DEPARTMENT OF COMPUTER APPLICATION

TKM COLLEGE OF ENGINEERING

KOLLAM – 691005

20MCA134 ADVANCED DATABASE MANAGEMENT SYSTEMS LAB

PRACTICAL RECORD BOOK

Second Semester

MCA 2021-2022

Submitted by:

NAME: GOPIKA S RAJ

ROLL NO: TKM21MCA-2021

DEPARTMENT OF COMPUTER APPLICATION

TKM COLLEGE OF ENGINEERING

KOLLAM – 691005

Certificate

This is a bonafide record of the work done by GOPIKA S RAJ (TKM21MCA-2021) in the Second Semester in ADVANCED DATABASE MANAGEMENT SYSTEMS LAB, Course(20MCA134) towards the partial fulfillment of the degree of Master of Computer Applications during the academic year 2021-2022.

Staff Member in-charge

Examiner

……..

……..

INDEX

INTRODUCTION TO SQL

Pronounced as SEQUEL: Structured English QUERY Language

- Pure non-procedural query language
- Designed and developed by IBM, Implemented by Oracle
- 1978 System/R IBM- 1st Relational DBMS
- 1979 Oracle and Ingres
- 1982 SQL/DS and DB2 IBM
- Accepted by both ANSI + ISO as Standard Query Language for any RDBMS
- SQL86 (SQL1) : first by ANSI and ratified by ISO (SQL-87), minor revision on 89 (SQL-89)
- SQL92 (SQL2) : major revision
- SQL99 (SQL3) : add recursive query, trigger, some OO features, and non-scholar type
- SQL2003 : XML, Window functions, and sequences (Not free)
  Supports all the three sublanguages of DBMS: DDL, DML, DCL
- Supports Aggregate functions, String Manipulation functions, Set theory operations, Date Manipulation functions, rich set of operators (IN, BETWEEN, LIKE, IS NULL,EXISTS)
- Supports REPORT writing features and Forms for designing GUI based applications

Experiment 1

AIM

Consider Dept table

DEPTNO	DNAME	LOC

Perform the following:

1. Rename the table dept as department
2. Add a new column PINCODE with not null constraints to the existing table DEPT
3. All constraints and views that reference the column are dropped automatically, along with the column.
4. Rename the column DNAME to DEPT_NAME in dept table
5. Change the data type of column loc as CHAR with size 10
6. Delete table

CODE:

```
CREATE DATABASE Departments;

USE Departments;
CREATE TABLE Dept(
DEPTNO int NOT NULL,
DNAME varchar(30) NOT NULL,
LOC varchar(30) NOT NULL
);
SELECT * FROM dept;
SELECT * FROM department;
```

Query 1:

```
RENAME TABLE dept TO department;
```

Query 2:

```
ALTER TABLE department ADD COLUMN PINCODE int(6) NOT NULL;
```

Query 3:

```
ALTER TABLE department DROP COLUMN PINCODE;
```

Query 4:

```
ALTER TABLE department CHANGE DNAME DEPT_NAME varchar(30);
```

Query 5:

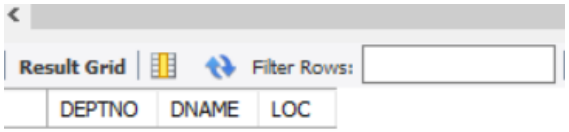
ALTER TABLE department MODIFY COLUMN LOC char(10);

Query 6:

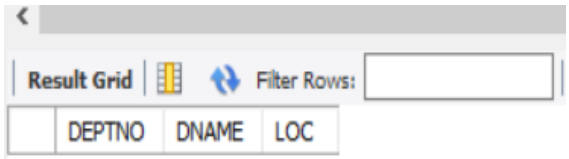
DROP TABLE department;

OUTPUT

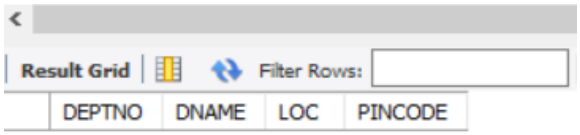
Query 1:



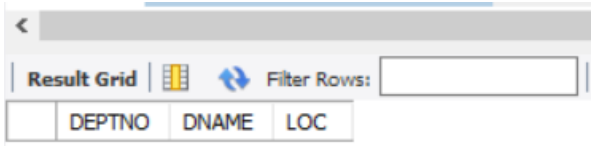
Query 2:



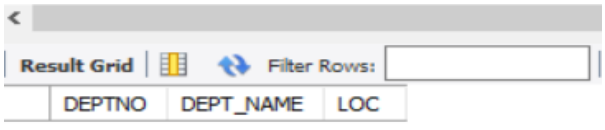
Query 3:



Query 4:



Query 5:



RESULT

Query processed successfully and output obtained.

Experiment 2

AIM

Consider the MOVIE DATABASE

Movies

title	director	myear	rating
Fargo	Coen	1996	8.2
Raising Arizona	Coen	1987	7.6
Spiderman	Raimi	2002	7.4
Wonder Boys	Hanson	2000	7.6

Actors

actor	ayear
Cage	1964
Hanks	1956
Maguire	1975
McDormand	1957

Acts

actor	title
Cage	Raising Arizona
Maguire	Spiderman
Maguire	Wonder Boys
McDormand	Fargo
McDormand	Raising Arizona
McDormand	Wonder Boys

Directors

director	dyear
Coen	1954
Hanson	1945
Raimi	1959

Write following relational algebra queries for a given set of relations.

1. Find movies made after 1997
2. Find movies made by Hanson after 1997
3. Find all movies and their ratings
4. Find all actors and directors
5. Find Coen's movies with McDormand

CODE:

```
CREATE DATABASE MOVIE;
USE MOVIE;
CREATE TABLE directors ( director varchar(20) NOT NULL, dyear int NOT NULL,
        PRIMARY KEY(director) );
CREATE TABLE movies ( title varchar(30) NOT NULL, director varchar(20) NOT NULL,
        myear int NOT NULL, rating float NOT NULL, PRIMARY KEY(title),
        FOREIGN KEY (director) REFERENCES directors (director) );
CREATE TABLE actors ( actor varchar(20) NOT NULL, ayear int NOT NULL,
        PRIMARY KEY(actor) );
CREATE TABLE acts ( actor varchar(20) NOT NULL, title varchar(30) NOT NULL,
```

```
    FOREIGN KEY (actor) REFERENCES actors (actor),
    FOREIGN KEY (title) REFERENCES movies (title) );

INSERT INTO directors(director,dyear) VALUES("Coen",1954);
INSERT INTO directors(director,dyear) VALUES("Hanson",1945);
INSERT INTO directors(director,dyear) VALUES("Raimi",1959);

INSERT INTO movies(title,director,myear,rating) VALUES("Fargo","Coen",1996,8.2);
INSERT INTO movies(title,director,myear,rating) VALUES("Raising
Arizona","Coen",1987,7.6);
INSERT INTO movies(title,director,myear,rating) VALUES("Spiderman","Raimi",2002,7.4);
INSERT INTO movies(title,director,myear,rating) VALUES("Wonder
Boys","Hanson",2000,7.6);

INSERT INTO actors(actor,ayear) VALUES("Cage",1964);
INSERT INTO actors(actor,ayear) VALUES("Hanks",1956);
INSERT INTO actors(actor,ayear) VALUES("Maguire",1975);
INSERT INTO actors(actor,ayear) VALUES("McDormand",1957);

INSERT INTO acts(actor,title) VALUES("Cage","Raising Arizona");
INSERT INTO acts(actor,title) VALUES("Maguire","Spiderman");
INSERT INTO acts(actor,title) VALUES("Maguire","Wonder Boys");
INSERT INTO acts(actor,title) VALUES("McDormand","Fargo");
INSERT INTO acts(actor,title) VALUES("McDormand","Raising Arizona");
INSERT INTO acts(actor,title) VALUES("McDormand","Wonder Boys");

SELECT * FROM movies;
SELECT * FROM directors;
SELECT * FROM actors;
SELECT * FROM acts;
```

Query 1:

```
SELECT title FROM movies WHERE myear>1997;
```

Query 2:

```
SELECT title FROM movies WHERE director="Hanson" AND myear>1997;
```

Query 3:

```
SELECT title,rating FROM movies;
```

Query 4:

```
CREATE VIEW actdir AS SELECT actors.actor,directors.director FROM actors,directors;
SELECT * FROM actdir;
```

Query 5:

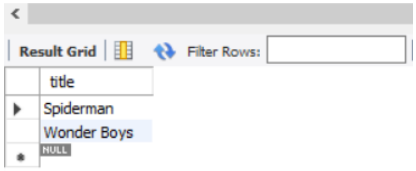
```
ALTER TABLE department MODIFY COLUMN LOC char(10);
```

Query 6:

```
SELECT movies.title FROM movies,acts WHERE director="Coen" AND actor="McDormand"
AND movies.title=acts.title;
```

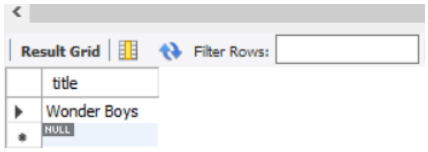
OUTPUT

Query 1:



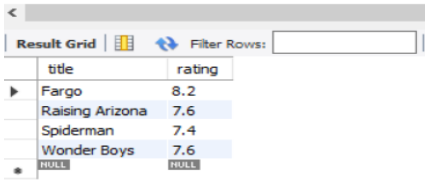
title
Spiderman
Wonder Boys
NULL

Query 2:



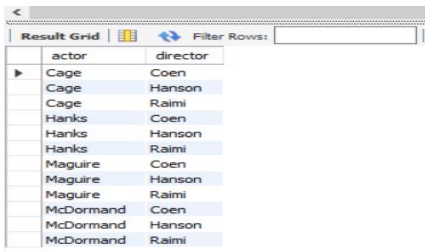
title
Wonder Boys
NULL

Query 3:



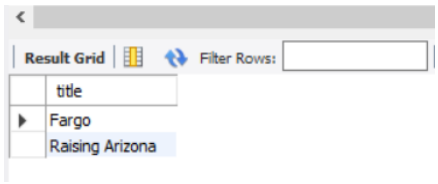
title	rating
Fargo	8.2
Raising Arizona	7.6
Spiderman	7.4
Wonder Boys	7.6
NULL	NULL

Query 4:



actor	director
Cage	Coen
Cage	Hanson
Cage	Raimi
Hanks	Coen
Hanks	Hanson
Hanks	Raimi
Maguire	Coen
Maguire	Hanson
Maguire	Raimi
McDormand	Coen
McDormand	Hanson
McDormand	Raimi

Query 5:



title
Fargo
Raising Arizona

RESULT

Query processed successfully and output obtained.

Experiment 3

AIM

Consider Employee table

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E101	Amit	oduction	45000	12-Mar-00	Bangalore
E102	Amit	HR	70000	03-Jul-02	Bangalore
E103	sunita	anagemer	120000	11-Jan-01	mysore
E105	sunita	IT	67000	01-Aug-01	mysore
E106	mahesh	Civil	145000	20-Sep-03	Mumbai

Perform the following

1. Display all the fields of employee table
2. Retrieve employee number and their salary
3. Retrieve average salary of all employee
4. Retrieve number of employee
5. Retrieve distinct number of employee
6. Retrieve total salary of employee group by employee name and count similar names
7. Retrieve total salary of employee which is greater than >120000
8. Display name of employee in descending order
9. Display details of employee whose name is AMIT and salary greater than 50000

CODE:

```
CREATE DATABASE EMPLOYEES;
USE EMPLOYEES;
CREATE TABLE EMPLOYEE( EMPNO char(4) not null, EMPNAME varchar(30) not null,
   DEPT varchar(30) not null, SALARY int(8) not null, DOJ date not null,
   BRANCH varchar(20) not null, PRIMARY KEY(EMPNO) );

INSERT INTO EMPLOYEE  VALUES("E101","Amit","Production",45000,"2000-03-
12","Banglore");
INSERT INTO EMPLOYEE  VALUES("E102","Amit","HR",70000,"2002-07-
03","Banglore");
INSERT INTO EMPLOYEE  VALUES("E103","Sunitha","Management",120000,"2001-01-
11","Mysore");
INSERT INTO EMPLOYEE  VALUES("E105","Sunitha","IT",67000,"2001-08-
01","Mysore");
INSERT INTO EMPLOYEE  VALUES("E106","Mahesh","Civil",145000,"2003-09-
20","Mumbai");
```

Query 1:

SELECT * FROM EMPLOYEE;

Query 2:

SELECT EMPNO,SALARY FROM EMPLOYEE;

Query 3:

SELECT AVG(SALARY) FROM EMPLOYEE;

Query 4:

SELECT COUNT(*) FROM EMPLOYEE;

Query 5:

SELECT DISTINCT EMPNO FROM EMPLOYEE;

Query 6:

SELECT SUM(SALARY),EMPNAME,COUNT(EMPNAME) AS OCCURENCE FROM EMPLOYEE GROUP BY EMPNAME;

Query 7:

SELECT SUM(SALARY) FROM EMPLOYEE WHERE SALARY>120000;

Query 8:

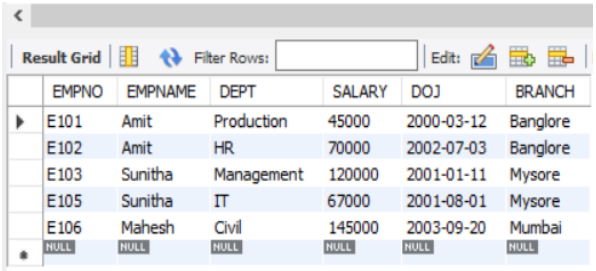
SELECT EMPNAME FROM EMPLOYEE ORDER BY EMPNAME DESC;

Query 9:

SELECT * FROM EMPLOYEE WHERE EMPNAME="Amit" AND SALARY>50000;

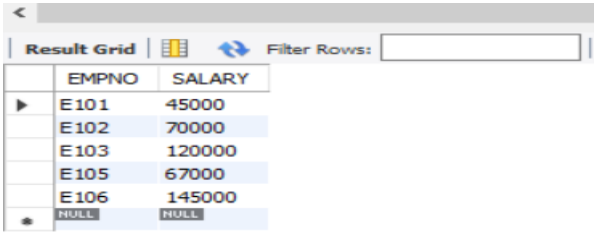
OUTPUT

Query 1:



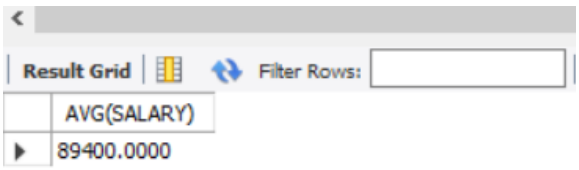
EMPNO	EMPNAME	DEPT	SALARY	DOJ	BRANCH
E101	Amit	Production	45000	2000-03-12	Banglore
E102	Amit	HR	70000	2002-07-03	Banglore
E103	Sunitha	Management	120000	2001-01-11	Mysore
E105	Sunitha	IT	67000	2001-08-01	Mysore
E106	Mahesh	Civil	145000	2003-09-20	Mumbai
NULL	NULL	NULL	NULL	NULL	NULL

