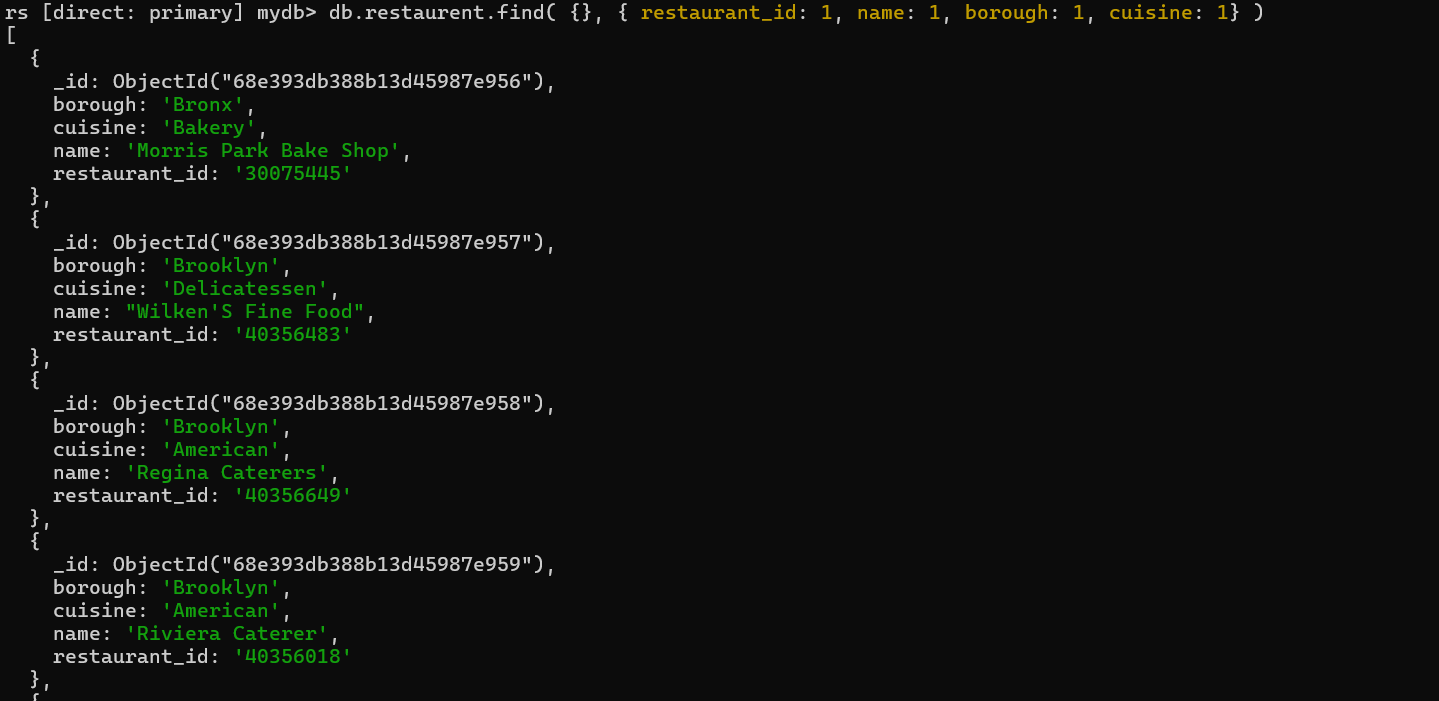
1. **Write a MongoDB query to display all the documents in the collection restaurants**

**db.restaurent.find().pretty()**

****

1. **Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for all the documents in the collection restaurant.**

**db.restaurent.find( {}, { restaurant\_id: 1, name: 1, borough: 1, cuisine: 1} )**

****

**3. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine, but exclude the field \_id for all the documents in the collection restaurant.**

