Create a database called COMPANY consisting of two tables: - EMP & DEPT

EMP:

Column name	Data type	Description
EMPNO	Number	Employee number
ENAME	Varchar	Employee name
JOB	Char	Designation
MGR	Number	Manager’s Emp. Number
HIREDATE	Date	Date of joining
SAL	Number	Basic Salary
COMM	Number	Commission
DEPTNO	Number	Department Number

DEPT:

Column name	Data type	Description
DEPTNO	Number	Department number
DNAME	Varchar	Department name
LOC	Varchar	Location of department

Data for EMP

7369	Smith	Clerk	7902	17/12/80	800	300	20
7499	Allen	Salesman	7698	20/2/81	1600	300	30

Data for DEPT table

10	Accounting	New York
20	Research	Dallas

30	Sales	Chicago
40	Operations	Boston

```
mysql> select * from emp;

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

| empno | ename | job      | mgr  | hiredate   | sal  | comm | deptno |

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

|  7369 | smith | clerk    | 7902 | 1980-12-17 |  800 |  300 |     20 |

|  7499 | allen | salesman | 7698 | 1981-02-20 | 1600 |  300 |     30 |

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

mysql> select * from dept;

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

| deptno | dname      | loc      |

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

|     10 | accounting | new york |

|     20 | research   | dallas   |

|     30 | sales      | chicago  |

|     40 | operations | boston   |

+--------+------------+----------+
```

Solve below queries by using MySQL

1. List employees not belonging to department 30, 40, or 10.
   mysql> select * from emp where deptno not in (30,40,10);
   ```
   +-------+-------+-------+------+------------+------+------+--------+
   | empno | ename | job   | mgr  | hiredate   | sal  | comm | deptno |
   +-------+-------+-------+------+------------+------+------+--------+
   |  7369 | smith | clerk | 7902 | 1980-12-17 |  800 |  300 |     20 |
   +-------+-------+-------+------+------------+------+------+--------+
   ```
2. List the different designations in the company.
   mysql> select job from emp;
   ```
   +----------+
   ```

```
| job      |
+----------+
| clerk    |
| salesman |
+----------+
```

3. List the names of employees who are not eligible for commission.
mysql> select * from emp where comm is null;
4. List employees whose names either start or end with "S".
mysql> select * from emp where ename like 's%' or ename like '%s';

```
+-------+-------+-------+------+------------+------+------+--------+
| empno | ename | job   | mgr  | hiredate   | sal  | comm | deptno |
+-------+-------+-------+------+------------+------+------+--------+
|  7369 | smith | clerk | 7902 | 1980-12-17 |  800 |  300 |     20 |
+-------+-------+-------+------+------------+------+------+--------+
```

5. List employees whose names have letter "A" as second letter" in their names.
mysql> select * from emp where ename like '_s%';
6. List the number of employees working with the company.
mysql> select count(*) from emp;

```
+----------+
| count(*) |
+----------+
|        2 |
+----------+
```

7. List the number of employees and average salary for employees in department 20.
mysql> select count(*),avg(sal) from emp where deptno = 20;

```
+----------+----------+
| count(*) | avg(sal) |
+----------+----------+
|        1 | 800.0000 |
+----------+----------+
```

8. List name, salary and PF amount of all employees. (PF is calculated as 10% of basic salary)
mysql> select ename,sal,sal*0.1 as pf from emp;

```
+-------+------+-------+
| ename | sal  | pf    |
+-------+------+-------+
| smith |  800 |  80.0 |
| allen | 1600 | 160.0 |
+-------+------+-------+
```

9. List names of employees who are more than 2 years old in the company.
mysql> select * from emp where datediff(curdate(),hiredate)/365 > 2;

```
+-------+-------+----------+------+------------+------+------+--------+
| empno | ename | job      | mgr  | hiredate   | sal  | comm | deptno |
+-------+-------+----------+------+------------+------+------+--------+
|  7369 | smith | clerk    | 7902 | 1980-12-17 |  800 |  300 |     20 |
|  7499 | allen | salesman | 7698 | 1981-02-20 | 1600 |  300 |     30 |
+-------+-------+----------+------+-----------+------+------+--------+
```

10. List the employee details in the ascending order of their basic salary.

mysql> SELECT * from emp ORDER BY sal ASC;

```
+-------+-------+----------+------+------------+------+------+--------+
| empno | ename | job      | mgr  | hiredate   | sal  | comm | deptno |
+-------+-------+----------+------+------------+------+------+--------+
|  7369 | smith | clerk    | 7902 | 1980-12-17 |  800 |  300 |     20 |
|  7499 | allen | salesman | 7698 | 1981-02-20 | 1600 |  300 |     30 |
+-------+-------+----------+------+------------+------+------+--------+
```

11. List the department numbers and number of employees in each department.

mysql> select deptno,count(*) as count from emp group by deptno;

```
+--------+-------+
| deptno | count |
+--------+-------+
|     20 |     1 |
|     30 |     1 |
+--------+-------+
```

12. List the total salary, maximum and minimum salary and average salary of the employees, for department 20.

mysql> select sum(sal),max(sal),min(sal),avg(sal) from emp where deptno = 20;

```
+----------+----------+----------+----------+
| sum(sal) | max(sal) | min(sal) | avg(sal) |
+----------+----------+----------+----------+
|      800 |      800 |      800 | 800.0000 |
+----------+----------+----------+----------+
```

13. Display the employees who have more salary as that of Smith

mysql> select * from emp having sal > (select sal from emp where ename = 'smith');

```
+-------+-------+----------+------+------------+------+------+--------+
| empno | ename | job      | mgr  | hiredate   | sal  | comm | deptno |
+-------+-------+----------+------+------------+------+------+--------+
|  7499 | allen | salesman | 7698 | 1981-02-20 | 1600 |  300 |     30 |
+-------+-------+----------+------+------------+------+------+--------+
```

A2_1

Solve DDL Queries using following database

EMP:

Column name	Data type	Description
EMPNO	Number	Employee number
ENAME	Varchar	Employee name
JOB	Char	Designation
MGR	Number	Manager's Emp. Number
HIREDATE	Date	Date of joining
SAL	Number	Basic Salary
COMM	Number	Commission
DEPTNO	Number	Department Number

DEPT:

Column name	Data type	Description
DEPTNO	Number	Department number
DNAME	Varchar	Department name
LOC	Varchar	Location of department

Data for EMP

7369	Smith	Clerk	7902	17/12/80	800	300	20
7499	Allen	Salesman	7698	20/2/81	1600	300	30

Data for DEPT table

10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston

Create above tables with appropriate constraints like primary key, unique,default,foreign key, check constraints, not null

1. Create a database Called COMPANY consisting of two tables - EMP & DEPT
   mysql> create database company;
   mysql> use company;

   mysql> create table dept(deptno int,dname varchar(20) NOT NULL,loc varchar(20) NOT NULL,PRIMARY KEY(deptno));

   mysql> create table emp(empno int,ename varchar(20) not null,job varchar(20) not null,mgr int,hiredate date,sal decimal(15,2) check (sal>=0),comm int check (comm>=0),deptno int references dept(deptno),primary key(empno));
2. Alter EMP table to add one more column AADHAR CARD no.(integer)
   mysql> alter table emp add aadhar int;
   mysql> alter table emp add unique(aadhar);

3. Alter EMP table to change datatype of AADHAR CARD from integer to varchar.
   mysql> alter table emp modify aadhar varchar(15);
```
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| empno    | int(11)       | NO   | PRI | NULL    |       |
| ename    | varchar(20)   | NO   |     | NULL    |       |
| job      | varchar(20)   | NO   |     | NULL    |       |
| mgr      | int(11)       | YES  |     | NULL    |       |
| hiredate | date          | YES  |     | NULL    |       |
| sal      | decimal(15,2) | YES  |     | NULL    |       |
| comm     | int(11)       | YES  |     | NULL    |       |
| deptno   | int(11)       | YES  |     | NULL    |       |
| aadhar   | varchar(15)   | YES  | UNI | NULL    |       |
+----------+---------------+------+-----+---------+-------+
```
4. Drop AADHAR CARD column
   mysql> select * from emp;
```
+-------+-------+----------+------+------------+---------+------+--------+
| empno | ename | job      | mgr  | hiredate   | sal     | comm | deptno |
+-------+-------+----------+------+------------+---------+------+--------+
|  7369 | smith | clerk    | 7902 | 1980-12-17 |  800.00 |  300 |     20 |
|  7499 | allen | salesman | 7698 | 1981-02-20 | 1600.00 |  300 |     30 |
+-------+-------+----------+------+------------+---------+------+--------+
```
5. Truncate Department table.
   mysql> truncate table dept;
   mysql> select * from dept;
   Empty set (0.00 sec)
6. Drop Department table
   mysql> drop table dept;
7. Create Department table with Deptno as Auto increment option.
   mysql> CREATE TABLE dept (deptno int NOT NULL AUTO_INCREMENT, dname varchar(50), loc varchar(50), PRIMARY KEY (deptno));
8. Alter employee table to add department no as foreign key.
   mysql> alter table emp add foreign key(deptno) references dept(deptno);
9. Create index on Salary
   mysql> create index idx_sal on emp(sal);
10. Create view for Employees who are belongs to department 20.
```
mysql> create view empof20
    -> as select * from emp where deptno = 20;
```

mysql> select * from empof20;

```
+-------+-------+-------+------+------------+--------+------+--------+
| empno | ename | job   | mgr  | hiredate   | sal    | comm | deptno |
+-------+-------+-------+------+------------+--------+------+--------+
|  7369 | smith | clerk | 7902 | 1980-12-17 | 800.00 |  300 |     20 |
+-------+-------+-------+------+------------+--------+------+--------+
```

Assignment 3_1

1. Create following Tables

cust_mstr(cust_no,fname,lname)

add_dets(cust_no,add1,add2,state,city,pincode)