Query 2:



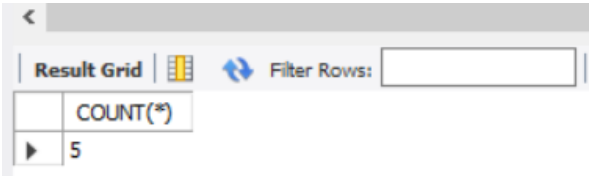
EMPNO	SALARY
E101	45000
E102	70000
E103	120000
E105	67000
E106	145000
NULL	NULL

Query 3:

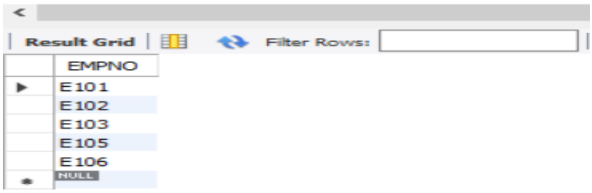


AVG(SALARY)
89400.0000

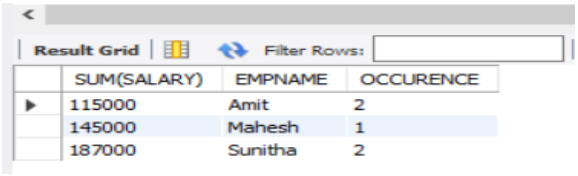
Query 4:



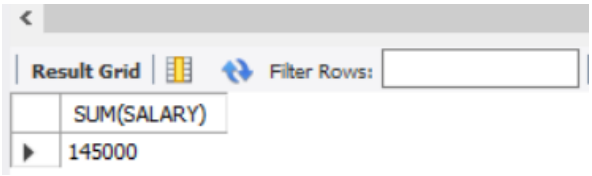
Query 5:



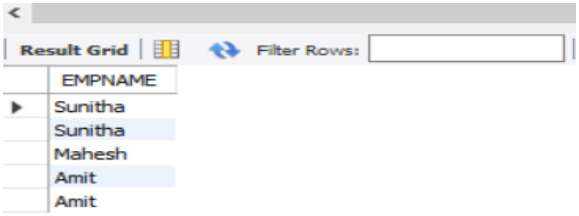
Query 6:



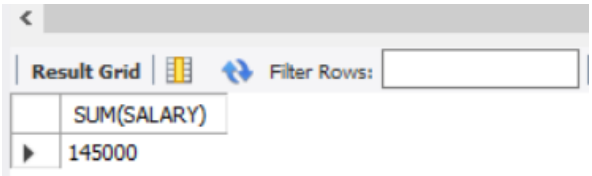
Query 7:



Query 8:



Query 9:



RESULT

Query processed successfully and output obtained.

Experiment 4

AIM

Apply DCL and TCL commands to impose restrictions on database.

CODE:

DCL:

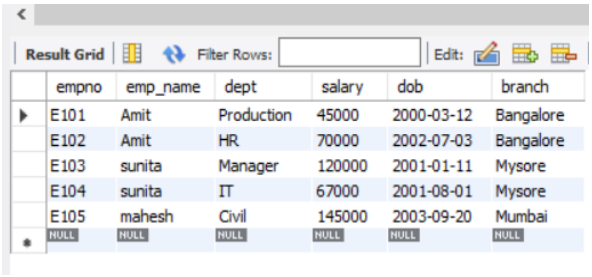
```
CREATE TABLE employee(empno VARCHAR(20) NOT NULL PRIMARY KEY,
        emp_name VARCHAR(20) NOT NULL,dept VARCHAR(20) NOT NULL,
        salary INT NOT NULL,dob DATE NOT NULL,branch VARCHAR(20) NOT NULL);
DESCRIBE employee;
INSERT INTO employee VALUES ('E101','Amit','Production',45000,'2000-03-12','Bangalore');
INSERT INTO employee VALUES ('E102','Amit','HR',70000,'2002-07-03','Bangalore');
INSERT INTO employee VALUES ('E103','sunita','Manager',120000,'2001-01-11','Mysore');
INSERT INTO employee VALUES ('E104','sunita','IT',67000,'2001-08-01','Mysore');
INSERT INTO employee VALUES ('E105','mahesh','Civil',145000,'2003-09-20','Mumbai');
SELECT * FROM employee;
delete from employee where empno="E101";

use employee1;
GRANT DELETE ON employee TO 'heylo'@'localhost';
REVOKE DELETE ON employee FROM 'heylo'@'localhost';
REVOKE DELETE ON *.* FROM 'heylo'@'localhost';
SHOW GRANTS FOR 'heylo'@'localhost';
```

TCL:

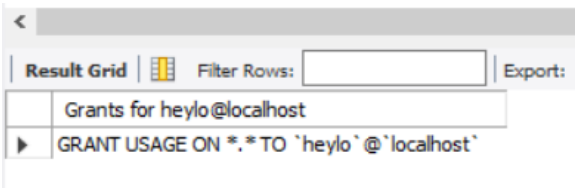
```
CREATE DATABASE tcl;
USE tcl;
CREATE TABLE dept(deptno varchar(20) not null,dname varchar(20) not null,loc varchar(20)
not null,primary key(deptno));
INSERT INTO dept VALUES ("d001","finance","kollam");
INSERT INTO dept VALUES ("d002","it","ernakulam");
INSERT INTO dept VALUES ("d003","management","thrissur");
set autocommit=0;
INSERT INTO dept VALUES ("d004","it","kozhikode");
savepoint b;
rollback;
select *from dept;
INSERT INTO dept VALUES ("d005","finance","kozhikode");
savepoint c;
INSERT INTO dept VALUES ("d006","finance","malappuram");
savepoint d;
rollback to c;
commit;
```

OUTPUT

DCL:

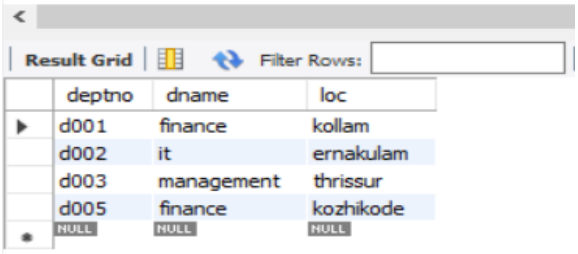


empno	emp_name	dept	salary	dob	branch
E101	Amit	Production	45000	2000-03-12	Bangalore
E102	Amit	HR	70000	2002-07-03	Bangalore
E103	sunita	Manager	120000	2001-01-11	Mysore
E104	sunita	IT	67000	2001-08-01	Mysore
E105	mahesh	Civil	145000	2003-09-20	Mumbai
NULL	NULL	NULL	NULL	NULL	NULL



Grants for heylo@localhost
GRANT USAGE ON *.* TO `heylo`@`localhost`

TCL:



deptno	dname	loc
d001	finance	kollam
d002	it	ernakulam
d003	management	thrissur
d005	finance	kozhikode
NULL	NULL	NULL

RESULT

Query processed successfully and output obtained.

Experiment 5

AIM

.Consider the schema for MovieDatabase:
ACTOR (**Act_id**, Act_Name, Act_Gender)
DIRECTOR (**Dir_id**, Dir_Name, Dir_Phone)
MOVIES (**Mov_id**, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST (**Act_id, Mov_id**, Role)
RATING **(Mov_id,** Rev_Stars)

Write SQL queries to
1. List the titles of all movies directed by 'Hitchcock'.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
5. Update rating of all movies directed by 'Steven Spielberg' to5.

CODE:

```
CREATE DATABASE clock;

USE clock;
CREATE TABLE actor(Act_id int, Act_Name varchar(50), Act_Gender varchar(10),PRIMARY
KEY(Act_id));
CREATE TABLE director (Dir_id int, Dir_Name varchar(50), Dir_Phone int,PRIMARY
KEY(Dir_id));
CREATE TABLE movies (Mov_id int, Mov_Title varchar(50), Mov_Year int, Mov_Lang
varchar(20), Dir_id int,PRIMARY KEY(Mov_id),FOREIGN KEY(Dir_id) REFERENCES
director(Dir_id) ON DELETE CASCADE) ;
CREATE TABLE movies_cast (Act_id int, Mov_id int,Role varchar(20),FOREIGN KEY(Act_id)
REFERENCES actor(Act_id) ON DELETE CASCADE,FOREIGN KEY(Mov_id) REFERENCES
movies(Mov_id) ON DELETE CASCADE);
CREATE TABLE rating (Mov_id int, Rev_Stars varchar(20),FOREIGN KEY(Mov_id)
REFERENCES movies(Mov_id) ON DELETE CASCADE) ;

INSERT INTO actor(Act_id,Act_Name,Act_Gender) VALUES (101,'kate','female');
INSERT INTO actor(Act_id,Act_Name,Act_Gender) VALUES (102,'leo','male');
INSERT INTO actor(Act_id,Act_Name,Act_Gender) VALUES (103,'joan','female');
INSERT INTO actor(Act_id,Act_Name,Act_Gender) VALUES (104,'frances','female');
INSERT INTO actor(Act_id,Act_Name,Act_Gender) VALUES (105,'tyre','male');
```

```
select * from actor;

INSERT INTO director(Dir_id,Dir_Name,Dir_phone) VALUES (301,'james
cameron','1982654329');
INSERT INTO director(Dir_id,Dir_Name,Dir_phone) VALUES (302,'Hitchcock','8907654312');
INSERT INTO director(Dir_id,Dir_Name,Dir_phone) VALUES (303,'Steven
Spielberg','8907654534');
INSERT INTO director(Dir_id,Dir_Name,Dir_phone) VALUES (304,'alejan','8456654534');

select * from director;


INSERT INTO movies(Mov_id,Mov_Title,Mov_Year,Mov_Lang,Dir_id) VALUES
(201,'titanic',1997,'english',301);
INSERT INTO movies(Mov_id,Mov_Title,Mov_Year,Mov_Lang,Dir_id) VALUES
(202,'rebecca',1940,'english',302);
INSERT INTO movies(Mov_id,Mov_Title,Mov_Year,Mov_Lang,Dir_id) VALUES
(203,'AI',2001,'english',303);
INSERT INTO movies(Mov_id,Mov_Title,Mov_Year,Mov_Lang,Dir_id) VALUES (204,'Ready
player one',2018,'english',303);
INSERT INTO movies(Mov_id,Mov_Title,Mov_Year,Mov_Lang,Dir_id) VALUES
(205,'Revanant',2016,'english',304);
select * from movies;


INSERT INTO movies_cast(Act_id,Mov_id,Role) VALUES (101,201,'Rose');
INSERT INTO movies_cast(Act_id,Mov_id,Role) VALUES (102,201,'jack');
INSERT INTO movies_cast(Act_id,Mov_id,Role) VALUES (102,205,'hugh glass');
INSERT INTO movies_cast(Act_id,Mov_id,Role) VALUES (103,202,'mrs.de winter');
INSERT INTO movies_cast(Act_id,Mov_id,Role) VALUES (104,203,'monica');
INSERT INTO movies_cast(Act_id,Mov_id,Role) VALUES (105,204,'wade watts');


select * from movies_cast;

INSERT INTO rating(Mov_id,Rev_Stars) VALUES (201,4.8);
INSERT INTO rating(Mov_id,Rev_Stars) VALUES (202,3);
INSERT INTO rating(Mov_id,Rev_Stars) VALUES (203,4.4);
INSERT INTO rating(Mov_id,Rev_Stars) VALUES (204,4.5);
INSERT INTO rating(Mov_id,Rev_Stars) VALUES (205,4.6);


select * from rating;
```

Query 1:

```
SELECT movies.Mov_Title,director.Dir_name from movies inner join director on
movies.dir_id=director.dir_id where director.dir_id=302 ;
```

Query 2:

```
SELECT mov_title FROM movies WHERE mov_id IN (
SELECT mov_id FROM  movies_cast WHERE act_id IN (
SELECT act_id FROM actor WHERE act_id IN (
SELECT act_id FROM  movies_cast GROUP BY act_id HAVING COUNT(act_id)>1)));
```

Query 3:

```
SELECT actor.Act_Name FROM actor inner join movies_cast on actor.act_id=movies_cast.act_id
WHERE mov_id in(SELECT Mov_id FROM movies where Mov_year<2000 or Mov_year>2015);
```

Query 4:

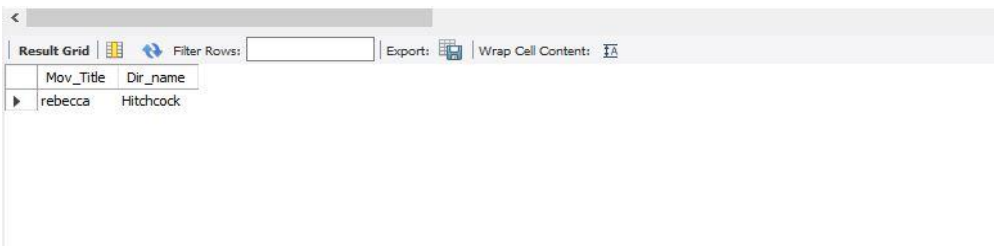
```
SELECT mov_title,MAX(rev_stars)
FROM movies
INNER JOIN rating USING (mov_id)
GROUP BY mov_title
HAVING MAX(rev_stars)>0
ORDER BY mov_title;
```

Query 5:

```
SET SQL_SAFE_UPDATES=0;
UPDATE rating
SET rev_stars=5
WHERE mov_id IN (SELECT mov_id FROM movies
WHERE dir_id IN (SELECT dir_id
FROM director
WHERE dir_name='steven spielberg'));
```

OUTPUT:

Query 1:



Query 2:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

mov_title
titanic
Revanant

Query 3:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Act_Name
kate
leo
joan
tyre
leo

Query 4:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

mov_title	MAX(rev_stars)
AI	4.4
Ready player one	4.5
rebecca	3
Revanant	4.6
titanic	4.8

Query 5:

```
SET SQL_SAFE_UPDATES=0;
UPDATE rating
SET rev_stars=5
WHERE mov_id IN (SELECT mov_id FROM movies
WHERE dir_id IN (SELECT dir_id
FROM director
WHERE dir_name='steven spielberg'));
```

ion Output

	Time	Action	Message
8	22:08:23	SET SQL_SAFE_UPDATES=0	0 row(s) affected
9	22:08:23	UPDATE rating SET rev_stars=5 WHERE mov_id IN (SELECT mov_id FROM movies WHE...	0 row(s) affected Rows matched: 2 Changed: 0 Warnings: 0

RESULT

Query processed successfully and output obtained.

Experiment 6

AIM

Queries using aggregate functions(COUNT,AVG,MIN,MAX,SUM),Group by,Order by,Having.

