1. To list all records with sal > 2000 and comm>200

Ans :

mysql> select \*

-> from emp

-> where sal > 2000 and comm > 200;

Empty set (0.00 sec)

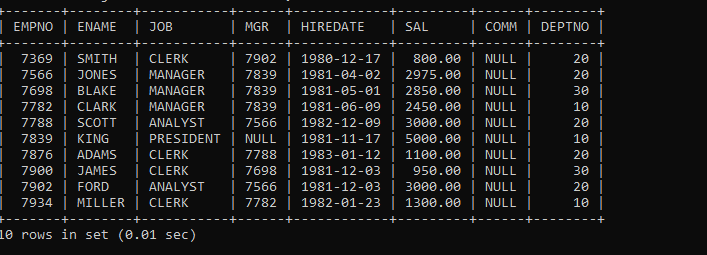
1. To list all record with job=’Clerk’ or sal>2000

Ans :

mysql> select \*

-> from emp

-> where job='clerk' or sal>2000;



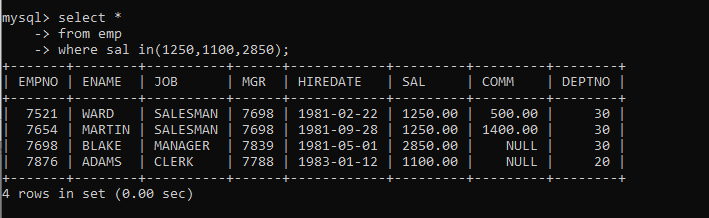
1. To list all the record with sal=1250 or 1100 or 2850

Ans:-

mysql> select \*

-> from emp

-> where sal in(1250,1100,2850);



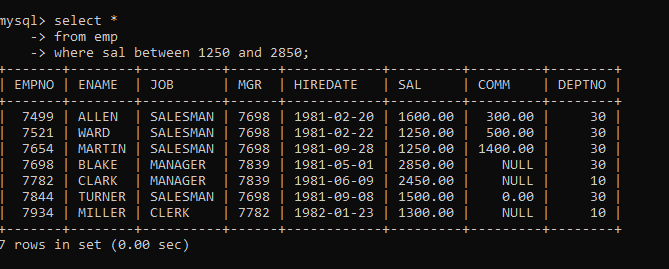
1. To list all employees with sal>1250 and <2850

Ans:-

mysql> select \*

-> from emp

-> where sal between 1250 and 2850;



5. To list all employees with name ends with AS

Ans:-

mysql> select \*

-> from emp

-> where ename like '%AS';

Empty set (0.04 sec)

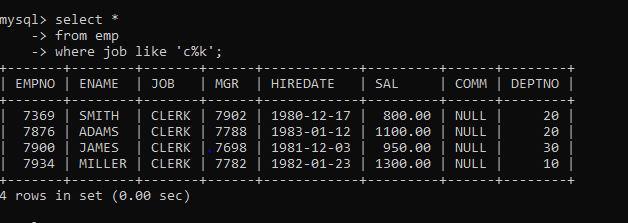
6.To list all employees with job starts with C and ends with K

Ans:-

mysql> select \*

-> from emp

-> where job like 'c%k';



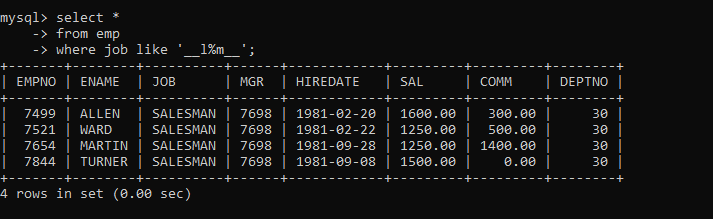
7. To list all employees with job contains L at third position and M at third last position

Ans:-

mysql> select \*

-> from emp

-> where job like '\_\_l%m\_\_';



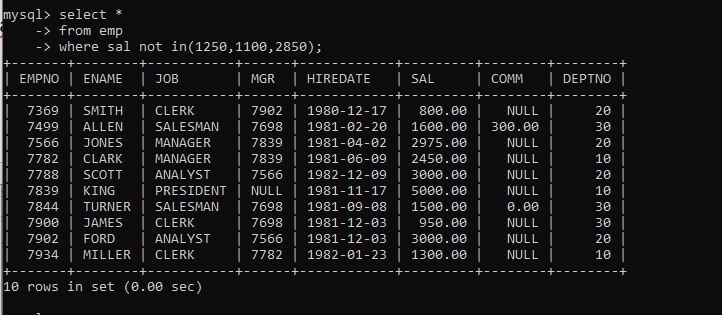
8. To list all the record with sal not equal to 1250 or 1100 or 2850

Ans:-

mysql> select \*

-> from emp

-> where sal not in(1250,1100,2850);



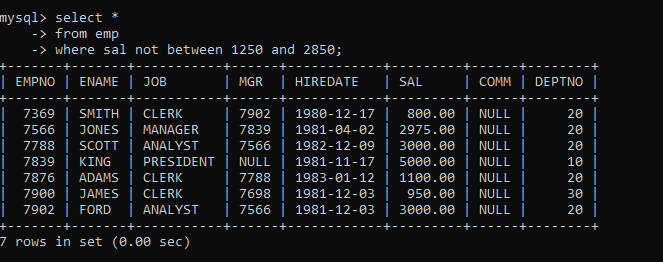
9. To list all employees with sal not >1250 and <2850

Ans:-

mysql> select \*

-> from emp

-> where sal not between 1250 and 2850;



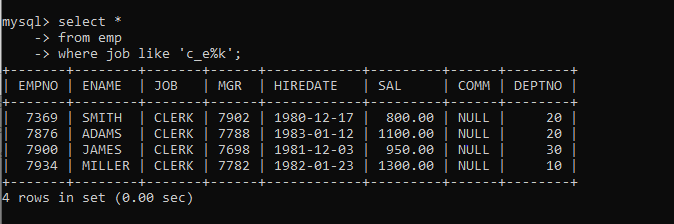
10. To list all employees with job starts with C , E at 3rd position and ends with K

Ans:-

mysql> select \*

-> from emp

-> where job like 'c\_e%k';



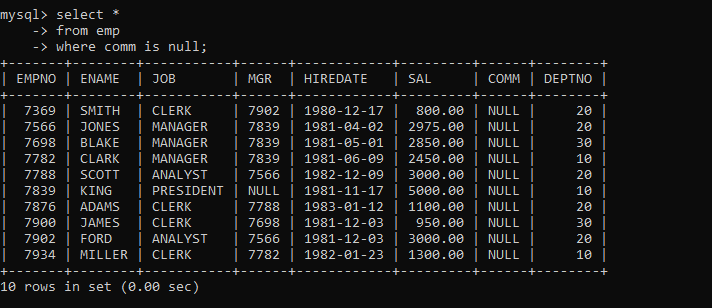
11. To list all rows with comm is null

Ans:-

mysql> select \*

-> from emp

-> where comm is null;



12. To list all employees with sal is null and name starts with ‘S

Ans:-

mysql> select \*

-> from emp

-> where sal is null and ename like 's%';

Empty set (0.00 sec)

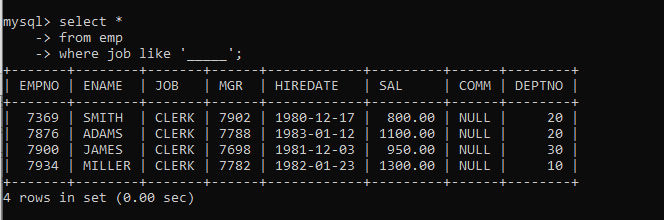
13. To list all employees with job contains 5 characters

Ans:-

mysql> select \*

-> from emp

-> where job like '\_\_\_\_\_';



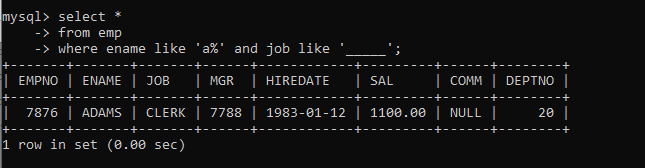
14. To list all employees with name contain ‘A’ at 1 position and job Contains 5 characters.

Ans:-

mysql> select \*

-> from emp

-> where ename like 'a%' and job like '\_\_\_\_\_';



Q2. Solve the following

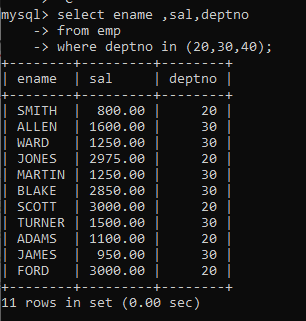
1. Retrieve the details (Name, Salary and dept no) of the emp who are working in department code 20, 30 and 40.

Ans :

mysql> select ename ,sal,deptno

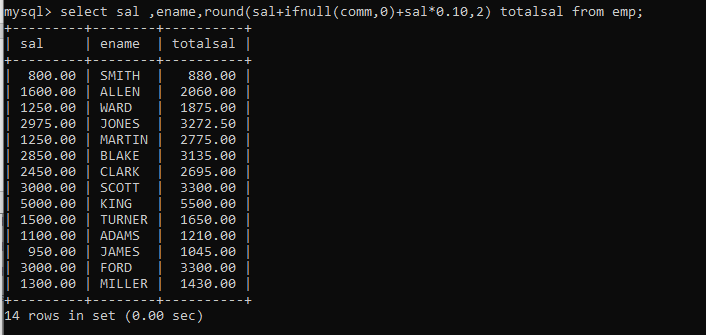
-> from emp

-> where deptno in (20,30,40);



2. Display the total salary of all employees . Total salary will be calculated as sal+comm+sal\*0.10

Ans: mysql> select sal ,ename,round(sal+ifnull(comm,0)+sal\*0.10,2) totalsal from emp;

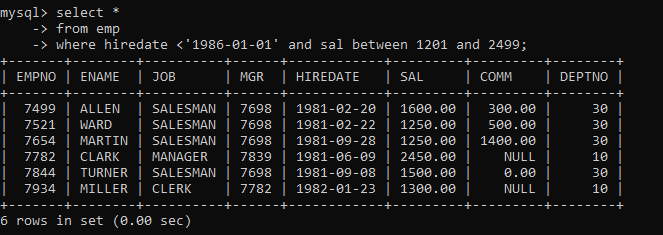


3. List the Name and job of the emp who have joined before 1 jan 1986 and whose salary range is between 1200 and 2500. Display the columns with user defined Column headers.

Ans :

mysql> select \*

-> from emp

-> where hiredate <'1986-01-01' and sal between 1201 and 2499; 

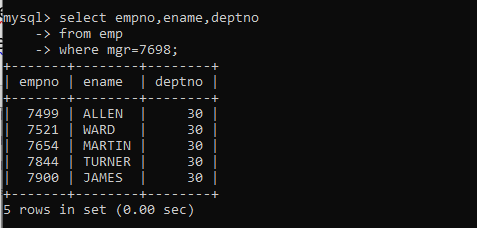
4. List the emp no, name, and department number of the emp works under manager with id 7698

Ans:-

mysql> select empno,ename,deptno

-> from emp

-> where mgr=7698;



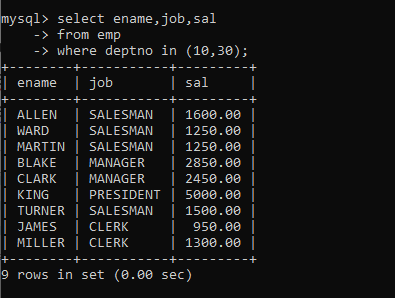
1. List the name, job, and salary of the emp who are working in departments 10 and 30.

Ans:-

mysql> select ename,job,sal

-> from emp

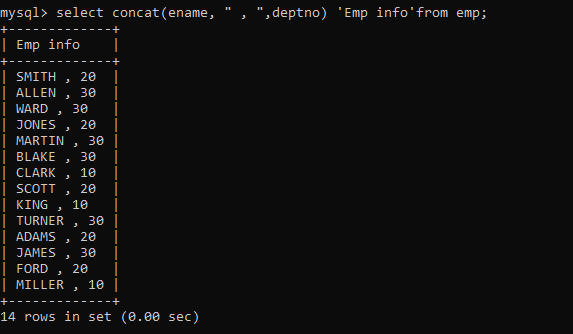
-> where deptno in (10,30);



6. Display name concatenated with dept code separated by comma and space. Name the column as ‘Emp info’.

Ans:-

mysql> select concat(ename, " , ",deptno) 'Emp info'from emp;



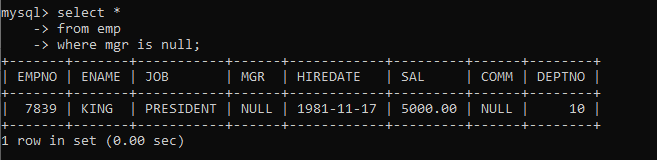
7. Display the emp details who do not have manager.

Ans :

mysql> select \*

-> from emp

-> where mgr is null;



8. Write a query which will display name, department no and date of joining of all employee who were joined

January 1, 1981 and March 31, 1983. Sort it based on date of joining (ascending). ;

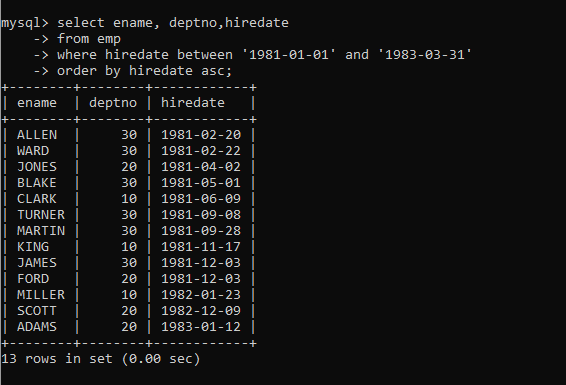
Ans :

mysql> select ename, deptno,hiredate

-> from emp

-> where hiredate between '1981-01-01' and '1983-03-31'

-> order by hiredate asc;



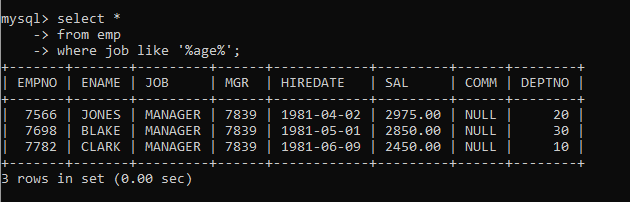
9. Display the employee details where the job contains word ‘AGE’ anywhere in the Job

Ans:-

mysql> select \*

-> from emp

-> where job like '%age%';

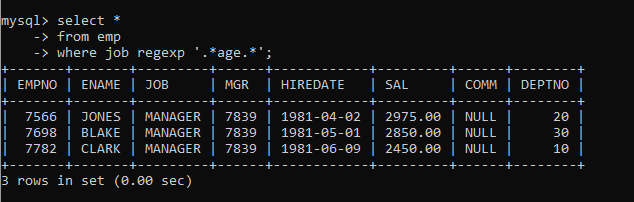


Or

mysql> select \*

-> from emp

-> where job regexp '.\*age.\*';



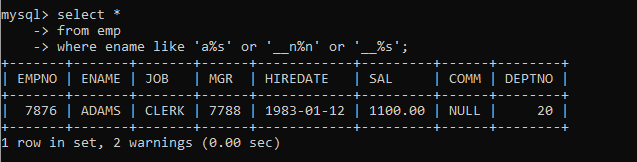
11. List the details of the employee , whose names start with ‘A’ and end with ‘S’ or whose names contains N as the second or third character, and ending with either ‘N’ or ‘S’.

Ans:-

mysql> select \*

-> from emp

-> where ename like 'a%s' or '\_\_n%n' or '\_\_%s';

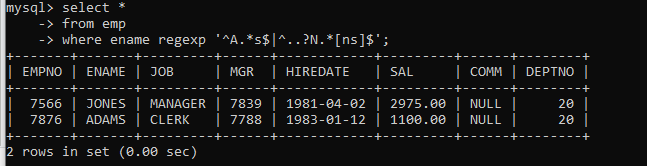


Or

mysql> select \*

-> from emp

-> where ename regexp '^A.\*S$|^..?N.\*[NS]$';



12. List the names of the emp having ‘\_’ character in their name.

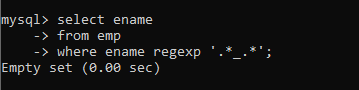
Ans:-

mysql> select ename

-> from emp

-> where ename regexp '.\*\_.\*';

Empty set (0.00 sec)



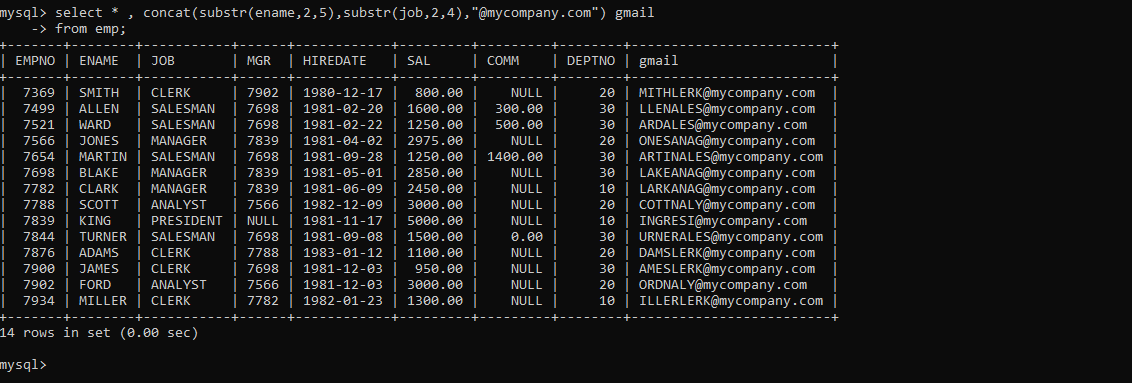
Single Row functions :-

1. To list all employees and their email, to generate email use 2 to 5 characters from ename Concat it with 2 to 4 characters in job and then concat it with ‘@mycompany.com’

Ans:-

mysql> select \* , concat(substr(ename,2,5),substr(job,2,4),"@mycompany.com") gmail

-> from emp;



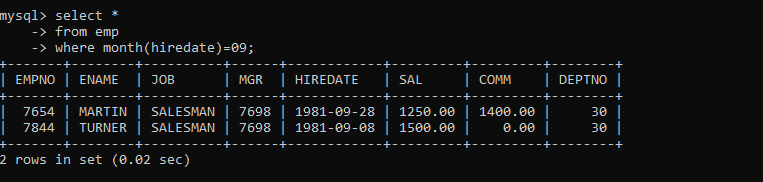
2. List all employees who joined in September.

Ans:-

mysql> select \*

-> from emp

-> where month(hiredate)=09;



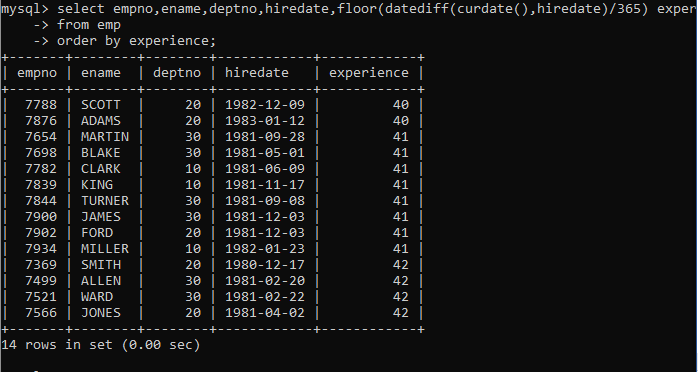
3. List the empno, name, and department number of the emp who have experience of 18 or more years and sort them based on their experience.

Ans:-

mysql> select empno,ename,deptno,hiredate,floor(datediff(curdate(),hiredate)/365) experience

-> from emp

-> order by experience;



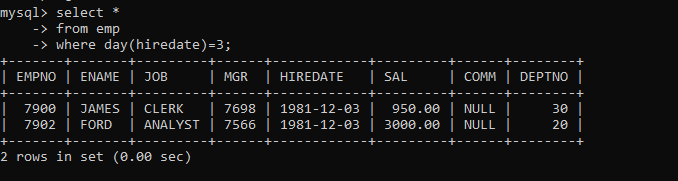
4. Display the employee details who joined on 3rd of any month or any year

Ans:-

mysql> select \*

-> from emp

-> where day(hiredate)=3;



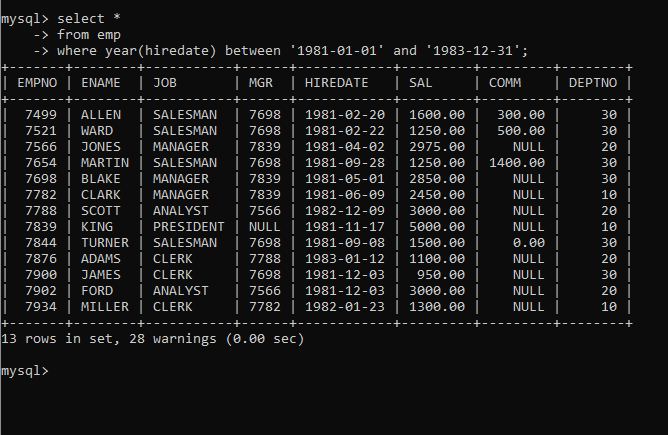
5.display all employees who joined between years 1981 to 1983.

Ans:-

mysql> select \*

-> from emp

-> where year(hiredate) between '1981-01-01' and '1983-12-31';



Group functions

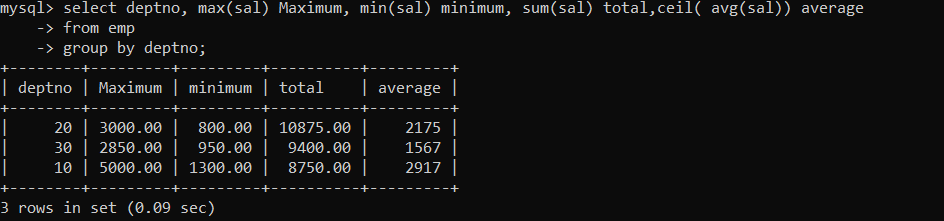
1. Display the Highest, Lowest, Total & Average salary of all employee. Label the columns Maximum, Minimum, Total and Average respectively for each Department. Also round the result to the nearest whole number.

Ans:-

mysql> select deptno, max(sal) Maximum, min(sal) minimum, sum(sal) total,ceil( avg(sal)) average

-> from emp

-> group by deptno;



1. Display Department no and number of managers working in that department. Label the column as ‘Total Number of Managers’ for each department.