**db.restaurent.find( {}, { restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0} )**

****

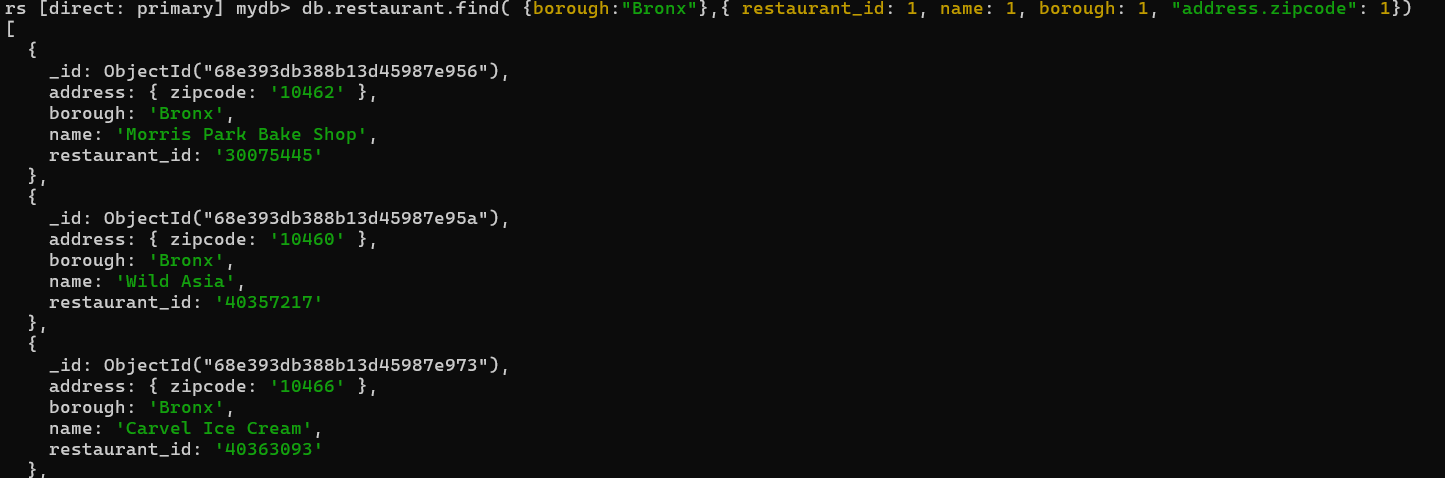
1. **Write a MongoDB query to display the fields restaurant\_id, name, borough and zip code, but exclude the field \_id for all the documents in the collection restaurant.**

**db.restaurant.find( {}, { restaurant\_id: 1, name: 1, borough: 1, "address.zipcode": 1, \_id: 0 } )**

****

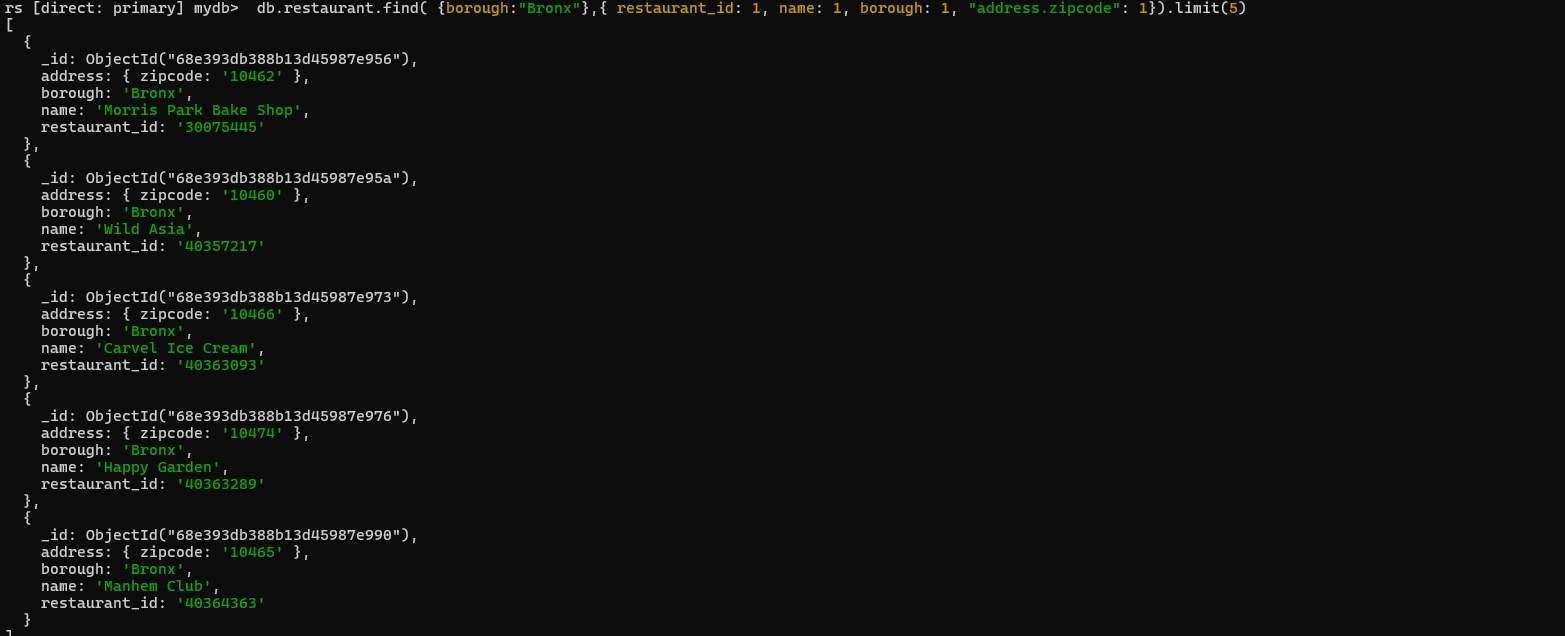
1. **Write a MongoDB query to display all the restaurant which is in the borough Bronx**