E_ID	E_NAME	AGE	SALARY
101	ANU	22	9000
102	Shane	29	8000
103	Rohan	34	6000
104	Scott	44	10000
105	Tiger	35	8000
106	Alex	27	7000
107	Abhi	29	8000

1. Create Employee table containing all Records.
2. Count number of employee names from employee table.
3. Find the Maximum age from employee table
4. Find the Minimum age from employee table.
5. Display the Sum of age employee table.
6. Display the Average of age from Employee table
7. Create a View for age in employee table
8. Display views
9. Find grouped salaries of employees.
10. Find salaries of employee in Ascending Order
11. Find salaries of employee in Descending Order

CODE:

```
CREATE DATABASE EMP;
USE EMP;
CREATE TABLE EMPP(E_ID int not null, E_NAME varchar(20) not null, AGE int not null,
   SALARY int not null, PRIMARY KEY(E_ID) );

INSERT INTO EMPP(E_ID,E_NAME,AGE,SALARY) VALUES(101,"Anu",22,9000);
INSERT INTO EMPP(E_ID,E_NAME,AGE,SALARY) VALUES(102,"Shane",29,8000);
INSERT INTO EMPP(E_ID,E_NAME,AGE,SALARY) VALUES(103,"Rohan",34,6000);
INSERT INTO EMPP(E_ID,E_NAME,AGE,SALARY) VALUES(104,"Scott",44,10000);
INSERT INTO EMPP(E_ID,E_NAME,AGE,SALARY) VALUES(105,"Tiger",35,8000);
INSERT INTO EMPP(E_ID,E_NAME,AGE,SALARY) VALUES(106,"Alex",27,7000);
INSERT INTO EMPP(E_ID,E_NAME,AGE,SALARY) VALUES(107,"Abhi",29,8000);
```

Query 1:

```
SELECT * FROM EMPP;
```

Query 2:

SELECT COUNT(*) FROM EMPP;

Query 3:

SELECT MAX(AGE) FROM EMPP;

Query 4:

SELECT MIN(AGE) FROM EMPP;

Query 5:

SELECT SUM(AGE) FROM EMPP;

Query 6:

SELECT AVG(AGE) FROM EMPP;

Query 7:

CREATE VIEW AGES AS SELECT AGE FROM EMPP;

Query 8:

SELECT COUNT(*) FROM EMPP;

Query 9:

SELECT SALARY FROM EMPP GROUP BY SALARY;

Query 10:

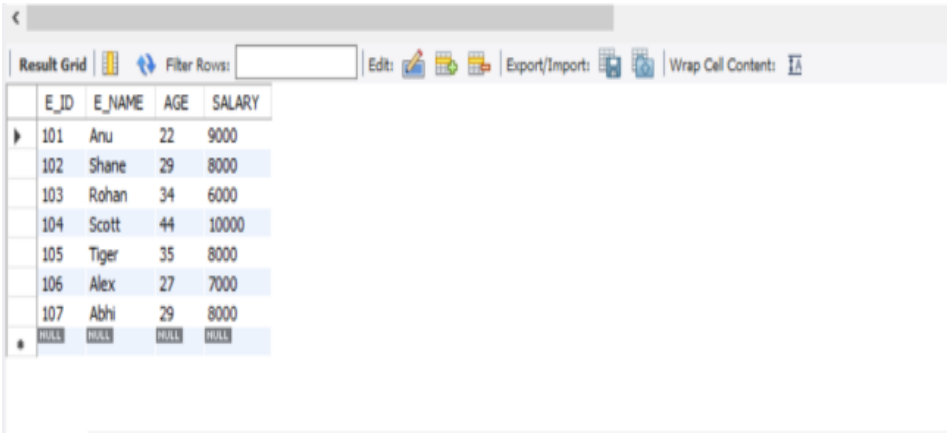
SELECT SALARY FROM EMPP ORDER BY SALARY ASC;

Query 11:

SELECT SALARY FROM EMPP ORDER BY SALARY DESC;

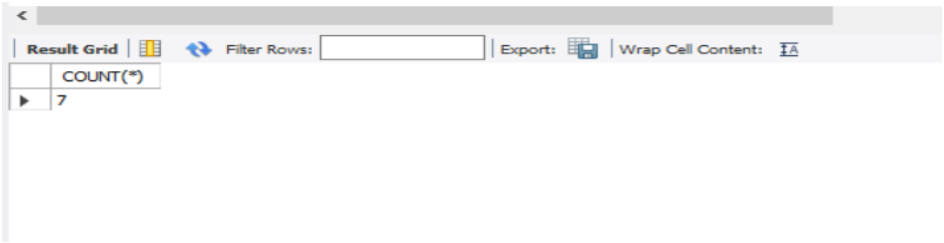
OUTPUT

Query 1:

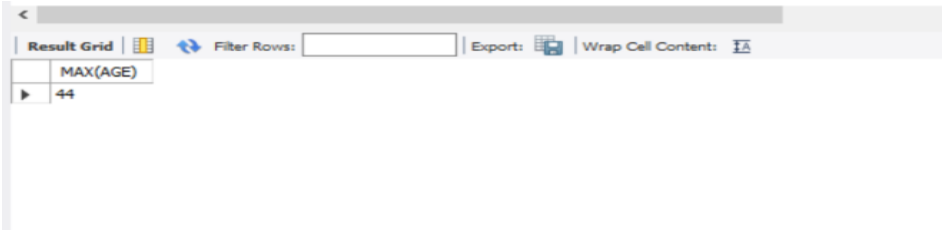


E_ID	E_NAME	AGE	SALARY
101	Anu	22	9000
102	Shane	29	8000
103	Rohan	34	6000
104	Scott	44	10000
105	Tiger	35	8000
106	Alex	27	7000
107	Abhi	29	8000
NULL	NULL	NULL	NULL

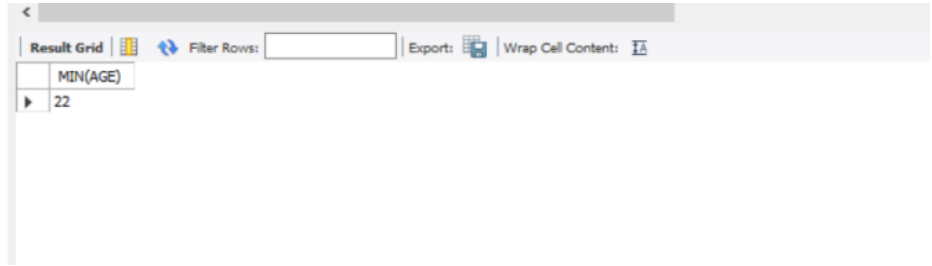
Query 2:



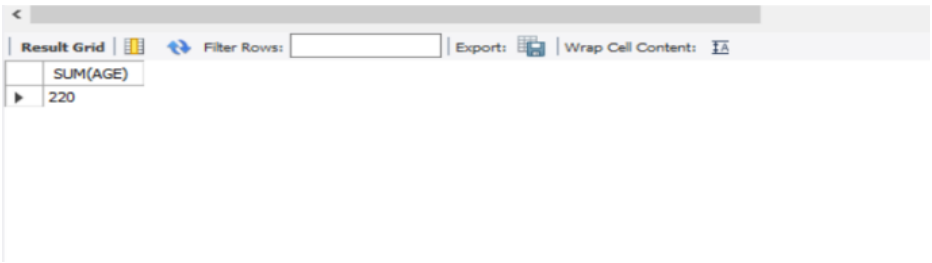
Query 3:



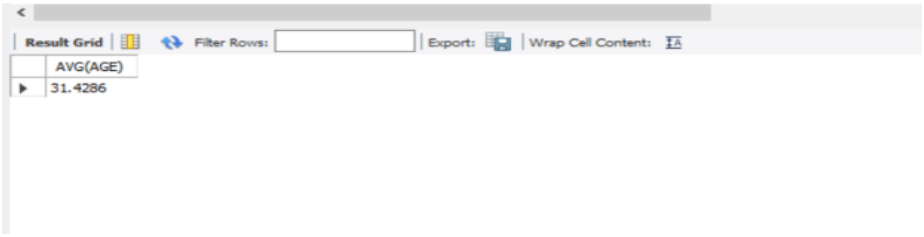
Query 4:



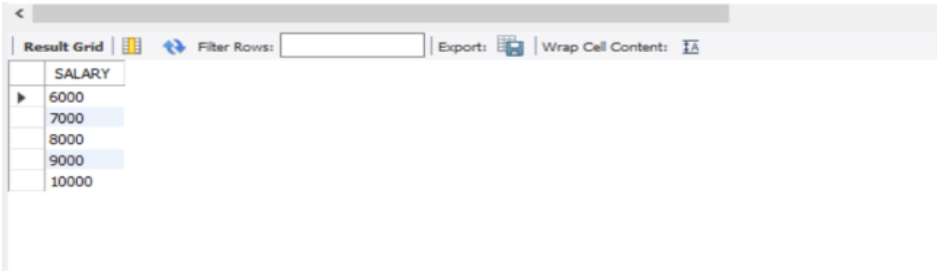
Query 5:



Query 6:

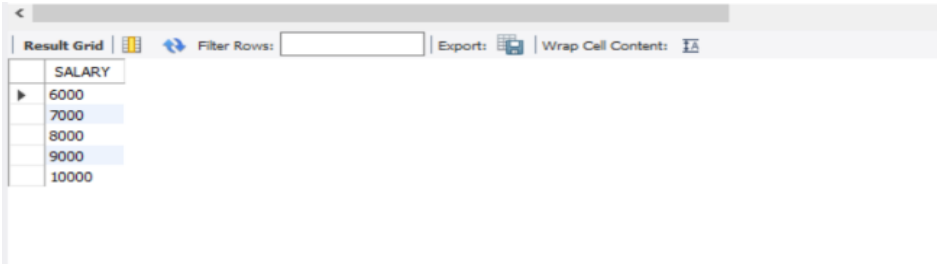


Query 8:



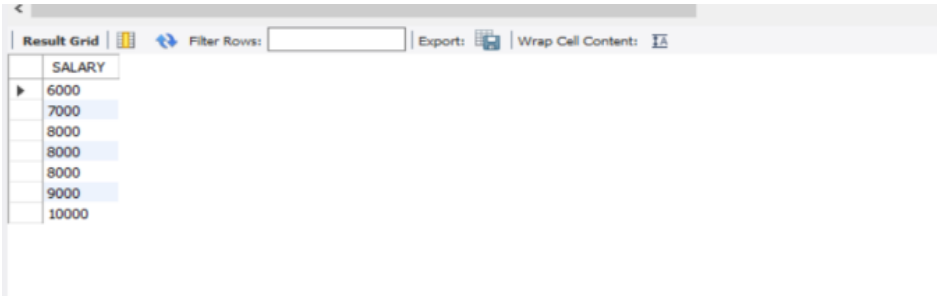
SALARY
6000
7000
8000
9000
10000

Query 9:



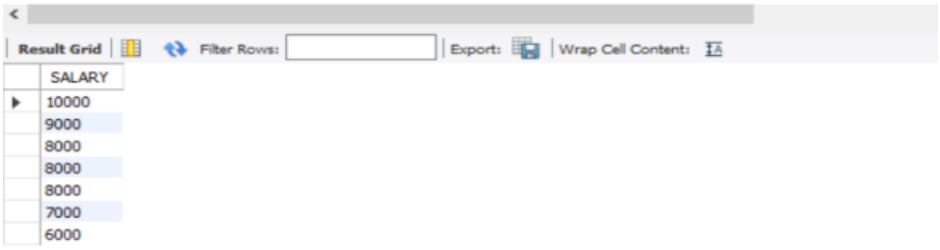
SALARY
6000
7000
8000
9000
10000

Query 10:



SALARY
6000
7000
8000
8000
8000
9000
10000

Query 11:



SALARY
10000
9000
8000
8000
8000
7000
6000

RESULT

Query processed successfully and output obtained.

Experiment 7

AIM

Given an integer i, write a PL/SQL procedure to insert the tuple (i, 'xxx') into a given relation

CODE

```
CREATE DATABSE studentdb;
USE studentdb;
CREATE TABLE T2(rollno int,name varchar(10),primary key(rollno));
call stud('101','Stefi');
call stud('102','Sreelaya');
select * from T2;

///STORED PROCEDURE///

CREATE DEFINER=`root`@`localhost` PROCEDURE `stud`(rollno int,name varchar(10))
BEGIN
insert into T2 values(rollno,name);
END
```

OUTPUT

RESULT

Query processed successfully and output obtained.

Experiment 8

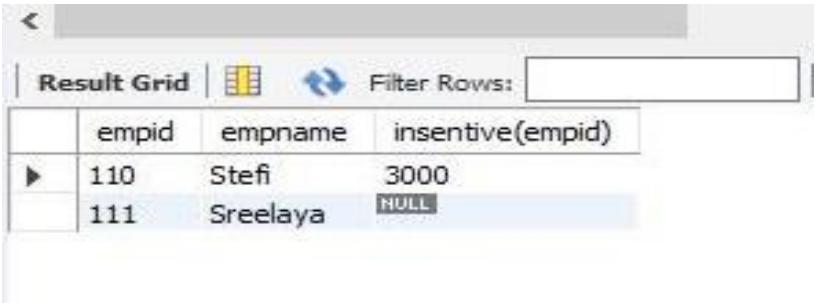
AIM

To write a PL/SQL block to calculate the incentive of an employee whose ID is 110

CODE

```
CREATE DATABASE employeedb2;
USE employeedb2;
CREATE TABLE E1(empid int,empname varchar(10),salary int,primary key(empid));
INSERT INTO  E1(empid,empname,salary)VALUES('110','Stefi',2000);
INSERT INTO  E1(empid,empname,salary)VALUES('111','Sreelaya',50000);
SELECT * from E1;
SELECT empid,empname,insentive(empid) from E1;

///FUNCTION///

CREATE DEFINER=`root`@`localhost` FUNCTION `insentive`(empid int) RETURNS
varchar(20) CHARSET latin1
BEGIN
DECLARE i VARCHAR(20);
IF (empid=110)
THEN SET i=3000;
END IF;
RETURN  i ;
END
```

OUTPUT



RESULT

Query processed successfully and output obtained.