Retrieve the address of customer Fname as 'xyz' and Lname as 'pqr'

mysql> select * from cust_mstr;

```
+--------+----------+-----------+
| custno | fname    | lname     |
+--------+----------+-----------+
|      1 | xyz      | pqr       |
|      2 | shubh    | nagargoje |
|      3 | alex     | goot      |
|      4 | hannibal | lecter    |
+--------+----------+-----------+
```

mysql> select * from add_dets;

```
+--------+--------------+---------------+-------------+------+---------+
| custno | add1         | add2          | state       | city | pincode |
+--------+--------------+---------------+-------------+------+---------+
|      1 | bavdhan      | khurd         | maharashtra | pune |  411021 |
|      2 | model colony | shivaji nagar | maharashtra | pune |  411038 |
|      3 | model colony | shivaji nagar | maharashtra | pune |  411038 |
|      4 | bavdhan      | khurd         | maharashtra | pune |  411038 |
+--------+--------------+---------------+-------------+------+---------+
```

mysql> select add1,add2 from add_dets inner join cust_mstr on add_dets.custno = cust_mstr.custno where cust_mstr.fname = 'xyz' and cust_mstr.lname = 'pqr';

```
+---------+-------+
| add1    | add2  |
+---------+-------+
| bavdhan | khurd |
+---------+-------+
```

2.Create following Tables

cust_mstr(custno,fname,lname)

acc_fd_cust_dets(custno,acc_fd_no)

fd_dets(acc_fd_no,amt)

List the customer holding fixed deposit of amount more than 5000

mysql> select * from acc_fd_cust_dets;

```
+--------+-----------+
| custno | acc_fd_no |
+--------+-----------+
|      2 |       243 |
|      4 |       244 |
+--------+-----------+
```

mysql> select * from fd_dets;

```
+-----------+---------+
| acc_fd_no | amt     |
+-----------+---------+
|       243 | 7000.00 |
|       244 | 4000.00 |
+-----------+---------+
```

mysql> select a.custno,a.fname,a.lname,c.amt from cust_mstr as a inner join acc_fd_cust_dets as b on a.custno=b.custno inner join fd_dets as c on b.acc_fd_no = c.acc_fd_no where c.amt > 5000;

```
+--------+-------+-----------+---------+
| custno | fname | lname     | amt     |
+--------+-------+-----------+---------+
|      2 | shubh | nagargoje | 7000.00 |
+--------+-------+-----------+---------+
```

3. Create following Tables

emp_mstr(e_mpno,f_name,l_name,m_name,dept,desg,branch_no) branch_mstr(name,branch_no)

List the employee details along with branch names to which they belong

>select * fom emp_mstr inner join branch_mstr

on emp_mstr.branch_no=branch_mstr.branch_no;

mysql> select * from emp_mstr;

```
+-------+---------+-----------+-------+------------+-----------+------+
| empno | fname   | lname     | mname | dept       | desg      | bno  |
+-------+---------+-----------+-------+------------+-----------+------+
|   101 | shubham | nagargoje | manik | automation | developer |    1 |
|   102 | shivam  | nagargoje | manik | analytics  | analyst   |    2 |
+-------+---------+-----------+-------+------------+-----------+------+
```

mysql> select * from branch_mstr;

```
+---------------+------+---------+
| name          | bno  | pincode |
+---------------+------+---------+
| fortune plaza |    1 |  411021 |
| hitec park    |    2 |  411001 |
+---------------+------+---------+
```

mysql> select a.*,b.name from emp_mstr as a inner join branch_mstr as b on a.bno = b.bno;

```
+-------+---------+-----------+-------+------------+-----------+------+---------------+
| empno | fname   | lname     | mname | dept       | desg      | bno  | name          |
+-------+---------+-----------+-------+------------+-----------+------+---------------+
|   101 | shubham | nagargoje | manik | automation | developer |    1 | fortune plaza |
|   102 | shivam  | nagargoje | manik | analytics  | analyst   |    2 | hitec park    |
+-------+---------+-----------+-------+------------+-----------+------+---------------+
```

4. Create following Tables

emp_mstr(emp_no,f_name,l_name,m_name,dept)

cntc_dets(code_no,cntc_type,cntc_data)

List the employee details along with contact details using left outer join & right join

mysql> select a.*,b.cntc_type,b.cntc_data from emp_mstr as a left join cntc_dets as b on a.empno = b.empno

-> union

-> select a.*,b.cntc_type,b.cntc_data from emp_mstr as a right join cntc_dets as b on a.empno = b.empno;

```
+-------+---------+-----------+-------+------------+-----------+------+-----------+-------------------+
| empno | fname   | lname     | mname | dept       | desg      | bno  | cntc_type | cntc_data         |
+-------+---------+-----------+-------+------------+-----------+------+-----------+-------------------+
|   101 | shubham | nagargoje | manik | automation | developer |    1 | email     | shubham@gmail.com |
|   102 | shivam  | nagargoje | manik | analytics  | analyst   |    2 | mobile    | 9067777444        |
+-------+---------+-----------+-------+------------+-----------+------+-----------+-------------------+
```

5. Create following Tables

cust_mstr(cust_no,fname,lname)

add_dets(code_no,pincode)

List the customer who do not have bank branches in their vicinity.

mysql> select a.custno,a.fname,a.lname from (select a.*,c.pincode from cust_mstr as a inner join add_dets as b on a.custno = b.custno left join branch_mstr as c on b.pincode = c.pincode) as a where pincode is null;

```
+--------+----------+-----------+
| custno | fname    | lname     |
+--------+----------+-----------+
|      2 | shubh    | nagargoje |
|      3 | alex     | goot      |
|      4 | hannibal | lecter    |
+--------+----------+-----------+
```

6. a) Create View on borrower table by selecting any two columns and perform insert update delete operations

mysql> create view borrower as select fname,lname from cust_mstr;

mysql> select * from borrower;

```
+----------+-----------+
| fname    | lname     |
+----------+-----------+
| xyz      | pqr       |
| shubh    | nagargoje |
| alex     | goot      |
| hannibal | lecter    |
+----------+-----------+
```

<u>mysql> insert into borrower values("rupali","phad");</u>

<u>mysql> update borrower set fname = "walter" where fname = "alex";</u>

<u>mysql> delete from borrower where fname = "hannibal";</u>

<u>mysql> select * from borrower;</u>

```
+--------+-----------+
| fname  | lname     |
+--------+-----------+
| xyz    | pqr       |
| shubh  | nagargoje |
| walter | goot      |
| rupali | phad      |
+--------+-----------+
```

b) Create view on borrower and depositor table by selecting any one column from each table

perform insert update delete operations

<u>mysql> select * from combine_view;</u>

```
+-----------+-----------+
| lname     | desg      |
+-----------+-----------+
| pqr       | developer |
| pqr       | analyst   |
| nagargoje | developer |
| nagargoje | analyst   |
| goot      | developer |
| goot      | analyst   |
| phad      | developer |
| phad      | analyst   |
+-----------+-----------+
```

mysql> insert into combine_view (lname) values("kulkarni");

mysql> insert into combine_view (desg) values("programmer");

mysql> update combine_view set lname = "karad" where lname = "kulkarni";

mysql> select * from combine_view;

```
+-----------+------------+
| lname     | desg       |
+-----------+------------+
| pqr       | developer  |
| pqr       | analyst    |
| pqr       | programmer |
| nagargoje | developer  |
| nagargoje | analyst    |
| nagargoje | programmer |
| goot      | developer  |
| goot      | analyst    |
| goot      | programmer |
| phad      | developer  |
| phad      | analyst    |
| phad      | programmer |
| karad     | developer  |
| karad     | analyst    |
| karad     | programmer |
+-----------+------------+
```

CUSTOMER (<u>CUST_ID</u>, CUST_NAME, ANNUAL_REVENUE, CUST_TYPE)
CUST_ID must be between 100 and 999
ANNUAL_REVENUE defaults to 10000
SHIPMENT (<u>SHIPMENT_ID</u>, CUST_ID, WEIGHT, TRUCK_ID,DESTINATION, SHIP_DATE)
Foreign Key: CUST_ID REFERENCES CUSTOMER,
Foreign Key: TRUCK_# REFERENCES TRUCK
Foreign Key: DESTINATION REFERENCES CITY
WEIGHT defaults to 10
TRUCK (<u>TRUCK_#</u>, DRIVER_NAME)
CITY (<u>CITY_NAME</u>, POPULATION)

Data For Truck Table

Truck #	**Driver Name**
10	Allen
11	Sham
12	Ram
13	Jason

Data For City Table

City Name	**Population**
Pune	6000000
Mumbai	40000000
Aurangabad	3000000
Chennai	8000000

Data For Customer Table

CUST_ID	**CUST_NAME**	**ANNUAL_REVENUE**	**CUST_TYPE**
101	**Rajesh Shah**	**100000**	**Speciality Wholesaler**
153	**Sanjay Surana**	**3400000**	**Retailer**
184	**Komal Malviya**	**96000**	**Drop Ship Wholesalers**
599	**Nitesh Jagdale**	**7800**	**Retailer**
785	**Saurabh Deshpande**	**500000**	**On-line Wholesaler**
986	**Satish Kumar**	**30000**	**Retailer**
200	**Kuber Khanna**	**20083**	**Drop Ship Wholesalers**

Data For Shipment Table

SHIPMENT_ID	**CUST_ID**	**WEIGHT in Kg**	**TRUCK_ID**	**DESTINATION**	**SHIP_DATE**

<u>23463434</u>	**101**	**200**	**10**	**Pune**	**2012-04-10**
<u>58259259</u>	**153**	**20**	**11**	**Mumbai**	**2016-05-31**
<u>39639066</u>	**599**	**180**	**12**	**Pune**	**2007-12-02**
<u>79840975</u>	**101**	**345**	**11**	**Aurangabad**	**2005-06-22**
<u>69045867</u>	**785**	**896**	**10**	**Pune**	**2016-11-30**
<u>72134714</u>	**184**	**23.4**	**13**	**Chennai**	**2017-09-11**
<u>86487655</u>	**599**	**14.56**	**12**	**Chennai**	**2013-05-25**
<u>86248124</u>	**785**	**313.34**	**11**	**Pune**	**2010-01-09**