**db.restaurant.find( {borough:"Bronx"},{ restaurant\_id: 1, name: 1, borough: 1, "address.zipcode": 1})**

****

1. **Write a MongoDB query to display the first 5 restaurant which is in the borough Bronx.**

**db.restaurant.find( {borough:"Bronx"},{ restaurant\_id: 1, name: 1, borough: 1, "address.zipcode": 1}).limit(5)**

****

**7.Write a MongoDB query to display the next 5 restaurants after skipping first 5 which are in the borough Bronx.**

**db.restaurant.find({ borough: "Bronx" }, { restaurant\_id: 1, name: 1, borough: 1}).skip(5).limit(5)**

****

**8.Write a MongoDB query to find the restaurants who achieved a score more than 90.**

**db.restaurant.find({ "grades.score": { $gt: 90 } },{ restaurant\_id: 1, name: 1, borough: 1, "grades.score": 1, })**

****

**9. Write a MongoDB query to find the restaurants that achieved a score, more than 80 but less than 100.**

**db.restaurant.find({ "grades.score": { $gt: 80 ,$lt:100} },{ restaurant\_id: 1, name: 1, borough: 1, "grades.score": 1, })**

****

**10. Write a MongoDB query to find the restaurants which locate in latitude value less than 95.754168.**

**db.restaurant.find({ "address.coord.1": { $lt: 95.754168 } },{ restaurant\_id: 1, name: 1, "address.coord": 1 })**

****

**11. Write a MongoDB query to find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.