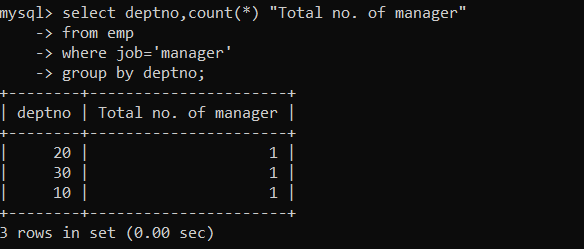
Ans:-

mysql> select deptno,count(\*) "Total no. of manager"

-> from emp

-> where job='manager'

-> group by deptno;



8. Get the Department number, and sum of Salary of all non managers where the sum is greater than 20000

Ans:-

practice DQL statement

Write SQL statement for the following

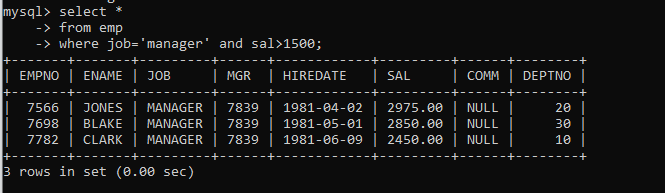
1. To find all managers with salary >1500

Ans:-

mysql> select \*

-> from emp

-> where job='manager' and sal>1500;



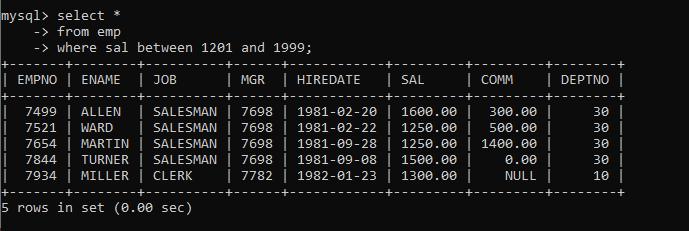
2. list all employees with sal >1200 and < 2000

Ans:-

mysql> select \*

-> from emp

-> where sal between 1201 and 1999;

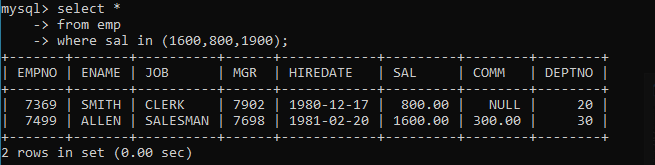


3. list all employees with sal is 1600 or sal is 800 or sal is 1900

Ans:-mysql> select \*

-> from emp

-> where sal in (1600,800,1900);

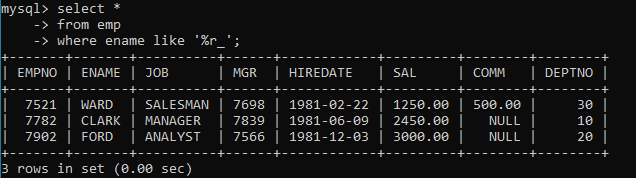


4. list all employees with R at second last position in name

Ans :mysql> select \*

-> from emp

-> where ename like '%r\_';

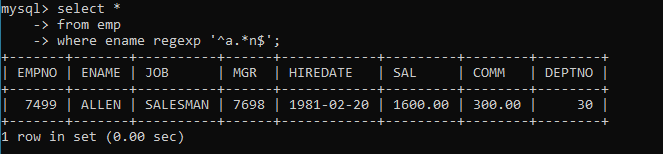


5. List all employees with name starts with A and ends with N

Ans : mysql> select \*

-> from emp

-> where ename regexp '^a.\*n$';



Q2. Solve following

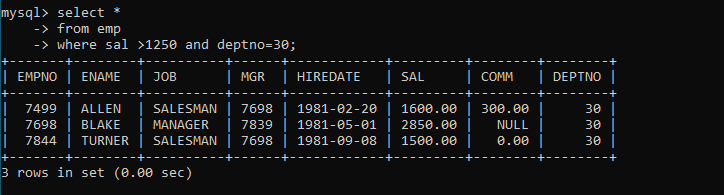
1. list all employees with salary > 1250 and dept no=30

Ans:-

mysql> select \*

-> from emp

-> where sal >1250 and deptno=30;



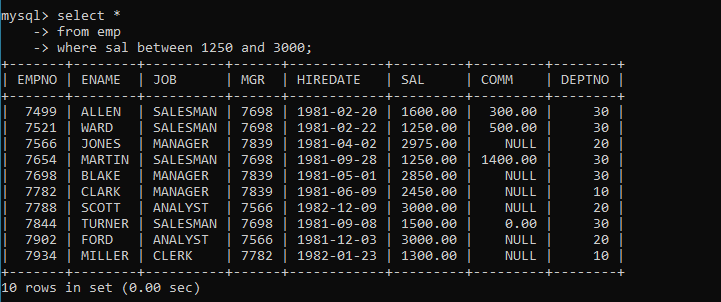
2. list all employees with salary >=1250 and <= 3000

Ans:-

mysql> select \*

-> from emp

-> where sal between 1250 and 3000;



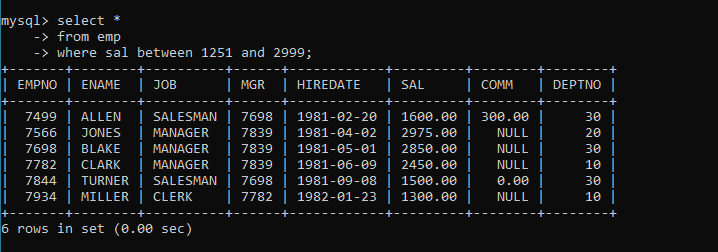
3. list all employees with salary >1250 and < 3000

Ans:-

mysql> select \*

-> from emp

-> where sal between 1251 and 2999;



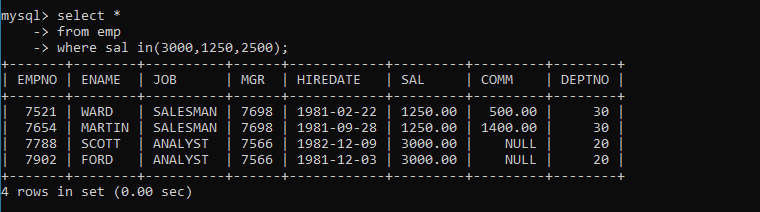
4. list all employees with salary either equal to 3000 or 1250 or 2500

Ans:-

mysql> select \*

-> from emp

-> where sal in(3000,1250,2500);



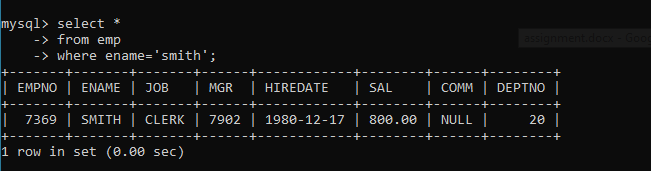
5. list all employee with name=SMITH

Ans:-

mysql> select \*

-> from emp

-> where ename='smith';



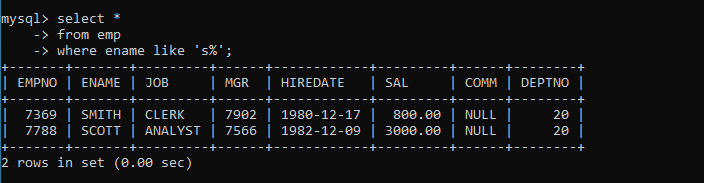
6. list all employees with name starting with S

Ans:-

mysql> select \*

-> from emp

-> where ename like 's%';



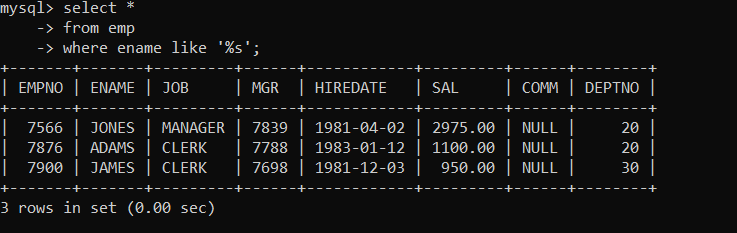
7. list all employees with name ending with S

Ans:-

mysql> select \*

-> from emp

-> where ename like '%s';



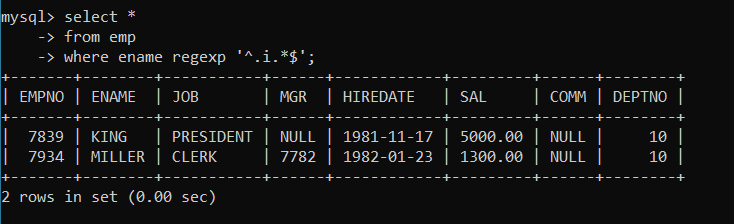
8. list all employees with name contains I at 2nd position

Ans:-

mysql> select \*

-> from emp

-> where ename regexp '^.i.\*$';



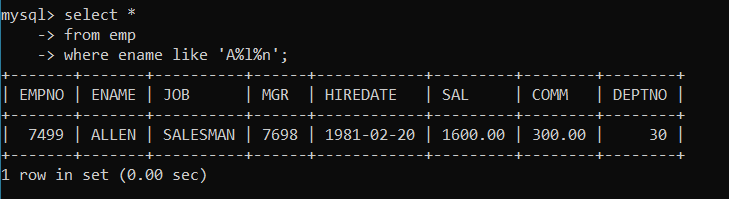
9. list all employees with name starts with A ends with N and somewhere in between L is there

Ans:-

mysql> select \*

-> from emp

-> where ename like 'A%l%n';



10. list all employees with name starts with A and B at 3 rd position and P at second last position

Ans:-

mysql> select \*

-> from emp

-> where ename regexp '^a.b.\*p.';

Empty set (0.00 sec)

or

mysql> select \*

-> from emp

-> where ename like 'A\_b%p\_';

Empty set (0.00 sec)

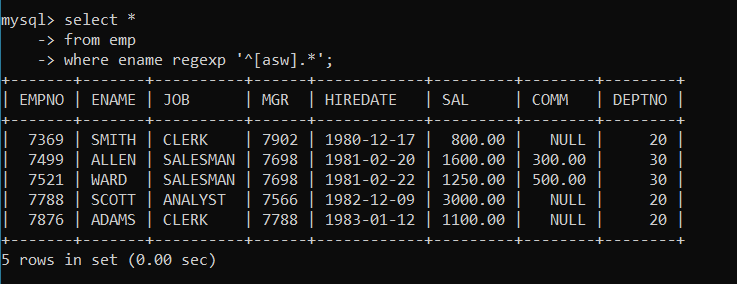
11. List all employees with name starts with either A or starts with S or starts with W

Ans:-

mysql> select \*

-> from emp

-> where ename regexp '^[asw].\*';



practice Aggregate functions

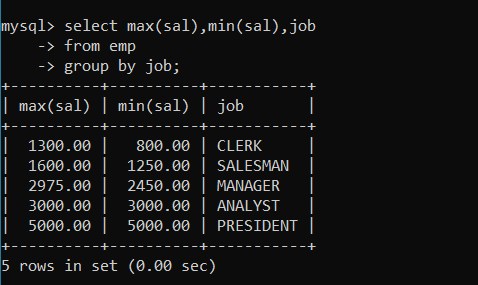
12. find max sal and min sal for each job

Ans:-

mysql> select max(sal),min(sal),job

-> from emp

-> group by job;

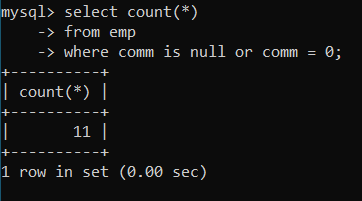


13. find how many employess have not received commission

Ans:-mysql> select count(\*)

-> from emp

-> where comm is null or comm = 0;

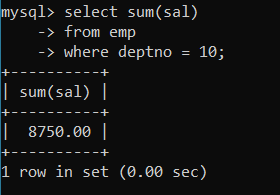


14. find sum of sal of all employees working in dept no 10

Ans :mysql> select sum(sal)

-> from emp

-> where deptno = 10;

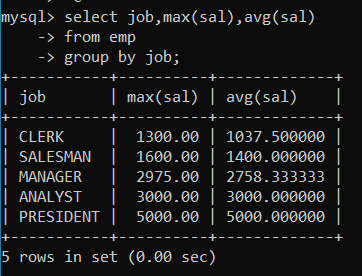


15. find maximum salary,average sal for each job in every department

Ans :mysql> select job,max(sal),avg(sal)

-> from emp

-> group by job;



16. find max salary for every department if deptno is > 15 and arrange data in deptno order.

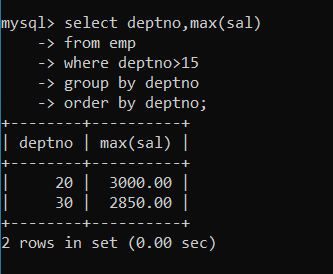
Ans : mysql> select deptno,max(sal)

-> from emp

-> where deptno>15

-> group by deptno

-> order by deptno;



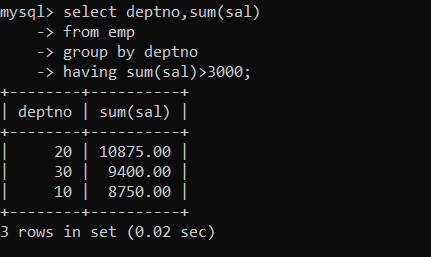
17. find sum salary for every department if sum is > 3000

Ans : mysql> select deptno,sum(sal)

-> from emp

-> group by deptno

-> having sum(sal)>3000;



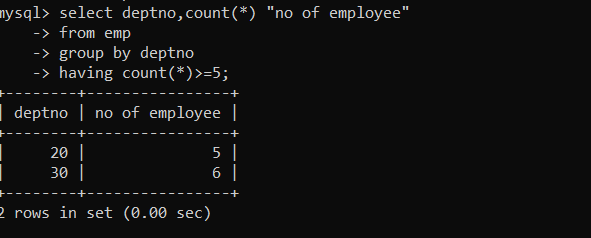
18. list all department which has minimum 5 employees

Ans:-mysql> select deptno,count(\*) "no of employee"

-> from emp

-> group by deptno

-> having count(\*)>=5;



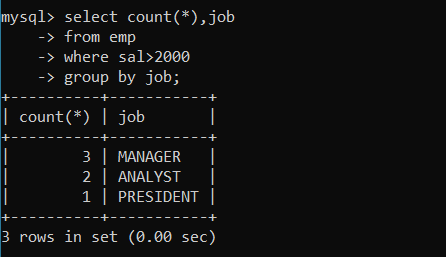
19. count how many employees earn salary more than 2000 in each job

Ans:-mysql> select count(\*),job

-> from emp

-> where sal>2000

-> group by job;



20. list all enames and jobs in small case letter

Ans:-mysql> select lower(ename),lower(job)

-> from emp;



21. list all names and jobs so that the length of name should be 15 if it is smaller then add spaces to left

Ans:-

mysql> select lpad(ename,15," ")"Employee name",Job from emp;

+-----------------+-----------+

| Employee name | Job |

+-----------------+-----------+

| SMITH | CLERK |

| ALLEN | SALESMAN |

| WARD | SALESMAN |

| JONES | MANAGER |

| MARTIN | SALESMAN |

| BLAKE | MANAGER |

| CLARK | MANAGER |

| SCOTT | ANALYST |

| KING | PRESIDENT |

| TURNER | SALESMAN |

| ADAMS | CLERK |

| JAMES | CLERK |

| FORD | ANALYST |

| MILLER | CLERK |

+-----------------+-----------+

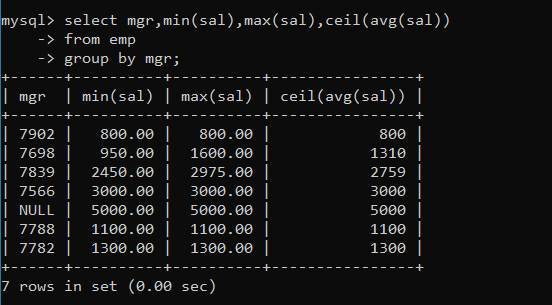
14 rows in set (0.00 sec)

22. display min sal,max sal, average sal for all employees working under same manager

Ans:-mysql> select mgr,min(sal),max(sal),ceil(avg(sal))

-> from emp

-> group by mgr;



23. find sum of total earnings(sal+comm), average of sal+comm for all employees who earn sal > 2000 and work in either dept no 10 or 20

Ans:-

mysql> select sum(sal+ifnull(comm,0))"Sum of total earnings",avg(sal+ifnull(comm,0))"Average of Emp in dept 10 or 20 and sal 2000+" from emp where sal>2000 and deptno in (10,20);

+-----------------------+-----------------------------------------------+

| Sum of total earnings | Average of Emp in dept 10 or 20 and sal 2000+ |

+-----------------------+-----------------------------------------------+

| 16425.00 | 3285.000000 |

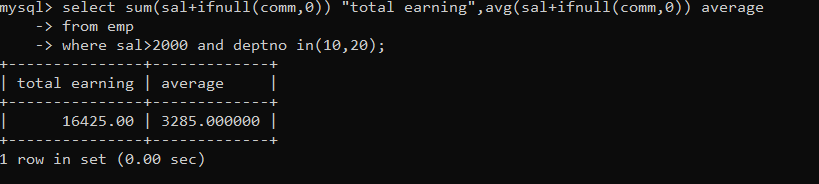
+-----------------------+-----------------------------------------------+

1 row in set (0.00 sec)

mysql> select sum(sal+ifnull(comm,0)) "total earning",avg(sal+ifnull(comm,0)) average

-> from emp

-> where sal>2000 and deptno in(10,20);



24. list all employees who joined in Aug 1980 and salary is >1500 and < 2500

Ans:-mysql> select \*

-> from emp

-> where month(hiredate)=08 and year(hiredate)=1980 and sal between 1501 and 2499;

Empty set (0.00 sec)

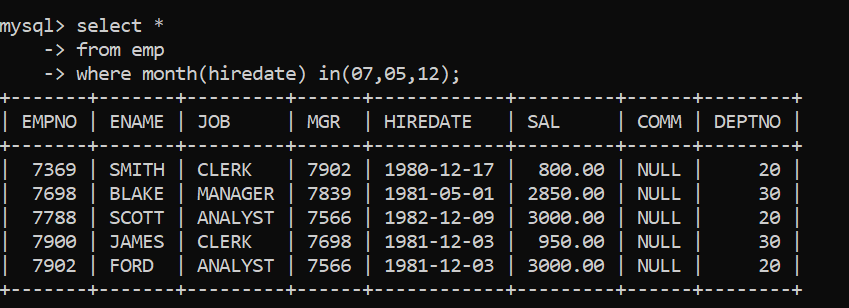
25. list all employees joined in either aug or may or dec

Ans:-

mysql> select \*

-> from emp

-> where month(hiredate) in(07,05,12);



26. display name and hiredate in dd/mm/yy format for all employees whose job is clerk and they earn some commission

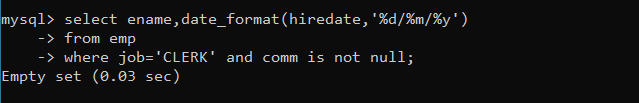
Ans:-

mysql> select ename,date\_format(hiredate,'%d/%m/%y')

-> from emp

-> where job='CLERK' and comm is not null;

Empty set (0.03 sec)



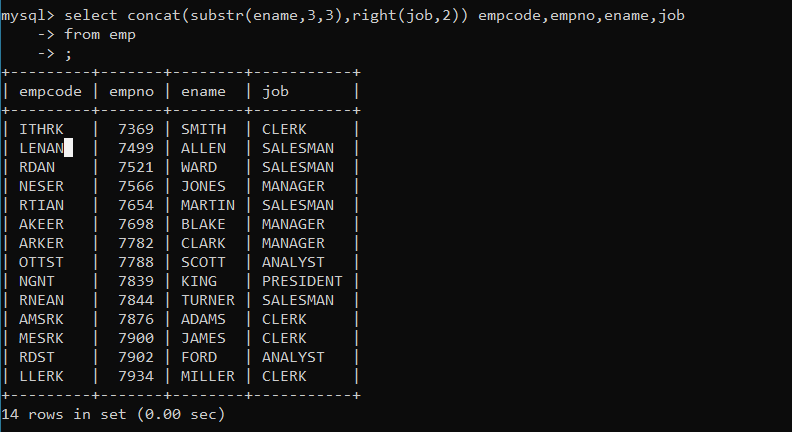
27. list empcode,empno,name and job for each employee. (note :empcode is 3 to 5 characters from name and last 2 characters of job)

Ans:-

mysql> select concat(substr(ename,3,3),right(job,2)) empcode,empno,ename,job

-> from emp

-> ;



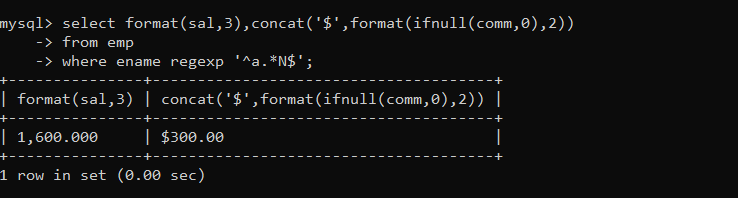
28. display thousand separator and $ symbol for commission if it is null then display it as 0 for all employees whose name starts with A and ends with N

Ans:-

mysql> select format(sal,3),concat('$',format(ifnull(comm,0),2))

-> from emp

-> where ename regexp '^a.\*N$';



29. Display empid,name,sal,comm,remark Remark should base on following conditions comm >= 600 "excellent Keep it up" if it < 600 or not null "good" otherwise "Need improvement"

Ans:-

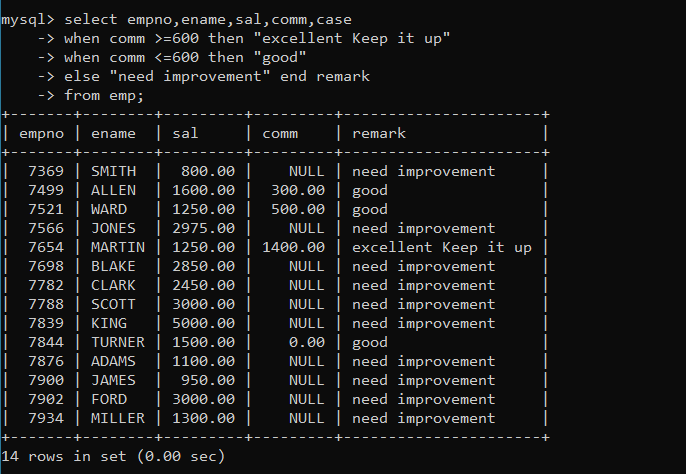
mysql> select empno,ename,sal,comm,case

-> when comm >=600 then "excellent Keep it up"

-> when comm <=600 then "good"

-> else "need improvement" end remark

-> from emp;



30. Display empid, name, deptno and department name by using following conditions.

dept 10 then "Hr"

if 20 then "Admin"

if 30 then "accounts"

otherwise purchase

Ans:-

mysql> select empno,ename,sal,comm,case

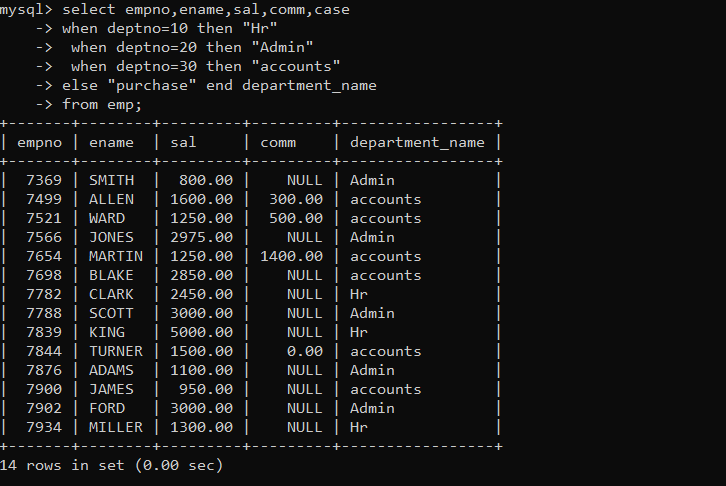
-> when deptno=10 then "Hr"

-> when deptno=20 then "Admin"

-> when deptno=30 then "accounts"

-> else "purchase" end department\_name

-> from emp;



Topic ----------------- create Table, DML , subquery and joins

31. Practice creating following tables

MySQL syntax:

create table mydept\_DBDA

(

deptid int primary key,

dname varchar(20) not null unique,

dloc varchar(20)

)

-

Oracle syntax:

create table mydept\_DBDA

(

deptid number primary key,

dname varchar2(20) not null unique,

dloc varchar2(20)

)

insert into mydept\_DBDA values(30,'Purchase','Mumbai');

MySql syntax:

create table myemployee

(

empno int primary key,

fname varchar(15) not null,

mname varchar(15),

lname varchar(15) not null,

sal float(9,2) check(sal >=1000),

doj date,

passportnum varchar(15) unique,

deptno int,

constraint fk\_deptno foreign key(deptno) references mydept\_DBDA(deptid) on

delete set null

on update cascade

)

Oracle syntax:

create table myemployee

(

empno number(5) primary key,

fname varchar2(15) not null,

mname varchar2(15),

lname varchar2(15) not null,

sal number(9,2) check(sal >=1000),

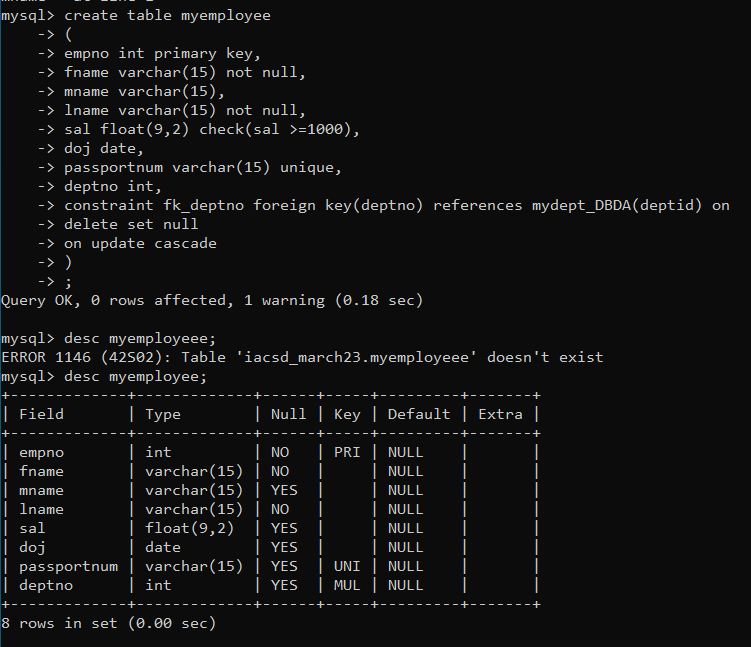
doj date default sysdate,

passportnum varchar2(15) unique,

deptno number constraint fk\_deptno references mydept\_DBDA(deptid) on delete

cascade

)



32. Create following tables Student, Course Student (sid,sname) ---------------- sid ---primary key Course(cid,cname)-------------- cid ---primary key Marks(studid,courseid,marks) Sample data for marks table studid,courseid,marks 1 1 99 1 3 98 2 1 95 2 2 97 create table marks( studid number, courseid number, marks number, constraint pk primary key(studid,courseid), constraint fk\_sid foreign key (studid) references student(sid) on delete cascade, constraint fk\_cid foreign key (courseid) references course(cid) )

Ans create table student(

-> sid int primary key,

-> sname varchar (20));

Query OK, 0 rows affected (0.05 sec)

create table Course(

-> cid int primary key,

-> cname varchar (20));

Query OK, 0 rows affected (0.04 sec)

mysql> create table Marks(

-> studid int,

-> courseid int,

-> marks int,

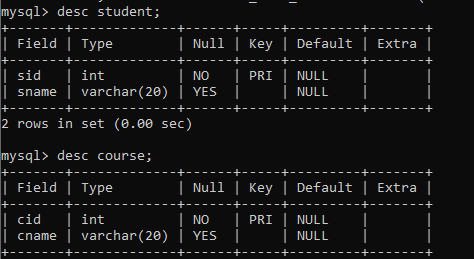
-> constraint pk primary key (studid,courseid),

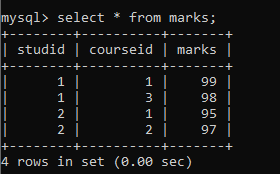
-> foreign key fk\_sid (studid) references student (sid),

-> foreign key fk\_cid (courseid) references course (cid)

-> );

Query OK, 0 rows affected (0.10 sec)





33. Create empty table emp10 with table structure same as emp table. create table emp10 as ( select \* from emp where 1=2; )

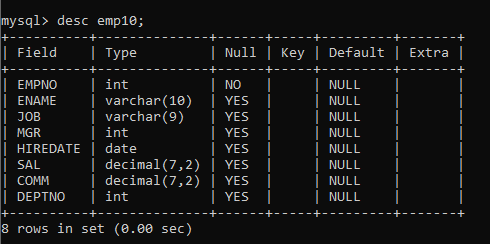
Ans :

create table emp10 as

-> (select \* from emp

-> where 1=2);

Query OK, 0 rows affected (0.09 sec)



34. Solve following using alter table add primary key constraint on emp,dept,salgrade emp ----→ empno dept---→ deptno salgrade---→ grade add foreign key constarint in emp deptno --->> dept(deptno) add new column in emp table netsal with constraint default 1000

35. Update employee sal ---- increase sal of each employee by 15 % sal +comm, change the job to manager and mgr to 7777 for all employees in deptno 10.