Solve following Queries:

Create above tables with appropriate constraints like primary key, unique,default,foreign key, check constraints, not null

```
mysql> create table customer(cust_id int(4),cust_name
varchar(20),annual_revenue int(10) default 10000,cust_type
varchar(30),PRIMARY KEY(cust_id));

mysql> create table truck(truck_id int,driver_name varchar(20),PRIMARY
KEY(truck_id));

mysql> create table city(city_name varchar(20),population
int(20),PRIMARY KEY(city_name));

mysql> create table shipment(shipment_id int(20),cust_id int(4),weight
int(4) default 10,truck_id int,destination varchar(20),ship_date
date,PRIMARY KEY(shipment_id),FOREIGN KEY(cust_id) references
customer(cust_id),FOREIGN KEY(truck_id) references
truck(truck_id),FOREIGN KEY(destination) references city(city_name));
```

1) Draw ER Diagram For above Example
2) Draw Schema Diagram of above Example.
3) What are the names of customers who have sent packages (shipments) to Mumbai City?

```
mysql> select cust_name from customer inner join shipment on customer.cust_id =
shipment.cust_id where shipment.destination = "Mumbai";

+---------------+
| cust_name     |
+---------------+
| Sanjay Surana |
+---------------+
```

4) What are the names and populations of cities that have received shipments weighing over 50 Kg?

mysql> select distinct city.* from city inner join shipment on city.city_name = shipment.destination where shipment.weight > 50;

```
+------------+------------+
| city_name  | population |
+------------+------------+
| Aurangabad |    3000000 |
| Pune       |    6000000 |
+------------+------------+
```

5) Who are the customers having over 500 in annual revenue who have sent shipments weighing less than 50 Kg?

mysql> select customer.* from customer inner join shipment on customer.cust_id = shipment.cust_id where customer.annual_revenue > 500 and shipment.weight < 50;

```
+---------+----------------+----------------+-----------------------+
| cust_id | cust_name      | annual_revenue | cust_type             |
+---------+----------------+----------------+-----------------------+
|     153 | Sanjay Surana  |        3400000 | Retailer              |
|     184 | Komal Malviya  |          96000 | Drop Ship Wholesalers |
|     599 | Nitesh Jagdale |           7800 | Retailer              |
+---------+----------------+----------------+-----------------------+
```

6) Who are the customers having over 1000 in annual revenue who have sent shipments weighing less than 10 kg or have sent a shipment to Mumbai?

mysql> select customer.* from customer inner join shipment on customer.cust_id = shipment.cust_id where customer.annual_revenue > 1000 and (shipment.weight < 10 or shipment.destination = "Mumbai");

```
+---------+---------------+----------------+-----------+
| cust_id | cust_name     | annual_revenue | cust_type |
+---------+---------------+----------------+-----------+
|     153 | Sanjay Surana |        3400000 | Retailer  |
+---------+---------------+----------------+-----------+
```

7) Who are the drivers who have delivered shipments for customers with annual revenue over 2000, to cities with populations over 1000?

mysql> select distinct truck.* from truck

```
    -> inner join shipment on truck.truck_id = shipment.truck_id
    -> inner join customer on customer.cust_id = shipment.cust_id
    -> inner join city on city.city_name = shipment.destination
    -> where customer.annual_revenue > 2000 and city.population > 1000;

+----------+-------------+
| truck_id | driver_name |
+----------+-------------+
|       10 | Allen       |
|       12 | Ram         |
|       11 | Sham        |
|       13 | Jason       |
+----------+-------------+
```

8) Display customers who have same Annual Revenue as "Sunil".

```
mysql> select * from customer where annual_revenue = (select annual_revenue
from customer where cust_name = "Sunil");

Empty set (0.00 sec)
```

9) Display shipments with weight greater than average weight of shipments.

```
mysql> select * from shipment where weight > (select avg(weight) from shipment);

+-------------+---------+--------+----------+-------------+------------+
| shipment_id | cust_id | weight | truck_id | destination | ship_date  |
+-------------+---------+--------+----------+-------------+------------+
|    69045867 |     785 | 896.00 |       10 | Pune        | 2016-11-30 |
|    79840975 |     101 | 345.00 |       11 | Aurangabad  | 2005-06-22 |
|    86248124 |     785 | 313.34 |       11 | Pune        | 2010-01-09 |
+-------------+---------+--------+----------+-------------+------------+
```

10) Display no of shipments destination wise and display only those with more than 5 count.

```
mysql> select destination,total from (select destination,count(destination) as total
from shipment group by destination) b where total > 5;

Empty set (0.00 sec)
```

1. Crete procedure to insert 3 rows in Location table. Location ID should be the next no at each insert.

```
create table loc(loc_id int,place varchar(20));
Table created

create sequence loc_seq;
Sequence created

create or replace procedure loct(nm1 varchar2)
is
id1 int;
begin
select loc_seq.nextval into id1 from dual;
insert into loc values(id1,nm1);
end;
Procedure created;

execute loct('karad');
execute loct('pune');
execute loct('mumbai');

statements Processed

select * from loc;

LOC_ID    PLACE
1         karad
2         pune
3         mumbai
```

2. Create a PL/SQL block that computes the commission amount for a given employee based on the employee's salary.
a. If the employee's salary is less than $5,000, display the bonus amount for the employee as 10% of the salary.
b. If the employee's salary is between $5,000 and $10,000, display the bonus amount for the employee as 15% of the salary.
c. If the employee's salary exceeds $10,000, display the bonus amount for the employee as 20% of the salary.
d. If the employee's salary is NULL, display the bonus amount for the employee as 0.
e. Test the PL/SQL block for each case using the following test cases, and check each bonus amount.

```
create table emp(ename varchar(20),eid int,salary int)
Table created.

insert into emp values('kac','04',0);
1 row(s) inserted.
```

```
insert into emp values('jac','01',4000);
1 row(s) inserted.

insert into emp values('hac','02',11000);
1 row(s) inserted.

insert into emp values('dac','03',8000);
1 row(s) inserted.

select * from emp

ENAME     EID       SALARY
kac       4         0
jac       1         4000
hac       2         11000
dac       3         8000



create or replace procedure ebonus(did in int)
as
bonus emp.eid%type;
dsalary emp.eid%type;
dname emp.ename%type;
begin
select emp.salary,emp.ename into dsalary,dname from emp where emp.eid=did;
if dsalary<5000 then
bonus:=dsalary*0.1;
elsif dsalary>=5000 and dsalary<=10000 then
bonus:=dsalary*0.15;
else
bonus:=dsalary*0.2;
end if;

 if dsalary is null then

 bonus:=0;
 end if;
dbms_output.put_line('Name : '||dname||',Salary : '||dsalary||',Bonus : '||bonus);

end;
Procedure created.

execute ebonus(1)
Name : jac,Salary : 4000,Bonus : 400

execute ebonus(2)
Name : hac,Salary : 11000,Bonus : 2200
```

```
execute ebonus(3)
Name : dac,Salary : 8000,Bonus : 1200

execute ebonus(4)
Name : kac,Salary : 0,Bonus : 0
```

3. Create a procedure, NEW_EMP, to insert a new employee into the EMPLOYEES table. The procedure should contain a call to the VALID_DEPTID function to check whether the department ID specified for the new employee exists in the DEPARTMENTS table.
a. Create a function VALID_DEPTID to validate a specified department ID. The function should return a integer value.

```
create table dept(dname);
insert into dept values('comp','civil','mech');

Table created

1 row(s) inserted.

create table emp(ename,dept_nm);
Table created;

select * from dept;
     dname
     civil
     comp
     mech

create or replace function valid_deptid(dnm1 varchar(20))
return number
is
nm1 varchar(20);
begin
select dept.dname into nm1 from dept where dept.dname=dnm1;
if(nm1 is null) then;
return 0;
else
return 1;
end if;
end;

Function created;

create or replace procedure new_emp(enm in varchar(20),dnm in varchar(20))
as
c int;
begin
c :=valid_deptid(dnm);
```

```
if c=1 then
insert into emp values(enm,dnm);
else
dbms_output.put_line('No such department Exists');
end if;
end;

Procedure created;

execute new_emp('jac','civil');
1 row(s) inserted

execute new_emp('bac','entc');
No such department Exists
```

4. Create a procedure to display ename, salary commission of employee. Pass the empid as argument to Procedure. Use Host variables

```
create table emp(eid int,ename varchar2(20),salary int )



Table created

insert into emp values('1','jac','10000';)

insert into emp values('1','bac',8000';)
insert into emp values('1','mac','4000';)

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

select * from emp;

emp_id    ename     salary
1         jac       10000
2         bac       8000
3         hac       4000



Variable dname varchar2(20);
Variable dsalary int;

create or replace procedure display(did in int)
as
```

```
sal int;
nm varchar(20);
begin
select emp.ename,emp.salary into nm,sal from emp where eid=did
:dsalary:=sal;
:dname :=nm;
dbms_output.put_line(' Emp _id :'||:did||' name :'||:dname||' salary :'||:dsalary);
end;

Procedure created

execute display(1);

Emp_id :1 name :jac salary :10000;
```

5. Create a function to calculate a tax of salary.
Pass salary of particular employee as input to function. Function should return the tax for it. Call the function in Select clause.

```
create table emp1(ename varchar(20),salary int)
Table created.

insert into emp1 values('jac ',1000);
insert into emp1 values('dac ',8000);
insert into emp1 values('bac ',6000);

1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.

select * from emp1;

     ENAME     SALARY
     jac  1000
     dac  8000
     bac       6000

create or replace function find_tax(sal in int)
return int
is
tax int;
begin
tax :=sal*0.18;
return (tax) ;
end;

Function created.
```

```
declare
sal1 varchar(20);
nam varchar(20);
d1 int;
begin
select emp1.salary,emp1.ename into sal1,nam from emp1 where emp1.salary=8000 ;
select find_tax(sal1) into d1 from emp1 where emp1.salary=8000  ;
dbms_output.put_line('Name :'||nam||',salary :'||sal1||',Tax :'||d1);
end;
```

```
Name :dac ,salary :8000,Tax :1440
```