**

**db.restaurant.find(**

**{**

**cuisine: { $ne: "American" },**

**"grades.score": { $gt: 70 },**

**"address.coord.1": { $lt: -65.754168 }**

**},**

**{**

**restaurant\_id: 1,**

**name: 1,**

**cuisine: 1,**

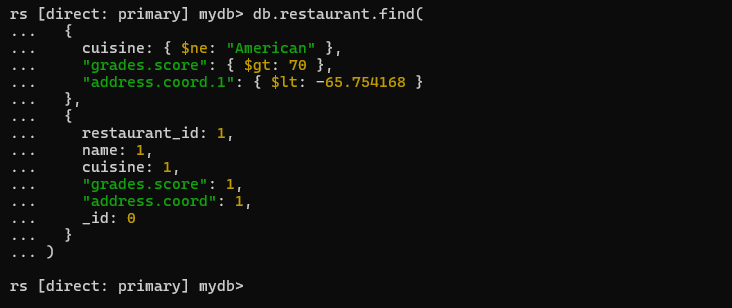
**"grades.score": 1,**

**"address.coord": 1,**

**\_id: 0**

**}**

**)**

****

**12. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than 65.754168.**

**db.restaurant.find(**

**{**

**cuisine: { $ne: "American" },**

**"grades.score": { $gt: 70 },**

**"address.coord.0": { $lt: 65.754168 }**

**},**

**{**

**restaurant\_id: 1,**

**name: 1,**

**cuisine: 1,**

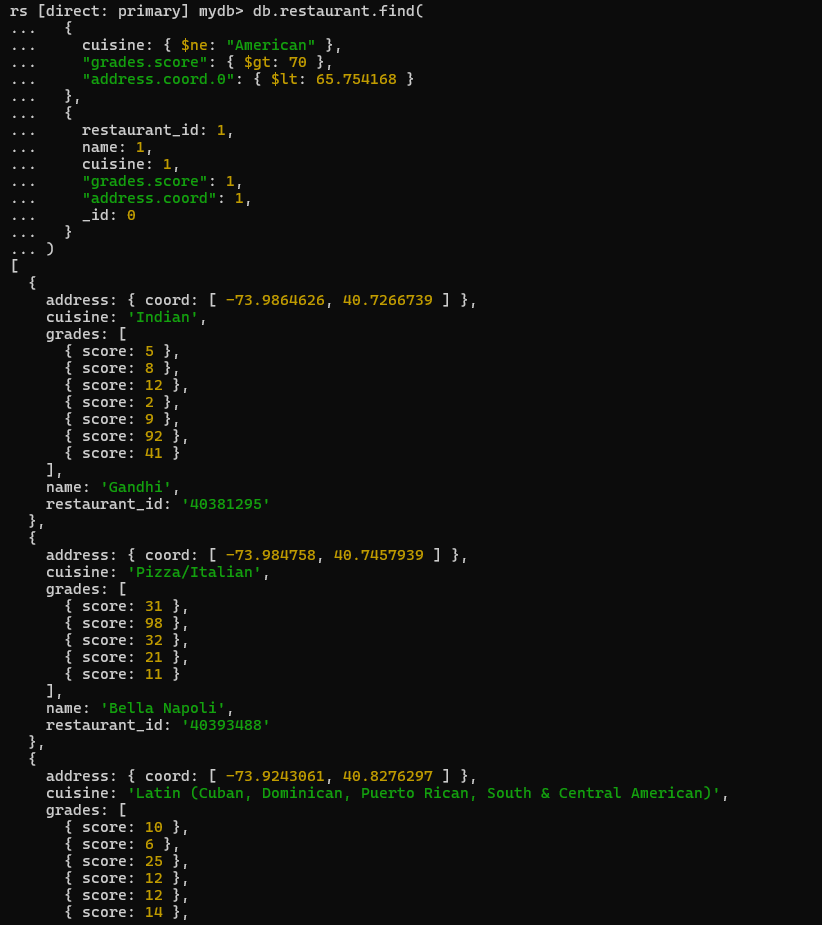
**"grades.score": 1,**

**"address.coord": 1,**

**\_id: 0**

**}**

**)**

**13. Write a MongoDB query to find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.**

**db.restaurant.find(**

**{**

**cuisine: { $ne: "American" },**

**"grades.grade": "A",**

**borough: { $ne: "Brooklyn" }**

**},**

**{**

**restaurant\_id: 1,**

**name: 1,**

**borough: 1,**

**cuisine: 1,**

**"grades.grade": 1,**

**\_id: 0**

**}**

**).sort({ cuisine: -1 })**

**14. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.**

**db.restaurant.find(**

**{ name: { $regex: /^Wil/} },**

**{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }**

**)**

****

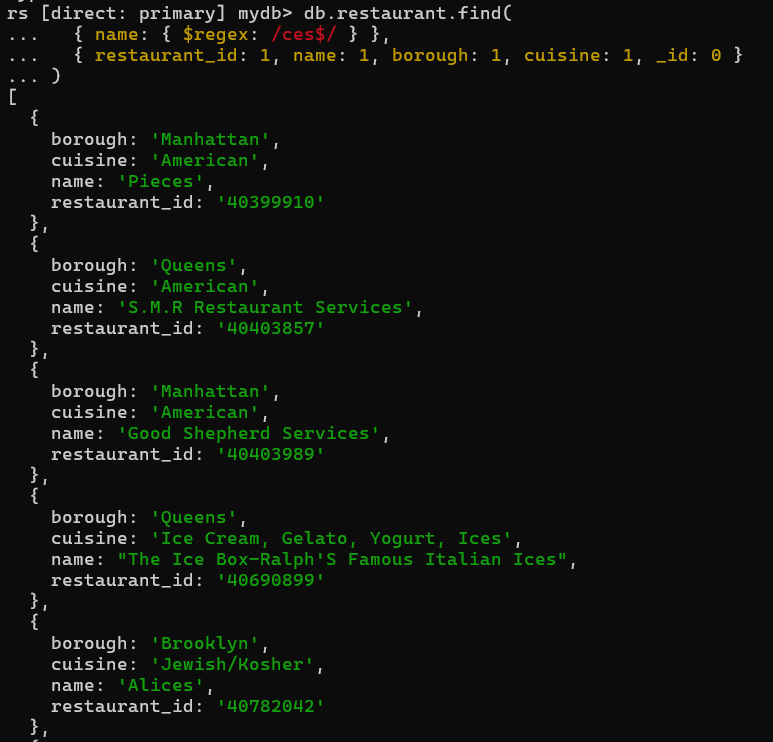
**15. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.**