Ans:-

36. change job of smith to senior clerk

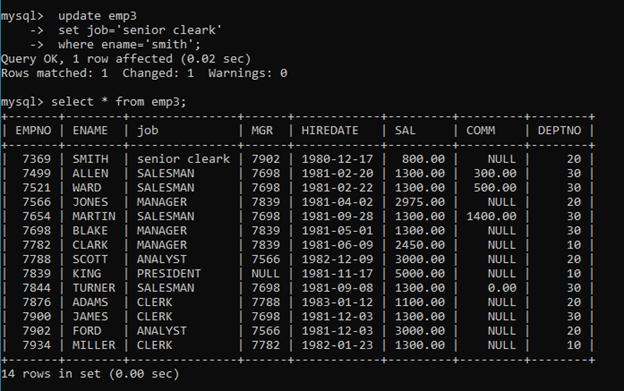
Ans :

update emp3

-> set job='senior cleark'

-> where ename='smith';

Query OK, 1 row affected (0.02 sec)



37. increase salary of all employees by 15% if they are earning some commission

38. list all employees with sal>smith's sal

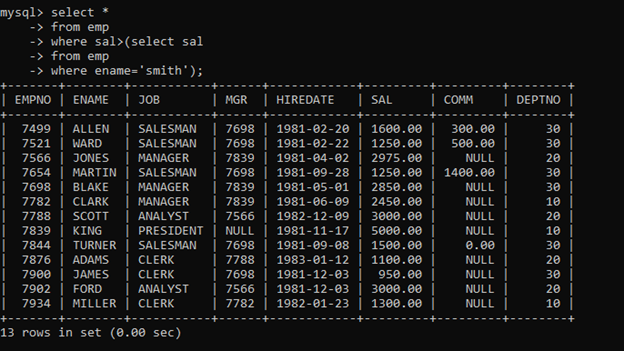
Ans : select \*

-> from emp

-> where sal>(select sal

-> from emp

-> where ename='smith');



39. list all employees who are working in smith's department

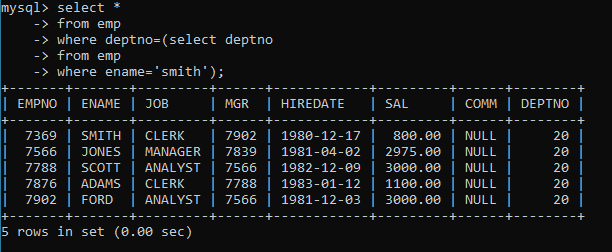
Ans : select \*

-> from emp

-> where deptno=(select deptno

-> from emp

-> where ename='smith');



40. list all employees with sal < rajan's sal and salary > revati's sal

Ans : select \*

-> from emp

-> where sal<any(select sal

-> from emp

-> where ename in ('rajan','revati'));

Empty set (0.02 sec)

41. delete all employees working in alan's department

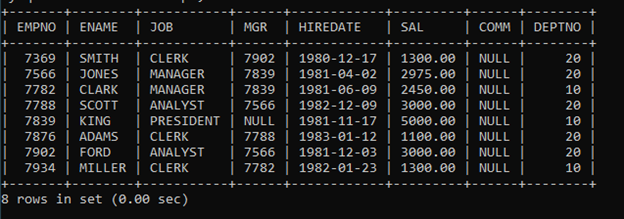
Ans delete from emp2

-> where deptno=(select e.deptno

-> from (select \*from emp2)e

-> where e.ename='allen');

Query OK, 6 rows affected (0.03 sec)



42. change salary of Alan to the salary of Miller.

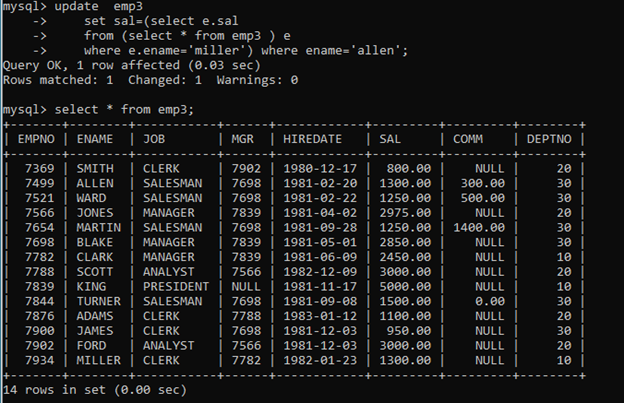
Ans : update emp3

-> set sal=(select e.sal

-> from (select \* from emp3 ) e

-> where e.ename='miller') where ename='allen';

Query OK, 1 row affected (0.03 sec)



43. change salary of all emplees who working in Ward’s department to the salary of Miller.

Ans :

mysql> update emp3

-> set sal=(select e.sal

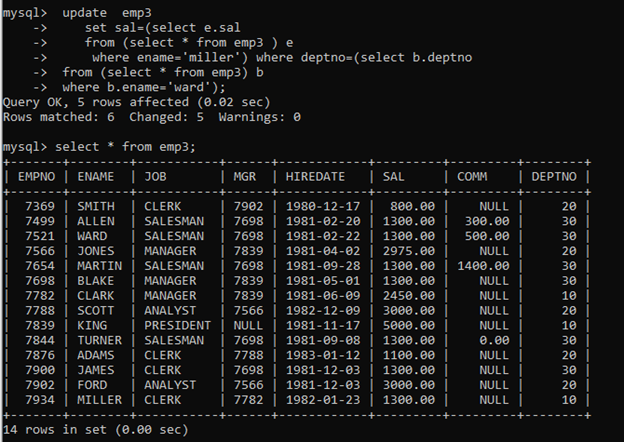
-> from (select \* from emp3 ) e

-> where ename='miller') where deptno=(select b.deptno

-> from (select \* from emp3) b

-> where b.ename='ward');

Query OK, 5 rows affected (0.02 sec)



44. list all employees with salary > either Smith's salary or alan's sal

Ans :

mysql> select \* from emp where sal>any(select sal from emp where ename='smith' or ename='allen');

+-------+--------+-----------+------+------------+---------+---------+--------+

| EMPNO | ENAME | job | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+-----------+------+------------+---------+---------+--------+

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1840.00 | 300.00 | 30 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1437.50 | 500.00 | 30 |

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1437.50 | 1400.00 | 30 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 2850.00 | NULL | 30 |

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10 |

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20 |

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1725.00 | 0.00 | 30 |

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 950.00 | NULL | 30 |

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20 |

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10 |

+-------+--------+-----------+------+------------+---------+---------+--------+

13 rows in set (0.00 sec)

45. list all employees who earn more than average sal of dept 10

Ans :

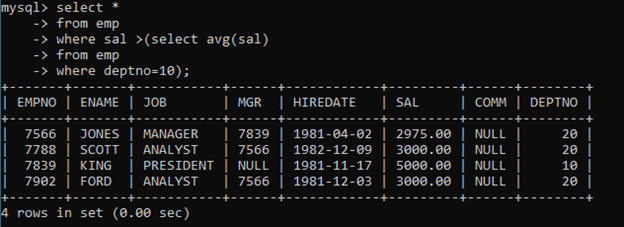
select \*

-> from emp

-> where sal >(select avg(sal)

-> from emp

-> where deptno=10);



46. list all employees who earn more than average sal of Alan's department

Ans : select \*

-> from emp

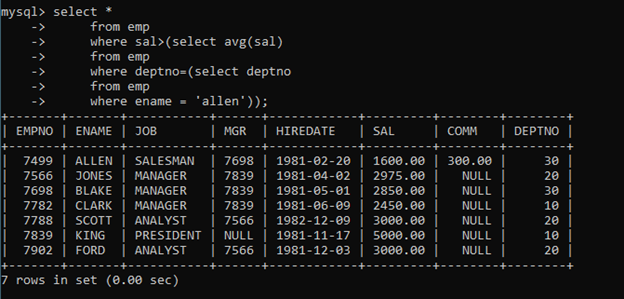
-> where sal>(select avg(sal)

-> from emp

-> where deptno=(select deptno

-> from emp

-> where ename = 'allen'));



47. list all employees who are working in purchase department

Ans :

mysql> select \* from emp where deptno=(select deptno from dept d where d.dname="SALES");

+-------+--------+----------+------+------------+---------+---------+--------+

| EMPNO | ENAME | job | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+----------+------+------------+---------+---------+--------+

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1840.00 | 300.00 | 30 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1437.50 | 500.00 | 30 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1437.50 | 1400.00 | 30 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 2850.00 | NULL | 30 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1725.00 | 0.00 | 30 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 950.00 | NULL | 30 |

+-------+--------+----------+------+------------+---------+---------+--------+

6 rows in set (0.00 sec)

mysql> select \* from emp where deptno=(select deptno from dept where dname="SALES");

+-------+--------+----------+------+------------+---------+---------+--------+

| EMPNO | ENAME | job | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+----------+------+------------+---------+---------+--------+

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1840.00 | 300.00 | 30 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1437.50 | 500.00 | 30 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1437.50 | 1400.00 | 30 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 2850.00 | NULL | 30 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1725.00 | 0.00 | 30 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 950.00 | NULL | 30 |

+-------+--------+----------+------+------------+---------+---------+--------+

6 rows in set (0.00 sec)

48. list all employees who earn more than average salary of their own department

Ans :select \*

-> from emp b

-> where sal>(select avg(sal)

-> from emp a

-> where a.deptno=b.deptno);

49. list all employees who earn sal < than their managers salary

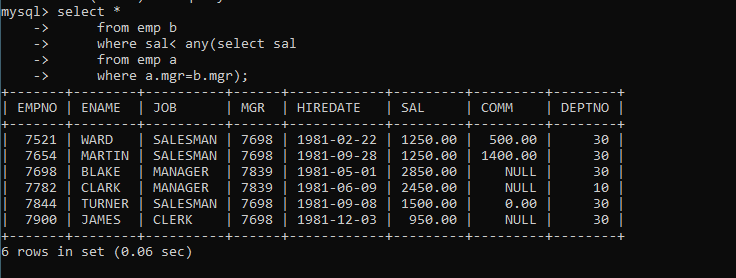
Ans :mysql> select \*

-> from emp b

-> where sal< any(select sal

-> from emp a

-> where a.mgr=b.mgr);

****

50. list all employees who are earning more than average salary of their job

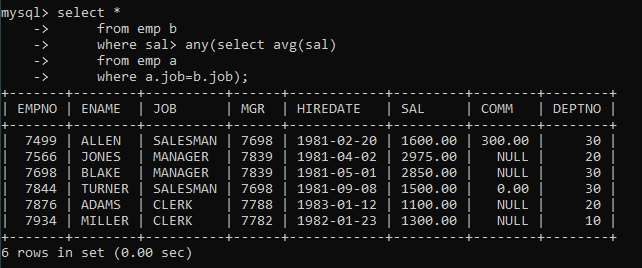
Ans :mysql> select \*

-> from emp b

-> where sal> any(select avg(sal)

-> from emp a

-> where a.job=b.job);

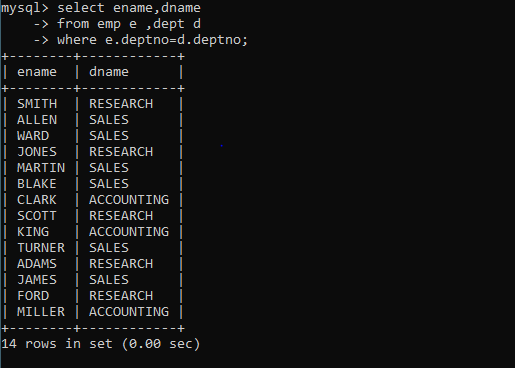


51. display employee name and department

Ans :mysql> select ename,dname

-> from emp e ,dept d

-> where e.deptno=d.deptno;

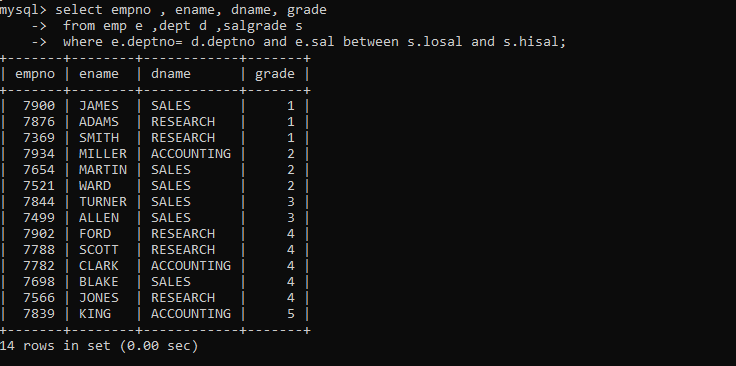


52. display empno,name,department name and grade (use emp,dept and salgrade table)

Ans:-mysql> select empno , ename, dname, grade

-> from emp e ,dept d ,salgrade s

-> where e.deptno= d.deptno and e.sal between s.losal and s.hisal;



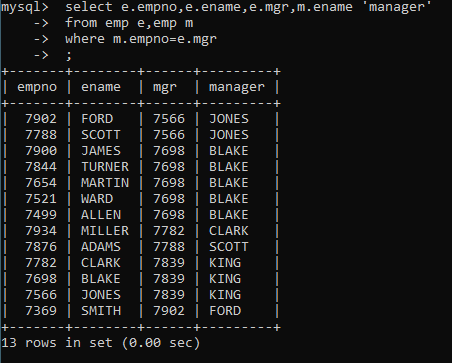
53. list all employees number,name, mgrno and manager name

Ans : mysql> select e.empno,e.ename,e.mgr,m.ename 'manager'

-> from emp e,emp m

-> where m.empno=e.mgr

-> ;



54. create following tables and solve following questions(primary keys are marked in yellow)

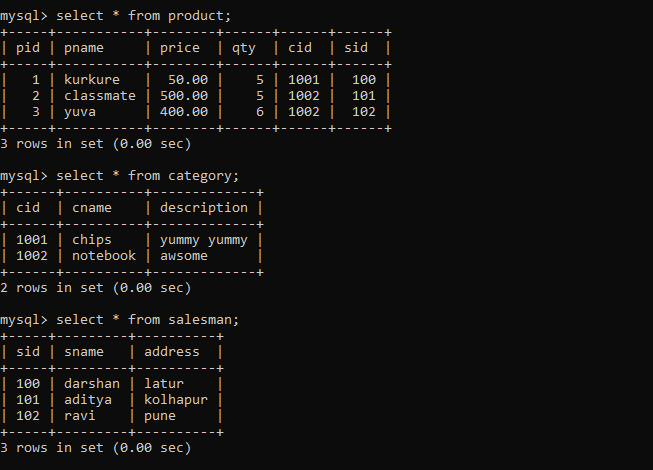
foreign keys are marked in green

product(pid,pname,price,qty,cid,sid)

salesman (sid,sname,address)

category(cid,cnam,descritpion)

1. list all product name,their category name and name of a person, who sold that product

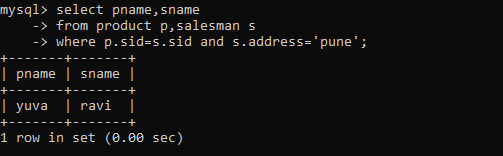


2. list all product name and salesman name for all salesman who stays in pune

mysql> select pname,sname

-> from product p,salesman s

-> where p.sid=s.sid and s.address='pune';



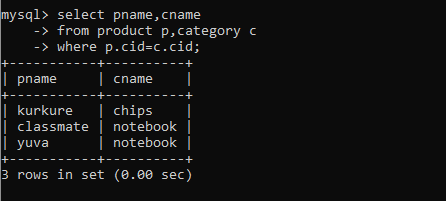
3. list all product name and category name

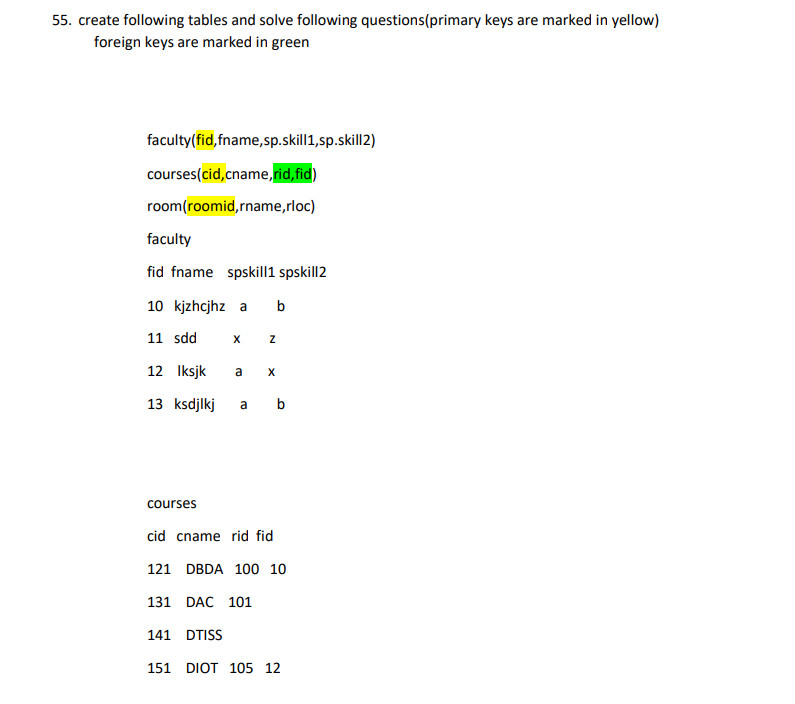
Ans:-

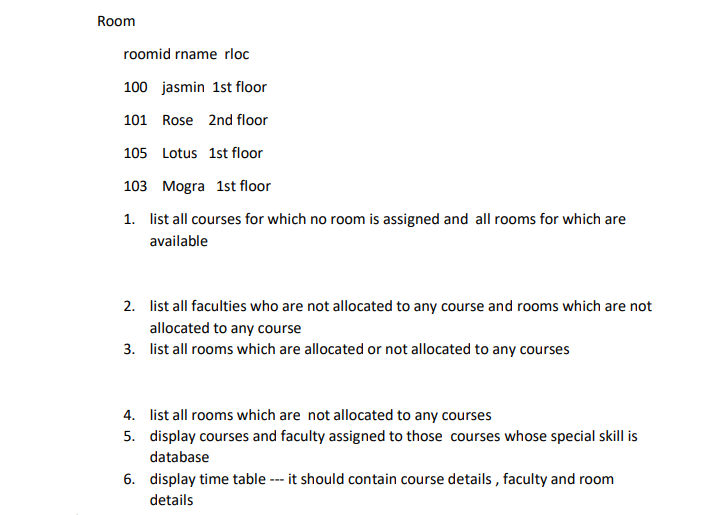
mysql> select pname,cname

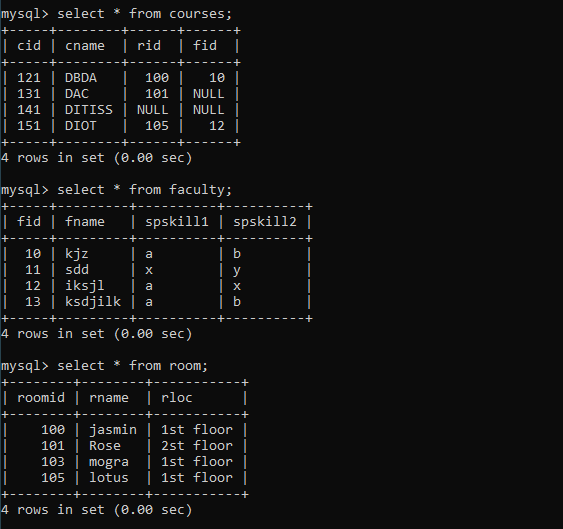
-> from product p,category c

-> where p.cid=c.cid;









1.Ans:-

1. list all courses for which no room is assigned and all rooms for which are

available

mysql> select c.cname,r.rname

-> from course c right join room r on c.rid=r.rid;

+-------+--------+

| cname | rname |

+-------+--------+

| DBDA | Jasmin |

| DAC | Rose |

| NULL | Mogra |

| DIOT | Lotus |

+-------+--------+

4 rows in set (0.00 sec)

2. list all faculties who are not allocated to any course and rooms which are not

allocated to any course

3. list all rooms which are allocated or not allocated to any courses

mysql> select r.rid,r.rname,c.cid,c.cname

-> from room r left join course c on r.rid=c.rid;

+-----+--------+------+-------+

| rid | rname | cid | cname |

+-----+--------+------+-------+

| 100 | Jasmin | 121 | DBDA |

| 101 | Rose | 131 | DAC |

| 103 | Mogra | NULL | NULL |

| 105 | Lotus | 151 | DIOT |

+-----+--------+------+-------+

4 rows in set (0.00 sec)

4. list all rooms which are not allocated to any courses

mysql> select r.rname,c.cname

-> from room r left join course c

-> on r.rid=c.rid

-> where c.cname is null;

+-------+-------+

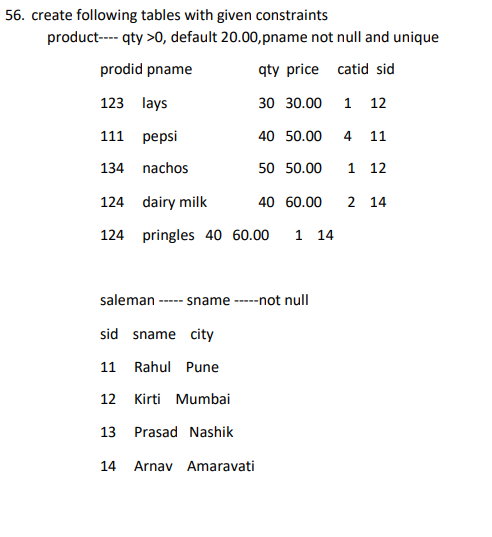
| rname | cname |

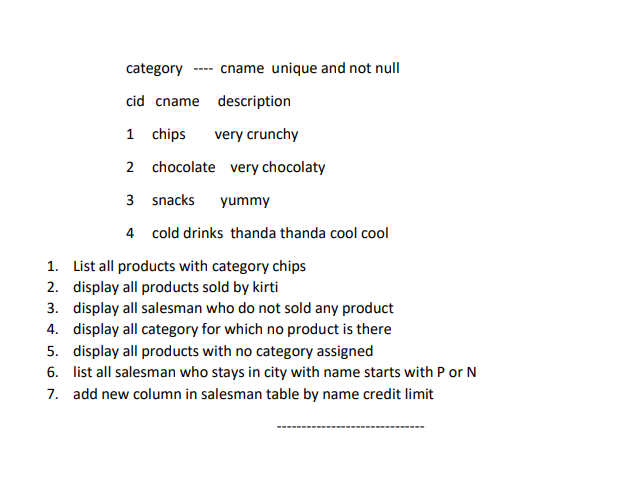
+-------+-------+

| Mogra | NULL |

+-------+-------+

1 row in set (0.00 sec)





Assignment 3 - on views and indexes Day-8

Vehicle

| Vid | Vname | Price | desc |
| --- | --- | --- | --- |
| 1 | Activa | 80000 | ksldjfjksj |
| 2 | Santro | 8,00000 | kdjfkjsd |
| 3 | Motor bike | 100000 | fdkdfj |

customer

| Custid | Cname | address |
| --- | --- | --- |
| 1 | Nilima | Pimpari |
| 2 | Ganesh | Pune |
| 3 | Pankaj | Mumbai |

salesman

| Sid | Sname | adress |
| --- | --- | --- |
| 10 | Rajesh | mumbai |
| 11 | Seema | Pune |
| 13 | Rakhi | pune |

cust-vehicle (customer is buying Many vehicle and 1 vehicle can be bought by many customers)

| Custid | Vid | Sid | Buy\_price |
| --- | --- | --- | --- |
| 1 | 1 | 10 | 75000 |
| 1 | 2 | 10 | 7,90,000 |
| 2 | 3 | 11 | 80000 |
| 3 | 3 | 11 | 75000 |
| 3 | 2 | 10 | 8,00000 |

1. create all given tables

Ans:-

mysql> create table vehicle

-> (

-> vid int primary key,

-> vname varchar(20),

-> vprice double(9,2),

-> description varchar(20)

-> );//

Query OK, 0 rows affected, 1 warning (0.10 sec)

mysql> insert into vehicle values(1,'Activa',80000,'Two')//

Query OK, 1 row affected (0.02 sec)

mysql> insert into vehicle values(2,'Santro',800000,'Three')//

Query OK, 1 row affected (0.02 sec)

mysql> insert into vehicle values(3,'Motorbike',100000,'Two')//

Query OK, 1 row affected (0.02 sec)

2. create index on vehicle table based on price

Ans:-mysql> create table customer1

-> (

-> custid int primary key,

-> cname varchar(20),

-> caddress varchar(20)