Explicit Cursor

1. RETRIEVE EMPLOYEES ONE BY ONE AND PRINT OUT A LIST OF THOSE EMPLOYEES CURRENTLY WORKING IN THE SALES DEPARTMENT (DEPARTMENT_ID = 80).

```
      create table employee(empid number, ename varchar(25), did
number);

      Table created.

      SQL> insert into employee values('1','A','10');

      1 row created.

      SQL> insert into employee values('2','B','20');

      1 row created.

      SQL> insert into employee values('3','C','30');

      1 row created.

      SQL> insert into employee values('4','D','40');

      1 row created.

      SQL> insert into employee values('5','E','80');

      1 row created.
```

```
      **s1.sql**

      create or replace procedure p1 is

      cursor c1 is

            select * from employee where did='80';
      v_rec c1%rowtype;

      begin
            open c1;
            loop
                  fetch c1 into v_rec;
                  exit when c1%NOTFOUND;
                  DBMS_OUTPUT.PUT_LINE(v_rec.empid || v_rec.ename ||
v_rec.did);
            end loop;
            close c1;
      end;
      /

      **

      SQL> start s1.sql
      Procedure created.

      SQL> execute p1
      5E80

      PL/SQL procedure successfully completed.
```

2. USE A CURSOR TO RETRIEVE EMPLOYEE NUMBERS AND NAMES AND POPULATE A DATABASE TABLE, TEMP_LIST, WITH THIS INFORMATION.

```
      SQL> create table templist(eid number,name varchar(25), deptid
number);
```

```
Table created.

**s1.sql**
create or replace procedure p1 is
      cursor c1 is
            select * from employee;
      v_rec c1%rowtype;

begin

      open c1;
      loop
            fetch c1 into v_rec;
            exit when c1%NOTFOUND;
            insert into templist values(v_rec.empid, v_rec.ename,
v_rec.did);
      end loop;
      close c1;
end;
/

****

SQL> start s1.sql
Procedure created.

SQL> execute p1
PL/SQL procedure successfully completed.

SQL> select * from templist;
         EID NAME                 DEPTID
```

```
---------- ------------------------- ----------
         1 A                                10
         2 B                                20
         3 C                                30
         4 D                                40
         5 E                                80
```

*****trigger question******

```
SQL> alter table employee add sal number;

Table altered.

SQL> update employee set sal='1000' where empid=1;

1 row updated.

SQL> update employee set sal=15000 where empid=2;

1 row updated.

SQL> update employee set sal=7000 where empid=3;

1 row updated.

SQL> update employee set sal=3000 where empid=4;

1 row updated.
```

```
SQL> update employee set sal=18000 where empid=5;

1 row updated.

SQL> select * from employee;

     EMPID ENAME                           DID        SAL
---------- ------------------------- ---------- ----------
         1 A                                 10       1000
         2 B                                 20      15000
         3 C                                 30       7000
         4 D                                 40       3000
         5 E                                 80      18000
```

s1.sql

```
create or replace trigger t1
before update on employee
for each row
begin
      if :NEW.SALARY=0 then
            RAISE_APPLICATION_ERROR(-20555,'error');
      end if;
end;
/
```

```
SQL> update employee set salary=0 where empid=1;
update employee set salary=0 where empid=11
```

```
            *
      ERROR at line 1:
      ORA-20555: error
      ORA-06512: at "STUDENT.T1", line 3
      ORA-04088: error during execution of trigger 'STUDENT.T1'
```

****

3. CREATE A PL/SQL BLOCK THAT DETERMINES THE TOP EMPLOYEES WITH RESPECT TO SALARIES.

ACCEPT A NUMBER N FROM THE USER WHERE N REPRESENTS THE NUMBER OF TOP N EARNERS FROM THE

EMPLOYEES TABLE. FOR EXAMPLE, TO VIEW THE TOP FIVE EARNERS, ENTER 5.

THERE SHOULD BE NO DUPLICATION IN THE SALARIES. IF TWO EMPLOYEES EARN THE SAME SALARY, THE SALARY SHOULD BE PICKED UP ONLY ONCE.

TEST A VARIETY OF SPECIAL CASES, SUCH AS N = 0 OR WHERE N IS GREATER THAN THE NUMBER

OF EMPLOYEES IN THE EMPLOYEES TABLE. EMPTY THE TOP_DOGS TABLE AFTER EACH TEST. THE OUTPUT SHOWN REPRESENTS THE FIVE HIGHEST SALARIES IN THE EMPLOYEES TABLE

```
      ***s1.sql***

            create or replace procedure p1(v_num in int) is
                  cursor c1 is select distinct sal from emp order by sal
desc;
                  v_row c1%rowtype;
            begin
                  open c1;
                  for i in 1..v_num loop
```

```
                fetch c1 into v_row;
                exit when c1%NOTFOUND;
                dbms_output.put_line(to_char(v_row.sal));
           end loop;
           if(c1%rowcount< v_num) then
                RAISE_APPLICATION_ERROR(-20505,'count exceeded');
           end if;
      end;
      /
****

SQL> select * from emp;

     EMPID ENAME                       SAL
---------- -------------------- ----------
         1 A                          1000
         2 B                          5000
         3 C                          2000
         4 D                         10000
         5 E                          3500
         6 F                          8700
         7 G                         15000
         8 H                          9000
         9 I                          5000

9 rows selected.

SQL> execute p1(6);
15000
10000
```

```
9000

8700

5000

3500


PL/SQL procedure successfully completed.
```

IMPLICIT CURSOR.

```
SQL> select * from employee;

       EID ENAME                    SALARY     DEPTNO
---------- -------------------- ---------- ----------
       100 xyz                        1000    10
       200 abc                        1500    40
       300 pqr                        2330    80
       400 uvw                        3500    80
       500 ksi                        6500    30
       600 ajs                        4500    90
       700 mvp                        6666    50

7 rows selected.
```

1. DELETE THE EMP WHO ARE WORKING IN DEPARTMENT 80.PRINT THE NO OF ROWS DELETED.

```
***s.sql***

create or replace procedure p is
```

```
      begin
           delete from employee where deptno=80;
           dbms_output.put_line('Number of rows deleted : 
'||to_char(sql%rowcount));
      end;
      /

      SQL> start s.sql;
      Procedure created.

      SQL> execute p;
      Number of rows deleted : 2

      PL/SQL procedure successfully completed.
```

2. RAISE THE SALARY OF EMPLOYEE WORKING IN DEPARTMENT 10.DISPLAY THE NO OF ROWS UPDATED.

```
      ***s.sql***

      create or replace procedure p is
      begin
           update employee set salary=salary*1.1 where deptno=10;
           dbms_output.put_line('Number of rows updated : 
'||to_char(sql%rowcount));
      end;
      /

      SQL> start s.sql;
```

```
Procedure created.

SQL> execute p;
Number of rows updated : 1

PL/SQL procedure successfully completed.
```

TRIGGERS

1) CHANGES TO THE DATA ARE ALLOWED ON THE TABLES ONLY DURING NORMAL OFFICE HOURS OF 08:45 AM UNTIL 5:30 PM MONDAY TO FRIDAY

CREATE A STORED PROCEDURE CALLED SECURE_DML THAT PREVENTS THE DML STATEMENT FROM EXECUTING DURING OUTSIDE OF NORMA;L OFFICE HOURS RETURNING A MESSAGE "U MAY ONLY MAKE THE CHANGES DURING NORMAL OFFICE HOURS"

CREATE A STATEMENT TRIGGER ON JOB TABLE THAT CALLS THE ABOVE PROCEDURE

```
SQL> select * from emp;

ENAME                      EID        SAL JOB
------------------------- ---------- ---------- -----------------
--------
Sid                          10     10000 manager
satish                              20     20000 admin
abc                          30     30000 clerk


***s1.sql***

create or replace trigger t
```

```
      before insert or update or delete on emp
      begin
            if((to_char(sysdate,'dy') in ('sat', 'sun')) or
(to_char(sysdate,'HH24:MI')) not between '08:00' and '17:30') then
            raise_application_error(-20500,'Not Allowed');
            end if;
      end;
      /

      SQL> start s.sql;
      Trigger created.      //DAY IS THU,TIME IS 09:30

      SQL> insert into emp values('Azim', '40', '1200000', 'ADMIN');
      1 row created.

      SQL> start s.sql;
      Trigger created.      //DAY IS SAT, TIME IS 09:30

      SQL> insert into emp values('Mon', '50', '13244', 'clerk');
      insert into emp values('Mon', '50', '13244', 'clerk')
                  *
      ERROR at line 1:
      ORA-20500: Not Allowed
      ORA-06512: at "STUDENT.T", line 3
      ORA-04088: error during execution of trigger 'STUDENT.T'


      SQL> insert into emp values('Mon', '50', '13244', 'clerk');
      1 row created.

      SQL> start s.sql;
```

```
Trigger created.

SQL> insert into emp values('M', '32','3453','clerk');
1 row created.

SQL> select * from emp;

ENAME                  EID      SAL   JOB
------------------------- ---------- ----------
Sid                         10     10000  manager
satish                        20    20000  admin
abc                         30     30000  clerk
Azim                    40  1200000  ADMIN
Mon                         50     13244  clerk
M                           32      3453  clerk

6 rows selected.

SQL> select to_char(sysdate,'HH24:MI') from dual;

TO_CH
-----
07:33

SQL> start s.sql;
Trigger created.

SQL> insert into emp values('A', '34', '45274', 'clerk' );
1 row created.
```

```
SQL> start s.sql;

Trigger created.      //DAY IS THU, TIME IS 07:30


SQL> insert into emp values('D','45','45426','admin');

insert into emp values('D','45','45426','admin')

             *

ERROR at line 1:

ORA-20500: Not Allowed

ORA-06512: at "STUDENT.T", line 3

ORA-04088: error during execution of trigger 'STUDENT.T'
```

2) EMPLOYEE SHOULD RECEIVE AN AUTOMATIC INCREASES IN THE SALARY IF THE MINIMUM SALARY FOR THE JOB IS INCREASED

CREATE A STORED PROCEDURE UPDATE _EMP_SAL TO UPDATE THE SALARY AMOUNT. THIS PROCEDURE ACCEPTS 2 PARAMETERS THE JOBID FOR WHICH THE SALARY HAS TO BE UPDATED AND THE NEW MINIMUM SALARY FOR THIS JOB. THIS PROCEDURE IS EXECUTED FROM THE TRIGGER ON THE JOBS TABLE.