**db.restaurant.find(**

**{ name: { $regex: /ces$/ } },**

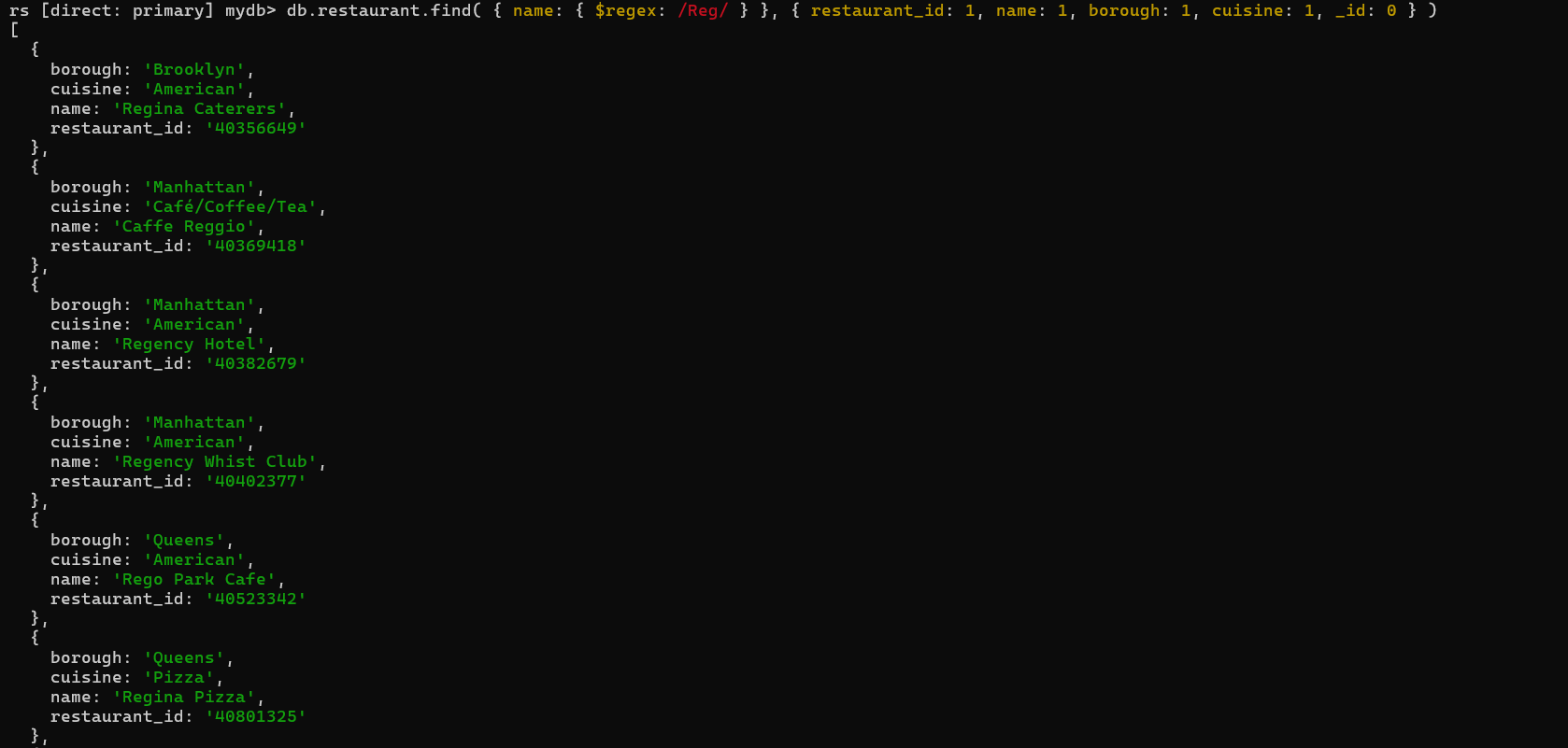
**{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 }**

**)**

****

**16. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.**

**db.restaurant.find( { name: { $regex: /Reg/ } }, { restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 } )**

**17. Write a MongoDB query to find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.**

**db.restaurant.find({ borough: "Bronx",cuisine: { $in: ["American", "Chinese"] }},{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 })**