-> )//

Query OK, 0 rows affected (0.08 sec)

mysql> insert into customer1 values(1,'Nilima','Pimpari')//

Query OK, 1 row affected (0.02 sec)

mysql> insert into customer1 values(2,'Ganesh','Pune')//

Query OK, 1 row affected (0.02 sec)

mysql> insert into customer1 values(3,'Pankaj','Mumbai')//

Query OK, 1 row affected (0.02 sec)

mysql> create table salesman1

-> (

-> sid int primary key,

-> sname varchar(20),

-> saddress varchar(20)

-> )//

Query OK, 0 rows affected (0.07 sec)

mysql> insert into salesman1 values(10,'Rajesh','Mumbai')//

Query OK, 1 row affected (0.02 sec)

mysql> insert into salesman1 values(11,'Seema','Pune')//

Query OK, 1 row affected (0.02 sec)

mysql> insert into salesman1 values(13,'Rakhi','Pune')//

Query OK, 1 row affected (0.02 sec)

mysql> create table cust\_vehicle

-> (

-> custid int,

-> vid int,

-> sid int,

-> buy\_price double(9,2),

-> primary key (custid,vid),

-> foreign key (custid) references customer1(custid),

-> foreign key (vid) references vehicle (vid),

-> foreign key (sid) references salesman1 (sid)

->

-> )//

Query OK, 0 rows affected, 1 warning (0.13 sec)

mysql> insert into cust\_vehicle values(1,1,10,75000)//

Query OK, 1 row affected (0.03 sec)

mysql> insert into cust\_vehicle values(1,2,10,790000)//

Query OK, 1 row affected (0.02 sec)

mysql> insert into cust\_vehicle values(2,3,11,80000)//

Query OK, 1 row affected (0.02 sec)

mysql> insert into cust\_vehicle values(3,3,11,75000)//

Query OK, 1 row affected (0.02 sec)

mysql> insert into cust\_vehicle values(3,2,10,800000)//

Query OK, 1 row affected (0.02 sec)

mysql> create index price\_vehicle

-> on vehicle(vprice)//

Query OK, 0 rows affected (0.22 sec)

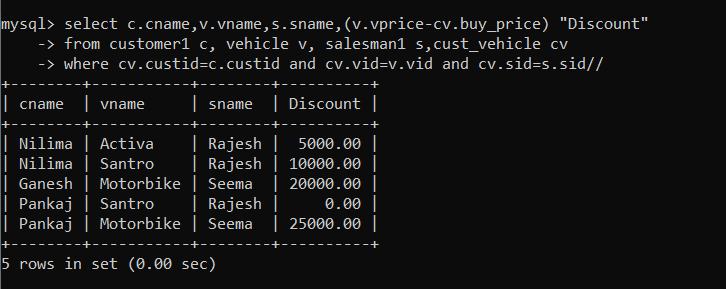
Records: 0 Duplicates: 0 Warnings: 0

3. find all customer name,vehicle name, salesman name, discount earn by all customer

mysql> select c.cname,v.vname,s.sname,(v.vprice-cv.buy\_price) "Discount"

-> from customer1 c, vehicle v, salesman1 s,cust\_vehicle cv

-> where cv.custid=c.custid and cv.vid=v.vid and cv.sid=s.sid//



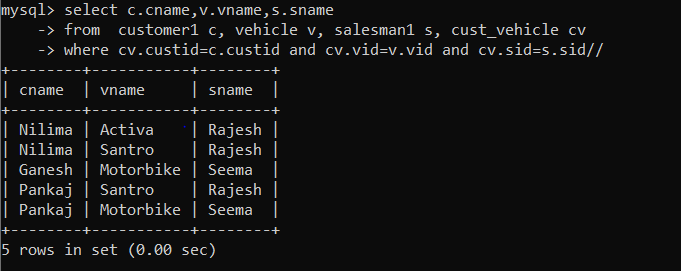
4. find all customer name,vehicle name,salesman name for all salesman who stays in pune

Ans:-

mysql> select c.cname,v.vname,s.sname

-> from customer1 c, vehicle v, salesman1 s, cust\_vehicle cv

-> where cv.custid=c.custid and cv.vid=v.vid and cv.sid=s.sid//



5. find how many customers bought motor bike

Ans:-

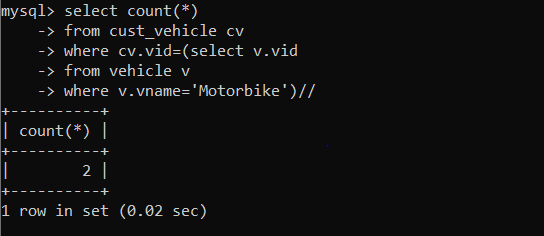
mysql> select count(\*)

-> from cust\_vehicle cv

-> where cv.vid=(select v.vid

-> from vehicle v

-> where v.vname='Motorbike')//



6. create a view find\_discount which displays output

-------to create view create view find\_discount as

select cname,vname,price,buying\_price,price-buying\_price “discount”

from customer c inner join cust\_vehicle cv on c.custid=cv.cid inner join vehicle v on v.vid=cv.vid

--------to display discount

select \* from find\_discount;

Ans:-

mysql> create view find\_discount

-> as

-> select c.cname,v.vname,v.vprice,cv.buy\_price,(v.vprice-cv.buy\_price) "Discount"

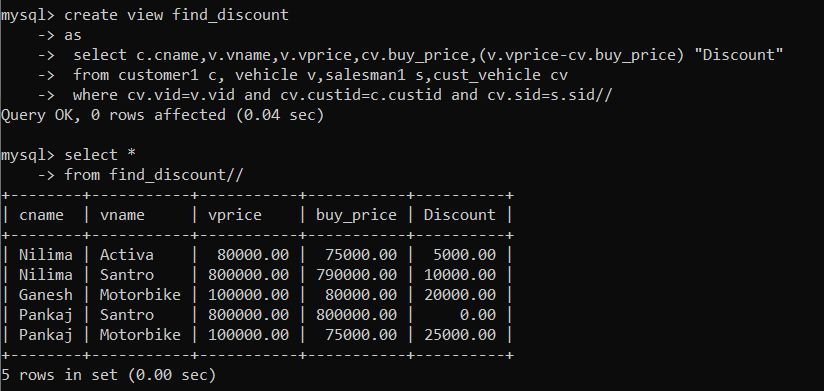
-> from customer1 c, vehicle v,salesman1 s,cust\_vehicle cv

-> where cv.vid=v.vid and cv.custid=c.custid and cv.sid=s.sid//

Query OK, 0 rows affected (0.04 sec)

mysql> select \*

-> from find\_discount//

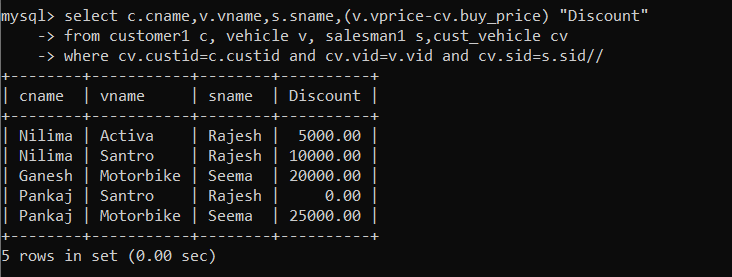


7. find all customer name, vehicle name, salesman name, discount earn by all customer

Ans:-mysql> select c.cname,v.vname,s.sname,(v.vprice-cv.buy\_price) "Discount"

-> from customer1 c, vehicle v, salesman1 s,cust\_vehicle cv

-> where cv.custid=c.custid and cv.vid=v.vid and cv.sid=s.sid//



8. create view my\_hr to display empno,ename,job,comm for all employees who earn commission

Ans:-mysql> create view my\_hr

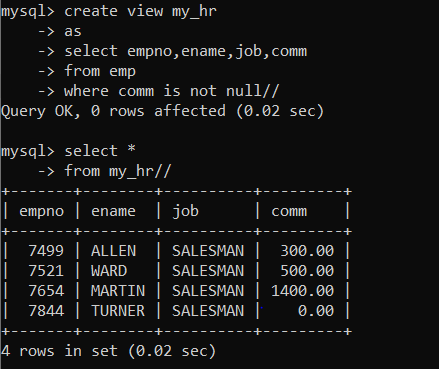
-> as

-> select empno,ename,job,comm

-> from emp

-> where comm is not null//

Query OK, 0 rows affected (0.02 sec)



9. create view mgr30 to display all employees from department 30

Ans:-

mysql> create view mgr30

-> as

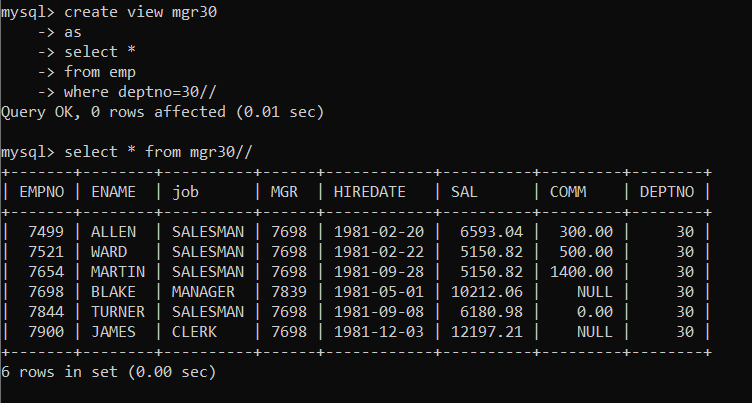
-> select \*

-> from emp

-> where deptno=30//

Query OK, 0 rows affected (0.01 sec)

mysql> select \* from mgr30//



10. insert 3 employees in view mgr30 check whether insertion is possible

Ans:-

mysql> insert into mgr30 values(1001,'akash','manager',7007,'1986-11-12',5250,750,30)//

Query OK, 1 row affected (0.02 sec)

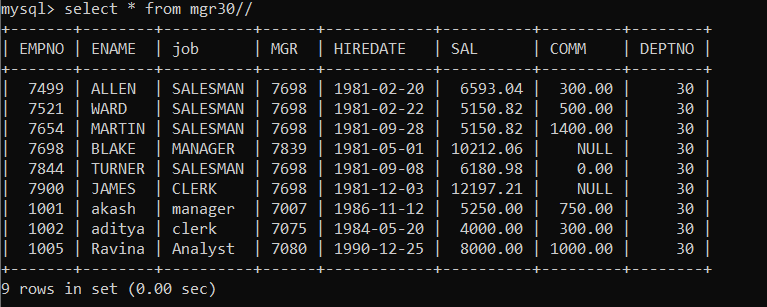
mysql> insert into mgr30 values(1002,'aditya','clerk',7075,'1984-05-20',4000,300,30)//

Query OK, 1 row affected (0.03 sec)

mysql> insert into mgr30 values(1005,'Ravina','Analyst',7080,'1990-12-25',8000,1000,30)//

Query OK, 1 row affected (0.01 sec)

mysql> select \* from mgr30//



11. insert 3 records in dept and display all records from dept

Ans:-

mysql> insert into dept values(10,'Development','San Fransisco')//

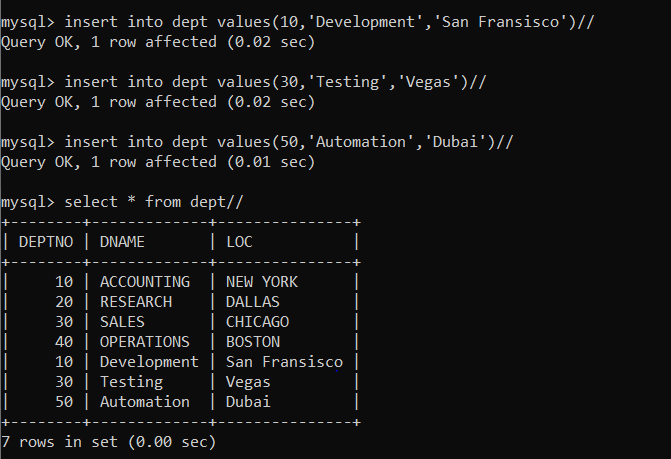
Query OK, 1 row affected (0.02 sec)

mysql> insert into dept values(30,'Testing','Vegas')//

Query OK, 1 row affected (0.02 sec)

mysql> insert into dept values(50,'Automation','Dubai')//

Query OK, 1 row affected (0.01 sec)



12. use rollback command check what happens

Ans:-

mysql> rollback//

Query OK, 0 rows affected (0.00 sec)

13. do the following

insert row in emp with empno 100 insert row in emp with empno 101 insert row in emp with empno 102 add savepoint A

insert row in emp with empno 103 insert row in emp with empno 104 insert row in emp with empno 105 add savepoint B

delete emp with empno 100 delete emp with emp no 104 rollback upto svaepoint B

check what all records will appear in employee table rollback upto A

check what all records will appear in employee table commit all changes

check what all records will appear in employee table check whether you can roll back the contents.

Ans:-

mysql> insert into emp values(100,'Shubham','Clerk',7698,'2000-11-25',20000,500,

20);

mysql> insert into emp values(101,'Akash','Clerk',7698,'2005-11-25',20000,500,20

);

mysql> insert into emp values(102,'Abhi','Clerk',7698,'2005-11-25',20000,500,20)

;

mysql> savepoint a;

mysql> insert into emp values(103,'Deepak','Manager',7698,'2020-11-25',80000,500

,50);

mysql> insert into emp values(104,'Akanksha','Manager',7698,'2020-10-25',80000,5

00,40);

mysql> insert into emp values(105,'Om','Manager',7698,'2020-10-25',80000,500,30)

;

mysql> savepoint b;

Query OK, 0 rows affected (0.00 sec)

mysql> delete from emp

-> where empno=100;

mysql> delete from emp

-> where empno=104;

mysql> rollback to b;

Query OK, 0 rows affected (0.00 sec)

mysql> select \* from emp;

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO

|

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 1300.00 | NULL | 20

|

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1300.00 | 300.00 | 30

|

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1300.00 | 500.00 | 30

|

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20

|

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1300.00 | 1400.00 | 30

|

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 1300.00 | NULL | 30

|

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10

|

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20

|

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10

|

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1300.00 | 0.00 | 30

|

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20

|

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 1300.00 | NULL | 30

|

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20

|

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10

|

| 1000 | Bhagyesh | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 30

|

| 1000 | Ashish | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 1002 | Suyash | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 100 | Shubham | Clerk | 7698 | 2000-11-25 | 20000.00 | 500.00 | 20

|

| 101 | Akash | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

| 102 | Abhi | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

| 103 | Deepak | Manager | 7698 | 2020-11-25 | 80000.00 | 500.00 | 50

|

| 104 | Akanksha | Manager | 7698 | 2020-10-25 | 80000.00 | 500.00 | 40

|

| 105 | Om | Manager | 7698 | 2020-10-25 | 80000.00 | 500.00 | 30

|

+-------+----------+-----------+------+------------+----------+---------+-------

-+

rollback upto A

check what all records will appear in employee table

mysql> rollback to a;

Query OK, 0 rows affected (0.00 sec)

mysql> select \* from emp;

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO

|

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 1300.00 | NULL | 20

|

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1300.00 | 300.00 | 30

|

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1300.00 | 500.00 | 30

|

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20

|

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1300.00 | 1400.00 | 30

|

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 1300.00 | NULL | 30

|

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10

|

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20

|

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10

|

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1300.00 | 0.00 | 30

|

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20

|

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 1300.00 | NULL | 30

|

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20

|

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10

|

| 1000 | Bhagyesh | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 30

|

| 1000 | Ashish | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 1002 | Suyash | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 100 | Shubham | Clerk | 7698 | 2000-11-25 | 20000.00 | 500.00 | 20

|

| 101 | Akash | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

| 102 | Abhi | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

+-------+----------+-----------+------+------------+----------+---------+-------

commit all changes

check what all records will appear in employee table

mysql> commit;

Query OK, 0 rows affected (0.07 sec)

mysql> select \* from emp;

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO

|

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 1300.00 | NULL | 20

|

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1300.00 | 300.00 | 30

|

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1300.00 | 500.00 | 30

|

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20

|

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1300.00 | 1400.00 | 30

|

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 1300.00 | NULL | 30

|

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10

|

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20

|

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10

|

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1300.00 | 0.00 | 30

|

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20

|

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 1300.00 | NULL | 30

|

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20

|

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10

|

| 1000 | Bhagyesh | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 30

|

| 1000 | Ashish | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 1002 | Suyash | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 100 | Shubham | Clerk | 7698 | 2000-11-25 | 20000.00 | 500.00 | 20

|

| 101 | Akash | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

| 102 | Abhi | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

+-------+----------+-----------+------+------------+----------+---------+-------

-+

check whether you can roll back the contents.

mysql> rollback;

Query OK, 0 rows affected (0.00 sec)

mysql> select \* from emp;

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO

|

+-------+----------+-----------+------+------------+----------+---------+-------

-+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 1300.00 | NULL | 20

|

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1300.00 | 300.00 | 30

|

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1300.00 | 500.00 | 30

|

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20

|

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1300.00 | 1400.00 | 30

|

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 1300.00 | NULL | 30

|

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10

|

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20

|

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10

|

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1300.00 | 0.00 | 30

|

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20

|

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 1300.00 | NULL | 30

|

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20

|

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10

|

| 1000 | Bhagyesh | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 30

|

| 1000 | Ashish | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 1002 | Suyash | Developer | 7698 | 2022-12-05 | 50000.00 | 0.00 | 20

|

| 100 | Shubham | Clerk | 7698 | 2000-11-25 | 20000.00 | 500.00 | 20

|

| 101 | Akash | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

| 102 | Abhi | Clerk | 7698 | 2005-11-25 | 20000.00 | 500.00 | 20

|

+-------+----------+-----------+------+------------+----------+---------+-------

-+

14. create a procedure getMin(deptno,minsal) to find minimum salary of given table.

Ans:-

st.emp.DEPTNO'; this is incompatible with sql\_mode=only\_full\_group\_by

mysql> delimiter //

mysql> create procedure get\_min2(pdeptno int,out pmin float)

-> begin

-> select deptno,min(sal) into pdeptno,pmin

-> from emp

-> where deptno=pdeptno

-> group by deptno;

-> end//

Query OK, 0 rows affected (0.01 sec)

mysql> delimiter;

-> ^C

mysql> call get\_min2(10,@min)//

Query OK, 1 row affected (0.00 sec)

mysql> select @min//

+------+

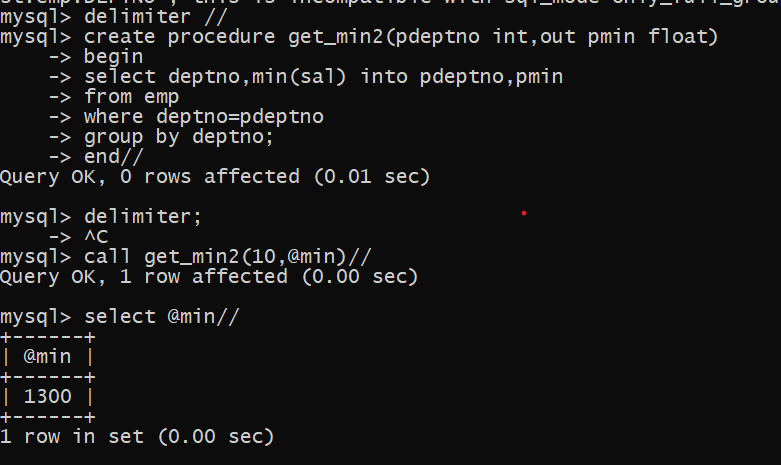
| @min |

+------+

| 1300 |

+------+

1 row in set (0.00 sec)



Assignment 4 –on PLSQL –day9

Solve the following

1. 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

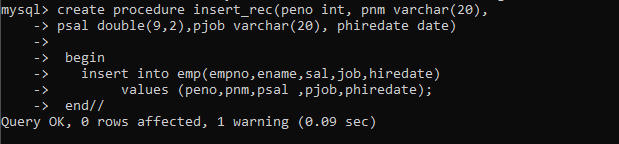
create procedure insert\_rec(peno int,pnm varchar(20),psal decimal(9,2),pjob varchar(20),phiredate date)

begin

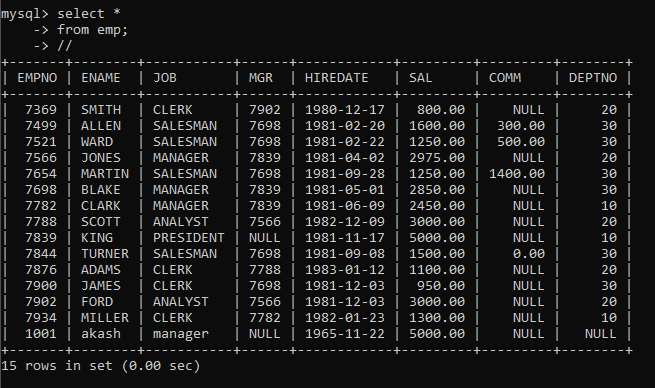
insert into emp(empno,ename,sal,job,hiredate) values(peno,pnm,psal,pjob,phiredate)

end//

Ans:-



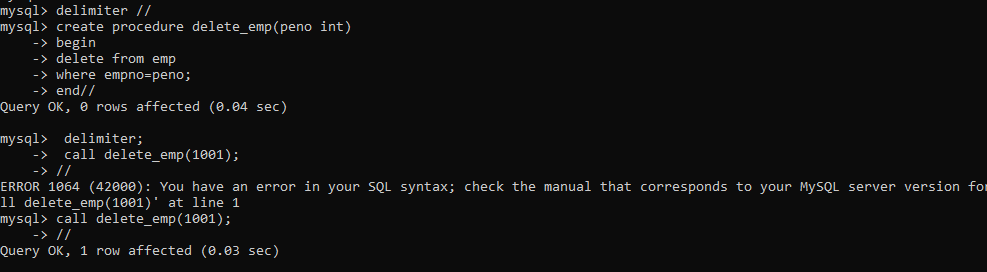




2. write a procedure to delete record from employee table. the procedure should accept empno as input parameter.

write delete statement inside procedure delete\_emp to delete one record from emp table

Ans:-



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

> given salary. pass salary as a parameter to procedure

Ans:-

mysql> delimiter //

mysql> create procedure dis\_emp5(psal double(9,2))

-> begin

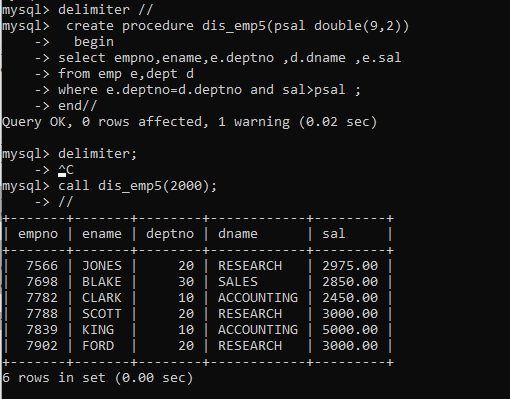
-> select empno,ename,e.deptno ,d.dname ,e.sal

-> from emp e,dept d

-> where e.deptno=d.deptno and sal>psal ;

-> end//

Query OK, 0 rows affected, 1 warning (0.02 sec)



4. write a procedure to find min,max,avg of salary and number of employees in the given deptno.

deptno --à in parameter

min,max,avg and count ---à out type parameter

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

Ans:-

mysql> delimiter //

mysql> create procedure dis\_val1( in pdeptno int ,out pmin float,out pmax float,

-> out pavg float ,out pcount int )

-> begin

-> select deptno, min(sal),max(sal),avg(sal),count(sal) into pdeptno,pmin,pmax,pavg,pcount

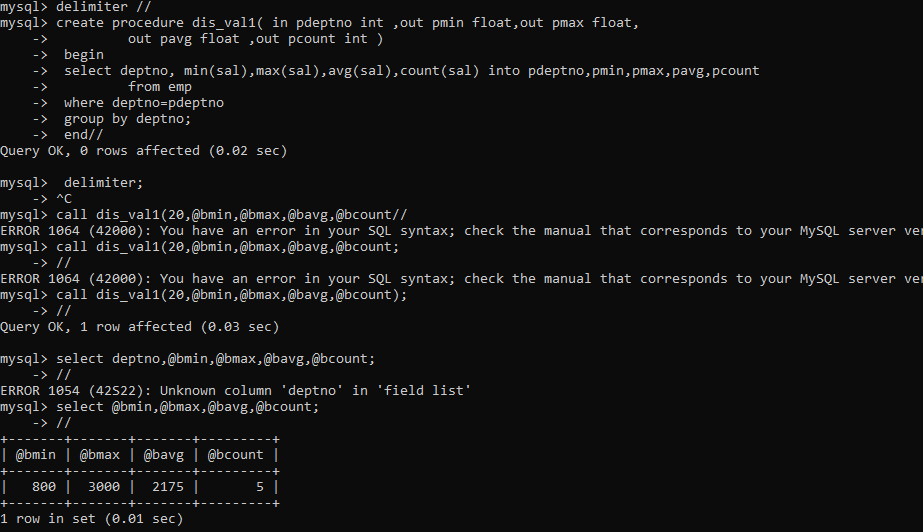
-> from emp

-> where deptno=pdeptno

-> group by deptno;

-> end//

Query OK, 0 rows affected (0.02 sec)



5. write a procedure to display all pid,pname,cid,cname and salesman name(use product,category and salesman table)

Ans:-

mysql> delimiter //

mysql> create procedure dis\_pro()

-> begin

-> select p.pid,p.pname,p.cid,c.cname,s.sname

-> from product p,category c,salesman s

-> where p.cid=c.cid and p.sid=s.sid;

-> end//

Query OK, 0 rows affected (0.02 sec)

mysql> call dis\_pro()//

+-----+---------+------+------------+----------+

| pid | pname | cid | cname | sname |

+-----+---------+------+------------+----------+

| 10 | lays | 1 | chips | joy |

| 14 | pepsi | 2 | cold drink | shah |

| 15 | miranda | 2 | cold drink | sam |

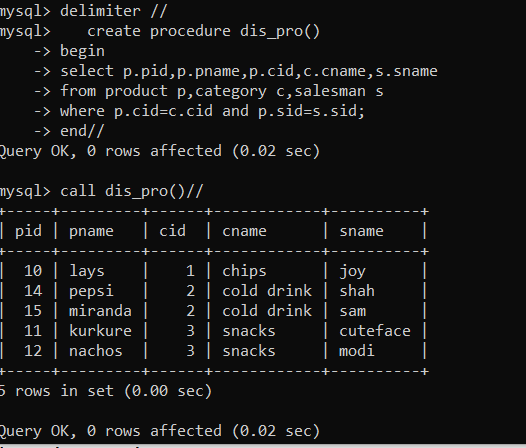
| 11 | kurkure | 3 | snacks | cuteface |

| 12 | nachos | 3 | snacks | modi |

+-----+---------+------+------------+----------+

5 rows in set (0.00 sec)

Query OK, 0 rows affected (0.02 sec)



6. write a procedure to display all vehicles bought by a customer. pass cutome name as a parameter.(use vehicle,salesman,custome and relation table)

7. 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.