CREATE A ROW TRIGGER NAMED UPDATE_EMP_TRIGGER ON THE JOB'S TABLE THAT INVOKES THE PROCEDURE UPDATE_EMP_SAL WHEN THE MINIMUM SALARY IN JOB'S TABLE IS UPDATED FOR THE SPECIFIED JOBID.

```
SQL> create table job(jobid int primary key, minsal int);

Table created.


SQL> insert into job values('10','500');

1 row created.


SQL> insert into job values('20','2000');

1 row created.
```

```
SQL> insert into job values('30','4500');

1 row created.


SQL> insert into job values('40','9000');

1 row created.


SQL> select * from job;

     JOBID     MINSAL
---------- ----------
        10        500
        20       2000
        30       4500
        40       9000


SQL> update emp set jobid=10 where empid in (1);

1 row updated.


SQL> update emp set jobid=20 where empid in (3,5);

2 rows updated.


SQL> update emp set jobid=30 where empid in (2,6,9);

3 rows updated.


SQL> update emp set jobid=40 where empid in (4,7,8);

3 rows updated.


SQL> select * from emp;


     EMPID ENAME                       SAL      JOBID
---------- -------------------- ---------- ----------
```

```
         1 A                              1000          10
         2 B                              5000          30
         3 C                              2000          20
         4 D                             10000          40
         5 E                              3500          20
         6 F                              8700          30
         7 G                             15000          40
         8 H                              9000          40
         9 I                              5000          30

9 rows selected.
```

s1.sql

```
        create or replace procedure p1(v_jobid in int, v_minsal in
int) is
        cursor c2 is select * from emp;
        v_fetch c2%rowtype;

        begin
             open c2;
             loop
                  fetch c2 into v_fetch;
                  exit when c2%NOTFOUND;
                  if(v_fetch.jobid = v_jobid) then
                       if(v_fetch.sal < v_minsal) then
                            update emp set sal=v_minsal;
                       end if;
                  end if;
             end loop;
```

```
           close c2;

      end;
      /
****

**t.sql**

      create or replace trigger t1
      after update on job for each row
      begin
           execute p1(:new.jobid,:new.minsal);
      end;
      /
****
```

Create Institute Database and Create Student collection with following keys
1. Student Id
2. Student Name
3. Branch
4. Address :{Area, City ,Pin code}
5. Subjects: [{subject name: "DBMS"   ,
                score: 67
                } ,
                {subject name: "TOC"   ,
                score: 56
                }   ]
6. Area of Interest: [" DBMS","Networking".....]

* Enter Subject Names as: 1. DBMSA 2. TOC 3. DC & WSN 4. OSD 5. FC&A

1. Create database Institute.

```
> use Institute
```

```
switched to db Institute
```

2. Create collection Students.

```
> db.createCollection("Students")
```

```
{ "ok" : 1 }
```

3. Insert 10 document with above mentioned structure.

```
>
db.Students.insert([{"stud_id":1,"stud_name":"shubh","branch":"comput
er","address":["kothrud","pune",411038],"subjects":[{"sub_name":"dbms
","score":67},{"sub_name":"toc","score":89}],"area_of_interest":["dbms",
"networking"]},{"stud_id":2,"stud_name":"ruchik","branch":"computer"
,"address":["bavdhan","pune",411021],"subjects":[{"sub_name":"cn","sc
ore":99}],"area_of_interest":["python","mongodb"]},{"stud_id":3,"stud_n
ame":"saurabh","branch":"IT","address":["bavdhan","pune",411021],"s
ubjects":[{"sub_name":"dbms","score":67},{"sub_name":"toc","score":89
}],"area_of_interest":["machine
learning","ai"]},{"stud_id":4,"stud_name":"jacob","branch":"computer",
"address":["mulund","mumbai",400001],"subjects":[{"sub_name":"ads",
"score":67},{"sub_name":"os","score":89}],"area_of_interest":["python",
"mongodb"]},{"stud_id":5,"stud_name":"rushi","branch":"computer","a
ddress":["swargate","pune",411005],"subjects":[{"sub_name":"ads","scor
e":76},{"sub_name":"os","score":78}],"area_of_interest":["data
analytics"]},{"stud_id":6,"stud_name":"shivam","branch":"IT","address"
:["bavdhan","pune",411021],"subjects":[{"sub_name":"ads","score":76},{
"sub_name":"os","score":78}],"area_of_interest":["dbms","gamification"]
```

},{"stud_id":7,"stud_name":"arnav","branch":"computer","address":["juhu","mumbai",400013],"subjects":[{"sub_name":"ads","score":76},{"sub_name":"microprocessor","score":78}],"area_of_interest":["data analytics","os"]},{"stud_id":8,"stud_name":"lucifer","branch":"IT","address":["juhu","mumbai",400013],"subjects":[{"sub_name":"ads","score":76},{"sub_name":"microprocessor","score":78}],"area_of_interest":["python","mongodb"]},{"stud_id":9,"stud_name":"tanmay","branch":"computer","address":["midc","aurangabad",421001],"subjects":[{"sub_name":"ads","score":76},{"sub_name":"microprocessor","score":78}],"area_of_interest":["machine learning"]},{"stud_id":10,"stud_name":"hannibal","branch":"IT","address":["midc","aurangabad",421001],"subjects":[{"sub_name":"ads","score":76},{"sub_name":"microprocessor","score":78}],"area_of_interest":["big data"]}])

4. Display all students information.

```
> db.Students.find().pretty()
{
        "_id" : ObjectId("5ba7de27f12348da7e1a5d44"),
        "stud_id" : 1,
        "stud_name" : "shubh",
        "branch" : "computer",
        "address" : [
                "kothrud",
                "pune",
                411038
        ],
        "subjects" : [
                {
                        "sub_name" : "dbms",
                        "score" : 67
                },
                {
                        "sub_name" : "toc",
                        "score" : 89
                }
        ],
        "area_of_interest" : [
                "dbms",
                "networking"
        ]
}
{
        "_id" : ObjectId("5ba7de27f12348da7e1a5d45"),
        "stud_id" : 2,
        "stud_name" : "ruchik",
		"branch" : "computer",
		"address" : [
			"bavdhan",
			"pune",
			411021
		],
		"subjects" : [
			{
				"sub_name" : "cn",
				"score" : 99
			}
		],
		"area_of_interest" : [
			"python",
			"mongodb"
		]
	}
	{
		"_id" : ObjectId("5ba7de27f12348da7e1a5d46"),
		"stud_id" : 3,
		"stud_name" : "saurabh",
		"branch" : "IT",
		"address" : [
			"bavdhan",
			"pune",
			411021
		],
		"subjects" : [
			{
				"sub_name" : "dbms",
				"score" : 67
			},
			{
				"sub_name" : "toc",
				"score" : 89
			}
		],
		"area_of_interest" : [
			"machine learning",
			"ai"
		]
	}
	{
		"_id" : ObjectId("5ba7de27f12348da7e1a5d47"),
		"stud_id" : 4,
		"stud_name" : "jacob",
		"branch" : "computer",
		"address" : [
                "mulund",
                "mumbai",
                400001
        ],
        "subjects" : [
                {
                        "sub_name" : "ads",
                        "score" : 67
                },
                {
                        "sub_name" : "os",
                        "score" : 89
                }
        ],
        "area_of_interest" : [
                "python",
                "mongodb"
        ]
}
{
        "_id" : ObjectId("5ba7de27f12348da7e1a5d48"),
        "stud_id" : 5,
        "stud_name" : "rushi",
        "branch" : "computer",
        "address" : [
                "swargate",
                "pune",
                411005
        ],
        "subjects" : [
                {
                        "sub_name" : "ads",
                        "score" : 76
                },
                {
                        "sub_name" : "os",
                        "score" : 78
                }
        ],
        "area_of_interest" : [
                "data analytics"
        ]
}
{
        "_id" : ObjectId("5ba7de27f12348da7e1a5d49"),
        "stud_id" : 6,
        "stud_name" : "shivam",
        "branch" : "IT",
        "address" : [
                "bavdhan",
                "pune",
                411021
        ],
        "subjects" : [
                {
                        "sub_name" : "ads",
                        "score" : 76
                },
                {
                        "sub_name" : "os",
                        "score" : 78
                }
        ],
        "area_of_interest" : [
                "dbms",
                "gamification"
        ]
}
{
        "_id" : ObjectId("5ba7de27f12348da7e1a5d4a"),
        "stud_id" : 7,
        "stud_name" : "arnav",
        "branch" : "computer",
        "address" : [
                "juhu",
                "mumbai",
                400013
        ],
        "subjects" : [
                {
                        "sub_name" : "ads",
                        "score" : 76
                },
                {
                        "sub_name" : "microprocessor",
                        "score" : 78
                }
        ],
        "area_of_interest" : [
                "data analytics",
                "os"
        ]
}
{
        "_id" : ObjectId("5ba7de27f12348da7e1a5d4b"),
        "stud_id" : 8,
		"stud_name" : "lucifer",
		"branch" : "IT",
		"address" : [
			"juhu",
			"mumbai",
			400013
		],
		"subjects" : [
			{
				"sub_name" : "ads",
				"score" : 76
			},
			{
				"sub_name" : "microprocessor",
				"score" : 78
			}
		],
		"area_of_interest" : [
			"python",
			"mongodb"
		]
	}
	{
		"_id" : ObjectId("5ba7de27f12348da7e1a5d4c"),
		"stud_id" : 9,
		"stud_name" : "tanmay",
		"branch" : "computer",
		"address" : [
			"midc",
			"aurangabad",
			421001
		],
		"subjects" : [
			{
				"sub_name" : "ads",
				"score" : 76
			},
			{
				"sub_name" : "microprocessor",
				"score" : 78
			}
		],
		"area_of_interest" : [
			"machine learning"
		]
	}
	{
		"_id" : ObjectId("5ba7de27f12348da7e1a5d4d"),
        "stud_id" : 10,
        "stud_name" : "hannibal",
        "branch" : "IT",
        "address" : [
                "midc",
                "aurangabad",
                421001
        ],
        "subjects" : [
                {
                        "sub_name" : "ads",
                        "score" : 76
                },
                {
                        "sub_name" : "microprocessor",
                        "score" : 78
                }
        ],
        "area_of_interest" : [
                "big data"
        ]
}
```