****

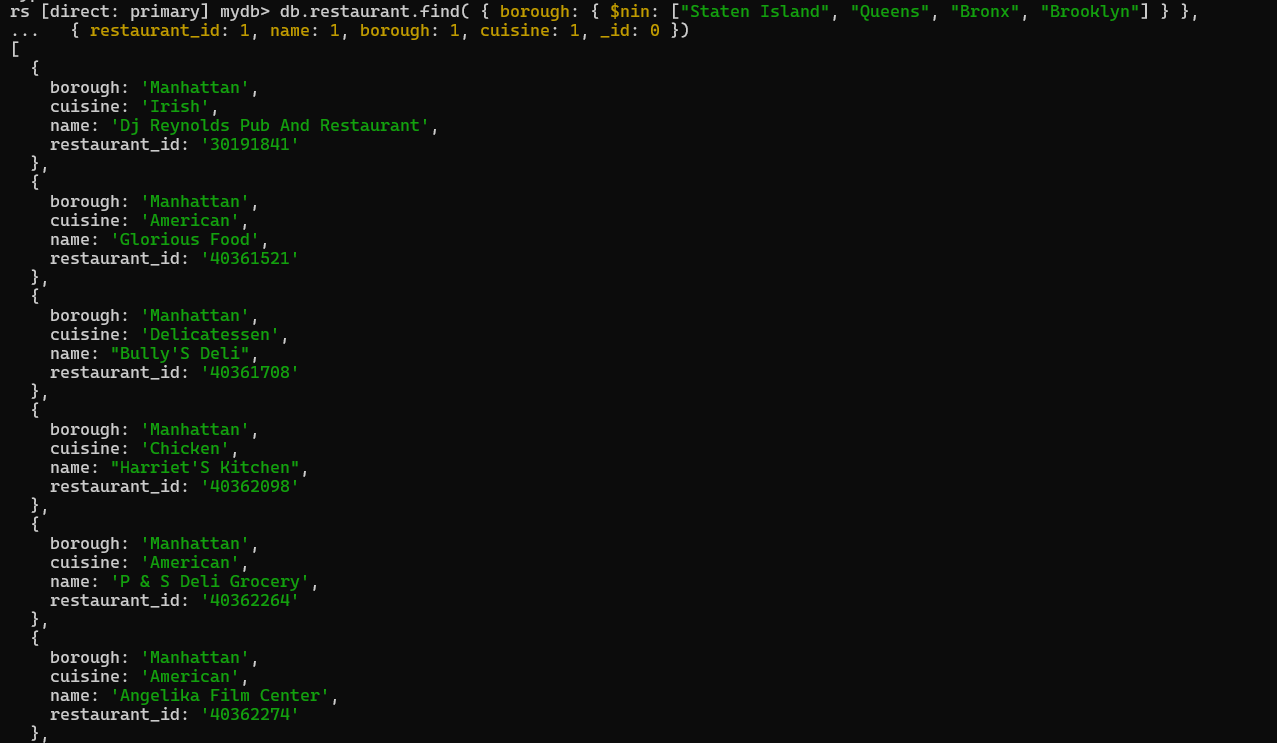
**18. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn.**

**db.restaurant.find({ borough: { $in: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 })**

**19. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.**

**db.restaurant.find( { borough: { $nin: ["Staten