Experiment 9

AIM

To create the Book database and do the following: (Consider the attributes based on the question given)

book(book_name, author_name, price,quantity)

a. Write a query to update the quantity by double in the table book.
b. List all the book_name whose price is greater than those of book named "Database for Dummies"
c. Retrieve the list of author_name whose first letter is 'a' along with the book_name and price (Explore more about *Like* keyword)
d. Write a PL/SQL Procedure to find the total number of books of same author

CODE:

```
CREATE DATABASE books;
USE books;
CREATE TABLE book_info(book_name varchar (20),author varchar(20),price int,quantity int);
INSERT into book_info VALUES('randamoozham','MT',300,5);
INSERT into book_info VALUES ('ikigai','hector',500,7);
INSERT into book_info VALUES ('databse of dummies','xyz',250,7);
INSERT into book_info VALUES ('wings of flare','APJ',500,7);
INSERT into book_info VALUES ('oopol','MT',270,3);
SELECT * from book_info;
```

a) set sql_safe_updates=0;
UPDATE book_info set quantity=quantity*2;

b) SELECT book_name from book_info where price>(select price from book_info where book_name='databse of dummies');

c) SELECT author,book_name,price from book_info where author like 'a%';

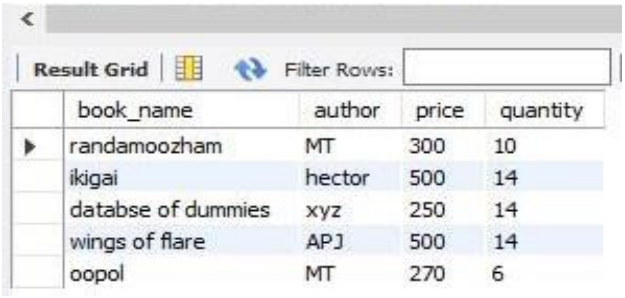
OUTPUT

Query a:

Result Grid | Filter Rows:

book_name	author	price	quantity
randamoozham	MT	300	5
ikigai	hector	500	7
databse of dummies	xyz	250	7
wings of flare	APJ	500	7
oopol	MT	270	3

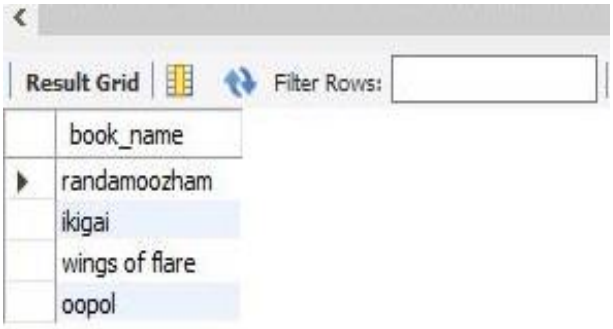
Query b:



Result Grid | Filter Rows:

book_name	author	price	quantity
randamoozham	MT	300	10
ikigai	hector	500	14
databse of dummies	xyz	250	14
wings of flare	APJ	500	14
oopol	MT	270	6

Query c:



Result Grid | Filter Rows:

book_name
randamoozham
ikigai
wings of flare
oopol

RESULT

Query processed successfully and output obtained.

Experiment 10

AIM

Create the Company database with the following tables and do the following:

Administration (employee_salary, development _cost, fund_ amount, turn_over,bonus)
Emp_details (emp_no, emp_name, DOB, address, doj, mobile_no, dept_no, salary).

a. Calculate the total and average salary amount of the employees of each department.

b. Display total salary spent for employees.

c. Develop a PL/SQL function to display total fund_amount spent by the administration department

CODE:

```
CREATE DATABASE company;
USE company;
CREATE TABLE Admins(
emp_sal double,
dvlp_cost double,
fund_amount double,
turn_over double,
bonus double);
CREATE TABLE Emp_details(
emp_no int,
emp_name varchar(20),
DOB date,
address varchar(20),
doj date,
mobile_no int8,
dept_no int,
salary double);
INSERT INTO Admins VALUES
(12000,25000,560000,65000,5000),
(70000,55000,860000,15000,1000),
(18000,45000,160000,75000,7000),
(10000,27000,520000,60000,5000),
(18000,27000,360000,35000,3000);
INSERT INTO Emp_details VALUES
(1,"hamna","1999-10-10","Street - 2 xyz","2020-10-10",9865986598,10,12000),
(2,"ansi","1997-10-10","Street - 2 abc","2020-10-10",9865986598,10,12200),
(3,"sree","1996-10-10","Street ","2020-10-10",9865986598,11,12500),
(4,"stef","1957-10-10","Street in","2020-10-10",9865986598,11,17200),
(5,"anu","1948-10-10","gared","2020-10-10",9865986598,12,12090),
```

(6,"shiva","1988-10-10","Sas","2020-10-10-",9865986598,12,12050);

a) SELECT dept_no,avg(salary) 'Average salary',sum(salary) 'Total Salary' FROM Emp_details GROUP BY dept_no;

b) SELECT sum(salary) 'SUM OF SALARY'FROM Emp_details;

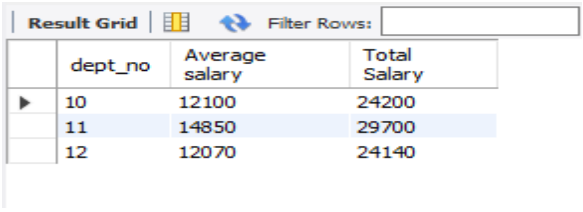
c) //FUCTION//

```
CREATE DEFINER=`root`@`localhost` FUNCTION `fund_total`() RETURNS double
BEGIN
DECLARE f DOUBLE;
 DECLARE i DOUBLE;
 SELECT SUM(fund_amount)
FROM Admins;
RETURN f;
END
```

```
//FUNCTION CALL//
SELECT fund_total() from Admins LIMIT 1;
```

OUTPUT

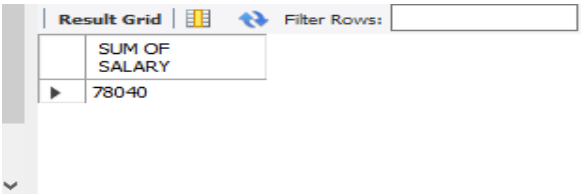
Query a:



Result Grid | Filter Rows:

dept_no	Average salary	Total Salary
10	12100	24200
11	14850	29700
12	12070	24140

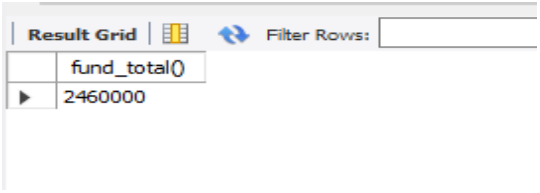
Query b:



Result Grid | Filter Rows:

SUM OF SALARY
78040

Query c:



Result Grid | Filter Rows:

fund_total()
2460000

RESULT

Query processed successfully and output obtained.

Experiment 11

AIM

To create a database containing table employee with employee details.Write PLSQL to update the experience level of employee as beginner,intermediate and advanced.

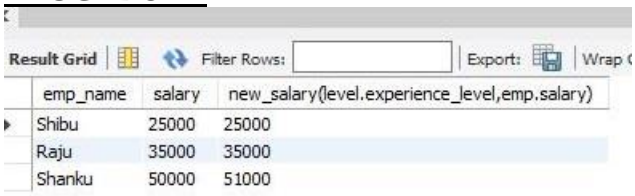
CODE:

```
CREATE DATABASE company;
USE company;
CREATE TABLE emp(emp_id int primary key,emp_name varchar(20),salary varchar(20));
CREATE TABLE dept(dept_id int primary key,emp_id int,designation varchar(20),experience int(10) ,foreign key(emp_id) references emp(emp_id));
INSERT INTO emp(emp_id,emp_name,salary)values(101,'Shibu',25000);
INSERT INTO emp(emp_id,emp_name,salary)values(102,'Raju',35000);
INSERT INTO emp(emp_id,emp_name,salary)values(103,'Shanku',50000);

SELECT * from emp;

INSERT INTO dept(dept_id,emp_id,designation,experience)values(201,101,'Peon',2);
INSERT INTO dept(dept_id,emp_id,designation,experience)values(202,102,'Clerk',6);
INSERT INTO dept(dept_id,emp_id,designation,experience)values(203,103,'Manager',12);

SELECT * from dept;

CREATE TABLE level(emp_id int,dept_id int,experience_level varchar(20),foreign key(emp_id) references emp(emp_id),foreign key(dept_id) references dept(dept_id));

call exp(2,101,201);
call exp(6,102,201);
call exp(12,103,203);

SELECT* from level;
SELECT emp.emp_name,emp.salary,new_salary(level.experience_level,emp.salary) from emp,level where emp.emp_id=level.emp_id;



////STORED PROCEDURE

CREATE DEFINER=`root`@`localhost` PROCEDURE `exp`(experience int,emp_id int,dept_id int)
BEGIN
```

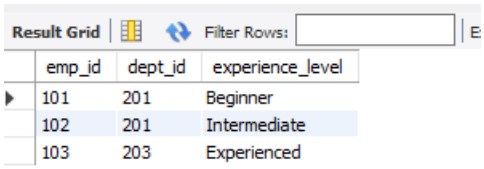
```
DECLARE
   levels varchar(45);

if (experience > 0 && experience<5)
   then set levels = 'beginner';
   insert into employe(emp_id,experience,salary,levels) values(emp_id,experience,salary,levels);
   end if;
   if( exp>=6 && exp <10)
   then set levels = 'intermediate';

    insert into employe(emp_id,experience,salary,levels)
values(emp_id,experience,salary,levels);
   end if;
   if (exp >= 10)
   then set levels = 'Experienced';

   insert into employe(emp_id,experience,salary,levels) values(emp_id,experience,salary,levels);
   end if;
END

////FUNCTION////

CREATE DEFINER=`root`@`localhost` FUNCTION `new_salary`(experience_level
varchar(20),sal varchar(10)) RETURNS int(11)
BEGIN
if(experience_level = 'Experienced')
then
return(sal+1000);
else
return(sal);
end if;

RETURN 1;
END
```

OUTPUT

PROCEDURE



FUNCTION



emp_id	dept_id	experience_level
101	201	Beginner
102	201	Intermediate
103	203	Experienced

RESULT

Query processed successfully and output obtained.

Experiment 12

AIM

Create a table bookings with attributes bk_id,name,source,dist on insert and update ,update billing details in bill table(id,bill)

CODE:

```
CREATE DATABSE train;
USE train;
CREATE TABLE booking(bk_id int(50),name varchar(50),source varchar(50),dist int(50));
CREATE TABLE bill(id int(50),bill int(50));
INSERT INTO booking values(10,&amp;quot;Farhana&amp;quot;,&amp;quot;kyj&amp;quot;,500);
SELECT * from bill;
SELECT * from booking;
INSERT INTO booking values(11,&amp;quot;Arun&amp;quot;,&amp;quot;kollam&amp;quot;,100);
UPDATE booking set dist=150 where bk_id=10;
set SQL_SAFE_UPDATES=0;
DELETE from bill;
DELETE from booking;
```

TRIGGER INSERT:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `train`.`booking_AFTER_INSERT`
AFTER INSERT ON `booking` FOR EACH ROW
BEGIN
if(new.dist&amp;gt;100 and new.dist&amp;lt;=200)
then
insert into bill set id=new.bk_id,bill=800;
end if;
if(new.dist&amp;lt;=100)
then
insert into bill set id=new.bk_id,bill=500;
end if;
if(new.dist&amp;gt;200)
then
insert into bill set id=new.bk_id,bill=0;
end if;
END
```

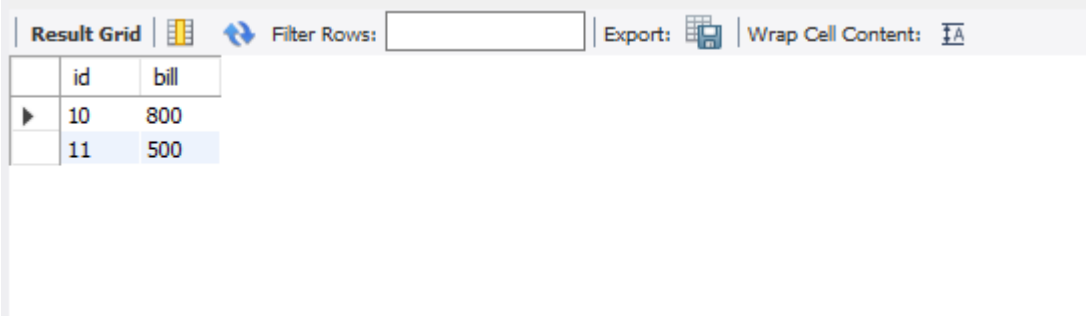
TRIGGER UPDATE:

```
CREATE DEFINER=`root`@`localhost` TRIGGER `train`.`booking_AFTER_UPDATE`
AFTER UPDATE ON `booking` FOR EACH ROW
BEGIN
if(new.dist&amp;gt;100 and new.dist&amp;lt;=200)
then
update bill set bill=800 where id=new.bk_id ;
```

```
end if;
if(new.dist&lt;=100)
then
update bill set bill=500 where id=new.bk_id;
end if;

if(new.dist&gt;200)
then
update bill set bill=0 where id=new.bk_id;
end if;
END
```

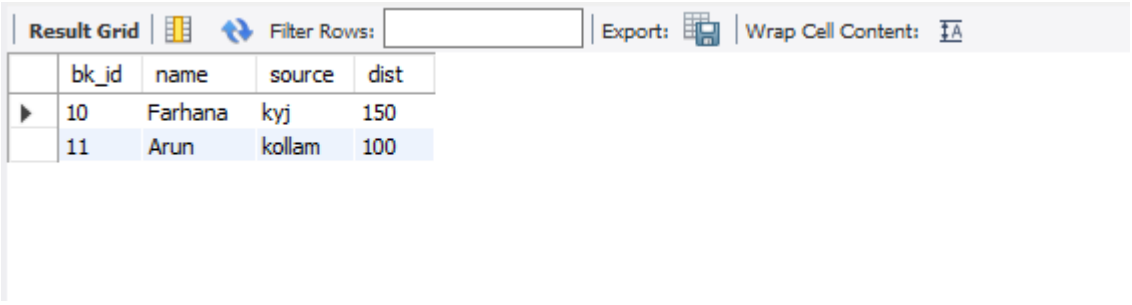
OUTPUT:

TRIGGER INSERT



id	bill
10	800
11	500

TRIGGER UPDATE



bk_id	name	source	dist
10	Farhana	kyj	150
11	Arun	kollam	100

RESULT

Query processed successfully and output obtained.

Experiment 13

AIM

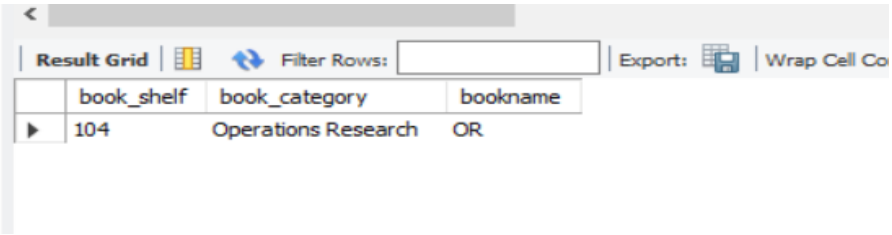
To write a program to implement cursor.

CODE