5. Update student branch from IT to Computer of studentid 3.

   > db.Students.update({"stud_id":3},{$set:{"branch":"IT"}})

   WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })

6. Add interest Python in studentid 5.

   > db.Students.update({"stud_id":5},{$push:{"area_of_interest":"python"}})

   WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

7. Add one subject name and its score for Student Id 8.

   > db.Students.update({"stud_id":8},{$push:{"subjects":{"sub_name":"coa","score":89}}})

   WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

8. Change City name from Mumbai to Delhi

   > db.Students.update({"stud_id":8,"address":"mumbai"},{$set:{"address.$":"delhi"}})

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

9. Remove record with student id 3.

> db.Students.remove({"stud_id":3})

```
WriteResult({ "nRemoved" : 1 })
```

10. Add new Key "Hobbies" with values

> db.Students.update({},{$set:{"hobbies":"coding"}},{upsert:false,multi:true})

```
WriteResult({ "nMatched" : 9, "nUpserted" : 0, "nModified" : 9 })
```

11. Display students staying in Pune city.

> db.Students.find({"address":"pune"})

```
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d44"), "stud_id" : 1, "stud_name" :
"shubh", "branch" : "computer", "address" : [ "kothrud", "pune", 411038 ],
"subjects" : [ { "sub_name" : "dbms", "score" : 67 }, { "sub_name" : "toc",
"score" : 89 } ], "area_of_interest" : [ "dbms", "networking" ], "hobbies" :
"coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d45"), "stud_id" : 2, "stud_name" :
"ruchik", "branch" : "computer", "address" : [ "bavdhan", "pune", 411021 ],
"subjects" : [ { "sub_name" : "cn", "score" : 99 } ], "area_of_interest" : [
"python", "mongodb" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d48"), "stud_id" : 5, "stud_name" :
"rushi", "branch" : "computer", "address" : [ "swargate", "pune", 411005 ],
"subjects" : [ { "sub_name" : "ads", "score" : 76 }, { "sub_name" : "os", "score" :
78 } ], "area_of_interest" : [ "data analytics", "python" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d49"), "stud_id" : 6, "stud_name" :
"shivam", "branch" : "IT", "address" : [ "bavdhan", "pune", 411021 ], "subjects" :
[ { "sub_name" : "ads", "score" : 76 }, { "sub_name" : "os", "score" : 78 } ],
"area_of_interest" : [ "dbms", "gamification" ], "hobbies" : "coding" }
```

12. Display students staying in Pune or Mumbai City.

> db.Students.find({$or:[{"address":"pune"},{"address":"mumbai"}]})

```
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d44"), "stud_id" : 1, "stud_name" :
"shubh", "branch" : "computer", "address" : [ "kothrud", "pune", 411038 ],
"subjects" : [ { "sub_name" : "dbms", "score" : 67 }, { "sub_name" : "toc",
"score" : 89 } ], "area_of_interest" : [ "dbms", "networking" ], "hobbies" :
"coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d45"), "stud_id" : 2, "stud_name" :
"ruchik", "branch" : "computer", "address" : [ "bavdhan", "pune", 411021 ],
"subjects" : [ { "sub_name" : "cn", "score" : 99 } ], "area_of_interest" : [
"python", "mongodb" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d47"), "stud_id" : 4, "stud_name" :
"jacob", "branch" : "computer", "address" : [ "mulund", "mumbai", 400001 ],
"subjects" : [ { "sub_name" : "ads", "score" : 67 }, { "sub_name" : "os", "score" :
89 } ], "area_of_interest" : [ "python", "mongodb" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d48"), "stud_id" : 5, "stud_name" :
"rushi", "branch" : "computer", "address" : [ "swargate", "pune", 411005 ],
"subjects" : [ { "sub_name" : "ads", "score" : 76 }, { "sub_name" : "os", "score" :
78 } ], "area_of_interest" : [ "data analytics", "python" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d49"), "stud_id" : 6, "stud_name" :
"shivam", "branch" : "IT", "address" : [ "bavdhan", "pune", 411021 ], "subjects" :
[ { "sub_name" : "ads", "score" : 76 }, { "sub_name" : "os", "score" : 78 } ],
"area_of_interest" : [ "dbms", "gamification" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d4a"), "stud_id" : 7, "stud_name" :
"arnav", "branch" : "computer", "address" : [ "juhu", "mumbai", 400013 ],
"subjects" : [ { "sub_name" : "ads", "score" : 76 }, { "sub_name" :
"microprocessor", "score" : 78 } ], "area_of_interest" : [ "data analytics", "os" ],
"hobbies" : "coding" }
```

13. Display students with area of interest Python and MongoDB.

```
> db.Students.find({$and:[{"area_of_interest":"python"},{"area_of_interest":"mongodb"}]})
```

```
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d45"), "stud_id" : 2, "stud_name" :
"ruchik", "branch" : "computer", "address" : [ "bavdhan", "pune", 411021 ],
"subjects" : [ { "sub_name" : "cn", "score" : 99 } ], "area_of_interest" : [
"python", "mongodb" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d47"), "stud_id" : 4, "stud_name" :
"jacob", "branch" : "computer", "address" : [ "mulund", "mumbai", 400001 ],
"subjects" : [ { "sub_name" : "ads", "score" : 67 }, { "sub_name" : "os", "score" :
89 } ], "area_of_interest" : [ "python", "mongodb" ], "hobbies" : "coding" }
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d4b"), "stud_id" : 8, "stud_name" :
"lucifer", "branch" : "IT", "address" : [ "juhu", "delhi", 400013 ], "subjects" : [ {
"sub_name" : "ads", "score" : 76 }, { "sub_name" : "microprocessor", "score" : 78
}, { "sub_name" : "coa", "score" : 89 } ], "area_of_interest" : [ "python",
"mongodb" ], "hobbies" : "coding" }
```

14. Display students with branch IT and area of interest.

```
> db.Students.find({$and:[{"branch":"IT"},{"area_of_interest":"gamification"}]})
```

```
{ "_id" : ObjectId("5ba7de27f12348da7e1a5d49"), "stud_id" : 6, "stud_name" :
"shivam", "branch" : "IT", "address" : [ "bavdhan", "pune", 411021 ], "subjects" :
[ { "sub_name" : "ads", "score" : 76 }, { "sub_name" : "os", "score" : 78 } ],
"area_of_interest" : [ "dbms", "gamification" ], "hobbies" : "coding" }
```

15. Drop collection.

```
> db.Students.drop()

true
```

Use Zips database

1. Import above database in MongoDB by using mongoimport.

```
shubham@shubham-Inspiron-14-3452:~$ sudo mongoimport --db zip --collection zipdata --file zips.json
2018-09-26T14:08:41.929+0530	connected to: localhost
2018-09-26T14:08:43.166+0530	imported 29353 documents
```

2. Export student collection into student.json.

```
> mongoexport-db student -c student -o stud.json
connected to: 127.0.0.1

exported 2 records
```

3. Study mongodump command

`mongodump` is a utility for creating a binary export of the contents of a database. `mongodump` can export data from either `mongod` or `mongos` instances.

`mongodump` can be a part of a backup strategy with `mongorestore` for partial backups based on a query, syncing from production to staging or development environments, or changing the storage engine of a standalone. However, the use of `mongodump` and `mongorestore` as a backup strategy can be problematic for sharded clusters and replica sets.

Part A: Indexing

1. Sort data using population in ascending order and display query plan using explain command.

```
> db.zipdata.find().sort({pop:1}).explain()
{
	"queryPlanner" : {
		"plannerVersion" : 1,
		"namespace" : "zip.zipdata",
		"indexFilterSet" : false,
		"parsedQuery" : {

		},
		"winningPlan" : {
			"stage" : "SORT",
			"sortPattern" : {
				"pop" : 1
			},
			"inputStage" : {
				"stage" : "SORT_KEY_GENERATOR",
				"inputStage" : {
					"stage" : "COLLSCAN",
                                        "direction" : "forward"
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "shubham-Inspiron-14-3452",
                "port" : 27017,
                "version" : "4.0.2",
                "gitVersion" : "fc1573ba18aee42f97a3bb13b67af7d837826b47"
        },
        "ok" : 1
}
```

2. Apply index on population in ascending order then sort data using population in ascending order and display query plan using explain command.

> db.zipdata.createIndex({pop:1})

```
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 2,
        "note" : "all indexes already exist",
        "ok" : 1
}
```

> db.zipdata.find().sort({pop:1}).explain()

```
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "zip.zipdata",
                "indexFilterSet" : false,
                "parsedQuery" : {

                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "pop" : 1
                                },
                                "indexName" : "pop_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "pop" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "pop" : [
                                                "[MinKey, MaxKey]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "shubham-Inspiron-14-3452",
                "port" : 27017,
                "version" : "4.0.2",
                "gitVersion" : "fc1573ba18aee42f97a3bb13b67af7d837826b47"
        },
        "ok" : 1
}
```

Part B: Aggregation
1. Display All Data.