Ans:-

mysql> delimiter //

mysql> create procedure dis\_info()

-> begin

-> select e.empno,e.ename,e.job,e.sal,e.deptno,case when

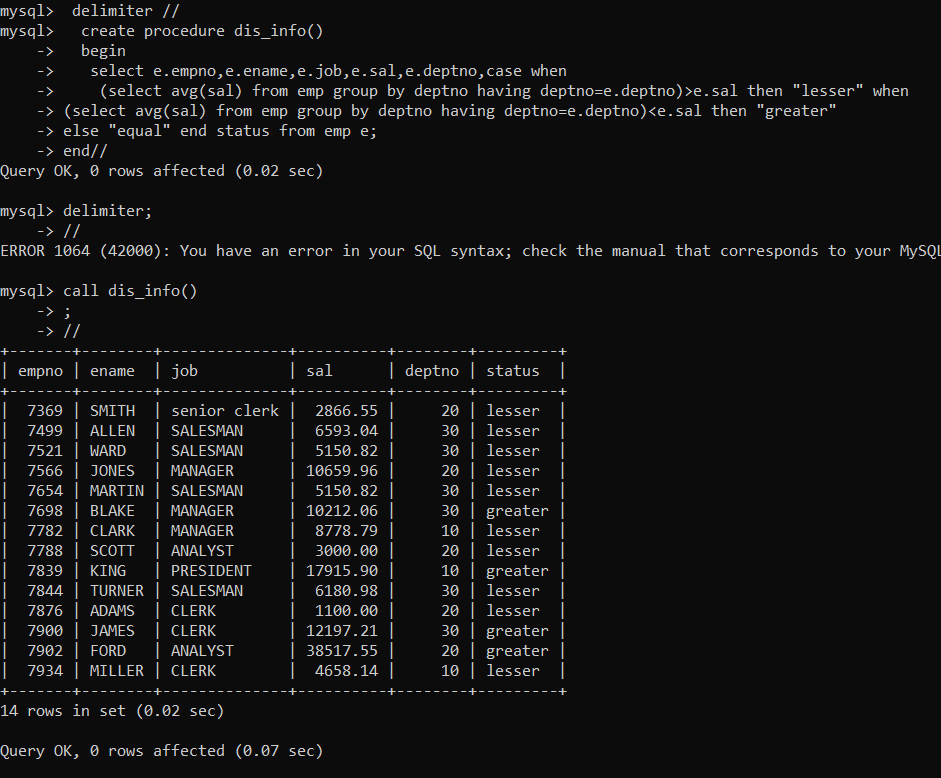
-> (select avg(sal) from emp group by deptno having deptno=e.deptno)>e.sal then "lesser" when

-> (select avg(sal) from emp group by deptno having deptno=e.deptno)<e.sal then "greater"

-> else "equal" end status from emp e;

-> end//

Query OK, 0 rows affected (0.02 sec)



8. 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

Ans:

delimiter //

create function retnexp2(hdate date) returns int

begin

declare pexp int default 0;

set pexp=(floor(datediff(curdate(),hdate)/365));

return pexp;

end//

delimiter //

create procedure updtsal()

begin

declare finished int default 0;

declare vhdate date;

declare updsal\_cur cursor for select hiredate from emp;

declare continue handler for not found set finished = 1;

open updsal\_cur;

l1: loop

fetch updsal\_cur into vhdate;

if finished = 1 then

leave l1;

end if;

if retnexp2(vhdate)>40 and retnexp2(vhdate)<=42 then

update emp set sal=sal\*1.2 where hiredate=vhdate;

elseif retnexp2(vhdate)>42 then

update emp set sal=sal\*1.25 where hiredate=vhdate;

end if;

end loop;

close updsal\_cur;

end //

delimiter ;

mysql> create procedure updtsal()

-> begin

-> declare finished int default 0;

-> declare vhdate date;

-> declare updsal\_cur cursor for select hiredate from emp;

-> declare continue handler for not found set finished = 1;

->

-> open updsal\_cur;

->

-> l1: loop

-> fetch updsal\_cur into vhdate;

->

-> if finished = 1 then

-> leave l1;

-> end if;

->

->

-> if retnexp2(vhdate)>40 and retnexp2(vhdate)<=42 then

-> update emp set sal=sal\*1.2 where hiredate=vhdate;

->

-> elseif retnexp2(vhdate)>42 then

-> update emp set sal=sal\*1.25 where hiredate=vhdate;

->

->

-> end if;

-> end loop;

->

-> close updsal\_cur;

-> end //

Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;

mysql> select \* from emp;

+-------+----------+-----------+------+------------+----------+---------+--------+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+----------+-----------+------+------------+----------+---------+--------+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 800.00 | NULL | 20 |

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1600.00 | 300.00 | 30 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1250.00 | 500.00 | 30 |

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1250.00 | 1400.00 | 30 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 2850.00 | NULL | 30 |

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10 |

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20 |

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1500.00 | 0.00 | 30 |

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 950.00 | NULL | 30 |

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20 |

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10 |

| 4561 | sour\_abh | president | NULL | 1981-04-02 | 99999.00 | 1000.00 | 50 |

+-------+----------+-----------+------+------------+----------+---------+--------+

15 rows in set (0.00 sec)

mysql> call updtsal;

ERROR 1264 (22003): Out of range value for column 'SAL' at row 15

mysql> delete from emp where ename='sour\_abh';

Query OK, 1 row affected (0.04 sec)

mysql> call updtsal;

Query OK, 0 rows affected (0.39 sec)

mysql> select \* from emp;

+-------+--------+-----------+------+------------+---------+---------+--------+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+-----------+------+------------+---------+---------+--------+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 1152.00 | NULL | 20 |

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 2304.00 | 300.00 | 30 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1800.00 | 500.00 | 30 |

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 3570.00 | NULL | 20 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1500.00 | 1400.00 | 30 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 3420.00 | NULL | 30 |

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2940.00 | NULL | 10 |

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20 |

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 6000.00 | NULL | 10 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1800.00 | 0.00 | 30 |

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 1368.00 | NULL | 30 |

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 4320.00 | NULL | 20 |

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1560.00 | NULL | 10 |

+-------+--------+-----------+------+------------+---------+---------+--------+

14 rows in set (0.00 sec)

**9. 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));**

**Ans:**

delimiter //

create function retnexp2(hdate date) returns int

begin

declare pexp int default 0;

set pexp=(floor(datediff(curdate(),hdate)/365));

return pexp;

end//

delimiter ;

mysql> create table emp\_allowance((select \*,floor(datediff(curdate(),hiredate)/365) exp from emp));

Query OK, 15 rows affected (0.25 sec)

Records: 15 Duplicates: 0 Warnings: 0

mysql> alter table emp\_allowance add column allowance decimal(9,2);

Query OK, 0 rows affected (0.10 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> select \* from emp\_allowance;

+-------+----------+-----------+------+------------+----------+---------+--------+------+-----------+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO | exp | allowance |

+-------+----------+-----------+------+------------+----------+---------+--------+------+-----------+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 800.00 | NULL | 20 | 42 | NULL |

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1600.00 | 300.00 | 30 | 42 | NULL |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1250.00 | 500.00 | 30 | 42 | NULL |

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20 | 42 | NULL |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1250.00 | 1400.00 | 30 | 41 | NULL |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 2850.00 | NULL | 30 | 41 | NULL |

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10 | 41 | NULL |

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20 | 40 | NULL |

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10 | 41 | NULL |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1500.00 | 0.00 | 30 | 41 | NULL |

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20 | 40 | NULL |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 950.00 | NULL | 30 | 41 | NULL |

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20 | 41 | NULL |

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10 | 41 | NULL |

| 4561 | sour\_abh | president | NULL | 1981-04-02 | 99999.00 | 1000.00 | 50 | 42 | NULL |

+-------+----------+-----------+------+------------+----------+---------+--------+------+-----------+

15 rows in set (0.00 sec)

mysql>

mysql> create procedure updtallwnc11()

-> begin

-> declare finished int default 0;

-> declare vhdate date;

-> declare updall\_cur cursor for select hiredate from emp\_allowance;

-> declare continue handler for not found set finished = 1;

->

-> open updall\_cur;

->

-> l1: loop

-> fetch updall\_cur into vhdate;

->

-> if finished = 1 then

-> leave l1;

-> end if;

->

-> update emp\_allowance

-> set allowance = retnexp2(vhdate)\*3000

-> where hiredate = vhdate;

-> end loop;

->

-> close updall\_cur;

-> end //

Query OK, 0 rows affected (0.04 sec)

mysql> delimiter ;

mysql> call updtallwnc11;

Query OK, 0 rows affected (0.23 sec)

mysql> select \* from emp\_allowance;

+-------+----------+-----------+------+------------+----------+---------+--------+------+-----------+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO | exp | allowance |

+-------+----------+-----------+------+------------+----------+---------+--------+------+-----------+

| 7369 | SMITH | CLERK | 7902 | 1980-12-17 | 800.00 | NULL | 20 | 42 | 126000.00 |

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 1600.00 | 300.00 | 30 | 42 | 126000.00 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1250.00 | 500.00 | 30 | 42 | 126000.00 |

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 2975.00 | NULL | 20 | 42 | 126000.00 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1250.00 | 1400.00 | 30 | 41 | 123000.00 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 2850.00 | NULL | 30 | 41 | 123000.00 |

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 2450.00 | NULL | 10 | 41 | 123000.00 |

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20 | 40 | 120000.00 |

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 5000.00 | NULL | 10 | 41 | 123000.00 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1500.00 | 0.00 | 30 | 41 | 123000.00 |

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20 | 40 | 120000.00 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 950.00 | NULL | 30 | 41 | 123000.00 |

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 3000.00 | NULL | 20 | 41 | 123000.00 |

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1300.00 | NULL | 10 | 41 | 123000.00 |

| 4561 | sour\_abh | president | NULL | 1981-04-02 | 99999.00 | 1000.00 | 50 | 42 | 126000.00 |

+-------+----------+-----------+------+------------+----------+---------+--------+------+-----------+

15 rows in set (0.00 sec)

10. 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

Ans:

delimiter //

create function retnctc(psal double(9,2),pdate date) returns double(9,2)

begin

declare pctc double(9,2);

declare spallwnc int default 0;

if retnexp2(pdate)<=40 then set spallwnc=psal\*1.1;

elseif retnexp2(pdate)<=41 then set spallwnc=psal\*1.2;

else set spallwnc=psal\*1.3;

end if;

set pctc=psal\*(1+(0.15+0.2+0.08))+spallwnc;

return pctc;

end//

delimiter ;

mysql> select retnctc(5000,'1980-12-03');

+----------------------------+

| retnctc(5000,'1980-12-03') |

+----------------------------+

| 13650.00 |

+----------------------------+

1 row in set (0.02 sec)

11. Write query to display empno,ename,sal,cost to company for all employees(note: use function written in question 10)

Ans:

mysql> select empno,ename,sal,retnctc(sal,hiredate)'cost to company' from emp;

+-------+--------+---------+-----------------+

| empno | ename | sal | cost to company |

+-------+--------+---------+-----------------+

| 7369 | SMITH | 800.00 | 2184.00 |

| 7499 | ALLEN | 1840.00 | 5023.20 |

| 7521 | WARD | 1437.50 | 3924.62 |

| 7566 | JONES | 2975.00 | 8122.25 |

| 7654 | MARTIN | 1437.50 | 3780.62 |

| 7698 | BLAKE | 2850.00 | 7495.50 |

| 7782 | CLARK | 2450.00 | 6443.50 |

| 7788 | SCOTT | 3000.00 | 7590.00 |

| 7839 | KING | 5000.00 | 13150.00 |

| 7844 | TURNER | 1725.00 | 4536.75 |

| 7876 | ADAMS | 1100.00 | 2783.00 |

| 7900 | JAMES | 950.00 | 2498.50 |

| 7902 | FORD | 3000.00 | 7890.00 |

| 7934 | MILLER | 1300.00 | 3419.00 |

+-------+--------+---------+-----------------+

14 rows in set (0.00 sec)

Q2. Write trigger

1. Write a tigger to store the old salary details in Emp \_Back (Emp \_Back has the same structure as emp table without any constraint) table.

(note :create emp\_back table before writing trigger)

----- to create emp\_back table

create table emp\_back( empno int,

ename varchar(20), oldsal decimal(9,2), newsal decimal(9,2)

)

(note :

execute procedure written in Q8 and

check the entries in EMP\_back table after execution of the procedure)

mysql> create table emp\_back

-> (

-> empno int,

-> ename varchar(20),

-> oldsal double(9,2),

-> newsal double(9,2),

-> action varchar(20),

-> user varchar(20),

-> actdate date

-> )//

Query OK, 0 rows affected, 2 warnings (0.10 sec)

mysql> delimiter //

mysql> create trigger details after update on emp

-> for each row

-> begin

-> insert into emp\_back values (old.empno,old.ename,old.sal,new.sal,'update',user(),curdate());

-> end//

Query OK, 0 rows affected (0.03 sec)

mysql>call updtsal;

Query OK, 0 rows affected (0.01 sec)

mysql> select \* from emp;

+-------+--------+--------------+------+------------+----------+---------+--------+

| EMPNO | ENAME | job | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+--------------+------+------------+----------+---------+--------+

| 7369 | SMITH | senior clerk | 7902 | 1980-12-17 | 2866.55 | NULL | 20 |

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 6593.04 | 300.00 | 30 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 5150.82 | 500.00 | 30 |

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 10659.96 | NULL | 20 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 5150.82 | 1400.00 | 30 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 10212.06 | NULL | 30 |

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 8778.79 | NULL | 10 |

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3000.00 | NULL | 20 |

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 17915.90 | NULL | 10 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 6180.98 | 0.00 | 30 |

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1100.00 | NULL | 20 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 12197.21 | NULL | 30 |

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 38517.55 | NULL | 20 |

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 4658.14 | NULL | 10 |

+-------+--------+--------------+------+------------+----------+---------+--------+

14 rows in set (0.00 sec)

mysql> select \* from emp\_back;

+-------+--------+----------+----------+--------+----------------+------------+

| empno | ename | oldsal | newsal | action | user | actdate |

+-------+--------+----------+----------+--------+----------------+------------+

| 7369 | SMITH | 2388.79 | 2866.55 | update | root@localhost | 2023-04-14 |

| 7499 | ALLEN | 5494.20 | 6593.04 | update | root@localhost | 2023-04-14 |

| 7521 | WARD | 4292.35 | 5150.82 | update | root@localhost | 2023-04-14 |

| 7566 | JONES | 8883.30 | 10659.96 | update | root@localhost | 2023-04-14 |

| 7654 | MARTIN | 4292.35 | 5150.82 | update | root@localhost | 2023-04-14 |

| 7698 | BLAKE | 8510.05 | 10212.06 | update | root@localhost | 2023-04-14 |

| 7782 | CLARK | 7315.66 | 8778.79 | update | root@localhost | 2023-04-14 |

| 7839 | KING | 14929.92 | 17915.90 | update | root@localhost | 2023-04-14 |

| 7844 | TURNER | 5150.82 | 6180.98 | update | root@localhost | 2023-04-14 |

| 7900 | JAMES | 8470.28 | 10164.34 | update | root@localhost | 2023-04-14 |

| 7902 | FORD | 26748.30 | 32097.96 | update | root@localhost | 2023-04-14 |

| 7900 | JAMES | 10164.34 | 12197.21 | update | root@localhost | 2023-04-14 |

| 7902 | FORD | 32097.96 | 38517.55 | update | root@localhost | 2023-04-14 |

| 7934 | MILLER | 3881.78 | 4658.14 | update | root@localhost | 2023-04-14 |

+-------+--------+----------+----------+--------+----------------+------------+

14 rows in set (0.00 sec)

2. Write a trigger which add entry in audit table when user tries to insert or delete records in employee table store empno,name,username and date on which operation performed and which action is done insert or delete. in emp\_audit table. create table before writing trigger.

create table empaudit( empno int;

ename varchar(20), username varchar(20); chdate date;

action varchar(20)

);

Ans:

mysql> create trigger trg before insert on emp

-> for each row

-> begin

-> insert into empaudit values(new.empno,new.ename,user(),curdate(),'insert');

-> end//

Query OK, 0 rows affected (0.03 sec)

mysql> create trigger trg1 before delete on emp

-> for each row

-> begin

-> insert into empaudit values(old.empno,old.ename,user(),curdate(),'delete');

-> end//

Query OK, 0 rows affected (0.03 sec)

mysql> delete from emp

-> where ename="Smith";

-> //

Query OK, 1 row affected (0.04 sec)

mysql> select \* from empaudit//

+-------+-------+----------------+------------+--------+

| empno | ename | username | chdate | action |

+-------+-------+----------------+------------+--------+

| 7369 | SMITH | root@localhost | 2023-04-22 | delete |

+-------+-------+----------------+------------+--------+

1 row in set (0.00 sec)

mysql> insert into emp (empno,ename) values(7369,"Smith");

-> //

Query OK, 1 row affected (0.02 sec)

mysql> select \* from empaudit//

+-------+-------+----------------+------------+--------+

| empno | ename | username | chdate | action |

+-------+-------+----------------+------------+--------+

| 7369 | SMITH | root@localhost | 2023-04-22 | delete |

| 7369 | Smith | root@localhost | 2023-04-22 | insert |

+-------+-------+----------------+------------+--------+

2 rows in set (0.00 sec)

mysql> select \* from emp//

+-------+--------+-----------+------+------------+---------+---------+--------+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+-----------+------+------------+---------+---------+--------+

| 7499 | ALLEN | SALESMAN | 7698 | 1981-02-20 | 2000.00 | 300.00 | 30 |

| 7521 | WARD | SALESMAN | 7698 | 1981-02-22 | 1562.50 | 500.00 | 30 |

| 7566 | JONES | MANAGER | 7839 | 1981-04-02 | 3718.75 | NULL | 20 |

| 7654 | MARTIN | SALESMAN | 7698 | 1981-09-28 | 1562.50 | 1400.00 | 30 |

| 7698 | BLAKE | MANAGER | 7839 | 1981-05-01 | 3562.50 | NULL | 30 |

| 7782 | CLARK | MANAGER | 7839 | 1981-06-09 | 3062.50 | NULL | 10 |

| 7788 | SCOTT | ANALYST | 7566 | 1982-12-09 | 3750.00 | NULL | 20 |

| 7839 | KING | PRESIDENT | NULL | 1981-11-17 | 6250.00 | NULL | 10 |

| 7844 | TURNER | SALESMAN | 7698 | 1981-09-08 | 1875.00 | 0.00 | 30 |

| 7876 | ADAMS | CLERK | 7788 | 1983-01-12 | 1375.00 | NULL | 20 |

| 7900 | JAMES | CLERK | 7698 | 1981-12-03 | 1484.38 | NULL | 30 |

| 7902 | FORD | ANALYST | 7566 | 1981-12-03 | 4687.50 | NULL | 20 |

| 7934 | MILLER | CLERK | 7782 | 1982-01-23 | 1625.00 | NULL | 10 |

| 7369 | Smith | NULL | NULL | NULL | NULL | NULL | NULL |

+-------+--------+-----------+------+------------+---------+---------+--------+

14 rows in set (0.00 sec)

3. Create table vehicle\_history. Write a trigger to store old vehicleprice and new vehicle price in history table before you update price in vehicle table

(note: use vehicle table).

create table vehicle\_history( vno int,

vname varchar(20), oldprice decimal(9,2), newprice decimal(9,2), chdate date, username varchar(20)

);

Ans:

create trigger veh before update on vehicle

for each row

begin

insert into vehicle\_history1 values(vid,vname,old.vprice,new.vprice,curdate(),user(),'update ');

end//

mysql> update vehicle

-> set vprice=100000

-> where vid=1;

-> //

Query OK, 1 row affected (0.03 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql> select \* from vehicle\_history1//

+------+-------+-----------+-----------+------------+----------------+--------+

| vid | vname | oldvprice | newvprice | chdate | username | action |

+------+-------+-----------+-----------+------------+----------------+--------+

| NULL | NULL | 80000.00 | 100000.00 | 2023-04-22 | root@localhost | update |

+------+-------+-----------+-----------+------------+----------------+--------+

1 row in set (0.00 sec)

Day 13 (Mongodb)

Assignment 1:-

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

Ans:-

db.samplerestaurant.find().pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a8"),

"address" : {

"building" : "351",

"coord" : [

-73.98513559999999,

40.7676919

],

"street" : "West 57 Street",

"zipcode" : "10019"

},

"borough" : "Manhattan",

"cuisine" : "Irish",

"grades" : [

{

"date" : ISODate("2014-09-06T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-07-22T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-07-31T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-12-29T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Dj Reynolds Pub And Restaurant",

"restaurant\_id" : "30191841"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a9"),

"address" : {

"building" : "469",

"coord" : [

-73.961704,

40.662942

],

"street" : "Flatbush Avenue",

"zipcode" : "11225"

},

"borough" : "Brooklyn",

"cuisine" : "Hamburgers",

"grades" : [

{

"date" : ISODate("2014-12-30T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2014-07-01T00:00:00Z"),

"grade" : "B",

"score" : 23

},

{

"date" : ISODate("2013-04-30T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-05-08T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Wendy'S",

"restaurant\_id" : "30112340"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3aa"),

"address" : {

"building" : "7114",

"coord" : [

-73.9068506,

40.6199034

],

"street" : "Avenue U",

"zipcode" : "11234"

},

"borough" : "Brooklyn",

"cuisine" : "Delicatessen",

"grades" : [

{

"date" : ISODate("2014-05-29T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2014-01-14T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-08-03T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2012-07-18T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-03-09T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2011-10-14T00:00:00Z"),

"grade" : "A",

"score" : 9

}

],

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ab"),

"address" : {

"building" : "6409",

"coord" : [

-74.00528899999999,

40.628886

],

"street" : "11 Avenue",

"zipcode" : "11219"

},

"borough" : "Brooklyn",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-07-18T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-07-30T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-02-13T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-08-16T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2011-08-17T00:00:00Z"),

"grade" : "A",

"score" : 11

}

],

"name" : "Regina Caterers",

"restaurant\_id" : "40356649"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ac"),

"address" : {

"building" : "7715",

"coord" : [

-73.9973325,

40.61174889999999

],

"street" : "18 Avenue",

"zipcode" : "11214"

},

"borough" : "Brooklyn",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-04-16T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-04-23T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2012-04-24T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2011-12-16T00:00:00Z"),

"grade" : "A",

"score" : 2

}

],

"name" : "C & C Catering Service",

"restaurant\_id" : "40357437"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ad"),

"address" : {

"building" : "1839",

"coord" : [

-73.9482609,

40.6408271

],

"street" : "Nostrand Avenue",

"zipcode" : "11226"

},

"borough" : "Brooklyn",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"grades" : [

{

"date" : ISODate("2014-07-14T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-07-10T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2012-07-11T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2012-02-23T00:00:00Z"),

"grade" : "A",

"score" : 8

}

],

"name" : "Taste The Tropics Ice Cream",

"restaurant\_id" : "40356731"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ae"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-28T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-06-19T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2012-06-15T00:00:00Z"),

"grade" : "A",

"score" : 3

}

],

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3af"),

"address" : {

"building" : "2780",

"coord" : [

-73.98241999999999,

40.579505

],

"street" : "Stillwell Avenue",

"zipcode" : "11224"

},

"borough" : "Brooklyn",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-06-10T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-06-05T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2012-04-13T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-10-12T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Riviera Caterer",

"restaurant\_id" : "40356018"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b0"),

"address" : {

"building" : "1269",

"coord" : [

-73.871194,

40.6730975

],

"street" : "Sutter Avenue",

"zipcode" : "11208"

},

"borough" : "Brooklyn",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2014-09-16T00:00:00Z"),

"grade" : "B",

"score" : 21

},

{

"date" : ISODate("2013-08-28T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2013-04-02T00:00:00Z"),

"grade" : "C",

"score" : 56

},

{

"date" : ISODate("2012-08-15T00:00:00Z"),

"grade" : "B",

"score" : 27

},

{

"date" : ISODate("2012-03-28T00:00:00Z"),

"grade" : "B",

"score" : 27

}

],

"name" : "May May Kitchen",

"restaurant\_id" : "40358429"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b1"),

"address" : {

"building" : "97-22",

"coord" : [

-73.8601152,

40.7311739

],

"street" : "63 Road",

"zipcode" : "11374"