Island", "Queens", "Bronx", "Brooklyn"] } },**

**{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 })**

****

**20. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.**

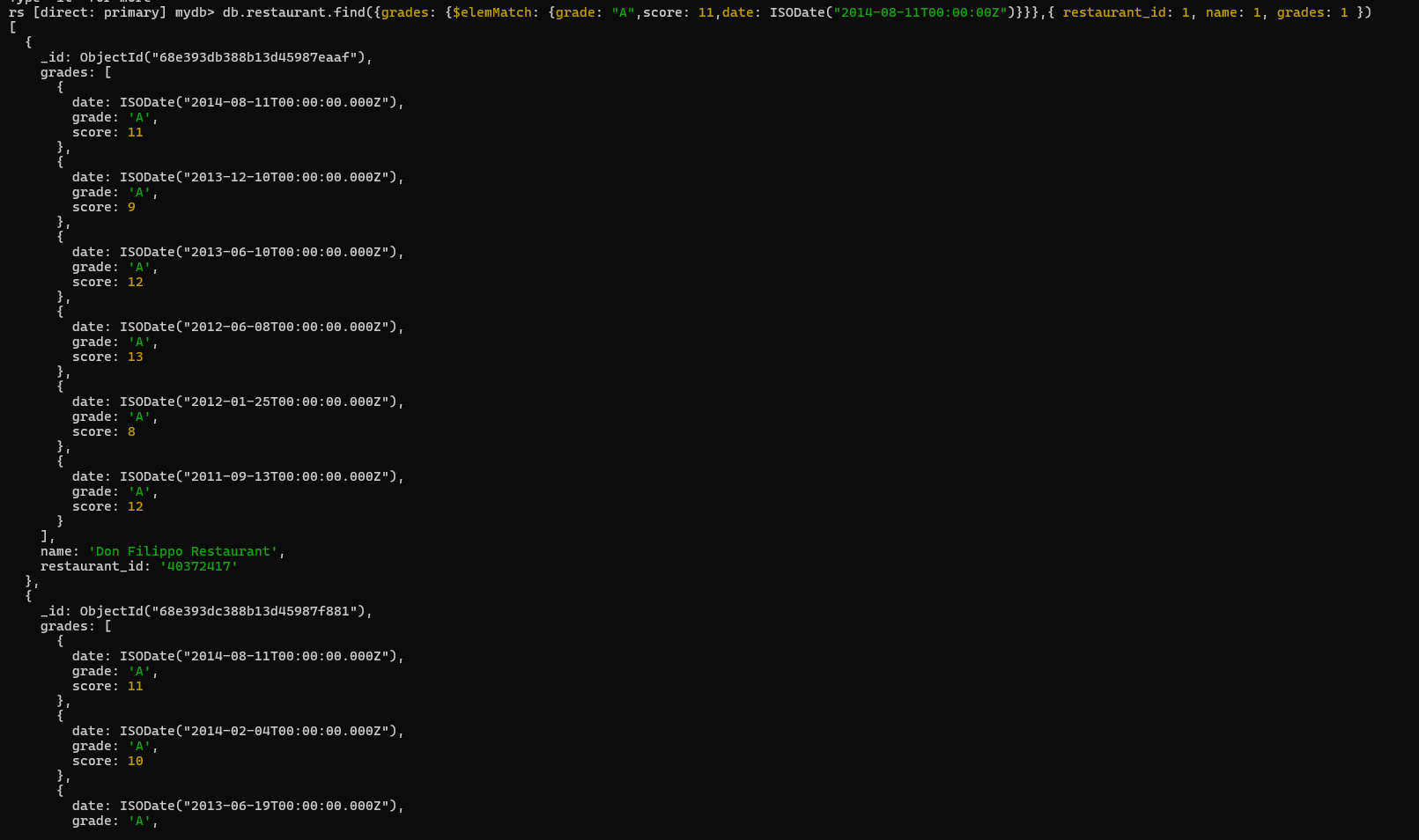
**db.restaurant.find({ "grades.score": { $lte: 10 } }, { restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, "grades.score": 1, \_id: 0 })**