```
CREATE DATABASE college;
USE college;
CREATE TABLE library(shelf_no int,category varchar(10),book_name varchar(20));
INSERT INTO library values(101,'Topology','Real Analysis');
INSERT INTO library values(102,'Algebra','Linear Algebra');
INSERT INTO library values(103,'Analysis','Complex Analysis');
INSERT INTO library values(104,'OR','Operations Research');
INSERT INTO library values(106,'NumberSys','AbstractAlg');

CREATE TABLE book_by_order(book_shelf int,book_category varchar(20),bookname
varchar(20));
SELECT* from library;

call book_details();

/*
CREATE DEFINER=`root`@`localhost` PROCEDURE `book_details`()
BEGIN
declare book_shelf int;
declare bookname varchar(20);
declare book_category varchar(10);
declare C_finished integer default 0;
declare C1 cursor for select shelf_no,category,book_name from library;
declare continue handler for not found set C_finished = 1;
open C1;
book_details:loop
if C_finished=1 then
leave book_details;
end if;
if C_finished = 0 then
Fetch from C1 into book_shelf,book_category,bookname;
if book_category = 'OR' then
insert into book_by_order values(book_shelf,bookname,book_category);
end if;
end if;
end loop;
close C1;
```

```
END
*/
```

OUTPUT



RESULT

Query processed successfully and output obtained.

Experiment 14

AIM

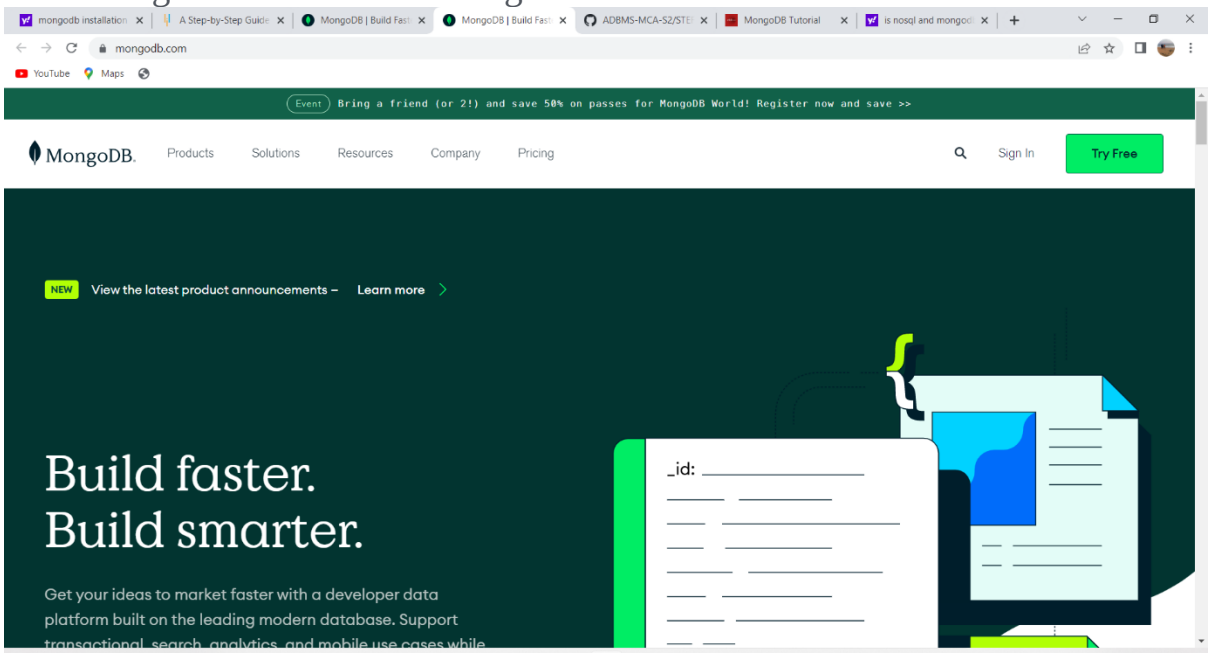
To understand the installation and configuration of NOSQL databases.

CODE

MongoDB is a cross-platform, document oriented NoSql database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

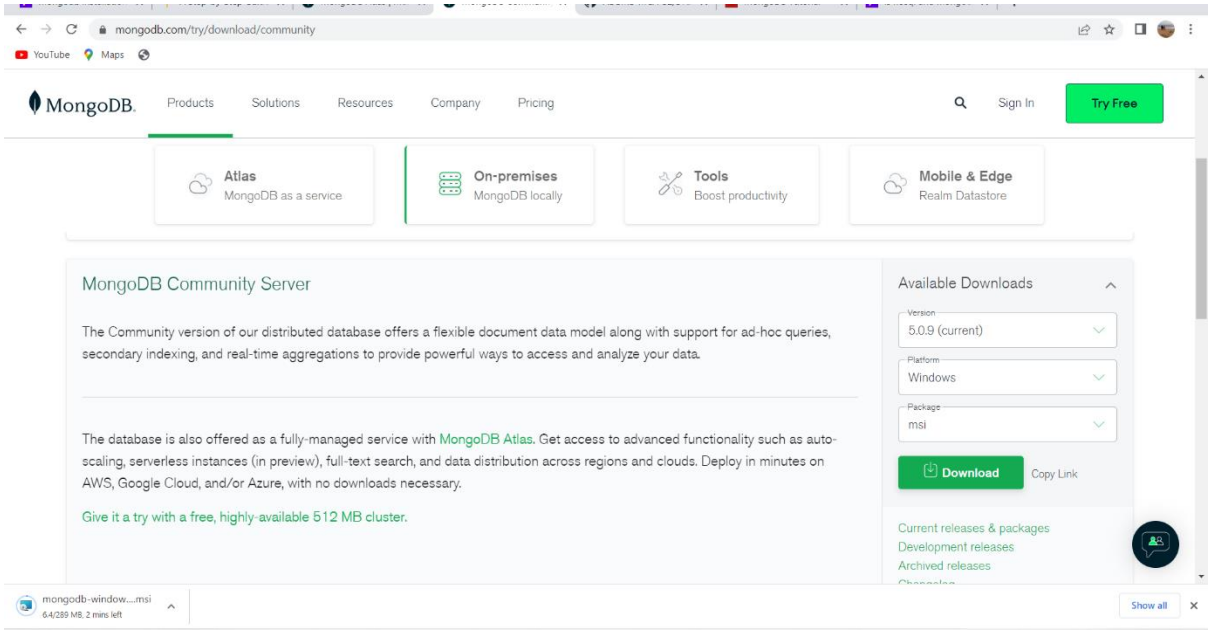
STEP 1:

Navigate to the official MongoDB website.



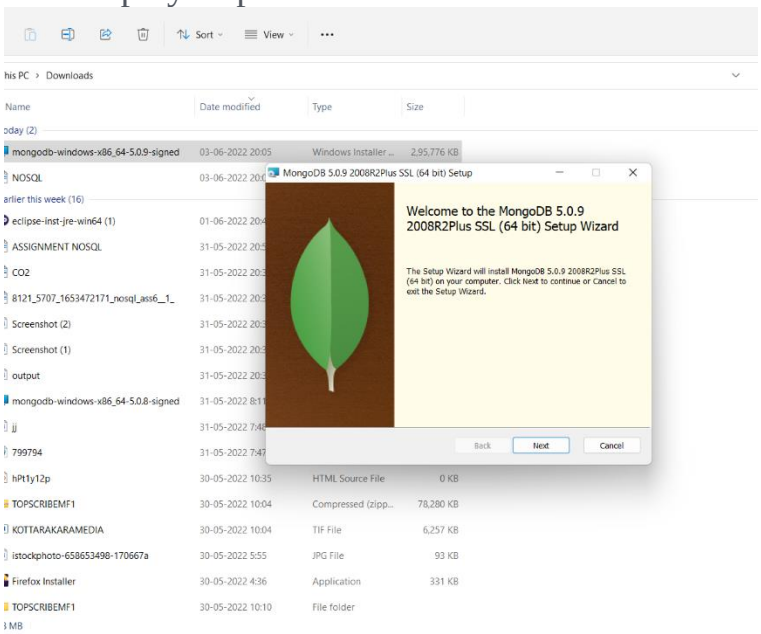
STEP 2:

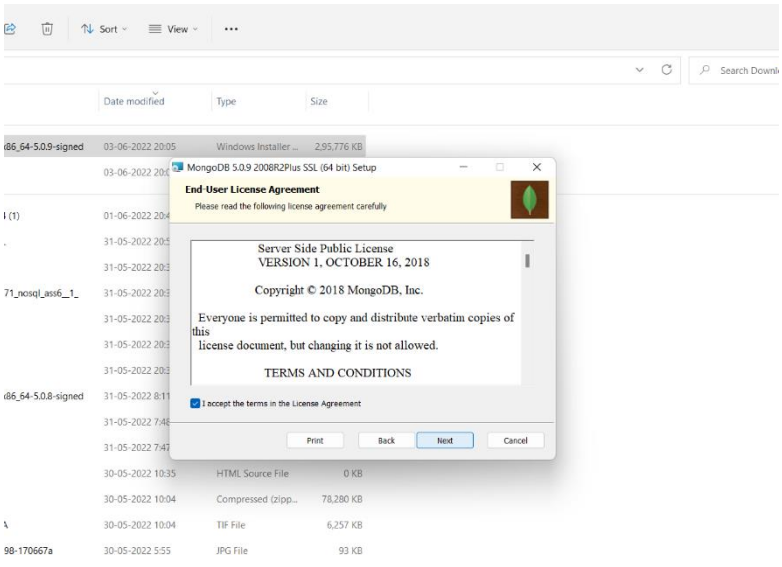
Under the products section, click on the Community server version. Make sure that the specifications to the right of the screen are correct. At the time of writing, the latest version is 4.4.5. Ensure that the platform is Windows, and the package is MSI. Go ahead and click on download.

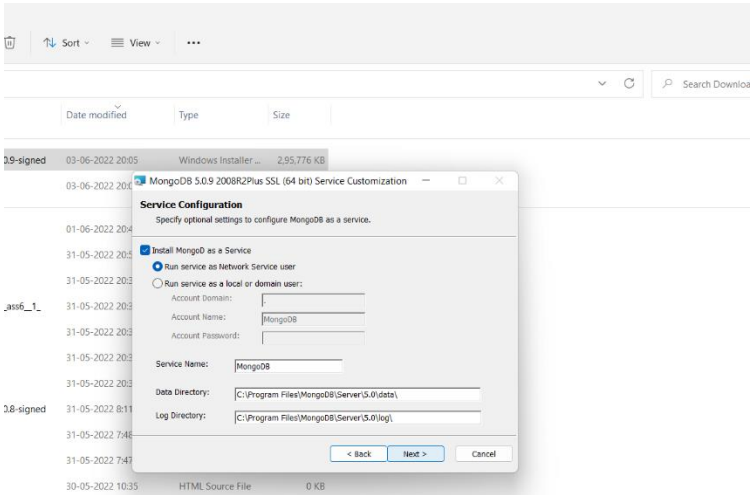


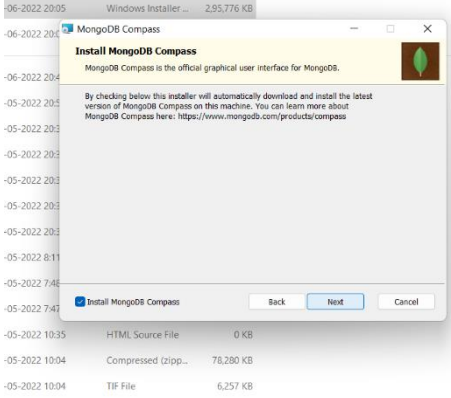
STEP 3:

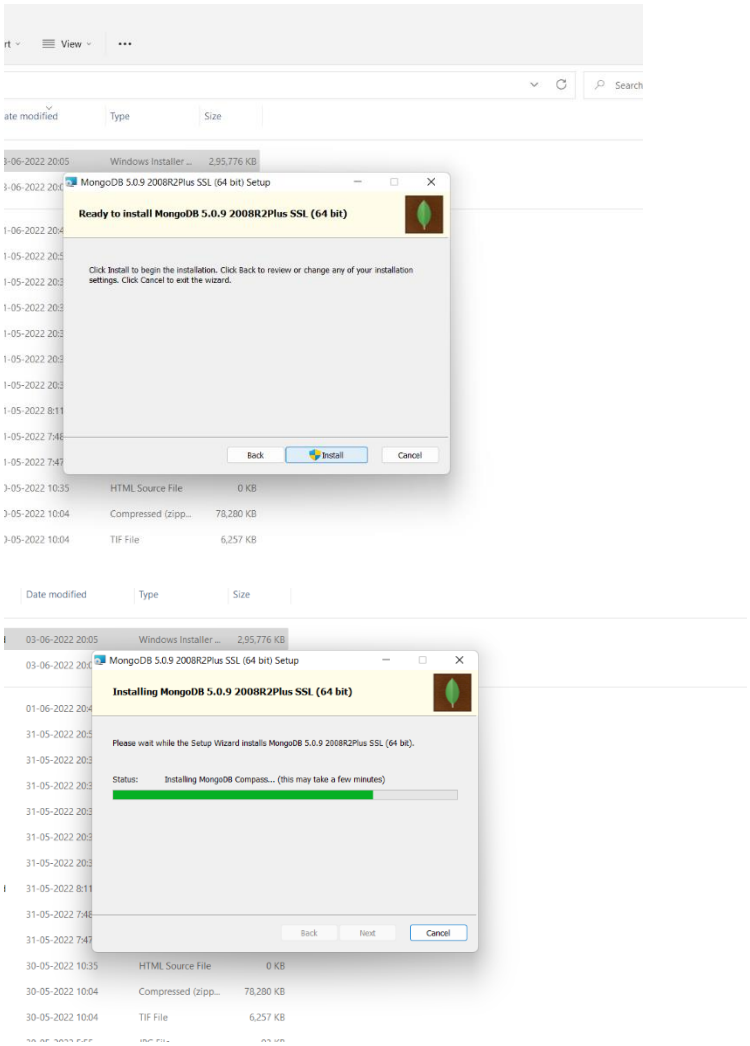
You can find the downloaded file in the downloads directory. Install the software step by step.




Sort
View
Search Downl
Date modified
Type
Size
MongoDB 5.0.9 2008R2Plus SSL (64 bit) Setup
End-User License Agreement
Please read the following license agreement carefully
Server Side Public License
VERSION 1, OCTOBER 16, 2018
Copyright © 2018 MongoDB, Inc.
Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.
TERMS AND CONDITIONS
I accept the terms in the License Agreement
Print
Back
Next
Cancel

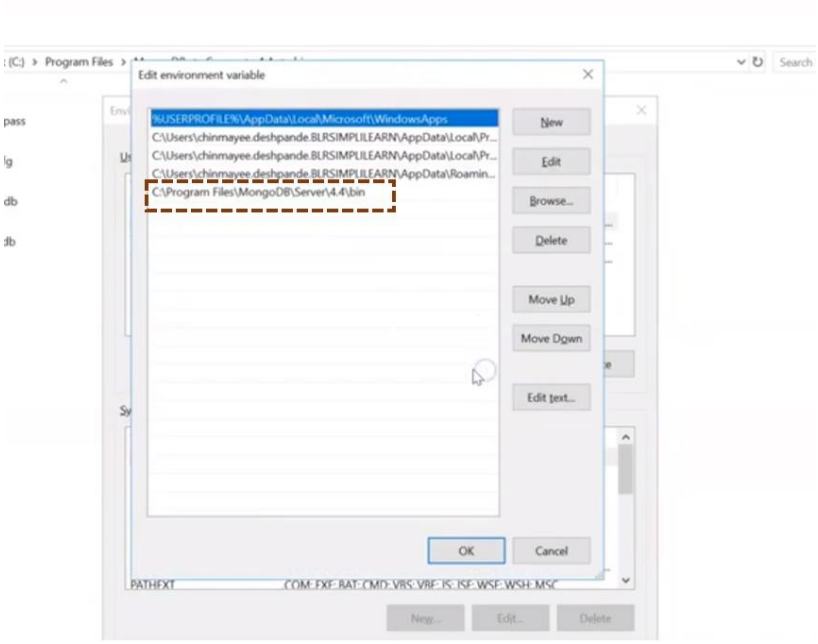

Sort
View
Search Downloa
Date modified
Type
Size
MongoDB 5.0.9 2008R2Plus SSL (64 bit) Service Customization
Service Configuration
Specify optional settings to configure MongoDB as a service.
Install MongoD as a Service
Run service as Network Service user
Run service as a local or domain user:
Account Domain:
Account Name:
MongoDB
Account Password:
Service Name:
MongoDB
Data Directory:
C:\Program Files\MongoDB\Server\5.0\data\
Log Directory:
C:\Program Files\MongoDB\Server\5.0\log\
< Back
Next >
Cancel


View
Type
Size
MongoDB Compass
Install MongoDB Compass
MongoDB Compass is the official graphical user interface for MongoDB.
By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/compass
Install MongoDB Compass
Back
Next
Cancel



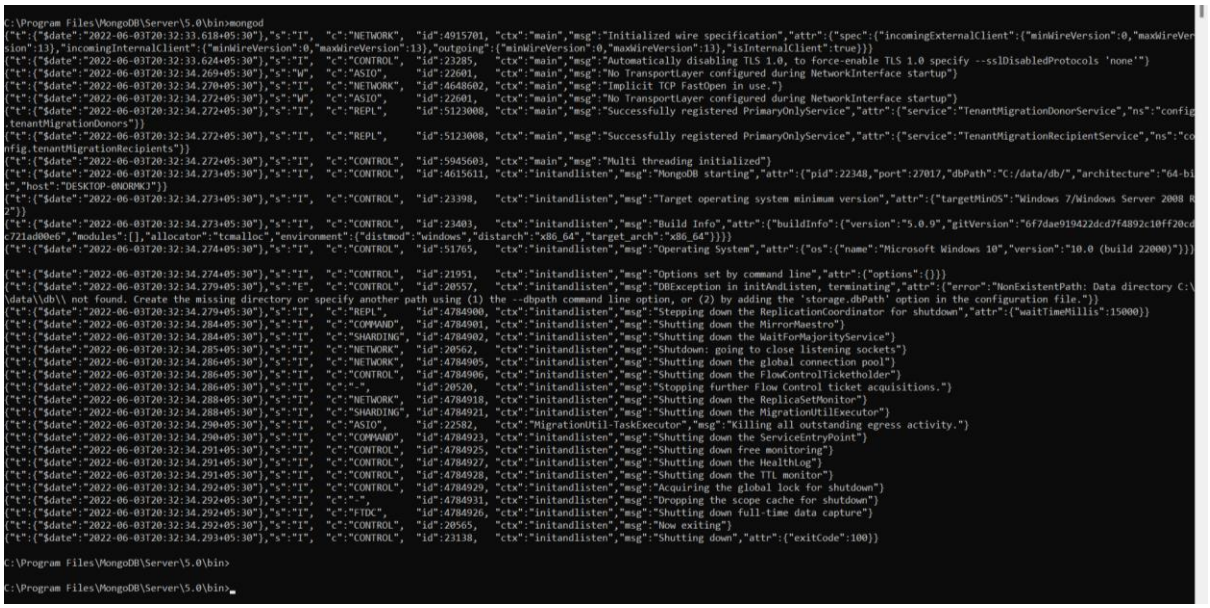
STEP:4

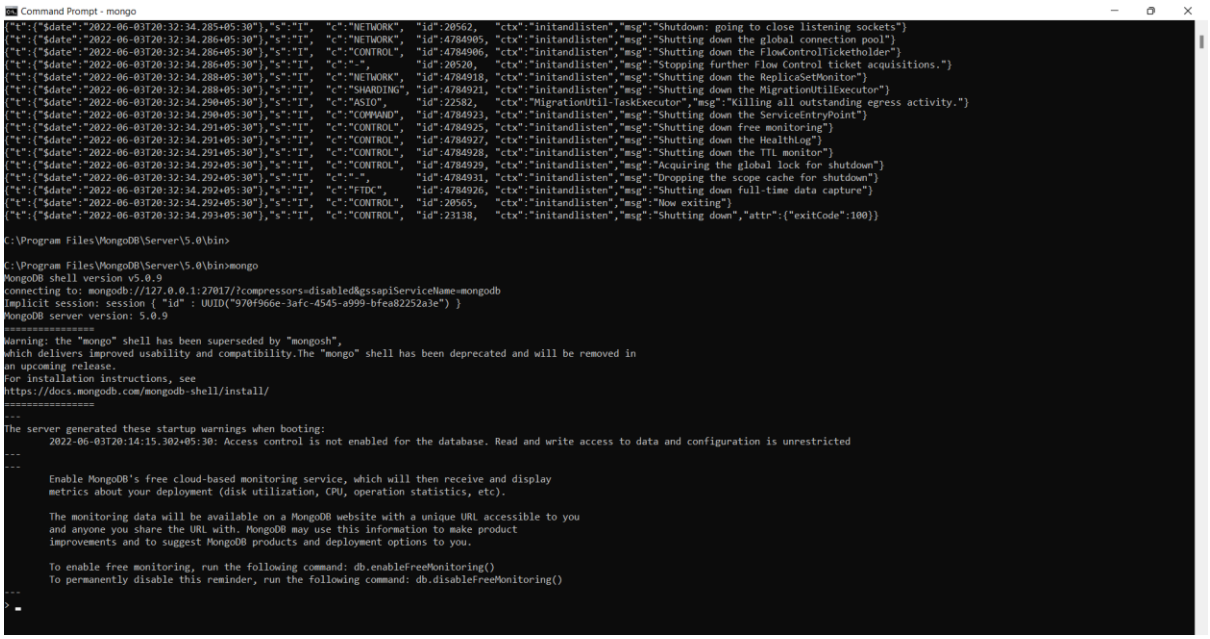
create an environment variable for the executable file so that we don't have to change the directory structure every time we want to execute the file.



STEP:5

After creating an environment path, we can open the command prompt and type mongod and then mongo.




Command Prompt - mongo
{"t":{"$date":"2022-06-03T20:32:34.285+05:30"},"s":"I", "c":"NETWORK", "id":20562, "ctx":"initandlisten","msg":"Shutdown: going to close listening sockets"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I", "c":"NETWORK", "id":4784905, "ctx":"initandlisten","msg":"Shutting down the global connection pool"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I", "c":"CONTROL", "id":4784906, "ctx":"initandlisten","msg":"Shutting down the FlowControlTicketholder"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I", "c":"-", "id":20520, "ctx":"initandlisten","msg":"Stopping further Flow Control ticket acquisitions."}
{"t":{"$date":"2022-06-03T20:32:34.288+05:30"},"s":"I", "c":"NETWORK", "id":4784918, "ctx":"initandlisten","msg":"Shutting down the ReplicaSetMonitor"}
{"t":{"$date":"2022-06-03T20:32:34.288+05:30"},"s":"I", "c":"SHARDING", "id":4784921, "ctx":"initandlisten","msg":"Shutting down the MigrationUtilExecutor"}
{"t":{"$date":"2022-06-03T20:32:34.290+05:30"},"s":"I", "c":"ASIO", "id":22582, "ctx":"MigrationUtil-TaskExecutor","msg":"Killing all outstanding egress activity."}
{"t":{"$date":"2022-06-03T20:32:34.290+05:30"},"s":"I", "c":"COMMAND", "id":4784923, "ctx":"initandlisten","msg":"Shutting down the ServiceEntryPoint"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I", "c":"CONTROL", "id":4784925, "ctx":"initandlisten","msg":"Shutting down free monitoring"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I", "c":"CONTROL", "id":4784927, "ctx":"initandlisten","msg":"Shutting down the HealthLog"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I", "c":"CONTROL", "id":4784928, "ctx":"initandlisten","msg":"Shutting down the TTL monitor"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I", "c":"CONTROL", "id":4784929, "ctx":"initandlisten","msg":"Acquiring the global lock for shutdown"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I", "c":"-", "id":4784931, "ctx":"initandlisten","msg":"Dropping the scope cache for shutdown"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I", "c":"FTDC", "id":4784926, "ctx":"initandlisten","msg":"Shutting down full-time data capture"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I", "c":"CONTROL", "id":20565, "ctx":"initandlisten","msg":"Now exiting"}
{"t":{"$date":"2022-06-03T20:32:34.293+05:30"},"s":"I", "c":"CONTROL", "id":23138, "ctx":"initandlisten","msg":"Shutting down","attr":{"exitCode":100}}
C:\Program Files\MongoDB\Server\5.0\bin>
C:\Program Files\MongoDB\Server\5.0\bin>mongo
MongoDB shell version v5.0.9
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("970f966e-3afc-4545-a999-bfea82252a3e") }
MongoDB server version: 5.0.9
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
The server generated these startup warnings when booting:
2022-06-03T20:14:15.302+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
>

Experiment 15

AIM

To build sample collection/documents to perform query operations.

CODE:

```
test> use student
switched to db student
student> db.createCollection("stud")
{ ok: 1 }
student> db.stud.insert({"srn":101,"sname":"Adharsh","degree":"Bsc","semester":4,"cgpa":6.7})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62a453a8de96bb8621ae7cf1") }
}
student> db.stud.insert({"srn":102,"sname":"Binoy","degree":"Bca","semester":4,"cgpa":9.7})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62a453cbde96bb8621ae7cf2") }
}
student> db.stud.insert({"srn":103,"sname":"Rahul","degree":"Bca","semester":6,"cgpa":6.3})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62a453e7de96bb8621ae7cf3") }
}
student> db.stud.insert({"srn":104,"sname":"Arun","degree":"Bca","semester":6,"cgpa":7.3})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62a45401de96bb8621ae7cf4") }
}
student> db.stud.insert({"srn":105,"sname":"Amal","degree":"Bsc","semester":6,"cgpa":5.3})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("62a45419de96bb8621ae7cf5") }
}
```

OUTPUT

RESULT

Query processed successfully and output obtained.

Experiment 16

AIM

To perform CRUD operations on the student database.

1. Display all the documents.

2. Display all the students in Bca.

3. Display all the students in ascending order.

4. Display all the top three students.

5. Display the students 1,2,3.

6. Display the degree of the student Rahul.

7.Display the student details of 3,4,5 in descending order of cgpa

8. Display the number of sttudents in Bca

9. Display all the degree without the _id.

10. Display the distinct degree.

CODE:

Query 1:

```
student> db.stud.find().pretty()
```

Query 2:

```
student> db.stud.find({"degree":"Bca"}).pretty()
```

Query 3:

```
student> db.stud.find({},{sname:1,_id:0}).sort({sname:1})
```

Query 4:

```
student> db.stud.find({},{sname:1,_id:0}).limit(3).sort({cgpa:-1})
```

Query 5:

```
student> db.stud.find().skip(2).limit(3)
```

Query 6:

```
db.stud.find({sname:'Rahul'},{degree:1,_id:0}).pretty()
```

Query 7:

```
db.stud.find().skip(2).limit(3).sort({cgpa:-1})
```

Query 8:

```
db.stud.countDocuments({degree:'Bca'})
```

Query 9:

db.stud.find({},{degree:1,_id:0}).pretty()

Query 10:

db.stud.distinct("degree")

Query 11:

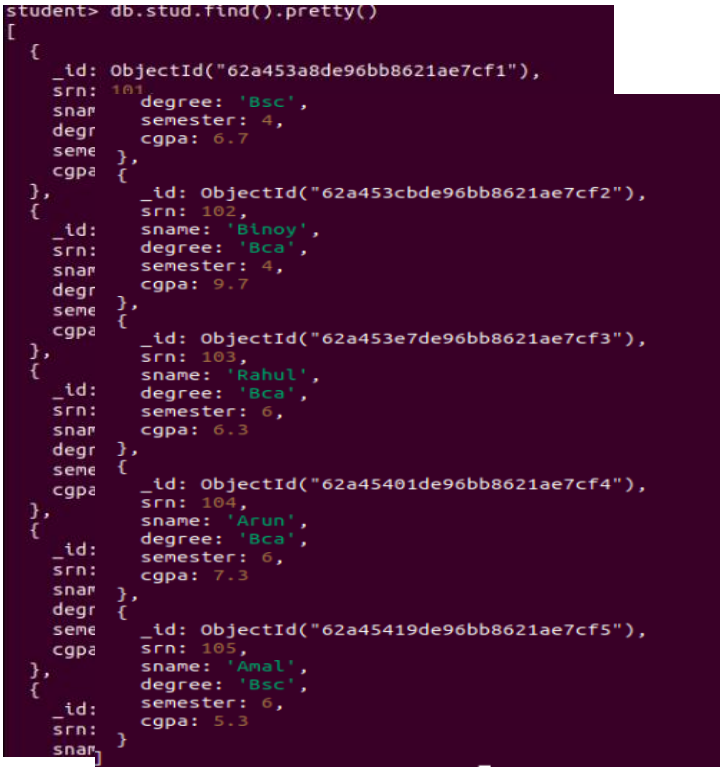
db.stud.find({$and:[{cgpa:{$gt:6,$lt:7}},{degree:'Bca'}]})

Query 12:

db.stud.find({$and:[{degree:'Bca'},{semester:6}]},{sname:1,_id:0,semester:1})

OUTPUT

Query 1:



Query 2:

```
student> db.stud.find({"degree":"Bca"}).pretty()
[
  {
    _id: ObjectId("62a453cbde96bb8621ae7cf2"),
    srn: 102,
    sname: 'Binoy',
    degree: 'Bca',
    semester: 4,
    cgpa: 9.7
  },
  {
    _id: ObjectId("62a453e7de96bb8621ae7cf3"),
    srn: 103,
    sname: 'Rahul',
    degree: 'Bca',
    semester: 6,
    cgpa: 6.3
  },
  {
    _id: ObjectId("62a45401de96bb8621ae7cf4"),
    srn: 104,
    sname: 'Arun',
    degree: 'Bca',
    semester: 6,
    cgpa: 7.3
  }
]
```

Query 3:

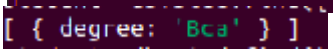
```
student> db.stud.find({},{sname:1, _id:0}).sort({sname:1})
[
  { sname: 'Adharsh' },
  { sname: 'Amal' },
  { sname: 'Arun' },
  { sname: 'Binoy' },
  { sname: 'Rahul' }
]
```

Query 4:

[{ sname: 'Binoy' }, { sname: 'Arun' }, { sname: 'Adharsh' }]

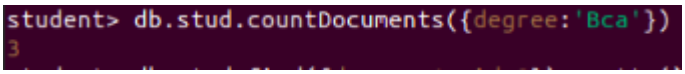
Query 5:

Query 6:

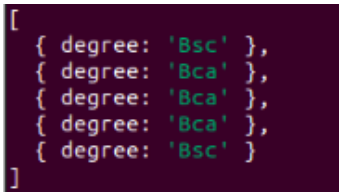


Query 7:

Query 8:



Query 9:



Query 10:

```
[ 'Bca', 'Bsc' ]
```

Query 11:

```
[
  {
    _id: ObjectId("62a453e7de96bb8621ae7cf3"),
    srn: 103,
    sname: 'Rahul',
    degree: 'Bca',
    semester: 6,
    cgpa: 6.3
  }
]
```

Query 12:

```
[
  {
    _id: ObjectId("62a453e7de96bb8621ae7cf3"),
    srn: 103,
    sname: 'Rahul',
    degree: 'Bca',
    semester: 6,
    cgpa: 6.3
  },
  {
    _id: ObjectId("62a45401de96bb8621ae7cf4"),
    srn: 104,
    sname: 'Arun',
    degree: 'Bca',
    semester: 6,
    cgpa: 7.3
  }
]
```

RESULT

Query processed successfully and output obtained.

Experiment 17

AIM:

Create an employee database with the field, eid, ename, dept, desig, salary, yoj, address: {dno, street, locality, city}

1.Display the all the epmployee with salary in the range(5000,7500)
2.Display all the employee with design developments
3.Display the salary of Rahul
4.Display the city of employee
5.Update the salary of developers by 5000
6.Add field age to Rahul
7.Remove yoj from Rahul
8.Add an array field project to Rahul
9.Add p2,p3 project to Rahul
10.Remove p3 from Rahul
11.Add a new embedded object 'contact' with email and phone as array object to Rahul

CODE:

Query 1:

```
Use emp;
Db.emp.find({salary:{$gt50000,$lt:75000}},{_id:0,ename:1})
```

Query 2:

```
db.emp.find({desg:'dev'},{_id:0,ename:1})
```

Query 3:

```
db.emp.find({ename:'Rahul'},{ename:1,salary:1,_id:0})
```

Query 4:

```
db.emp.find({},{ename:1,address:{city:1},_id:0})
```

Query 5:

```
db.emp.updateMany({desg:'dev'},{$inc:{salary:5000}})
```

Query 6:

```
db.emp.updateMany({},{$set:{age:25}})
```

Query 7:

```
db.emp.updateOne({ename:'Rahul'},{$unset:{yoj:""}})
```

Query 8:

```
db.emp.updateOne({ename:'Rahul'},{$push:{project:'p1'}})
```

Query 9:

```
db.emp.updateOne({ename:'Rahul'},{$addToSet:{project:{$each:['p2','p3']}}})
```

Query 10:

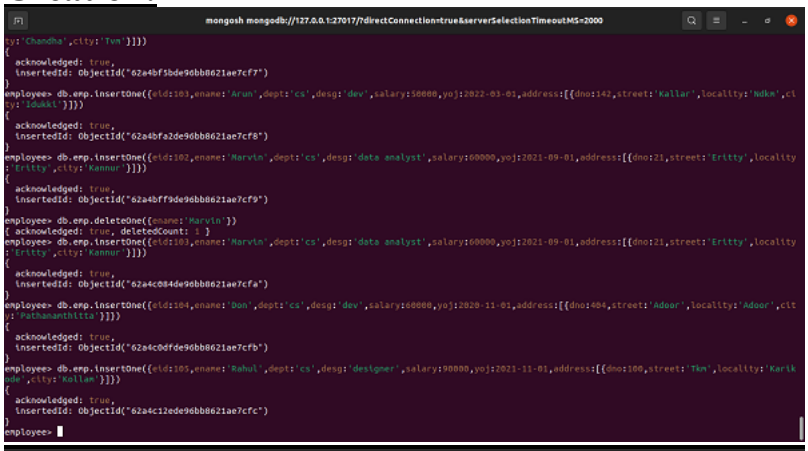
```
db.emp.updateOne({ename:'Rahul'},{$pull:{project:'p3'}})
```

Query 11:

db.emp.updateOne({ename:'Rahul'},{$push:{contacts:{phone:626762232,email:'rahul@gmail.com'}}})

OUTPUT:

Creation:



Query 1:

```
employee> db.emp.find({salary:{$gt:50000,$lt:75000}},{_id:0,ename:1})
[ { ename: 'Adharsh' }, { ename: 'Marvin' }, { ename: 'Don' } ]
employee>
```

Query 2:

```
employee> db.emp.find({desg:'dev'},{ename:1,_id:0})
[ { ename: 'Binoy' }, { ename: 'Arun' }, { ename: 'Don' } ]
employee>
```

Query 3:

```
employee> db.emp.find({ename:'Rahul'},{ename:1,salary:1,_id:0})
[ { ename: 'Rahul', salary: 90000 } ]
employee>
```

Query 4:

```
employee> db.emp.find({},{ename:1,address:{city:1},_id:0})
[
  { ename: 'Binoy', address: [ { city: 'Kollam' } ] },
  { ename: 'Adharsh', address: [ { city: 'Tvm' } ] },
  { ename: 'Arun', address: [ { city: 'Idukki' } ] },
  { ename: 'Marvin', address: [ { city: 'Kannur' } ] },
  { ename: 'Don', address: [ { city: 'Pathanamthitta' } ] },
  { ename: 'Rahul', address: [ { city: 'Kollam' } ] }
]
employee>
```

Query 5:

```
employee> db.emp.updateMany({desg:'dev'},{$inc:{salary:5000}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

Query 6:

```
employee> db.emp.updateMany({},{$set:{age:25}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 6,
  modifiedCount: 6,
  upsertedCount: 0
}
```

Query 7:

```
employee> db.emp.updateOne({ename:'Rahul'},{$unset:{yoj:""}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Query 8:

```
employee> db.emp.updateOne({ename:'Rahul'},{$push:{project:'p1'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Query 9:

```
employee> db.emp.updateOne({ename:'Rahul'},{$addToSet:{project:{$each:['p2','p3']}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.emp.find({ename:'Rahul'},{})
[
  {
    _id: ObjectId("62a4c12ede96bb8621ae7cfc"),
    eid: 105,
    ename: 'Rahul',
    dept: 'cs',
    desg: 'designer',
    salary: 90000,
    address: [
      { dno: 100, street: 'Tkm', locality: 'Karikode', city: 'Kollam' }
    ],
    age: 25,
    project: [ 'p1', 'p2', 'p3' ]
  }
]
```

Query 10:

```
employee> db.emp.updateOne({ename:'Rahul'},{$pull:{project:'p3'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.emp.find({ename:'Rahul'},{})
[
  {
    _id: ObjectId("62a4c12ede96bb8621ae7cfc"),
    eid: 105,
    ename: 'Rahul',
    dept: 'cs',
    desg: 'designer',
    salary: 90000,
    address: [
      { dno: 100, street: 'Tkm', locality: 'Karikode', city: 'Kollam' }
    ],
    age: 25,
    project: [ 'p1', 'p2' ]
  }
]
employee>
```

Query 11:

```
employee> db.emp.updateOne({ename:'Rahul'},{$push:{contacts:{phone:626762232,email:'rahul@gmail.com'}}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
employee> db.emp.find({ename:'Rahul'},{})
[
  {
    _id: ObjectId("62a4c12ede96bb8621ae7cfc"),
    eid: 105,
    ename: 'Rahul',
    dept: 'cs',
    desg: 'designer',
    salary: 90000,
    address: [
      { dno: 100, street: 'Tkn', locality: 'Karikode', city: 'Kollan' }
    ],
    age: 25,
    project: [ 'p1', 'p2' ],
    contacts: [ { phone: 626762232, email: 'rahul@gmail.com' } ]
  }
]
```

RESULT

Query processed successfully and output obtained.

Experiment 18

AIM:

Create database candidate and collection details.

1.Query customer who are either male or younger than 25 using aggregate method.
2.Calculate total purchase amount for male and female using aggregate method.
3.Select customer who are older than 25 and calculate the average purchase amount for males and females.
4.Sort the data based on average amount.

CODE:

Query 1:

```
use candidate;
db.details.find({$or:[{gender:'M'},{age:{$lt:25}}]},{name:1,_id:0})
```

Query 2:

```
db.details.aggregate([{$group:{_id:"$gender","amount":{$sum:"$amount"}}}])
```

Query 3:

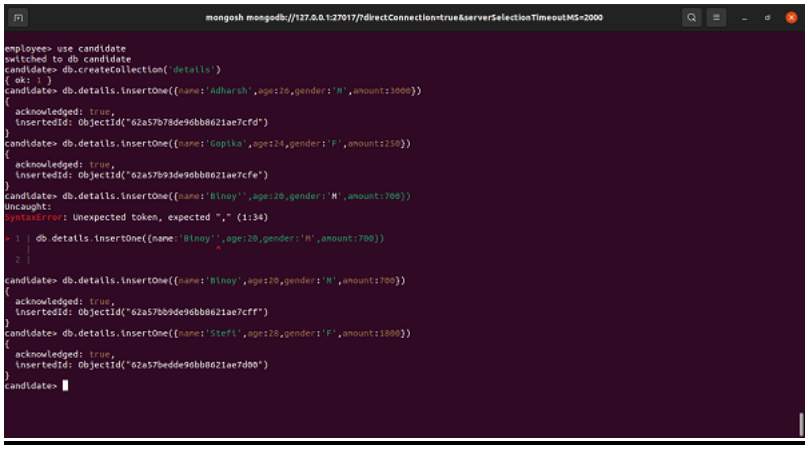
```
db.details.aggregate([{$match:{age:{$gt:25}}},{$group:{_id:'$gender',amount:{$avg:'$amount'
}}}])
```

Query 4:

```
db.details.aggregate([{$group:{_id:'$gender',amount:{$avg:'$amount'}}},{$sort:{amount:1}}])
db.details.aggregate([{ $group: { _id: '$gender', amount: { $avg: '$amount' } } }, { $sort:
{ amount: -1 } }])
```

OUTPUT:

Creation:



Query 1:

```
candidate> db.details.find({$or:[{gender:'M'},{age:{$lt:25}}]},{name:1,_id:0})
[ { name: 'Adharsh' }, { name: 'Gopika' }, { name: 'Binoy' } ]
candidate> db.details.find({$or:[{gender:'M'},{age:{$lt:25}}]})
[
  {
    _id: ObjectId("62a57b78de96bb8621ae7cfd"),
    name: 'Adharsh',
    age: 26,
    gender: 'M',
    amount: 3000
  },
  {
    _id: ObjectId("62a57b93de96bb8621ae7cfe"),
    name: 'Gopika',
    age: 24,
    gender: 'F',
    amount: 250
  },
  {
    _id: ObjectId("62a57bb9de96bb8621ae7cff"),
    name: 'Binoy',
    age: 20,
    gender: 'M',
    amount: 700
  }
]
```

Query 2:

```
candidate> db.details.aggregate([{$group:{_id:"$gender","amount":{$sum:"$amount"}}}])
[ { _id: 'M', amount: 3700 }, { _id: 'F', amount: 2050 } ]
```

Query 3:

```
candidate> db.details.insertOne({name:'Arya',age:27,gender:'F',amount:100})
{
  acknowledged: true,
  insertedId: ObjectId("62a58605de96bb8621ae7d02")
}
candidate> db.details.aggregate([{$match:{age:{$gt:25}}},{$group:{_id:'$gender',amount:{$avg:'$amount'}}}])
[ { _id: 'M', amount: 3000 }, { _id: 'F', amount: 950 } ]
candidate>
```

Query 4:

```
candidate> db.details.aggregate([{$group:{_id:'$gender',amount:{$avg:'$amount'}}},{$sort:{amount:1}}])
[ { _id: 'F', amount: 716.6666666666666 }, { _id: 'M', amount: 1850 } ]
candidate> db.details.aggregate([{ $group: { _id: '$gender', amount: { $avg: '$amount' } } }, { $sort: { amount: -1 } }])
[ { _id: 'M', amount: 1850 }, { _id: 'F', amount: 716.6666666666666 } ]
candidate>
```

RESULT

Query processed successfully and output obtained.

Experiment 19

AIM:

Create database college and collection student.

1.Display the details of the students who have their name starting with the letter c using $regex operator.
2. Display the details of the students who have their name ending with the letter r using $regex operator.
3. Display the details of the students who having cs as their department.
4. Remove the details of the students who having ec as their department.