<u>> db.zipdata.find()</u>

```
{ "_id" : "01007", "city" : "BELCHERTOWN", "loc" : [ -72.410953, 42.275103 ], "pop" :
10579, "state" : "MA" }
{ "_id" : "01005", "city" : "BARRE", "loc" : [ -72.108354, 42.409698 ], "pop" : 4546, "state" :
"MA" }
{ "_id" : "01008", "city" : "BLANDFORD", "loc" : [ -72.936114, 42.182949 ], "pop" : 1240,
"state" : "MA" }
{ "_id" : "01010", "city" : "BRIMFIELD", "loc" : [ -72.188455, 42.116543 ], "pop" : 3706,
"state" : "MA" }
{ "_id" : "01011", "city" : "CHESTER", "loc" : [ -72.988761, 42.279421 ], "pop" : 1688, "state" :
"MA" }
{ "_id" : "01012", "city" : "CHESTERFIELD", "loc" : [ -72.833309, 42.38167 ], "pop" : 177,
"state" : "MA" }
{ "_id" : "01002", "city" : "CUSHMAN", "loc" : [ -72.51565, 42.377017 ], "pop" : 36963, "state"
: "MA" }
{ "_id" : "01013", "city" : "CHICOPEE", "loc" : [ -72.607962, 42.162046 ], "pop" : 23396,
"state" : "MA" }
```

```
{ "_id" : "01001", "city" : "AGAWAM", "loc" : [ -72.622739, 42.070206 ], "pop" : 15338,
"state" : "MA" }
{ "_id" : "01020", "city" : "CHICOPEE", "loc" : [ -72.576142, 42.176443 ], "pop" : 31495,
"state" : "MA" }
{ "_id" : "01027", "city" : "MOUNT TOM", "loc" : [ -72.679921, 42.264319 ], "pop" : 16864,
"state" : "MA" }
{ "_id" : "01028", "city" : "EAST LONGMEADOW", "loc" : [ -72.505565, 42.067203 ], "pop" :
13367, "state" : "MA" }
{ "_id" : "01031", "city" : "GILBERTVILLE", "loc" : [ -72.198585, 42.332194 ], "pop" : 2385,
"state" : "MA" }
{ "_id" : "01032", "city" : "GOSHEN", "loc" : [ -72.844092, 42.466234 ], "pop" : 122, "state" :
"MA" }
{ "_id" : "01033", "city" : "GRANBY", "loc" : [ -72.520001, 42.255704 ], "pop" : 5526, "state" :
"MA" }
{ "_id" : "01034", "city" : "TOLLAND", "loc" : [ -72.908793, 42.070234 ], "pop" : 1652, "state"
: "MA" }
{ "_id" : "01035", "city" : "HADLEY", "loc" : [ -72.571499, 42.36062 ], "pop" : 4231, "state" :
"MA" }
{ "_id" : "01036", "city" : "HAMPDEN", "loc" : [ -72.431823, 42.064756 ], "pop" : 4709, "state"
: "MA" }
{ "_id" : "01038", "city" : "HATFIELD", "loc" : [ -72.616735, 42.38439 ], "pop" : 3184, "state" :
"MA" }
{ "_id" : "01039", "city" : "HAYDENVILLE", "loc" : [ -72.703178, 42.381799 ], "pop" : 1387,
"state" : "MA" }
Type "it" for more
```

2. Display total no of documents in the collection.

```
> db.zipdata.count()
29353
```

3. Display total no of documents in the collection with city “BARRE”.

```
> db.zipdata.count({"city":"BARRE"})
2
```

4. Display total no of documents in the collection state wise.

```
> db.zipdata.aggregate({$group:{_id:"$state",count:{$sum:1}}})
{ "_id" : "CA", "count" : 1516 }
{ "_id" : "MT", "count" : 314 }
{ "_id" : "MS", "count" : 363 }
{ "_id" : "FL", "count" : 804 }
{ "_id" : "AR", "count" : 578 }
{ "_id" : "GA", "count" : 635 }
{ "_id" : "WA", "count" : 484 }
{ "_id" : "SC", "count" : 350 }
{ "_id" : "MN", "count" : 882 }
{ "_id" : "NE", "count" : 574 }
{ "_id" : "MD", "count" : 420 }
{ "_id" : "TN", "count" : 582 }
{ "_id" : "DE", "count" : 53 }
{ "_id" : "DC", "count" : 24 }
{ "_id" : "AZ", "count" : 270 }
{ "_id" : "ME", "count" : 410 }
{ "_id" : "OR", "count" : 384 }
{ "_id" : "AL", "count" : 567 }
{ "_id" : "PA", "count" : 1458 }
{ "_id" : "RI", "count" : 69 }
Type "it" for more
```

5. Display total population in each state.

```
> db.zipdata.aggregate(({$group:{_id:"$state",population:{$sum:"$pop"}}}))
{ "_id" : "CA", "population" : 29754890 }
{ "_id" : "MT", "population" : 798948 }
{ "_id" : "MS", "population" : 2573216 }
{ "_id" : "FL", "population" : 12686644 }
{ "_id" : "AR", "population" : 2350725 }
{ "_id" : "GA", "population" : 6478216 }
{ "_id" : "WA", "population" : 4866692 }
{ "_id" : "SC", "population" : 3486703 }
{ "_id" : "MN", "population" : 4372982 }
{ "_id" : "NE", "population" : 1578139 }
{ "_id" : "MD", "population" : 4781379 }
{ "_id" : "TN", "population" : 4876457 }
{ "_id" : "DE", "population" : 666168 }
{ "_id" : "DC", "population" : 606900 }
{ "_id" : "AZ", "population" : 3665228 }
{ "_id" : "ME", "population" : 1226648 }
{ "_id" : "OR", "population" : 2842321 }
{ "_id" : "AL", "population" : 4040587 }
{ "_id" : "PA", "population" : 11881643 }
{ "_id" : "RI", "population" : 1003218 }
Type "it" for more
```

6. To return all states with a population greater than 10 million.

```
> db.zipdata.aggregate( [    { $group: {   _id: "$state", totalPop: { $sum: "$pop" } } },    {
$match: { totalPop: { $gte: 10*1000*1000 } } } ] )
{ "_id" : "CA", "totalPop" : 29754890 }
{ "_id" : "FL", "totalPop" : 12686644 }
{ "_id" : "PA", "totalPop" : 11881643 }
{ "_id" : "NY", "totalPop" : 17990402 }
{ "_id" : "OH", "totalPop" : 10846517 }
{ "_id" : "IL", "totalPop" : 11427576 }
{ "_id" : "TX", "totalPop" : 16984601 }
```

7. To return the average populations for cities in each state.

```
> db.zipdata.aggregate( [ { $group: { _id: { state: "$state", city: "$city" }, pop: { $sum: "$pop" } } },{ $group: { _id: "$_id.state", avgCityPop: { $avg: "$pop" } } } ] )
{ "_id" : "DC", "avgCityPop" : 303450 }
{ "_id" : "DE", "avgCityPop" : 14481.91304347826 }
{ "_id" : "RI", "avgCityPop" : 19292.653846153848 }
{ "_id" : "NJ", "avgCityPop" : 15775.89387755102 }
{ "_id" : "MT", "avgCityPop" : 2593.987012987013 }
{ "_id" : "CA", "avgCityPop" : 27756.42723880597 }
{ "_id" : "KS", "avgCityPop" : 3819.884259259259 }
{ "_id" : "MO", "avgCityPop" : 5672.195338512764 }
{ "_id" : "NH", "avgCityPop" : 5232.320754716981 }
{ "_id" : "OK", "avgCityPop" : 6155.743639921722 }
{ "_id" : "NE", "avgCityPop" : 3034.882692307692 }
{ "_id" : "CO", "avgCityPop" : 9981.075757575758 }
{ "_id" : "LA", "avgCityPop" : 10465.496277915632 }
{ "_id" : "ID", "avgCityPop" : 4320.811158798283 }
{ "_id" : "IL", "avgCityPop" : 9954.334494773519 }
{ "_id" : "AL", "avgCityPop" : 7907.2152641878665 }
{ "_id" : "OR", "avgCityPop" : 8262.561046511628 }
{ "_id" : "MD", "avgCityPop" : 12615.775725593667 }
{ "_id" : "AR", "avgCityPop" : 4175.355239786856 }
{ "_id" : "FL", "avgCityPop" : 27400.958963282937 }
Type "it" for more
```

Map Reduce operation by Using Zips database

Display total no of documents in the collection state wise.