},

"borough" : "Queens",

"cuisine" : "Jewish/Kosher",

"grades" : [

{

"date" : ISODate("2014-11-24T00:00:00Z"),

"grade" : "Z",

"score" : 20

},

{

"date" : ISODate("2013-01-17T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-08-02T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2011-12-15T00:00:00Z"),

"grade" : "B",

"score" : 25

}

],

"name" : "Tov Kosher Kitchen",

"restaurant\_id" : "40356068"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b2"),

"address" : {

"building" : "1",

"coord" : [

-73.96926909999999,

40.7685235

],

"street" : "East 66 Street",

"zipcode" : "10065"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-07T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2013-05-03T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2012-04-30T00:00:00Z"),

"grade" : "A",

"score" : 6

},

{

"date" : ISODate("2011-12-27T00:00:00Z"),

"grade" : "A",

"score" : 0

}

],

"name" : "1 East 66Th Street Kitchen",

"restaurant\_id" : "40359480"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b3"),

"address" : {

"building" : "265-15",

"coord" : [

-73.7032601,

40.7386417

],

"street" : "Hillside Avenue",

"zipcode" : "11004"

},

"borough" : "Queens",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"grades" : [

{

"date" : ISODate("2014-10-28T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-09-18T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-09-20T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40361322"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b4"),

"address" : {

"building" : "8825",

"coord" : [

-73.8803827,

40.7643124

],

"street" : "Astoria Boulevard",

"zipcode" : "11369"

},

"borough" : "Queens",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-11-15T00:00:00Z"),

"grade" : "Z",

"score" : 38

},

{

"date" : ISODate("2014-05-02T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-03-02T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2012-02-10T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Brunos On The Boulevard",

"restaurant\_id" : "40356151"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b5"),

"address" : {

"building" : "705",

"coord" : [

-73.9653967,

40.6064339

],

"street" : "Kings Highway",

"zipcode" : "11223"

},

"borough" : "Brooklyn",

"cuisine" : "Jewish/Kosher",

"grades" : [

{

"date" : ISODate("2014-11-10T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-10-10T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-10-04T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2012-05-21T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2011-12-30T00:00:00Z"),

"grade" : "B",

"score" : 19

}

],

"name" : "Seuda Foods",

"restaurant\_id" : "40360045"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b6"),

"address" : {

"building" : "1007",

"coord" : [

-73.856077,

40.848447

],

"street" : "Morris Park Ave",

"zipcode" : "10462"

},

"borough" : "Bronx",

"cuisine" : "Bakery",

"grades" : [

{

"date" : ISODate("2014-03-03T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-09-11T00:00:00Z"),

"grade" : "A",

"score" : 6

},

{

"date" : ISODate("2013-01-24T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-11-23T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2011-03-10T00:00:00Z"),

"grade" : "B",

"score" : 14

}

],

"name" : "Morris Park Bake Shop",

"restaurant\_id" : "30075445"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b7"),

"address" : {

"building" : "2206",

"coord" : [

-74.1377286,

40.6119572

],

"street" : "Victory Boulevard",

"zipcode" : "10314"

},

"borough" : "Staten Island",

"cuisine" : "Jewish/Kosher",

"grades" : [

{

"date" : ISODate("2014-10-06T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2014-05-20T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-04-04T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-01-24T00:00:00Z"),

"grade" : "A",

"score" : 9

}

],

"name" : "Kosher Island",

"restaurant\_id" : "40356442"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b8"),

"address" : {

"building" : "203",

"coord" : [

-73.97822040000001,

40.6435254

],

"street" : "Church Avenue",

"zipcode" : "11218"

},

"borough" : "Brooklyn",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"grades" : [

{

"date" : ISODate("2014-02-10T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-01-02T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-01-09T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2011-11-07T00:00:00Z"),

"grade" : "P",

"score" : 12

},

{

"date" : ISODate("2011-07-21T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40360076"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b9"),

"address" : {

"building" : "129-08",

"coord" : [

-73.839297,

40.78147

],

"street" : "20 Avenue",

"zipcode" : "11356"

},

"borough" : "Queens",

"cuisine" : "Delicatessen",

"grades" : [

{

"date" : ISODate("2014-08-16T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-08-27T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-09-20T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2011-09-29T00:00:00Z"),

"grade" : "A",

"score" : 10

}

],

"name" : "Sal'S Deli",

"restaurant\_id" : "40361618"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ba"),

"address" : {

"building" : "3406",

"coord" : [

-73.94024739999999,

40.7623288

],

"street" : "10 Street",

"zipcode" : "11106"

},

"borough" : "Queens",

"cuisine" : "Delicatessen",

"grades" : [

{

"date" : ISODate("2014-03-19T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2013-03-13T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-03-27T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2011-04-05T00:00:00Z"),

"grade" : "A",

"score" : 7

}

],

"name" : "Steve Chu'S Deli & Grocery",

"restaurant\_id" : "40361998"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3bb"),

"address" : {

"building" : "18",

"coord" : [

-73.996984,

40.72589

],

"street" : "West Houston Street",

"zipcode" : "10012"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-04-03T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-04-05T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2012-03-21T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2011-04-27T00:00:00Z"),

"grade" : "A",

"score" : 5

}

],

"name" : "Angelika Film Center",

"restaurant\_id" : "40362274"

2. Write a MongoDB query to display the fields restaurant\_id, name, borough and cuisine for

all the documents in the collection restaurant.

Ans:-

> db.samplerestaurant.find({},{restaurant\_id:1,name:1,borough:1,cuisine:1}).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a8"),

"borough" : "Manhattan",

"cuisine" : "Irish",

"name" : "Dj Reynolds Pub And Restaurant",

"restaurant\_id" : "30191841"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a9"),

"borough" : "Brooklyn",

"cuisine" : "Hamburgers",

"name" : "Wendy'S",

"restaurant\_id" : "30112340"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3aa"),

"borough" : "Brooklyn",

"cuisine" : "Delicatessen",

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ab"),

"borough" : "Brooklyn",

"cuisine" : "American",

"name" : "Regina Caterers",

"restaurant\_id" : "40356649"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ac"),

"borough" : "Brooklyn",

"cuisine" : "American",

"name" : "C & C Catering Service",

"restaurant\_id" : "40357437"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ad"),

"borough" : "Brooklyn",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"name" : "Taste The Tropics Ice Cream",

"restaurant\_id" : "40356731"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ae"),

"borough" : "Bronx",

"cuisine" : "American",

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3af"),

"borough" : "Brooklyn",

"cuisine" : "American",

"name" : "Riviera Caterer",

"restaurant\_id" : "40356018"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b0"),

"borough" : "Brooklyn",

"cuisine" : "Chinese",

"name" : "May May Kitchen",

"restaurant\_id" : "40358429"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b1"),

"borough" : "Queens",

"cuisine" : "Jewish/Kosher",

"name" : "Tov Kosher Kitchen",

"restaurant\_id" : "40356068"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b2"),

"borough" : "Manhattan",

"cuisine" : "American",

"name" : "1 East 66Th Street Kitchen",

"restaurant\_id" : "40359480"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b3"),

"borough" : "Queens",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40361322"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b4"),

"borough" : "Queens",

"cuisine" : "American",

"name" : "Brunos On The Boulevard",

"restaurant\_id" : "40356151"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b5"),

"borough" : "Brooklyn",

"cuisine" : "Jewish/Kosher",

"name" : "Seuda Foods",

"restaurant\_id" : "40360045"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b6"),

"borough" : "Bronx",

"cuisine" : "Bakery",

"name" : "Morris Park Bake Shop",

"restaurant\_id" : "30075445"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b7"),

"borough" : "Staten Island",

"cuisine" : "Jewish/Kosher",

"name" : "Kosher Island",

"restaurant\_id" : "40356442"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b8"),

"borough" : "Brooklyn",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40360076"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b9"),

"borough" : "Queens",

"cuisine" : "Delicatessen",

"name" : "Sal'S Deli",

"restaurant\_id" : "40361618"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ba"),

"borough" : "Queens",

"cuisine" : "Delicatessen",

"name" : "Steve Chu'S Deli & Grocery",

"restaurant\_id" : "40361998"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3bb"),

"borough" : "Manhattan",

"cuisine" : "American",

"name" : "Angelika Film Center",

"restaurant\_id" : "40362274"

}

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.

Ans:-

> db.samplerestaurant.find({},{restaurant\_id:1,name:1,borough:1,cuisine:1,\_id:0}).pretty();

{

"borough" : "Manhattan",

"cuisine" : "Irish",

"name" : "Dj Reynolds Pub And Restaurant",

"restaurant\_id" : "30191841"

}

{

"borough" : "Brooklyn",

"cuisine" : "Hamburgers",

"name" : "Wendy'S",

"restaurant\_id" : "30112340"

}

{

"borough" : "Brooklyn",

"cuisine" : "Delicatessen",

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

{

"borough" : "Brooklyn",

"cuisine" : "American",

"name" : "Regina Caterers",

"restaurant\_id" : "40356649"

}

{

"borough" : "Brooklyn",

"cuisine" : "American",

"name" : "C & C Catering Service",

"restaurant\_id" : "40357437"

}

{

"borough" : "Brooklyn",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"name" : "Taste The Tropics Ice Cream",

"restaurant\_id" : "40356731"

}

{

"borough" : "Bronx",

"cuisine" : "American",

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

{

"borough" : "Brooklyn",

"cuisine" : "American",

"name" : "Riviera Caterer",

"restaurant\_id" : "40356018"

}

{

"borough" : "Brooklyn",

"cuisine" : "Chinese",

"name" : "May May Kitchen",

"restaurant\_id" : "40358429"

}

{

"borough" : "Queens",

"cuisine" : "Jewish/Kosher",

"name" : "Tov Kosher Kitchen",

"restaurant\_id" : "40356068"

}

{

"borough" : "Manhattan",

"cuisine" : "American",

"name" : "1 East 66Th Street Kitchen",

"restaurant\_id" : "40359480"

}

{

"borough" : "Queens",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40361322"

}

{

"borough" : "Queens",

"cuisine" : "American",

"name" : "Brunos On The Boulevard",

"restaurant\_id" : "40356151"

}

{

"borough" : "Brooklyn",

"cuisine" : "Jewish/Kosher",

"name" : "Seuda Foods",

"restaurant\_id" : "40360045"

}

{

"borough" : "Bronx",

"cuisine" : "Bakery",

"name" : "Morris Park Bake Shop",

"restaurant\_id" : "30075445"

}

{

"borough" : "Staten Island",

"cuisine" : "Jewish/Kosher",

"name" : "Kosher Island",

"restaurant\_id" : "40356442"

}

{

"borough" : "Brooklyn",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40360076"

}

{

"borough" : "Queens",

"cuisine" : "Delicatessen",

"name" : "Sal'S Deli",

"restaurant\_id" : "40361618"

}

{

"borough" : "Queens",

"cuisine" : "Delicatessen",

"name" : "Steve Chu'S Deli & Grocery",

"restaurant\_id" : "40361998"

}

{

"borough" : "Manhattan",

"cuisine" : "American",

"name" : "Angelika Film Center",

"restaurant\_id" : "40362274"

}

4. 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.

And:-

> db.samplerestaurant.find({},{restaurant\_id:1,name:1,borough:1,"address.zipcode":1,\_id:0}).pretty();

{

"address" : {

"zipcode" : "10019"

},

"borough" : "Manhattan",

"name" : "Dj Reynolds Pub And Restaurant",

"restaurant\_id" : "30191841"

}

{

"address" : {

"zipcode" : "11225"

},

"borough" : "Brooklyn",

"name" : "Wendy'S",

"restaurant\_id" : "30112340"

}

{

"address" : {

"zipcode" : "11234"

},

"borough" : "Brooklyn",

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

{

"address" : {

"zipcode" : "11219"

},

"borough" : "Brooklyn",

"name" : "Regina Caterers",

"restaurant\_id" : "40356649"

}

{

"address" : {

"zipcode" : "11214"

},

"borough" : "Brooklyn",

"name" : "C & C Catering Service",

"restaurant\_id" : "40357437"

}

{

"address" : {

"zipcode" : "11226"

},

"borough" : "Brooklyn",

"name" : "Taste The Tropics Ice Cream",

"restaurant\_id" : "40356731"

}

{

"address" : {

"zipcode" : "10460"

},

"borough" : "Bronx",

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

{

"address" : {

"zipcode" : "11224"

},

"borough" : "Brooklyn",

"name" : "Riviera Caterer",

"restaurant\_id" : "40356018"

}

{

"address" : {

"zipcode" : "11208"

},

"borough" : "Brooklyn",

"name" : "May May Kitchen",

"restaurant\_id" : "40358429"

}

{

"address" : {

"zipcode" : "11374"

},

"borough" : "Queens",

"name" : "Tov Kosher Kitchen",

"restaurant\_id" : "40356068"

}

{

"address" : {

"zipcode" : "10065"

},

"borough" : "Manhattan",

"name" : "1 East 66Th Street Kitchen",

"restaurant\_id" : "40359480"

}

{

"address" : {

"zipcode" : "11004"

},

"borough" : "Queens",

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40361322"

}

{

"address" : {

"zipcode" : "11369"

},

"borough" : "Queens",

"name" : "Brunos On The Boulevard",

"restaurant\_id" : "40356151"

}

{

"address" : {

"zipcode" : "11223"

},

"borough" : "Brooklyn",

"name" : "Seuda Foods",

"restaurant\_id" : "40360045"

}

{

"address" : {

"zipcode" : "10462"

},

"borough" : "Bronx",

"name" : "Morris Park Bake Shop",

"restaurant\_id" : "30075445"

}

{

"address" : {

"zipcode" : "10314"

},

"borough" : "Staten Island",

"name" : "Kosher Island",

"restaurant\_id" : "40356442"

}

{

"address" : {

"zipcode" : "11218"

},

"borough" : "Brooklyn",

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40360076"

}

{

"address" : {

"zipcode" : "11356"

},

"borough" : "Queens",

"name" : "Sal'S Deli",

"restaurant\_id" : "40361618"

}

{

"address" : {

"zipcode" : "11106"

},

"borough" : "Queens",

"name" : "Steve Chu'S Deli & Grocery",

"restaurant\_id" : "40361998"

}

{

"address" : {

"zipcode" : "10012"

},

"borough" : "Manhattan",

"name" : "Angelika Film Center",

"restaurant\_id" : "40362274"

}

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

Ans:

> db.samplerestaurant.find({borough:'Bronx'}).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ae"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-28T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-06-19T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2012-06-15T00:00:00Z"),

"grade" : "A",

"score" : 3

}

],

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b6"),

"address" : {

"building" : "1007",

"coord" : [

-73.856077,

40.848447

],

"street" : "Morris Park Ave",

"zipcode" : "10462"

},

"borough" : "Bronx",

"cuisine" : "Bakery",

"grades" : [

{

"date" : ISODate("2014-03-03T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-09-11T00:00:00Z"),

"grade" : "A",

"score" : 6

},

{

"date" : ISODate("2013-01-24T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-11-23T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2011-03-10T00:00:00Z"),

"grade" : "B",

"score" : 14

}

],

"name" : "Morris Park Bake Shop",

"restaurant\_id" : "30075445"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3c8"),

"address" : {

"building" : "1006",

"coord" : [

-73.84856870000002,

40.8903781

],

"street" : "East 233 Street",

"zipcode" : "10466"

},

"borough" : "Bronx",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"grades" : [

{

"date" : ISODate("2014-04-24T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-09-05T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-02-21T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-07-03T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-07-11T00:00:00Z"),

"grade" : "A",

"score" : 5

}

],

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40363093"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ca"),

"address" : {

"building" : "1236",

"coord" : [

-73.8893654,

40.81376179999999

],

"street" : "238 Spofford Ave",

"zipcode" : "10474"

},

"borough" : "Bronx",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2013-12-30T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-01-08T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-06-12T00:00:00Z"),

"grade" : "B",

"score" : 15

}

],

"name" : "Happy Garden",

"restaurant\_id" : "40363289"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3dd"),

"address" : {

"building" : "277",

"coord" : [

-73.8941893,

40.8634684

],

"street" : "East Kingsbridge Road",

"zipcode" : "10458"

},

"borough" : "Bronx",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2014-03-03T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-09-26T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-03-19T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-08-29T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-08-17T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Happy Garden",

"restaurant\_id" : "40364296"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3df"),

"address" : {

"building" : "658",

"coord" : [

-73.81363999999999,

40.82941100000001

],

"street" : "Clarence Ave",

"zipcode" : "10465"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-06-21T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2012-07-11T00:00:00Z"),

"grade" : "A",

"score" : 10

}

],

"name" : "Manhem Club",

"restaurant\_id" : "40364363"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3f9"),

"address" : {

"building" : "2222",

"coord" : [

-73.84971759999999,

40.8304811

],

"street" : "Haviland Avenue",

"zipcode" : "10462"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-12-18T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2014-05-01T00:00:00Z"),

"grade" : "B",

"score" : 17

},

{

"date" : ISODate("2013-03-14T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-09-20T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-02-08T00:00:00Z"),

"grade" : "B",

"score" : 19

}

],

"name" : "The New Starling Athletic Club Of The Bronx",

"restaurant\_id" : "40364956"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c415"),

"address" : {

"building" : "72",

"coord" : [

-73.92506,

40.8275556

],

"street" : "East 161 Street",

"zipcode" : "10451"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-04-15T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-11-14T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2013-07-29T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-12-31T00:00:00Z"),

"grade" : "B",

"score" : 15

},

{

"date" : ISODate("2012-05-30T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-01-09T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-08-15T00:00:00Z"),

"grade" : "C",

"score" : 37

}

],

"name" : "Yankee Tavern",

"restaurant\_id" : "40365499"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c42c"),

"address" : {

"building" : "331",

"coord" : [

-73.87786539999999,

40.8724377

],

"street" : "East 204 Street",

"zipcode" : "10467"

},

"borough" : "Bronx",

"cuisine" : "Irish",

"grades" : [

{

"date" : ISODate("2014-08-26T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2014-03-26T00:00:00Z"),

"grade" : "B",

"score" : 23

},

{

"date" : ISODate("2013-09-11T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-12-18T00:00:00Z"),

"grade" : "B",

"score" : 27

},

{

"date" : ISODate("2011-10-20T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Mcdwyers Pub",

"restaurant\_id" : "40365893"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c440"),

"address" : {

"building" : "5820",

"coord" : [

-73.9002615,

40.885186

],

"street" : "Broadway",

"zipcode" : "10463"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-02-26T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-10-09T00:00:00Z"),

"grade" : "B",

"score" : 19

},

{

"date" : ISODate("2013-05-15T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-11-20T00:00:00Z"),

"grade" : "B",

"score" : 18

},

{

"date" : ISODate("2011-10-17T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-06-22T00:00:00Z"),

"grade" : "C",

"score" : 35

}

],

"name" : "The Punch Bowl",

"restaurant\_id" : "40366497"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c44c"),

"address" : {

"building" : "21",

"coord" : [

-73.9168424,

40.8401362

],

"street" : "East 170 Street",

"zipcode" : "10452"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-12-16T00:00:00Z"),

"grade" : "B",

"score" : 22

},

{

"date" : ISODate("2014-08-12T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2014-03-03T00:00:00Z"),

"grade" : "B",

"score" : 22

},

{

"date" : ISODate("2013-08-29T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-08-29T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-02-13T00:00:00Z"),

"grade" : "B",

"score" : 16

}

],

"name" : "Munchtime",

"restaurant\_id" : "40366748"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c44e"),

"address" : {

"building" : "4340",

"coord" : [

-73.8194559,

40.8899176

],

"street" : "Boston Road",

"zipcode" : "10475"

},

"borough" : "Bronx",

"cuisine" : "Pancakes/Waffles",

"grades" : [

{

"date" : ISODate("2014-09-22T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-05-16T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-10-30T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2013-06-05T00:00:00Z"),

"grade" : "B",

"score" : 24

},

{

"date" : ISODate("2012-05-14T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-01-04T00:00:00Z"),

"grade" : "A",

"score" : 7

}

],

"name" : "Ihop",

"restaurant\_id" : "40366833"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c452"),

"address" : {

"building" : "1191",

"coord" : [

-73.8513114,

40.8316981

],

"street" : "Castle Hill Avenue",

"zipcode" : "10462"

},

"borough" : "Bronx",

"cuisine" : "Café/Coffee/Tea",

"grades" : [

{

"date" : ISODate("2014-07-03T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-07-08T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-02-11T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-08-14T00:00:00Z"),

"grade" : "B",

"score" : 14

},

{

"date" : ISODate("2011-08-23T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Lulu'S Coffee Shop",

"restaurant\_id" : "40366938"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c459"),

"address" : {

"building" : "2894",

"coord" : [

-73.815175,

40.8137649

],

"street" : "Schurz Ave",

"zipcode" : "10465"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-10-31T00:00:00Z"),

"grade" : "B",

"score" : 18

},

{

"date" : ISODate("2014-02-20T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2013-06-07T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-05-17T00:00:00Z"),

"grade" : "A",

"score" : 2

}

],

"name" : "Marina Delray",

"restaurant\_id" : "40367161"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c47e"),

"address" : {

"building" : "4280",

"coord" : [

-73.86697,

40.89794699999999

],

"street" : "Katonah Ave",

"zipcode" : "10470"

},

"borough" : "Bronx",

"cuisine" : "Not Listed/Not Applicable",

"grades" : [

{

"date" : ISODate("2014-06-04T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-06-07T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2012-06-27T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2011-07-13T00:00:00Z"),

"grade" : "A",

"score" : 10

}

],

"name" : "The Lark'S Nest",

"restaurant\_id" : "40367946"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c47f"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "African",

"grades" : [

{

"date" : ISODate("2014-05-21T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-06-01T00:00:00Z"),

"grade" : "A",

"score" : 6

},

{

"date" : ISODate("2012-06-12T00:00:00Z"),

"grade" : "A",

"score" : 0

},

{

"date" : ISODate("2011-06-08T00:00:00Z"),

"grade" : "A",

"score" : 0

}

],

"name" : "African Terrace",

"restaurant\_id" : "40368021"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c480"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "Café/Coffee/Tea",

"grades" : [

{

"date" : ISODate("2014-05-21T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-05-29T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-06-18T00:00:00Z"),

"grade" : "A",

"score" : 8

}

],

"name" : "Terrace Cafe",

"restaurant\_id" : "40368018"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c481"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-21T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-05-28T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2012-06-18T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2011-06-07T00:00:00Z"),

"grade" : "A",

"score" : 9

}

],

"name" : "African Market (Baboon Cafe)",

"restaurant\_id" : "40368026"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c482"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-21T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-05-29T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-06-15T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2011-06-08T00:00:00Z"),

"grade" : "A",

"score" : 0

}

],

"name" : "Beaver Pond",

"restaurant\_id" : "40368025"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c483"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-21T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-05-22T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-05-17T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-06-08T00:00:00Z"),

"grade" : "A",

"score" : 5

}

],

"name" : "Cool Zone",

"restaurant\_id" : "40368022"

}

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

Ans:-

> db.samplerestaurant.find({borough:'Bronx'}).limit(5).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ae"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-28T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-06-19T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2012-06-15T00:00:00Z"),

"grade" : "A",

"score" : 3

}

],

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3b6"),

"address" : {

"building" : "1007",

"coord" : [

-73.856077,

40.848447

],

"street" : "Morris Park Ave",

"zipcode" : "10462"

},

"borough" : "Bronx",

"cuisine" : "Bakery",

"grades" : [

{

"date" : ISODate("2014-03-03T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-09-11T00:00:00Z"),

"grade" : "A",

"score" : 6

},

{

"date" : ISODate("2013-01-24T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-11-23T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2011-03-10T00:00:00Z"),

"grade" : "B",

"score" : 14

}

],

"name" : "Morris Park Bake Shop",

"restaurant\_id" : "30075445"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3c8"),

"address" : {

"building" : "1006",

"coord" : [

-73.84856870000002,

40.8903781

],

"street" : "East 233 Street",

"zipcode" : "10466"

},

"borough" : "Bronx",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"grades" : [

{

"date" : ISODate("2014-04-24T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-09-05T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-02-21T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-07-03T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-07-11T00:00:00Z"),

"grade" : "A",

"score" : 5

}

],

"name" : "Carvel Ice Cream",

"restaurant\_id" : "40363093"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ca"),

"address" : {

"building" : "1236",

"coord" : [

-73.8893654,

40.81376179999999

],

"street" : "238 Spofford Ave",

"zipcode" : "10474"

},

"borough" : "Bronx",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2013-12-30T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-01-08T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-06-12T00:00:00Z"),

"grade" : "B",

"score" : 15

}

],

"name" : "Happy Garden",

"restaurant\_id" : "40363289"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3dd"),

"address" : {

"building" : "277",

"coord" : [

-73.8941893,

40.8634684

],

"street" : "East Kingsbridge Road",

"zipcode" : "10458"

},

"borough" : "Bronx",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2014-03-03T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-09-26T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-03-19T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-08-29T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-08-17T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Happy Garden",

"restaurant\_id" : "40364296"

}

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

the borough Bronx.