CODE:

Query 1:

```
use college;
db.student.find({sname:{$regex:"^c"}});
```

Query 2:

```
db.student.find({sname:{$regex:"r$"}});
```

Query 3:

```
db.student.find({dept:{$regex:"cs"}});
```

Query 4:

```
db.student.remove({dept:{$regex:"ec"}});
```

OUTPUT:

```
> use college;
switched to db college
> db.createCollection("student");
{ "ok" : 1 }
> db.student.insert({srn:1,sname:"arun",dept:"cs",sem:1,cgpa:9})
WriteResult({ "nInserted" : 1 })
> db.student.insert({srn:2,sname:"adharsh",dept:"ec",sem:2,cgpa:8})
WriteResult({ "nInserted" : 1 })
> db.student.insert({srn:3,sname:"binoy",dept:"cs",sem:3,cgpa:7})
WriteResult({ "nInserted" : 1 })
> db.student.insert({srn:4,sname:"chandrakanth",dept:"mca",sem:2,cgpa:8})
WriteResult({ "nInserted" : 1 })
> db.student.insert({srn:5,sname:"amar",dept:"cs",sem:5,cgpa:9})
WriteResult({ "nInserted" : 1 })
> db.student.find()
{ "_id" : ObjectId("62d1218fd7efa24cfff14608"), "srn" : 1, "sname" : "arun", "dept" : "cs", "sem" : 1, "cgpa" : 9 }
{ "_id" : ObjectId("62d1219ed7efa24cfff14609"), "srn" : 2, "sname" : "adharsh", "dept" : "ec", "sem" : 2, "cgpa" : 8 }
{ "_id" : ObjectId("62d121acd7efa24cfff1460a"), "srn" : 3, "sname" : "binoy", "dept" : "cs", "sem" : 3, "cgpa" : 7 }
{ "_id" : ObjectId("62d121b8d7efa24cfff1460b"), "srn" : 4, "sname" : "chandrakanth", "dept" : "mca", "sem" : 2, "cgpa" : 8 }
{ "_id" : ObjectId("62d121c2d7efa24cfff1460c"), "srn" : 5, "sname" : "amar", "dept" : "cs", "sem" : 5, "cgpa" : 9 }
```

Query 1:

```
> db.student.find({sname:{$regex:"^c"}});
{ "_id" : ObjectId("62d121b8d7efa24cfff1460b"), "srn" : 4, "sname" : "chandrakanth", "dept" : "mca", "sem" : 2, "cgpa" : 8 }
```

Query 2:

```
> db.student.find({sname:{$regex:"r$"}});
{ "_id" : ObjectId("62d121c2d7efa24cfff1460c"), "srn" : 5, "sname" : "amar", "dept" : "cs", "sem" : 5, "cgpa" : 9 }
```

Query 3:

```
> db.student.find({dept:{$regex:"cs"}});
{ "_id" : ObjectId("62d1218fd7efa24cfff14608"), "srn" : 1, "sname" : "arun", "dept" : "cs", "sem" : 1, "cgpa" : 9 }
{ "_id" : ObjectId("62d121acd7efa24cfff1460a"), "srn" : 3, "sname" : "binoy", "dept" : "cs", "sem" : 3, "cgpa" : 7 }
{ "_id" : ObjectId("62d121c2d7efa24cfff1460c"), "srn" : 5, "sname" : "amar", "dept" : "cs", "sem" : 5, "cgpa" : 9 }
```

Query 4:

```
> db.student.remove({dept:{$regex:"ec"}});
WriteResult({ "nRemoved" : 1 })
> db.student.find();
{ "_id" : ObjectId("62d1218fd7efa24cfff14608"), "srn" : 1, "sname" : "arun", "dept" : "cs", "sem" : 1, "cgpa" : 9 }
{ "_id" : ObjectId("62d121acd7efa24cfff1460a"), "srn" : 3, "sname" : "binoy", "dept" : "cs", "sem" : 3, "cgpa" : 7 }
{ "_id" : ObjectId("62d121b8d7efa24cfff1460b"), "srn" : 4, "sname" : "chandrakanth", "dept" : "mca", "sem" : 2, "cgpa" : 8 }
{ "_id" : ObjectId("62d121c2d7efa24cfff1460c"), "srn" : 5, "sname" : "amar", "dept" : "cs", "sem" : 5, "cgpa" : 9 }
```

RESULT

Query processed successfully and output obtained.

Experiment 20

AIM:

To write queries to implement backup and monitoring in mongodb.

CODE:

Query 1:

mongodump

Query 2:

mongorestore ./dump/

Query 3:

mongodump --db=employee

Query 4:

mongorestore --db employee dump/employee

Query 5:

mongodump --db student --collection Details

Query 6:

mongorestore --db student --collection Details dump/student/Details.bson

Query 7:

mongostat

OUTPUT:

Query 1:

```
C:\Users\Arun>mongodump
2022-07-15T14:24:13.239+0530    writing admin.system.users to dump\admin\system.users.bson
2022-07-15T14:24:13.250+0530    done dumping admin.system.users (1 document)
2022-07-15T14:24:13.251+0530    writing admin.system.version to dump\admin\system.version.bson
2022-07-15T14:24:13.256+0530    done dumping admin.system.version (2 documents)
2022-07-15T14:24:13.256+0530    writing college.studlist to dump\college\studlist.bson
2022-07-15T14:24:13.257+0530    writing studentdb.students to dump\studentdb\students.bson
2022-07-15T14:24:13.258+0530    writing employee.employees to dump\employee\employees.bson
2022-07-15T14:24:13.259+0530    writing student.Details to dump\student\Details.bson
2022-07-15T14:24:13.262+0530    done dumping college.studlist (18 documents)
2022-07-15T14:24:13.262+0530    done dumping studentdb.students (10 documents)
2022-07-15T14:24:13.264+0530    done dumping employee.employees (3 documents)
2022-07-15T14:24:13.267+0530    done dumping student.Details (3 documents)
```

Query 2:

```
C:\Users\Arun>mongorestore ./dump/
2022-07-15T14:25:42.362+0530    preparing collections to restore from
2022-07-15T14:25:42.375+0530    reading metadata for employee.employees from dump\employee\employees.metadata.json
2022-07-15T14:25:42.401+0530    restoring employee.employees from dump\employee\employees.bson
2022-07-15T14:25:42.421+0530    finished restoring employee.employees (3 documents, 0 failures)
2022-07-15T14:25:42.421+0530    restoring users from dump\admin\system.users.bson
2022-07-15T14:25:42.456+0530    no indexes to restore for collection employee.employees
2022-07-15T14:25:42.456+0530    3 document(s) restored successfully. 0 document(s) failed to restore.
```

Query 3:

```
C:\Users\Arun>mongodump --db=employee
2022-07-15T14:26:17.220+0530    writing employee.employees to dump\employee\employees.bson
2022-07-15T14:26:17.230+0530    done dumping employee.employees (3 documents)
```

Query 4:

```
C:\Users\Arun>mongorestore --db=employee dump/employee
2022-07-15T14:26:43.748+0530    The --db and --collection flags are deprecated for this use-case; please use --nsInclude
 instead, i.e. with --nsInclude=${DATABASE}.${COLLECTION}
2022-07-15T14:26:43.754+0530    building a list of collections to restore from dump\employee dir
2022-07-15T14:26:43.756+0530    reading metadata for employee.employees from dump\employee\employees.metadata.json
2022-07-15T14:26:43.785+0530    restoring employee.employees from dump\employee\employees.bson
2022-07-15T14:26:43.809+0530    finished restoring employee.employees (3 documents, 0 failures)
2022-07-15T14:26:43.809+0530    no indexes to restore for collection employee.employees
2022-07-15T14:26:43.810+0530    3 document(s) restored successfully. 0 document(s) failed to restore.
```

Query 5:

```
C:\Users\Arun>mongodump --db student --collection Details
2022-07-15T14:29:14.758+0530    writing student.Details to dump\student\Details.bson
2022-07-15T14:29:14.767+0530    done dumping student.Details (3 documents)
```

Query 6:

```
C:\Users\Arun>mongorestore --db student --collection Details dump/student/Details.bson
2022-07-15T14:30:16.288+0530    checking for collection data in dump\student\Details.bson
2022-07-15T14:30:16.295+0530    reading metadata for student.Details from dump\student\Details.metadata.json
2022-07-15T14:30:16.321+0530    restoring student.Details from dump\student\Details.bson
2022-07-15T14:30:16.392+0530    finished restoring student.Details (3 documents, 0 failures)
2022-07-15T14:30:16.392+0530    no indexes to restore for collection student.Details
2022-07-15T14:30:16.394+0530    3 document(s) restored successfully. 0 document(s) failed to restore.
```

Query 7:

```
C:\Users\Arun>mongostat
insert query update delete getmore command dirty used flushes vsize   res qrw arw net_in net_out conn                tim
e
    *0    *0     *0     *0       0     0|0  0.0% 0.0%       0 5.49G 41.0M 0|0 0|0   111b   52.4k   35 Jul 15 15:02:37.63
0
    *0    *0     *0     *0       0     1|0  0.0% 0.0%       0 5.49G 41.0M 0|0 0|0   242b   52.8k   35 Jul 15 15:02:38.63
4
    *0    *0     *0     *0       0     0|0  0.0% 0.0%       0 5.49G 41.0M 0|0 0|0   111b   52.5k   35 Jul 15 15:02:39.63
8
2022-07-15T15:02:39.882+0530    signal 'interrupt' received; forcefully terminating
```

RESULT

Query processed successfully and output obtained.

Experiment 21

AIM:

Create database college and collection student. Write Queries to implement Users and roles

CODE:

Query 1:

use student

Query 2:

db.createUser({user:"amal",pwd:"1234",roles:[{role:"readwrite",db:"student"}]})

Query 3:

show users

Query 4:

show roles

OUTPUT:

Query 1:

```
> use student
switched to db student
```

Query 2:

```
> db.createUser({user:"amal",pwd:"1234",roles:[{role:"readWrite",db:"student"}]})
Successfully added user: {
        "user" : "amal",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "student"
                }
        ]
}
```

Query 3:

```
> show users
{
        "_id" : "student.amal",
        "userId" : UUID("92245e73-dce5-4437-ae44-37f8882bbb99"),
        "user" : "amal",
        "db" : "student",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "student"
                }
        ],
        "mechanisms" : [
                "SCRAM-SHA-1",
                "SCRAM-SHA-256"
        ]
}
```

Query 4:

```
> show roles
{
        "role" : "enableSharding",
        "db" : "student",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "dbAdmin",
        "db" : "student",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "read",
        "db" : "student",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "readWrite",
        "db" : "student",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "userAdmin",
        "db" : "student",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "dbOwner",
        "db" : "student",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
```

RESULT

Query processed successfully and output obtained.

Experiment 22

AIM:

Create database college and collection student. Write Queries to implement Replication

CODE:

Step 1:

Create a folder named Data in the C Drive.
Create rs1,rs2,rs3 as sub directories to Data folder.
Open a cmd in the bin folder of the mongo.

Query 1:

```
>mongo --port 27018
>start mongod -replSet qwerty -logpath \Data\rs1\1.log -dbpath \Data\rs1 --port 27018
>start mongod -replSet qwerty -logpath \Data\rs2\2.log -dbpath \Data\rs2 --port 27019
>start mongod -replSet qwerty -logpath \Data\rs3\3.log -dbpath \Data\rs3 --port 27020
```

Step 2:

Open a mongoshell.

Query 2:

```
>config={_id:"qwerty",members:[{_id:0,host:"localhost:27018"},{_id:1,host:"localhost:27019"
},{_id:2,host:"localhost:27020"}]}
> rs.initiate(config)
> show dbs
```

Step 3:

Open another mongosh

Query 3:

```
> mongo --port 27019
> rs.secondaryOk()
>show dbs
```

OUTPUT:

Query 1:

```
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replSet qwerty -logpath \Data\rs1\1.log -dbpath \Data\rs1 --port 27018

C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replSet qwerty -logpath \Data\rs2\2.log -dbpath \Data\rs2 --port 27019

C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replSet qwerty -logpath \Data\rs3\3.log -dbpath \Data\rs3 --port 27020
```

Query 2:

```
--config={_id:"qwerty",members:[{_id:0,host:"localhost:27018"},{_id:1,host:"localhost:27019"},{_id:2,host:"localhost:27020"}]}
{ config={_id:"qwerty",members:[{_id:0,host:"localhost:27018"},{_id:1,host:"localhost:27019"},{_id:2,host:"localhost:27020"}]}
        "_id" : "qwerty",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27018"
                },
                {
                        "_id" : 1,
                        "host" : "localhost:27019"
                },
                {
                        "_id" : 2,
                        "host" : "localhost:27020"
                }
        ]
}
> rs.initiate(config)
{ "ok" : 1 }
qwerty:SECONDARY> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
qwerty:PRIMARY>
```

Query 3:

```
qwerty:SECONDARY> rs.secondaryOk()
```

```
qwerty:SECONDARY> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
qwerty:SECONDARY>
```

RESULT

Query processed successfully and output obtained.

Experiment 23

AIM:

Create a collection college and comment and do the following basic indexing operations
1)create collection college
2)insert data
3)Get all indexes
4)Create new index and show all indexes
5)Drop index and show all indexes
6)create collection comment and insert data
8)Create a text Index and sow all indexes
9)With the help of text index search on the Collection

CODE:

Query 1:

```
db.createCollection("college")
```

Query 2:

```
db.college.insert({RegNo:2101,Name:"Raichal",Mark:[{cs:95,maths:88,phy:75,chem:85,eng:91
}]})
db.college.insert({RegNo:2102,Name:"Abina",Mark:[{cs:99,maths:82,phy:76,chem:95,eng:91}
]})
db.college.insert({RegNo:2103,Name:"Riya",Mark:[{cs:100,maths:85,phy:71,chem:91,eng:94}
]})
```

Query 3:

```
db.college.getIndexes()
```

Query 4:

```
db.college.createIndex({RegNo:1})
db.college.getIndexes()
```

Query 5:

```
db.college.dropIndex({RegNo:1})
db.college.getIndexes()
```

Query 6:

```
db.createCollection("comment")
db.comment.insert({name:"Abina",post:"hello"})
db.comment.insert({name:"Anu",post:"hai"})
db.comment.insert({name:"Sona",post:"hola"})
db.comment.createIndex({post:"text"})
```

Query 7:

```
db.comment.createIndex({post:"text"})
db.comment.getIndexes()
```

Query 8:

```
db.comment.find({$text:{$search:"\"hai\""}})
```

OUTPUT:

Query 1:

```
> db.createCollection("college")
{ "ok" : 1 }
```

Query 2:

```
> db.college.insert({RegNo:2101,Name:"Raichal",Mark:[{cs:95,maths:88,phy:75,chem:85,eng:91}]})
WriteResult({ "nInserted" : 1 })
> db.college.insert({RegNo:2102,Name:"Abina",Mark:[{cs:99,maths:82,phy:76,chem:95,eng:91}]})
WriteResult({ "nInserted" : 1 })
> db.college.insert({RegNo:2103,Name:"Riya",Mark:[{cs:100,maths:85,phy:71,chem:91,eng:94}]})
WriteResult({ "nInserted" : 1 })
```

Query 3:

```
> db.college.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

Query 4:

```
> db.college.createIndex({RegNo:1})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

Query 5:

```
> db.college.dropIndex({RegNo:1})
{ "nIndexesWas" : 2, "ok" : 1 }

> db.college.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

Query 6:

```
> db.createCollection("comment")
{ "ok" : 1 }

> db.comment.insert({name:"Anu",post:"hai"})
WriteResult({ "nInserted" : 1 })

> db.comment.insert({name:"Abina",post:"hello"})
WriteResult({ "nInserted" : 1 })
> db.comment.insert({name:"Sona",post:"hola"})
WriteResult({ "nInserted" : 1 })
> db.comment.createIndex({post:"text"})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

Query 7:

```
> db.comment.createIndex({post:"text"})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}

> db.comment.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_"
        },
        {
                "v" : 2,
                "key" : {
                        "_fts" : "text",
                        "_ftsx" : 1
                },
                "name" : "post_text",
                "weights" : {
                        "post" : 1
                },
                "default_language" : "english",
                "language_override" : "language",
                "textIndexVersion" : 3
        }
]
```

Query 8:

```
> db.comment.find({$text:{$search:"\"hai\""}})
{ "_id" : ObjectId("62a0733f42cba70d8eac4e68"), "name" : "Anu", "post" : "hai" }
```

RESULT

Query processed successfully and output obtained.