> var map=function(){if(this.state){emit(this.state,1);}}

> var red=function(key,values){return Array.sum(values);}

> var res=db.zipdata.mapReduce(map,red,{out:"count1"});

> db[res.result].find()

```
{ "_id" : "AK", "value" : 195 }
{ "_id" : "AL", "value" : 567 }
{ "_id" : "AR", "value" : 578 }
{ "_id" : "AZ", "value" : 270 }
{ "_id" : "CA", "value" : 1516 }
{ "_id" : "CO", "value" : 414 }
{ "_id" : "CT", "value" : 263 }
{ "_id" : "DC", "value" : 24 }
{ "_id" : "DE", "value" : 53 }
{ "_id" : "FL", "value" : 804 }
{ "_id" : "GA", "value" : 635 }
{ "_id" : "HI", "value" : 80 }
{ "_id" : "IA", "value" : 922 }
{ "_id" : "ID", "value" : 244 }
{ "_id" : "IL", "value" : 1237 }
{ "_id" : "IN", "value" : 676 }
{ "_id" : "KS", "value" : 715 }
{ "_id" : "KY", "value" : 809 }
{ "_id" : "LA", "value" : 464 }
{ "_id" : "MA", "value" : 474 }
Type "it" for more
{ "_id" : "MD", "value" : 420 }
{ "_id" : "ME", "value" : 410 }
{ "_id" : "MI", "value" : 876 }
{ "_id" : "MN", "value" : 882 }
{ "_id" : "MO", "value" : 994 }
{ "_id" : "MS", "value" : 363 }
{ "_id" : "MT", "value" : 314 }
{ "_id" : "NC", "value" : 705 }
{ "_id" : "ND", "value" : 391 }
{ "_id" : "NE", "value" : 574 }
{ "_id" : "NH", "value" : 218 }
{ "_id" : "NJ", "value" : 540 }
{ "_id" : "NM", "value" : 276 }
{ "_id" : "NV", "value" : 104 }
{ "_id" : "NY", "value" : 1595 }
{ "_id" : "OH", "value" : 1007 }
{ "_id" : "OK", "value" : 586 }
{ "_id" : "OR", "value" : 384 }
{ "_id" : "PA", "value" : 1458 }
{ "_id" : "RI", "value" : 69 }
Type "it" for more
> it
{ "_id" : "SC", "value" : 350 }
{ "_id" : "SD", "value" : 384 }
{ "_id" : "TN", "value" : 582 }
{ "_id" : "TX", "value" : 1671 }
{ "_id" : "UT", "value" : 205 }
{ "_id" : "VA", "value" : 816 }
{ "_id" : "VT", "value" : 243 }
{ "_id" : "WA", "value" : 484 }
{ "_id" : "WI", "value" : 716 }
{ "_id" : "WV", "value" : 656 }
{ "_id" : "WY", "value" : 140 }
> it
```

Create Blog Database and create posts collection with below keys

```
{
        Author:
        Title:
        tags: [  ]
        body :
        comment:[
                        {
                                User:
                                Comment_text:
                                likes:
                        },
                        {
                                User:
                                Comment_text:
                                likes:
                        }
                  ]
}
```

1. Insert 5 posts for 3 different authors.

> db.posts.insert([{"author":"shubham","title":"Twitter Sentiment Analysis","tags":["Machine Learning","Data Analytics"],"body":null,"comment":[{"user":"hannibal","comment_text":"Excellent!","likes":10}]},{"author":"AnilKumar","title":"Search Engine","tags":["information retrieval","data mining"],"body":null,"comment":[{"user":"borat","comment_text":"data set shoud be dynamic","likes":3}]},{"author":"SunilKumar","title":"Unity Engine","tags":["Game Development","Machine Learning"],"body":null,"comment":[{"user":"pablo","comment_text":"unreal engine 4 is more efficient","likes":2}]},{"author":"SunilKumar","title":"3D Ray Tracing","tags":["nvidia","iamai"],"body":null,"comment":[{"user":"shubh","comment_text":"Good!","likes":6}]},{"author":"AnilKumar","title":"TensorFlow for Image Classification","tags":["Image Processing","Data Analytics"],"body":null,"comment":[{"user":"andrewIng","comment_text":"performance is unacceptable","likes":4}]}])

```
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

2. Display tags key in inserted documents.

```
> db.posts.distinct("tags")
[
	"Data Analytics",
	"Machine Learning",
	"data mining",
	"information retrieval",
	"Game Development",
	"iamai",
	"nvidia",
	"Image Processing"
]
```

3. Count total no. of posts.

```
> db.posts.count()
5
```

4. Add new tag for a post.

```
> db.posts.update({"title":"Search Engine"},{$push:{"tags":"python"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

5. Delete a post of Author “SunilKumar”

```
> db.posts.remove({"author":"SunilKumar"},{ justOne:true})
WriteResult({ "nRemoved" : 1 })
```

6. Display all posts with Tags:Machine Learning

```
> db.posts.find({"tags":"Machine Learning"})
{ "_id" : ObjectId("5ba9b8302d4481e3d80a0269"), "author" : "shubham", "title" : "Twitter
Sentiment Analysis", "tags" : [ "Machine Learning", "Data Analytics" ], "body" : null,
"comment" : [ { "user" : "hannibal", "comment_text" : "Excellent!", "likes" : 10 } ] }
{ "_id" : ObjectId("5ba9b8302d4481e3d80a026b"), "author" : "SunilKumar", "title" : "Unity
Engine", "tags" : [ "Game Development", "Machine Learning" ], "body" : null, "comment" : [ {
"user" : "pablo", "comment_text" : "unreal engine 4 is more efficient", "likes" : 2 } ] }
```

7.Display Users who commented for Author “ SunilKumar”

```
> db.posts.distinct("comment.user",{"author":"SunilKumar"})
[ "pablo", "shubh" ]
```

8.Display comments with more than 4 likes

```
> db.posts.find( { "comment.likes": {$gt:4} },{"comment":1})
{ "_id" : ObjectId("5ba9b8302d4481e3d80a0269"), "comment" : [ { "user" : "hannibal",
"comment_text" : "Excellent!", "likes" : 10 } ] }
{ "_id" : ObjectId("5ba9b8302d4481e3d80a026c"), "comment" : [ { "user" : "shubh",
"comment_text" : "Good!", "likes" : 6 } ] }
```

9.Display comments with 0 likes

```
> db.posts.find( { "comment.likes": {$eq:0} },{"comment":1})
```

10. Add new comment to Author “AnilKumar”

```
>
db.posts.update({"author":"AnilKumar"},{$push:{"comment":{"user":"beerus","comment_text":"Nice!","likes":3}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Read the BLOG Database and display it using Java Forms.

1) Java Database Connectivity:

This Java program will accomplish the following operations.

- Connect to the MongoDB present at the localhost and port 27017.
- Connect to the database 'blog'. If such database does not exist, then it will create a new database with this name.
- Selection of the collection named as 'posts'.
- The added document was retrieved and printed on the console with the help of 'DBCursor' class as shown in the above program.

2) Code:

```
package mongoAssignment12;


import com.mongodb.BasicDBObject;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.DBCursor;

import com.mongodb.DBObject;

import com.mongodb.MongoClient;


public class mongoConnection {


	public static void main(String args[]) {

		try {

			/**** Connect to the MongoDB ****/

			MongoClient mongodb = new MongoClient("localhost", 27017);


			/**** Get database ****/

			// if database doesn't exists, MongoDB will create it for us

			@SuppressWarnings("deprecation")

			DB db = mongodb.getDB("blog");

			System.out.println("Connection to MongoDB database successfully");
```

```
                        /**
                         * Selecting Records from MongoDB
                         */
                        DBCollection coll = db.getCollection("posts");
                        System.out.println("Collection has selected successfully");
                        DBCursor cursor = coll.find();
                        int index = 1;

                        while (cursor.hasNext()) {
                                System.out.println("Document: " + index);
                                System.out.println(cursor.next());
                                index++;
                        }
                } catch (Exception e) {
                        System.err.println(e.getClass().getName() + ": " + e.getMessage());
                }
        }
}
```

3) Explaination Of Code:

Firstly, we are connecting to the MongoDB client through MongoClient class by passing the URL and port where MongoDB instance is running.

- The MongoDB instance is returned in the mongodb instance variable. It is used to invoke DB instance through 'mongodb.getDB ("blog")' method into 'db' instance variable.
- Next, we are going to select this collection 'posts' through 'db.getCollection ("posts");' method which returns the instance into 'coll' instance variable.
- Now, we are using 'DBCursor' class which returns the cursor to iterate over the documents present in the current collection through 'DBCursor cursor = coll.find ();' method.
- Lastly, with the help of 'DBCursor' instance, we are iterating over the available documents to display the document details which we had inserted in the last step into the collection 'MyCollection1'.
- The entire code is placed inside the try catch block in order to catch any possible exception thrown during runtime of this Java program.

4) Output:

```
Sep 25, 2018 11:50:45 AM com.mongodb.diagnostics.logging.JULLogger log

INFO: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN,
serverSelectionTimeout='30000 ms', maxWaitQueueSize=500}

Connection to MongoDB database successfully

Collection has selected successfully

Sep 25, 2018 11:50:45 AM com.mongodb.diagnostics.logging.JULLogger log

INFO: Cluster description not yet available. Waiting for 30000 ms before timing out

Sep 25, 2018 11:50:46 AM com.mongodb.diagnostics.logging.JULLogger log

INFO: Opened connection [connectionId{localValue:1, serverValue:2}] to localhost:27017

Sep 25, 2018 11:50:46 AM com.mongodb.diagnostics.logging.JULLogger log

INFO: Monitor thread successfully connected to server with description
ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true,
version=ServerVersion{versionList=[4, 0, 2]}, minWireVersion=0, maxWireVersion=7,
maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=10473138}

Sep 25, 2018 11:50:46 AM com.mongodb.diagnostics.logging.JULLogger log

INFO: Opened connection [connectionId{localValue:2, serverValue:3}] to localhost:27017

Document: 1

{ "_id" : { "$oid" : "5ba9b8302d4481e3d80a0269" }, "author" : "shubham", "title" : "Twitter Sentiment Analysis",
"tags" : ["Machine Learning", "Data Analytics"], "body" : null, "comment" : [{ "user" : "hannibal", "comment_text"
: "Excellent!", "likes" : 10.0 }] }

Document: 2

{ "_id" : { "$oid" : "5ba9b8302d4481e3d80a026a" }, "author" : "AnilKumar", "title" : "Search Engine", "tags" :
["information retrieval", "data mining", "python"], "body" : null, "comment" : [{ "user" : "borat", "comment_text" :
"data set shoud be dynamic", "likes" : 3.0 }, { "user" : "beerus", "comment_text" : "Nice!", "likes" : 3.0 }] }

Document: 3

{ "_id" : { "$oid" : "5ba9b8302d4481e3d80a026d" }, "author" : "AnilKumar", "title" : "TensorFlow for Image
Classification", "tags" : ["Image Processing", "Data Analytics"], "body" : null, "comment" : [{ "user" : "andrewIng",
"comment_text" : "performance is unacceptable", "likes" : 4.0 }] }
```

5) Conclusion:

In this tutorial, we have created a JDBC connection to the MongoDB database. Next, we selected this collection followed by iterating over all the documents present in this collection in order to display the available documents present inside the collection on the console.