Ans:-

> db.samplerestaurant.find({borough:'Bronx'}).limit(5).skip(5).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3df"),

"address" : {

"building" : "658",

"coord" : [

-73.81363999999999,

40.82941100000001

],

"street" : "Clarence Ave",

"zipcode" : "10465"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-06-21T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2012-07-11T00:00:00Z"),

"grade" : "A",

"score" : 10

}

],

"name" : "Manhem Club",

"restaurant\_id" : "40364363"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3f9"),

"address" : {

"building" : "2222",

"coord" : [

-73.84971759999999,

40.8304811

],

"street" : "Haviland Avenue",

"zipcode" : "10462"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-12-18T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2014-05-01T00:00:00Z"),

"grade" : "B",

"score" : 17

},

{

"date" : ISODate("2013-03-14T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-09-20T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-02-08T00:00:00Z"),

"grade" : "B",

"score" : 19

}

],

"name" : "The New Starling Athletic Club Of The Bronx",

"restaurant\_id" : "40364956"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c415"),

"address" : {

"building" : "72",

"coord" : [

-73.92506,

40.8275556

],

"street" : "East 161 Street",

"zipcode" : "10451"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-04-15T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-11-14T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2013-07-29T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-12-31T00:00:00Z"),

"grade" : "B",

"score" : 15

},

{

"date" : ISODate("2012-05-30T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-01-09T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-08-15T00:00:00Z"),

"grade" : "C",

"score" : 37

}

],

"name" : "Yankee Tavern",

"restaurant\_id" : "40365499"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c42c"),

"address" : {

"building" : "331",

"coord" : [

-73.87786539999999,

40.8724377

],

"street" : "East 204 Street",

"zipcode" : "10467"

},

"borough" : "Bronx",

"cuisine" : "Irish",

"grades" : [

{

"date" : ISODate("2014-08-26T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2014-03-26T00:00:00Z"),

"grade" : "B",

"score" : 23

},

{

"date" : ISODate("2013-09-11T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-12-18T00:00:00Z"),

"grade" : "B",

"score" : 27

},

{

"date" : ISODate("2011-10-20T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Mcdwyers Pub",

"restaurant\_id" : "40365893"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c440"),

"address" : {

"building" : "5820",

"coord" : [

-73.9002615,

40.885186

],

"street" : "Broadway",

"zipcode" : "10463"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-02-26T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-10-09T00:00:00Z"),

"grade" : "B",

"score" : 19

},

{

"date" : ISODate("2013-05-15T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-11-20T00:00:00Z"),

"grade" : "B",

"score" : 18

},

{

"date" : ISODate("2011-10-17T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-06-22T00:00:00Z"),

"grade" : "C",

"score" : 35

}

],

"name" : "The Punch Bowl",

"restaurant\_id" : "40366497"

}

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

Ans:-

> db.samplerestaurant.find({'grades.score':{$gt:90}}).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c508"),

"address" : {

"building" : "65",

"coord" : [

-73.9782725,

40.7624022

],

"street" : "West 54 Street",

"zipcode" : "10019"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-08-22T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-03-28T00:00:00Z"),

"grade" : "C",

"score" : 131

},

{

"date" : ISODate("2013-09-25T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-04-08T00:00:00Z"),

"grade" : "B",

"score" : 25

},

{

"date" : ISODate("2012-10-15T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-10-19T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Murals On 54/Randolphs'S",

"restaurant\_id" : "40372466"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c5a7"),

"address" : {

"building" : "345",

"coord" : [

-73.9864626,

40.7266739

],

"street" : "East 6 Street",

"zipcode" : "10003"

},

"borough" : "Manhattan",

"cuisine" : "Indian",

"grades" : [

{

"date" : ISODate("2014-09-15T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2014-01-14T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-05-30T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-04-24T00:00:00Z"),

"grade" : "P",

"score" : 2

},

{

"date" : ISODate("2012-10-01T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-04-06T00:00:00Z"),

"grade" : "C",

"score" : 92

},

{

"date" : ISODate("2011-11-03T00:00:00Z"),

"grade" : "C",

"score" : 41

}

],

"name" : "Gandhi",

"restaurant\_id" : "40381295"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c70b"),

"address" : {

"building" : "130",

"coord" : [

-73.984758,

40.7457939

],

"street" : "Madison Avenue",

"zipcode" : "10016"

},

"borough" : "Manhattan",

"cuisine" : "Pizza/Italian",

"grades" : [

{

"date" : ISODate("2014-12-24T00:00:00Z"),

"grade" : "Z",

"score" : 31

},

{

"date" : ISODate("2014-06-17T00:00:00Z"),

"grade" : "C",

"score" : 98

},

{

"date" : ISODate("2013-12-12T00:00:00Z"),

"grade" : "C",

"score" : 32

},

{

"date" : ISODate("2013-05-22T00:00:00Z"),

"grade" : "B",

"score" : 21

},

{

"date" : ISODate("2012-05-02T00:00:00Z"),

"grade" : "A",

"score" : 11

}

],

"name" : "Bella Napoli",

"restaurant\_id" : "40393488"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8f733"),

"address" : {

"building" : "1724",

"coord" : [

-73.94981,

40.780043

],

"street" : "2 Avenue",

"zipcode" : "10128"

},

"borough" : "Manhattan",

"cuisine" : "Indian",

"grades" : [

{

"date" : ISODate("2014-09-25T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2014-03-20T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-09-09T00:00:00Z"),

"grade" : "B",

"score" : 21

},

{

"date" : ISODate("2013-03-25T00:00:00Z"),

"grade" : "B",

"score" : 18

},

{

"date" : ISODate("2012-08-15T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-12-23T00:00:00Z"),

"grade" : "C",

"score" : 98

}

],

"name" : "Baluchi'S Indian Food",

"restaurant\_id" : "41569277"

}

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

less than 100.

Ans:-

> db.samplerestaurant.find({"grades.score":{$gt:80,$lt:100}}).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c508"),

"address" : {

"building" : "65",

"coord" : [

-73.9782725,

40.7624022

],

"street" : "West 54 Street",

"zipcode" : "10019"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-08-22T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-03-28T00:00:00Z"),

"grade" : "C",

"score" : 131

},

{

"date" : ISODate("2013-09-25T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-04-08T00:00:00Z"),

"grade" : "B",

"score" : 25

},

{

"date" : ISODate("2012-10-15T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-10-19T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Murals On 54/Randolphs'S",

"restaurant\_id" : "40372466"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c5a7"),

"address" : {

"building" : "345",

"coord" : [

-73.9864626,

40.7266739

],

"street" : "East 6 Street",

"zipcode" : "10003"

},

"borough" : "Manhattan",

"cuisine" : "Indian",

"grades" : [

{

"date" : ISODate("2014-09-15T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2014-01-14T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-05-30T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-04-24T00:00:00Z"),

"grade" : "P",

"score" : 2

},

{

"date" : ISODate("2012-10-01T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-04-06T00:00:00Z"),

"grade" : "C",

"score" : 92

},

{

"date" : ISODate("2011-11-03T00:00:00Z"),

"grade" : "C",

"score" : 41

}

],

"name" : "Gandhi",

"restaurant\_id" : "40381295"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c70b"),

"address" : {

"building" : "130",

"coord" : [

-73.984758,

40.7457939

],

"street" : "Madison Avenue",

"zipcode" : "10016"

},

"borough" : "Manhattan",

"cuisine" : "Pizza/Italian",

"grades" : [

{

"date" : ISODate("2014-12-24T00:00:00Z"),

"grade" : "Z",

"score" : 31

},

{

"date" : ISODate("2014-06-17T00:00:00Z"),

"grade" : "C",

"score" : 98

},

{

"date" : ISODate("2013-12-12T00:00:00Z"),

"grade" : "C",

"score" : 32

},

{

"date" : ISODate("2013-05-22T00:00:00Z"),

"grade" : "B",

"score" : 21

},

{

"date" : ISODate("2012-05-02T00:00:00Z"),

"grade" : "A",

"score" : 11

}

],

"name" : "Bella Napoli",

"restaurant\_id" : "40393488"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8cf77"),

"address" : {

"building" : "",

"coord" : [

-74.0163793,

40.7167671

],

"street" : "Hudson River",

"zipcode" : "10282"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-06-27T00:00:00Z"),

"grade" : "C",

"score" : 89

},

{

"date" : ISODate("2013-06-06T00:00:00Z"),

"grade" : "A",

"score" : 6

},

{

"date" : ISODate("2012-06-19T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "West 79Th Street Boat Basin Cafe",

"restaurant\_id" : "40756344"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d4b4"),

"address" : {

"building" : "7705",

"coord" : [

-73.8878704,

40.7435875

],

"street" : "Woodside Avenue",

"zipcode" : "11373"

},

"borough" : "Queens",

"cuisine" : "Thai",

"grades" : [

{

"date" : ISODate("2014-05-12T00:00:00Z"),

"grade" : "B",

"score" : 14

},

{

"date" : ISODate("2013-11-20T00:00:00Z"),

"grade" : "C",

"score" : 84

},

{

"date" : ISODate("2012-09-18T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-05-03T00:00:00Z"),

"grade" : "B",

"score" : 23

}

],

"name" : "Spicy Shallot",

"restaurant\_id" : "40979431"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d4e6"),

"address" : {

"building" : "466",

"coord" : [

-73.9747277,

40.7536114

],

"street" : "Lexington Avenue",

"zipcode" : "10017"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-10-14T00:00:00Z"),

"grade" : "C",

"score" : 84

},

{

"date" : ISODate("2014-02-14T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-07-11T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-01-29T00:00:00Z"),

"grade" : "C",

"score" : 36

},

{

"date" : ISODate("2012-08-15T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-03-22T00:00:00Z"),

"grade" : "B",

"score" : 27

}

],

"name" : "Bistro Caterers",

"restaurant\_id" : "40987023"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8e67b"),

"address" : {

"building" : "320",

"coord" : [

-73.9934047,

40.7544014

],

"street" : "West 37 Street",

"zipcode" : "10018"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-09-16T00:00:00Z"),

"grade" : "C",

"score" : 90

},

{

"date" : ISODate("2014-02-21T00:00:00Z"),

"grade" : "B",

"score" : 27

},

{

"date" : ISODate("2012-10-01T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-03-01T00:00:00Z"),

"grade" : "A",

"score" : 10

}

],

"name" : "Concrete Restaurant",

"restaurant\_id" : "41363541"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8ea6c"),

"address" : {

"building" : "222",

"coord" : [

-73.9579033,

40.7332745

],

"street" : "Franklin Street",

"zipcode" : "11222"

},

"borough" : "Brooklyn",

"cuisine" : "Italian",

"grades" : [

{

"date" : ISODate("2014-12-11T00:00:00Z"),

"grade" : "Z",

"score" : 18

},

{

"date" : ISODate("2013-10-31T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-03-20T00:00:00Z"),

"grade" : "B",

"score" : 24

},

{

"date" : ISODate("2012-07-18T00:00:00Z"),

"grade" : "B",

"score" : 18

},

{

"date" : ISODate("2012-01-12T00:00:00Z"),

"grade" : "C",

"score" : 81

}

],

"name" : "Anella",

"restaurant\_id" : "41410058"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8f733"),

"address" : {

"building" : "1724",

"coord" : [

-73.94981,

40.780043

],

"street" : "2 Avenue",

"zipcode" : "10128"

},

"borough" : "Manhattan",

"cuisine" : "Indian",

"grades" : [

{

"date" : ISODate("2014-09-25T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2014-03-20T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-09-09T00:00:00Z"),

"grade" : "B",

"score" : 21

},

{

"date" : ISODate("2013-03-25T00:00:00Z"),

"grade" : "B",

"score" : 18

},

{

"date" : ISODate("2012-08-15T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-12-23T00:00:00Z"),

"grade" : "C",

"score" : 98

}

],

"name" : "Baluchi'S Indian Food",

"restaurant\_id" : "41569277"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8f7ce"),

"address" : {

"building" : "116",

"coord" : [

-73.98978509999999,

40.7487912

],

"street" : "West 32 Street",

"zipcode" : "10001"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-02-11T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-10-10T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-04-29T00:00:00Z"),

"grade" : "C",

"score" : 82

},

{

"date" : ISODate("2012-04-30T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-12-21T00:00:00Z"),

"grade" : "B",

"score" : 19

},

{

"date" : ISODate("2011-07-20T00:00:00Z"),

"grade" : "C",

"score" : 29

}

],

"name" : "Cafe R",

"restaurant\_id" : "41574642"

}

{

"\_id" : ObjectId("6443942f421d36aff7b90c6c"),

"address" : {

"building" : "5616",

"coord" : [

-74.009993,

40.638865

],

"street" : "7 Avenue",

"zipcode" : "11220"

},

"borough" : "Brooklyn",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2014-11-24T00:00:00Z"),

"grade" : "C",

"score" : 86

},

{

"date" : ISODate("2014-06-24T00:00:00Z"),

"grade" : "B",

"score" : 20

},

{

"date" : ISODate("2014-01-07T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-06-13T00:00:00Z"),

"grade" : "A",

"score" : 10

}

],

"name" : "D & Y Restaurant",

"restaurant\_id" : "50000040"

}

{

"\_id" : ObjectId("6443942f421d36aff7b91f8f"),

"address" : {

"building" : "1898",

"coord" : [

-73.910439,

40.8499696

],

"street" : "Jerome Avenue",

"zipcode" : "10453"

},

"borough" : "Bronx",

"cuisine" : "Latin (Cuban, Dominican, Puerto Rican, South & Central American)",

"grades" : [

{

"date" : ISODate("2015-01-06T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2014-10-07T00:00:00Z"),

"grade" : "C",

"score" : 82

}

],

"name" : "La Potencia Restaurant",

"restaurant\_id" : "50014192"

}

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

95.754168.

Ans:-

> db.samplerestaurant.find({"address.coord.0":{$lt:-95.754168}}).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c9ef"),

"address" : {

"building" : "3707",

"coord" : [

-101.8945214,

33.5197474

],

"street" : "82 Street",

"zipcode" : "11372"

},

"borough" : "Queens",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-06-04T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-11-07T00:00:00Z"),

"grade" : "B",

"score" : 19

},

{

"date" : ISODate("2013-05-17T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-08-29T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-04-03T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-11-16T00:00:00Z"),

"grade" : "A",

"score" : 7

}

],

"name" : "Burger King",

"restaurant\_id" : "40534067"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8cd5d"),

"address" : {

"building" : "15259",

"coord" : [

-119.6368672,

36.2504996

],

"street" : "10 Avenue",

"zipcode" : "11357"

},

"borough" : "Queens",

"cuisine" : "Italian",

"grades" : [

{

"date" : ISODate("2014-09-04T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-03-26T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-03-04T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-09-27T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-04-20T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2011-11-23T00:00:00Z"),

"grade" : "C",

"score" : 34

}

],

"name" : "Cascarino'S",

"restaurant\_id" : "40668681"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d203"),

"address" : {

"building" : "60",

"coord" : [

-111.9975205,

42.0970258

],

"street" : "West Side Highway",

"zipcode" : "10006"

},

"borough" : "Manhattan",

"cuisine" : "Japanese",

"grades" : [

{

"date" : ISODate("2014-03-20T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-06-28T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-07-05T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2011-07-27T00:00:00Z"),

"grade" : "A",

"score" : 2

}

],

"name" : "Sports Center At Chelsea Piers (Sushi Bar)",

"restaurant\_id" : "40882356"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d3ed"),

"address" : {

"building" : "524",

"coord" : [

-122.3870832,

37.7606086

],

"street" : "Beach 20 Street",

"zipcode" : "11691"

},

"borough" : "Queens",

"cuisine" : "Pizza",

"grades" : [

{

"date" : ISODate("2014-09-24T00:00:00Z"),

"grade" : "Z",

"score" : 15

},

{

"date" : ISODate("2014-05-08T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-11-04T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-10-09T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-11-01T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Domino'S Pizza",

"restaurant\_id" : "40949724"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d472"),

"address" : {

"building" : "6822",

"coord" : [

-115.2372343,

36.1836718

],

"street" : "Fresh Meadows Lane",

"zipcode" : "11365"

},

"borough" : "Queens",

"cuisine" : "Pizza/Italian",

"grades" : [

{

"date" : ISODate("2014-03-08T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-02-13T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2012-07-16T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2011-07-12T00:00:00Z"),

"grade" : "A",

"score" : 5

}

],

"name" : "Angelo & Al'S Pizzeria",

"restaurant\_id" : "40969851"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8dbb3"),

"address" : {

"building" : "845",

"coord" : [

-120.4599938,

36.8606752

],

"street" : "U N Plaza",

"zipcode" : "10017"

},

"borough" : "Manhattan",

"cuisine" : "Japanese",

"grades" : [

{

"date" : ISODate("2014-10-02T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-03-21T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-04-16T00:00:00Z"),

"grade" : "B",

"score" : 16

},

{

"date" : ISODate("2012-05-08T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Megu Midtown",

"restaurant\_id" : "41167462"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8dd49"),

"address" : {

"building" : "4902",

"coord" : [

-95.9220496,

36.0912513

],

"street" : "East 49 Street",

"zipcode" : "11203"

},

"borough" : "Brooklyn",

"cuisine" : "Caribbean",

"grades" : [

{

"date" : ISODate("2014-06-17T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-06-18T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-01-31T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-08-09T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2012-07-24T00:00:00Z"),

"grade" : "P",

"score" : 2

},

{

"date" : ISODate("2012-03-08T00:00:00Z"),

"grade" : "B",

"score" : 15

},

{

"date" : ISODate("2011-10-03T00:00:00Z"),

"grade" : "A",

"score" : 7

}

],

"name" : "A&P Roti & Pastry Shop",

"restaurant\_id" : "41204137"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8e0f0"),

"address" : {

"building" : "20915",

"coord" : [

-99.0640336,

40.7284015

],

"street" : "East 56 Street",

"zipcode" : "10022"

},

"borough" : "Manhattan",

"cuisine" : "Italian",

"grades" : [

{

"date" : ISODate("2014-06-12T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2014-01-09T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2013-05-31T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-09-27T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2012-03-28T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-09-23T00:00:00Z"),

"grade" : "C",

"score" : 13

}

],

"name" : "Fiorini",

"restaurant\_id" : "41270541"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8e158"),

"address" : {

"building" : "5602",

"coord" : [

-119.565005,

36.3924905

],

"street" : "6 Avenue",

"zipcode" : "11220"

},

"borough" : "Brooklyn",

"cuisine" : "Middle Eastern",

"grades" : [

{

"date" : ISODate("2014-06-11T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-09-21T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-06-16T00:00:00Z"),

"grade" : "A",

"score" : 9

}

],

"name" : "Widdi Hall",

"restaurant\_id" : "41276825"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8e2fd"),

"address" : {

"building" : "13620",

"coord" : [

-104.2081266,

39.6086401

],

"street" : "38 Avenue",

"zipcode" : "11354"

},

"borough" : "Queens",

"cuisine" : "Café/Coffee/Tea",

"grades" : [

{

"date" : ISODate("2014-01-07T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2013-01-02T00:00:00Z"),

"grade" : "A",

"score" : 7

}

],

"name" : "Rose House",

"restaurant\_id" : "41303891"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8e440"),

"address" : {

"building" : "13357",

"coord" : [

-108.306687,

38.0890517

],

"street" : "41 Road",

"zipcode" : "11355"

},

"borough" : "Queens",

"cuisine" : "Bakery",

"grades" : [

{

"date" : ISODate("2014-04-14T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-04-10T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2012-04-05T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-11-09T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2011-06-22T00:00:00Z"),

"grade" : "A",

"score" : 9

}

],

"name" : "New Sun Mary Bakery",

"restaurant\_id" : "41323353"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8eeff"),

"address" : {

"building" : "24-28",

"coord" : [

-105.7857603,

39.9472999

],

"street" : "Hughes Avenue",

"zipcode" : "10458"

},

"borough" : "Bronx",

"cuisine" : "Mexican",

"grades" : [

{

"date" : ISODate("2014-12-04T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2014-06-28T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-05-02T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "El Oasis Bar Restaurant",

"restaurant\_id" : "41469687"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8f28f"),

"address" : {

"building" : "13617",

"coord" : [

-104.8340205,

39.7691992

],

"street" : "38 Avenue",

"zipcode" : "11354"

},

"borough" : "Queens",

"cuisine" : "Chinese/Japanese",

"grades" : [

{

"date" : ISODate("2014-12-04T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2014-06-25T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-12-11T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2013-06-19T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-06-27T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2011-12-07T00:00:00Z"),

"grade" : "B",

"score" : 17

}

],

"name" : "Minni Shabu Shabu",

"restaurant\_id" : "41513285"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8f438"),

"address" : {

"building" : "5371",

"coord" : [

-119.6684796,

36.3280082

],

"street" : "Kings Plaza Mall",

"zipcode" : "11234"

},

"borough" : "Brooklyn",

"cuisine" : "Ice Cream, Gelato, Yogurt, Ices",

"grades" : [

{

"date" : ISODate("2014-12-08T00:00:00Z"),

"grade" : "B",

"score" : 14

},

{

"date" : ISODate("2014-06-26T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-05-23T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-06-05T00:00:00Z"),

"grade" : "A",

"score" : 7

}

],

"name" : "Everything Yogurt/Gretel'S Pretzels",

"restaurant\_id" : "41539001"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8f69e"),

"address" : {

"building" : "15059",

"coord" : [

-114.6787351,

33.4631523

],

"street" : "34 Avenue",

"zipcode" : "11354"

},

"borough" : "Queens",

"cuisine" : "Delicatessen",

"grades" : [

{

"date" : ISODate("2014-09-04T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2013-08-29T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2013-04-08T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "George'S Sidestreet Deli & Grill",

"restaurant\_id" : "41563549"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8f90e"),

"address" : {

"building" : "30012",

"coord" : [

-100.707166,

47.13706149999999

],

"street" : "East 52 Street",

"zipcode" : "10022"

},

"borough" : "Manhattan",

"cuisine" : "Mediterranean",

"grades" : [

{

"date" : ISODate("2014-06-26T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-12-10T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-05-01T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-02-23T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2011-10-18T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Pierre Loti Midtown",

"restaurant\_id" : "41588029"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8ff12"),

"address" : {

"building" : "7608",

"coord" : [

-121.4158208,

38.5319584

],

"street" : "21St Ave",

"zipcode" : "11370"

},

"borough" : "Bronx",

"cuisine" : "Italian",

"grades" : [

{

"date" : ISODate("2014-10-01T00:00:00Z"),

"grade" : "C",

"score" : 32

},

{

"date" : ISODate("2014-03-13T00:00:00Z"),

"grade" : "B",

"score" : 18

},

{

"date" : ISODate("2013-05-21T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-11-27T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-02-27T00:00:00Z"),

"grade" : "A",

"score" : 2

}

],

"name" : "Antonio'S",

"restaurant\_id" : "41636621"

}

{

"\_id" : ObjectId("6443942f421d36aff7b90390"),

"address" : {

"building" : "1418",

"coord" : [

-97.40487,

27.7767669

],

"street" : "Elizabeth Street",

"zipcode" : "10013"

},

"borough" : "Manhattan",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2014-03-17T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-09-20T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-04-08T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-09-25T00:00:00Z"),

"grade" : "A",

"score" : 5

}

],

"name" : "Mini Express",

"restaurant\_id" : "41672159"

}

{

"\_id" : ObjectId("6443942f421d36aff7b903ab"),

"address" : {

"building" : "0",

"coord" : [

-157.8887924,

21.3158403

],

"street" : "Pier 86",

"zipcode" : "10036"

},

"borough" : "Manhattan",

"cuisine" : "Sandwiches/Salads/Mixed Buffet",

"grades" : [

{

"date" : ISODate("2014-12-30T00:00:00Z"),

"grade" : "Z",

"score" : 24

},

{

"date" : ISODate("2014-06-27T00:00:00Z"),

"grade" : "A",

"score" : 3

},

{

"date" : ISODate("2013-06-05T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2012-07-23T00:00:00Z"),

"grade" : "A",

"score" : 11

}

],

"name" : "Au Bon Pain",

"restaurant\_id" : "41673043"

}

{

"\_id" : ObjectId("6443942f421d36aff7b90464"),

"address" : {

"building" : "13933",

"coord" : [

-96.702326,

43.8332898

],

"street" : "245 Street",

"zipcode" : "11422"

},

"borough" : "Queens",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-11-19T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2014-05-30T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-03-27T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2012-10-17T00:00:00Z"),

"grade" : "A",

"score" : 4

}

],

"name" : "Checkers",

"restaurant\_id" : "41679636"

}

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.

Ans:-

> db.samplerestaurant.find({cuision:{$nin:["America"]},"grades.score":{$gt:70},"address.coord.0":{$lt:-65.754168}}).pretty();

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.

Ans:-

> db.samplerestaurant.find({cuision:{$nin:["America"]},"grades.score":{$gt:70},"address.coord.0":{$lt:-65.754168}}).limit(2).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c508"),

"address" : {

"building" : "65",

"coord" : [

-73.9782725,

40.7624022

],

"street" : "West 54 Street",

"zipcode" : "10019"

},

"borough" : "Manhattan",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-08-22T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-03-28T00:00:00Z"),

"grade" : "C",

"score" : 131

},

{

"date" : ISODate("2013-09-25T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-04-08T00:00:00Z"),

"grade" : "B",

"score" : 25

},

{

"date" : ISODate("2012-10-15T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2011-10-19T00:00:00Z"),

"grade" : "A",

"score" : 13

}

],

"name" : "Murals On 54/Randolphs'S",

"restaurant\_id" : "40372466"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c5a7"),

"address" : {

"building" : "345",

"coord" : [

-73.9864626,

40.7266739

],

"street" : "East 6 Street",

"zipcode" : "10003"

},

"borough" : "Manhattan",

"cuisine" : "Indian",

"grades" : [

{

"date" : ISODate("2014-09-15T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2014-01-14T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-05-30T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2013-04-24T00:00:00Z"),

"grade" : "P",

"score" : 2

},

{

"date" : ISODate("2012-10-01T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2012-04-06T00:00:00Z"),

"grade" : "C",

"score" : 92

},

{

"date" : ISODate("2011-11-03T00:00:00Z"),

"grade" : "C",

"score" : 41

}

],

"name" : "Gandhi",

"restaurant\_id" : "40381295"

}

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.