**21. Write a MongoDB query to find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.**

**db.restaurant.find({$or: [ { cuisine: { $nin: ["American", "Chinese"] } },{ name: { $regex: /^Wil/i } }]},{ restaurant\_id: 1, name: 1, borough: 1, cuisine: 1, \_id: 0 })**

** 22. Write a MongoDB query to find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates**

**db.restaurant.find({grades: {$elemMatch: {grade: "A",score: 11,date: ISODate("2014-08-11T00:00:00Z")}}},{ restaurant\_id: 1, name: 1, grades: 1 })**

****

**Lab 13**

**Create a Employee Collection add 5 documents:**

**Example: {emono:111,ename:”Deepali Vaidya”,sal:40000.00,dept:{deptno:12,dname:,”Hr”,dloc:”Mumbai}, Desg:”Analyst”,mgr:{name:”Satish”,num:111},project:[{name:”Project1”,Hrs:4},{name:”project- 2”,Hrs:4}]}**

**db.employee.insertMany([{**

**empno:111,**

**ename:"Deepali Vaidya",**

**sal: 40000.00,**

**dept: {deptno: 12, dname: "HR", dloc: "Mumbai"},**

**desg:"Analyst",**

**mgr:{name:"Satish",num:109},**

**project:[{name:"Project-1",HRS:4},{name:"Project-2",HRS: 4}]**

**},**

**{**

**empno:112,**

**ename:"Seema Patil",**

**sal: 20000.00,**

**dept: {deptno: 13, dname: "SALES", dloc: "PUNE"},**

**desg:"CLERK",**

**mgr:{name:"RAMESH",num:110},**

**project:[{name:"Project-3",HRS:4},{name:"Project-4",HRS: 4}]**

**},**

**{**

**empno:113,**

**ename:"RAHUL PATEL",**

**sal: 60000.00,**

**dept: {deptno: 12, dname: "HR", dloc: "Mumbai"},**

**desg:"DEVELOPER",**

**mgr:{name:"Satish",num:109},**

**project:[{name:"Project-1",HRS:4},{name:"Project-2",HRS: 4}]**

**},**

**{**

**empno:114,**

**ename:"KAVITA Vaidya",**

**sal: 30000.00,**

**dept: {deptno: 12, dname: "FINANCE", dloc: "DELHI"},**

**desg:"CLERK",**

**mgr:{name:"RAMESH",num:110},**

**project:[{name:"Project-3",HRS:4},{name:"Project-2",HRS: 4}]**

**},**

**{**

**empno:116,**

**ename:"RAJU RASTOGI",**

**sal: 50000.00,**

**dept: {deptno: 16, dname: "SECURITY", dloc: "DELHI"},**

**desg:"ANALYST",**

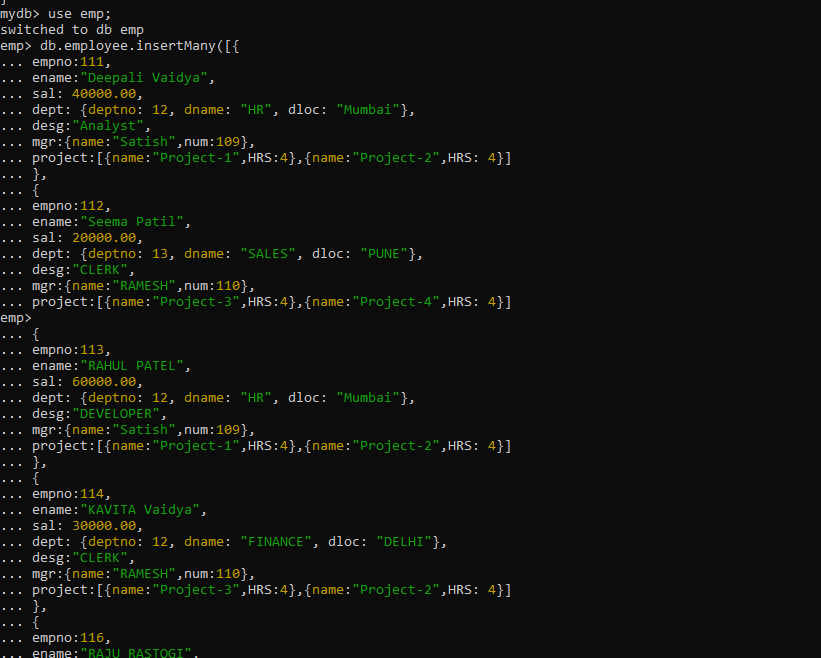
**mgr:{name:"RAMESH",num:110},**

**project:[{name:"Project-3",HRS:4},{name:"Project-2",HRS: 4}]**

**}**

**]);**

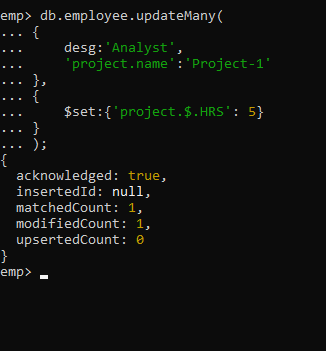
**db.employee.countDocuments({});**

****

1. **All Employee’s with the desg as ‘CLERK’ are now called as (AO) Administrative Officers. Update the Employee collection for this.**

****

1. **Change the number of hours for project-1 to 5 for all employees with designation analyst.**

****

1. **Add 2 projects project-3 and project-4 for employee whose name starts with ”Deep” with 2 hrs**

****

* **Add bonus rs 2000 for all employees with salary > 50000**
* **add bonus 1500 if salary 30000**
* **add bonus 1000 if salary <=30000**
* **Change manager name to Tushar for all employees whose manager is currently “satish” And manager number to 3333**
* **Increase salary of all employees from “purchase department” by 15000**
* **Decrease number of hrs by 2 for all employees who are working on project-2**
* **Delete project-2 from all employee document if they are working on the project for 4 hrs. db.emp.update({},{$pull:{“project.name”:”project2”,”project.Hrs”:4}},{multi:true}) or db.Emp.update({},{$pull:{project:{name:'project-2',Hrs:4}}},{multi:true}) or db.emp.update({},{$pull:{“project:{$elemMatch:{name:”project2”,”Hrs”:4}},{multi:true})**
* **Change the salary of employees to 10000 only if their salary is < 10000**
* **Increase bonus of all employees by 500 if the bonus is < 20000 or if employee belong to sales department**
* **Add 2 new project at position 2 for all employees with designation analyst or salary is equal to either 30000 or 33000 or 35000**
* **Delete last project of all employees with department name is “HR” and if the location is Mumbai**
* **Change designation of all employees to senior programmer if they are working on name:”Project-1” for 4 hrs**
* **Add list of hobbies in all employees document whose manager is Rajan or Revati**
* **Add list of skillset in all employee documents who are working on project-4 for 3 hrs or on project-3 for 4 hrs**
* **Add a new hobby as blogging at 3 position in hobbies array for all employess whose name starts with R or p and ends with j or s**
* **Increase salary by 10000 for all employees who are working on project-2 or project-3 or project-1**
* **Decrease bonus by 1000 rs And increase salary by 1000rs for all employees whose department location is Mumbai**