Ans:

> db.samplerestaurant.find({cuision:{$ne:'American'},'grades.grade':'A',"borough":{$nin:['Brooklyn']}}).sort({cuision:-1}).limit(2).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a8"),

"address" : {

"building" : "351",

"coord" : [

-73.98513559999999,

40.7676919

],

"street" : "West 57 Street",

"zipcode" : "10019"

},

"borough" : "Manhattan",

"cuisine" : "Irish",

"grades" : [

{

"date" : ISODate("2014-09-06T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2013-07-22T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2012-07-31T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-12-29T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Dj Reynolds Pub And Restaurant",

"restaurant\_id" : "30191841"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ae"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-28T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-06-19T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2012-06-15T00:00:00Z"),

"grade" : "A",

"score" : 3

}

],

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

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.

Ans:-

> db.samplerestaurant.find({name:{$in:[/^Wil.\*/]}},{restaurant\_id:1,name:1,borough:1,cuisine:1}).limit(2).pretty()

{

"\_id" : ObjectId("6443942e421d36aff7b8c3aa"),

"borough" : "Brooklyn",

"cuisine" : "Delicatessen",

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ae"),

"borough" : "Bronx",

"cuisine" : "American",

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

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.

Ans:-

> db.samplerestaurant.find({name:{$in:[/^.\*ces$/]}},{restaurant\_id:1,name:1,borough:1,cuisine:1}).limit(2).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c837"),

"borough" : "Manhattan",

"cuisine" : "American",

"name" : "Pieces",

"restaurant\_id" : "40399910"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c8f8"),

"borough" : "Queens",

"cuisine" : "American",

"name" : "S.M.R Restaurant Services",

"restaurant\_id" : "40403857"

}

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.

Ans:-

> db.samplerestaurant.find({name:{$in:[/^.\*Reg.\*$/]}},{restaurant\_id:1,name:1,borough:1,cuisine:1}).limit(2).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ab"),

"borough" : "Brooklyn",

"cuisine" : "American",

"name" : "Regina Caterers",

"restaurant\_id" : "40356649"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c4a8"),

"borough" : "Manhattan",

"cuisine" : "Café/Coffee/Tea",

"name" : "Caffe Reggio",

"restaurant\_id" : "40369418"

}

17. Write a MongoDB query to find the restaurants which belong to the borough Bronx and

prepared either American or Chinese dish.

Ans:-

> db.samplerestaurant.find({borough:'Bronx',$or:[{cuisine:'American'},{cuisine:'Chinese'}]}).limit(2).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ae"),

"address" : {

"building" : "2300",

"coord" : [

-73.8786113,

40.8502883

],

"street" : "Southern Boulevard",

"zipcode" : "10460"

},

"borough" : "Bronx",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-05-28T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-06-19T00:00:00Z"),

"grade" : "A",

"score" : 4

},

{

"date" : ISODate("2012-06-15T00:00:00Z"),

"grade" : "A",

"score" : 3

}

],

"name" : "Wild Asia",

"restaurant\_id" : "40357217"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3ca"),

"address" : {

"building" : "1236",

"coord" : [

-73.8893654,

40.81376179999999

],

"street" : "238 Spofford Ave",

"zipcode" : "10474"

},

"borough" : "Bronx",

"cuisine" : "Chinese",

"grades" : [

{

"date" : ISODate("2013-12-30T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2013-01-08T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-06-12T00:00:00Z"),

"grade" : "B",

"score" : 15

}

],

"name" : "Happy Garden",

"restaurant\_id" : "40363289"

}

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.

Ans:-

> db.samplerestaurant.find({$or:[{borough:'Staten Island'},{borough:'Bronx'},{borough:'Brooklyn'},{borough:'Queens'}]},{restaurant\_id:1,name:1,borough:1,cuisine:1}).limit(2).pretty()

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a9"),

"borough" : "Brooklyn",

"cuisine" : "Hamburgers",

"name" : "Wendy'S",

"restaurant\_id" : "30112340"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3aa"),

"borough" : "Brooklyn",

"cuisine" : "Delicatessen",

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

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.

And:-

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.

Ans:-> db.samplerestaurant.find({'grades.score':{$lt:10}},{restaurant\_id:1,name:1,borough:1,cuisine:1}).limit(3).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a8"),

"borough" : "Manhattan",

"cuisine" : "Irish",

"name" : "Dj Reynolds Pub And Restaurant",

"restaurant\_id" : "30191841"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3a9"),

"borough" : "Brooklyn",

"cuisine" : "Hamburgers",

"name" : "Wendy'S",

"restaurant\_id" : "30112340"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8c3aa"),

"borough" : "Brooklyn",

"cuisine" : "Delicatessen",

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

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'.

Ans:-

> db.samplerestaurant.find({cuisine:{$nin:['American','Chinese']},name:{$in:[/^Wil.\*$/]}},{restaurant\_id:1,name:1,borough:1,cuisine:1}).limit(3).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c3aa"),

"borough" : "Brooklyn",

"cuisine" : "Delicatessen",

"name" : "Wilken'S Fine Food",

"restaurant\_id" : "40356483"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d1b5"),

"borough" : "Bronx",

"cuisine" : "Pizza",

"name" : "Wilbel Pizza",

"restaurant\_id" : "40871979"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d308"),

"borough" : "Manhattan",

"cuisine" : "Seafood",

"name" : "Wild Edibles",

"restaurant\_id" : "40928482"

}

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

Ans:-

> db.samplerestaurant.find({"grades":{$elemMatch:{grade:"A",score:11,date: ISODate("2014-08-11T00:00:00Z")}}},{"restaurant\_id":1,"name":1,"grades":1}).pretty().limit(3);

{

"\_id" : ObjectId("6443942e421d36aff7b8c501"),

"grades" : [

{

"date" : ISODate("2014-08-11T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2013-12-10T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-06-10T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-06-08T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2012-01-25T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2011-09-13T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Don Filippo Restaurant",

"restaurant\_id" : "40372417"

}

{

"\_id" : ObjectId("6443942e421d36aff7b8d2d6"),

"grades" : [

{

"date" : ISODate("2014-08-11T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-02-04T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2013-06-19T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2012-06-25T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-02-08T00:00:00Z"),

"grade" : "A",

"score" : 5

}

],

"name" : "Cosi",

"restaurant\_id" : "40922983"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8de89"),

"grades" : [

{

"date" : ISODate("2014-08-11T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-02-07T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-12-05T00:00:00Z"),

"grade" : "A",

"score" : 8

},

{

"date" : ISODate("2012-07-13T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2012-02-08T00:00:00Z"),

"grade" : "A",

"score" : 9

}

],

"name" : "Subway",

"restaurant\_id" : "41230741"

}

23. Write a MongoDB query to find the restaurant Id, name and grades for those restaurants

where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate

"2014-08-11T00:00:00Z".

Ans:-

> db.samplerestaurant.find({'grades.1.grade':'A','grades.1.score':9,'grades.1.date':ISODate("2014-08-11T00:00:00Z")},{restaurant\_id:1,name:1,grades:1}).limit(2).pretty();

{

"\_id" : ObjectId("6443942e421d36aff7b8c9d0"),

"grades" : [

{

"date" : ISODate("2015-01-12T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2014-08-11T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2014-01-14T00:00:00Z"),

"grade" : "A",

"score" : 13

},

{

"date" : ISODate("2013-02-07T00:00:00Z"),

"grade" : "A",

"score" : 10

},

{

"date" : ISODate("2012-04-30T00:00:00Z"),

"grade" : "A",

"score" : 11

}

],

"name" : "Club Macanudo (Cigar Bar)",

"restaurant\_id" : "40526406"

}

{

"\_id" : ObjectId("6443942f421d36aff7b8e2c0"),

"grades" : [

{

"date" : ISODate("2015-01-15T00:00:00Z"),

"grade" : "A",

"score" : 11

},

{

"date" : ISODate("2014-08-11T00:00:00Z"),

"grade" : "A",

"score" : 9

},

{

"date" : ISODate("2013-06-18T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2013-01-24T00:00:00Z"),

"grade" : "A",

"score" : 2

},

{

"date" : ISODate("2012-06-11T00:00:00Z"),

"grade" : "A",

"score" : 11

}

],

"name" : "Shoolbred'S",

"restaurant\_id" : "41302014"

}

24. Write a MongoDB query to find the restaurant Id, name, address and geographical

location for those restaurants where 2nd element of coord array contains a value which is

more than 42 and upto 52

Ans:-

**> db.rest.find({'address.coord.1':{$gt:42,$lte:52}},{restaurant\_id:1,name:1,address:1}).pretty().limit(3).count()**

66

> db.rest.find({'address.coord.1':{$gt:42,$lte:52}},{restaurant\_id:1,name:1,address:1}).pretty().limit(3)

{

"\_id" : ObjectId("643c318fe09dff85147c6da6"),

"address" : {

"building" : "47",

"coord" : [

-78.877224,

42.89546199999999

],

"street" : "Broadway @ Trinity Pl",

"zipcode" : "10006"

},

"name" : "T.G.I. Friday'S",

"restaurant\_id" : "40387990"

}

{

**25. Write a MongoDB query to arrange the name of the restaurants in ascending order along**

**with all the columns.**

**> db.rest.find().sort({name:1}).pretty()**

{

"\_id" : ObjectId("643c3190e09dff85147ccc2f"),

"address" : {

"building" : "154",

"coord" : [

-73.9189064,

40.8654529

],

"street" : "Post Ave",

"zipcode" : "10034"

},

"borough" : "Manhattan",

"cuisine" : "Other",

"grades" : [ ],

"name" : "",

"restaurant\_id" : "50017887"

}

**26. Write a MongoDB query to arrange the name of the restaurants in descending along with**

**all the columns.**

**> db.rest.find().sort({name:-1}).pretty()**

{

"date" : ISODate("2013-04-17T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Zz'S Pizza & Grill",

"restaurant\_id" : "41702705"

**27. Write a MongoDB query to arranged the name of the cuisine in ascending order and for**

**that same cuisine borough should be in descending order.**

**> db.rest.find().sort({cuisine:1},{borough:-1}).pretty(**

{

"\_id" : ObjectId("643c318fe09dff85147c71f0"),

"address" : {

"building" : "1345",

"coord" : [

-73.959249,

40.768076

],

"street" : "2 Avenue",

"zipcode" : "10021"

},

"borough" : "Manhattan",

"cuisine" : "Afghan",

"grades" : [

**28. Write a MongoDB query to know whether all the addresses contains the street or not.**

**> db.rest.find({'address.street':{$exists:false}}).pretty().count()**

0

>

**29. Write a MongoDB query which will select all documents in the restaurants collection**

**where the coord field value is Double.**

**> db.rest.find({'address.coord':{$type: "double"}}).limit(3).pretty()**

{

"\_id" : ObjectId("643c318fe09dff85147c6b04"),

"address" : {

"building" : "351",

"coord" : [

-73.98513559999999,

40.7676919

],

"street" : "West 57 Street",

"zipcode" : "10019"

},

"borough" : "Manhattan",

"cuisine" : "Irish",

"grades" : [

**30. Write a MongoDB query which will select the restaurant Id, name and grades for those**

**restaurants which returns 0 as a remainder after dividing the score by 7.**

**> db.rest.find({'grades.score':{$mod:[7,0]}},{restaurent\_id:1,name:1,grades:1}).limit(2).pretty()**

{

"\_id" : ObjectId("643c318fe09dff85147c6b05"),

"grades" : [

{

"date" : ISODate("2014-06-10T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-06-05T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2012-04-13T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-10-12T00:00:00Z"),

**31. Write a MongoDB query to find the restaurant name, borough, longitude and attitude and**

**cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.**

**> db.rest.find({name:/mon/},{name:1,borough:1,'address.coord':1,cuisine:1}).limit(3).pretty()**

{

"\_id" : ObjectId("643c318fe09dff85147c6b95"),

"address" : {

"coord" : [

-73.98306099999999,

40.7441419

]

},

"borough" : "Manhattan",

"cuisine" : "American",

"name" : "Desmond'S Tavern"

}

> db.rest.find({name:/mon/},{name:1,borough:1,'address.coord':1,cuisine:1}).limit(3).pretty().count()

182

> db.rest.find({name:/.\*mon.\*/},{name:1,borough:1,'address.coord':1,cuisine:1}).limit(3).pretty().count()

182

>

**32. Write a MongoDB query to find the restaurant name, borough, longitude and latitude and**

**cuisine for those restaurants which contain 'Mad' as first three letters of its name.**

**> db.rest.find({name:/Mad/},{name:1,borough:1,'address.coord':1,cuisine:1}).limit(3).pretty()**

{

"\_id" : ObjectId("643c318fe09dff85147c6d2f"),

"address" : {

"coord" : [

-74.158427,

40.626607

]

},

"borough" : "Staten Island",

"cuisine" : "Spanish",

"name" : "Real Madrid Restaurant"

}

**Extra Commands:**

**>db.emp.drop()**

**> db.createCollection('emp',{capped:true,size:30000})**

Create a Employee Collection add 5 documents:

Example:

{empno:111,ename:”Deepali

Vaidya”,sal:40000.00,dept:{deptno:12,dname:,”Hr”,dloc:’Mumbai’},

Desg:”Analyst”,mgr:{name:”Satish”,num:111},project:[{name:”Project-

1”,Hrs:4},{name:”project- 2”,Hrs:4}]}

1.db.emp.insert({empno:111,ename:"Deepali",sal:40000.00,dept:{deptno:12,dname:"HR",dloc:"Mumbai"},Desg:"Analyst",mgr:{name:"Sathish",num:111},project:[{name:"Project-1",Hrs:4},{name:"Project-2",Hrs:4}]})

2.db.emp.insert({empno:112,ename:"Harshal",sal:35000.00,dept:{deptno:13,dname:"Finance",dloc:"Mumbai"},Desg:"Clerk",mgr:{name:"Revati",num:112},project:{name:"Project-1",Hrs:8}})

3.db.emp.insert({empno:113,ename:"Harshali",sal:50000.00,dept:{deptno:11,dname:"Development",dloc:"Mumbai"},Desg:"Architect",mgr:{name:"Ayush",num:113},project:[{name:"Project-1",Hrs:8}]})

4.db.emp.insert({empno:114,ename:"Rahul",sal:25000.00,dept:{deptno:11,dname:"Sales",dloc:"Mumbai"},Desg:"Developer",mgr:{name:"Suyash",num:114},project:[{name:"Project-1",Hrs:6},{name:"Project-2",Hrs:2}]})

5.db.emp.insert({empno:115,ename:"Virat",sal:300000.00,dept:{deptno:11,dname:"Purchase",dloc:"Mumbai"},Desg:"VP",project:[{name:"Project-1",Hrs:6},{name:"Project-2",Hrs:2}]})

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

Update the Employee collection for this.

Ans:

> db.emp.update({Desg:"Clerk"},{$set:{Desg:"(AO )Admin Officer"}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

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

Ans:

> db.emp.update({Desg:"Analyst",'project.name':'Project-1'},{$set:{'project.0.Hrs':5}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

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

Ans:

> db.emp.update({ename:/Deep/},{$push:{"project":{$each:[{name:"Project-3",Hrs:2},{name:"Project-4",Hrs:2}]}}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.emp.update({ename:/Deep/},{$push:{project:{$each:[{name:"Project-3",Hrs:2},{name:"Project-4",Hrs:2}]}}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

4. Add bonus rs 2000 for all employees with salary > 50000

updateMany is a method that allows updating multiple documents in a collection at once, so it doesn't require the multi: true option.

Ans:

> db.emp.update({sal:{$gt:50000}},{$inc:{sal:2000}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.emp.updateMany({sal:{$gt:50000}},{$inc:{sal:2000}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

>

5. Add bonus rs 1500 if salary <50000 and > 30000

Ans:

> db.emp.updateMany({sal:{$lt:50000,$gt:3000}},{$inc:{sal:1500}})

{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }

>

6. increment bounus by 1000 for all employees if salary <=30000

Ans:

> db.emp.updateMany({sal:{$gte:30000}},{$inc:{sal:1000}})

{ "acknowledged" : true, "matchedCount" : 4, "modifiedCount" : 4 }

>

7. Change manager name to Tushar for all employees whose manager is currently “sathish”

And manager number to 3333

Ans:

> db.emp.updateMany({'mgr.name':"Sathish"},{$set:{'mgr.name':"Tushar",'mgr.num':3333}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

8. Increase salary of all employees from “purchase department” by 15000

Ans:

> db.emp.updateMany({'dept.dname':"Purchase"},{$inc:{sal:15000}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

>

9. Decrease number of hrs by 2 for all employees who are working on project-2

Ans:

> db.emp.updateMany({'project.name':"Project-2"},{$inc:{'project.$.Hrs':-2}})

{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }

>

10. Delete project-3 from all employee document if they are working on the project for 4

Hrs.

Ans:

> db.emp.updateMany({"project.name":"Project-3","project.Hrs":2},{$unset:{"project.$":1}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 } {Deleted 1st appearance}

>

This will delete the document db.emp.deleteMany({'project':{elemMatch:{'name':"Project-2","Hrs":2}}})

11. Change the salary of employees to 10000 only if their salary is < 30000

Ans:

> db.emp.update({sal:{$lt:30000}},{$set:{sal:10000}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

12. Increase bonus of all employees by 500 if the bonus is <2000 or their salary is <

20000 or if employee belong to sales department

Ans:

> db.emp.updateMany({},{$set:{'bonus':500}})

{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 5 }

>

13. Add 2 new project at position 2 for all employees with designation analyst or salary is

equal to either 30000 or 33000 or 35000

Ans:

db.emp.update({$or:[{desg:"Analyst"},{sal:{$in:[10000,30000,33000,35000]}}]},{push:{project:{$each:[{name:"Project-7",Hrs:0,position:1},{name:"Project-8",Hrs:0,position:2}]}}},{multi:true})

WriteResult({

"nMatched" : 0,

"nUpserted" : 0,

"nModified" : 0,

"writeError" : {

"code" : 9,

"errmsg" : "multi update only works with $ operators"

}

})

> db.emp.update({$or:[{"Desg":"Analyst"},{sal:{$in:[10000,62000,40000]}}]},{$push:{project:{$each:[{name:"Project-7",Hrs:0},{name:"Project-8",Hrs:0}],$position:1}}},{multi:true})

WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })

14. Delete last project of all employees with department name is “HR” and if the location

is Mumbai

Ans:

> db.emp.update({'dept.dname':"HR",'dept.dloc':"Mumbai"},{$pop:{project:1}},{multi:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

15. Change designation of all employees to senior programmer if they are working on

name:”Project-1” for 6 hrs

Ans:

> db.emp.update({'project.name':"Project-1",'project.Hrs':6},{$set:{Desg:"Senior Programmer"}},{multi:true})

WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })

>

16. Add list of hobbies in all employees document whose manager is Rajan or Revati

Ans:

> db.emp.update({$or:[{'mgr.name':"Rajan"},{'mgr.name':"Revati"}]},{$set:{'Hobbies':["swim","chess","read"]}},{multi:true,upsert:true})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>

17. Add list of skillset in all employee documents who are working on project-4 for 3 hrs

or on project-3 for 4 hrs

Ans:

> db.emp.update({$or:[{'project.name':"Project-4",'project.Hrs':3},{'project.name':"Project-4",'project.Hrs':3}]},{$set:{'Skillset':["C++","Java"]}},{multi:true,upsert:true})

WriteResult({

"nMatched" : 0,

"nUpserted" : 1,

"nModified" : 0,

"\_id" : ObjectId("64419a16e09dff85147d36f7")

})

> db.emp.update({$or:[{'project.name':"Project-4",'project.Hrs':3},{'project.name':"Project-4",'project.Hrs':3}]},{$set:{'Skillset':["C++","Java"]}},{

multi:true})

WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })

>

18. 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

Ans:

> db.emp.update({ename:/^[Rp].\*[js]$/},{$push:{'Hobbies':{$each:["Blogging"],$position:2}}},{multi:true,upsert:true})

WriteResult({

"nMatched" : 0,

"nUpserted" : 1,

"nModified" : 0,

"\_id" : ObjectId("64419d1fe09dff85147d370a")

})

> db.emp.update({ename:/^[Rp].\*[js]$/},{$push:{'Hobbies':{$each:["Blogging"],$position:2}}},{multi:true})

WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })

>

19. Increase salary by 10000 for all employees who are working on project-2 or project-3

or project-1.

Ans:

> db.emp.updateMany({'project.name':{$in:["Project-1","Project-2","Project-3"]}},{$inc:{sal:10000}})

{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 5 }

>

19.1 Decrease bonus by 1000 rs And increase salary by 1000rs for all employees whose

department location is Mumbai.

Ans:

> db.emp.updateMany({'dept.dloc':"Mumbai"},{$inc:{bonus:-1000,sal:1000}})

{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 5 }

> db.emp.find().pretty()

{

"\_id" : ObjectId("6441735ab83879b38a19db69"),

"empno" : 111,

"ename" : "Deepali",

"sal" : 53500,

"dept" : {

"deptno" : 12,

"dname" : "HR",

"dloc" : "Mumbai"

},

"Desg" : "Analyst",

"mgr" : {

"name" : "Tushar",

"num" : 3333

},

"project" : [

{

"name" : "Project-1",

"Hrs" : 5

},

{

"name" : "Project-7",

"Hrs" : 0

},

{

"name" : "Project-8",

"Hrs" : 0

},

null,

{

"name" : "Project-4",

"Hrs" : 2

},

null

],

"bonus" : 0

}

{

"\_id" : ObjectId("6441736cb83879b38a19db6a"),

"empno" : 112,

"ename" : "Harshal",

"sal" : 48500,

"dept" : {

"deptno" : 13,

"dname" : "Finance",

"dloc" : "Mumbai"

},

"Desg" : "(AO )Admin Officer",

"mgr" : {

"name" : "Revati",

"num" : 112

},

"project" : {

"name" : "Project-1",

"Hrs" : 8

},

"bonus" : 0,

"Hobbies" : [

"swim",

"chess",

"read"

]

}

{

"\_id" : ObjectId("64417372b83879b38a19db6b"),

"empno" : 113,

"ename" : "Harshali",

"sal" : 62000,

"dept" : {

"deptno" : 11,

"dname" : "Development",

"dloc" : "Mumbai"

},

"Desg" : "Architect",

"mgr" : {

"name" : "Ayush",

"num" : 113

},

"project" : [

{

"name" : "Project-1",

"Hrs" : 8

}

],

"bonus" : 0

}

{

"\_id" : ObjectId("64417379b83879b38a19db6c"),

"empno" : 114,

"ename" : "Rahul",

"sal" : 21000,

"dept" : {

"deptno" : 11,

"dname" : "Sales",

"dloc" : "Mumbai"

},

"Desg" : "Senior Programmer",

"mgr" : {

"name" : "Suyash",

"num" : 114

},

"project" : [

{

"name" : "Project-1",

"Hrs" : 6

},

{

"name" : "Project-7",

"Hrs" : 0

},

{

"name" : "Project-8",

"Hrs" : 0

},

{

"name" : "Project-2",

"Hrs" : 0

}

],

"bonus" : 0

}

{

"\_id" : ObjectId("6441737fb83879b38a19db6d"),

"empno" : 115,

"ename" : "Virat",

"sal" : 331000,

"dept" : {

"deptno" : 11,

"dname" : "Purchase",

"dloc" : "Mumbai"

},

"Desg" : "Senior Programmer",

"project" : [

{

"name" : "Project-1",

"Hrs" : 6

},

{

"name" : "Project-2",

"Hrs" : 0

}

],

"bonus" : 0

}

{

"\_id" : ObjectId("64419a16e09dff85147d36f7"),

"Skillset" : [

"C++",

"Java"

]

}

{

"\_id" : ObjectId("64419d1fe09dff85147d370a"),

"Hobbies" : [

"Blogging"

]

}

20. Remove all employees working on project-1

Ans:

> db.emp.deleteMany({"project.name":"Project-1"})

{ "acknowledged" : true, "deletedCount" : 5 }

>

21. Replace document of employee with name “Rahul” to some new document

Ans:

> db.emp.replaceOne({ename:"Rahul"},{empno:120,ename:"Dipak",sal:55000,dept:{deptno:10,dname:"Sales",dloc:"Pune"},Desg:"HR",mgr:{name:"Akash",num:123

,project:{name:"Project-9",hrs:8}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

>

22. Change skill python to python 3.8 for all employees if python is there in the skillset

Ans:

> db.emp.update( { Skillset: "python" }, { $set: { "Skillset.$": "Python 3.8" } },{ multi: true, upsert: true});

23. Add 2 skills MongoDb and Perl at the end of skillset array for all employees who are

working at Mumbai location

Ans:

db.emp.update({'dept.dloc':"Mumbai"},{$push:{Skillset:{$each:["MongoDB","Perl"]}}},{multi:true})

24. Delete first hobby from hobby array for all employees who are working on project-1

or project-2

Ans:

db.emp.update({'project.name':{$in:["Project-1","Project-2"]}},{$pop:{Skillset:-1}},{multi:true})

25. Delete last hobby from hobbies array for all employees who are working on project

which is at 2 nd position in projects array for 4 hrs

Ans:

> db.emp.update({'project.1.hrs':4},{$pop:{Hobbies:1}},{multi:true})

WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })

>

26. Add 2 new projects at the end of array for all employees whose skillset contains Perl

or python

Ans:

> db.emp.update({Skillset:{$in:["Perl","Python"]},"project": {$type: "array"}}, {$push:{project:{$each:[{name:"Project-3",Hrs:3},{name:"Project-4",Hrs

:5}]}}},{multi:true})

WriteResult({ "nMatched" : 6, "nUpserted" : 0, "nModified" : 6 })

>

27. Change hrs to 6 for project-1 for all employees if they working on the project-1 for <

6 hrs. otherwise keep the existing value.

Ans:

db.emp.update({"project.name": "Project-1","project.Hrs": { $lt: 6 }},{$set: { "project.$.Hrs": 6 }},{multi:true})

WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })

28.creates location kv pair under project array

Ans:

>db.emp.update({ "project.name": "Project-2" },{ $set: { "project.$.loc": "pune" } })