**Huda Husain Petkar**

**M. Sc DS AI || L017 || ADBMS**

**Practical - 2**

USING (practical 1)

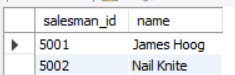
**1. Count the customers with grades above Bangalore’s average.**

(select count(customer\_id) from customer where grade > (Select avg(grade) from customer where city ='Bangalore'));



**2. Find the name and numbers of all salesmen who had more than one customer.**

select salesman\_id, name from salesman where salesman\_id In (select salesman\_id from customer group by salesman\_id having count(distinct(customer\_id)) >1 );



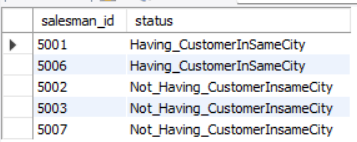
**3. List all salesmen and indicate those who have and don’t have customers in their cities**

**(Use UNION operation.)**

select salesman\_id, 'Having\_CustomerInSameCity' AS status from salesman where

salesman\_id in (select salesman\_id from customer where salesman.city = customer.city) union

select salesman\_id, 'Not\_Having\_CustomerInsameCity' As status from salesman where salesman\_id not in (select salesman\_id from customer where salesman.city = customer.city);



**4. Create a view that finds the salesman who has the customer with the highest order of a**

**day.**

create view v1 as

select salesman\_id from salesman

where salesman\_id in (Select salesman\_id from orrder where purch\_amt = (Select max(purch\_amt) from orrder where order\_date = '2016-10-05') );

select \* from v1;

create view view1 as

select salesman\_id from salesman

where salesman\_id in (Select salesman\_id from orrder where purch\_amt = (Select max(purch\_amt) from orrder where order\_date = order\_date) );

select \* from view1;





**5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders**

**must also be deleted**

Delete from salesman where salesman\_id = 1000;

2. Design ERD for the following schema and execute the following Queries on it:

Consider the schema for Movie Database:

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)

**Creation of Database:**

create database movie;

use movie;

**Creation of Tables:**

—--------------------- ACTOR ----------------------

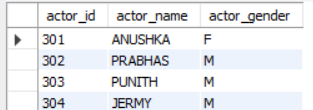
create table actor(actor\_id int, actor\_name varchar(25), actor\_gender varchar(2), primary key(actor\_id));

INSERT INTO ACTOR VALUES (301,'ANUSHKA','F'),

(302,'PRABHAS','M'),

(303,'PUNITH','M'),

(304,'JERMY','M');



---------------DIRECTOR-------------

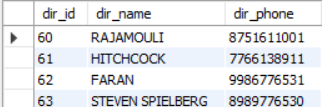
create table director(dir\_id int, dir\_name varchar(25), dir\_phone varchar(10), primary key(dir\_id));

INSERT INTO DIRECTOR VALUES (60,'RAJAMOULI','8751611001'),

(61,'HITCHCOCK','7766138911'),

(62,'FARAN','9986776531'),

(63,'STEVEN SPIELBERG','8989776530');



-----------------MOVIES ------------------------------

create table movies(mov\_id int, mov\_title varchar(30),mov\_year int, mov\_lang varchar(15), dir\_id int,

primary key(mov\_id), foreign key(dir\_id) references director(dir\_id));

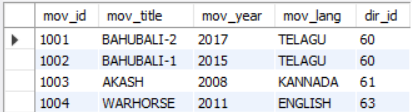
INSERT INTO MOVIES VALUES (1001,'BAHUBALI-2', 2017, 'TELAGU', 60);

INSERT INTO MOVIES VALUES (1002,'BAHUBALI-1', 2015, 'TELAGU', 60);

INSERT INTO MOVIES VALUES (1003,'AKASH', 2008, 'KANNADA', 61);

INSERT INTO MOVIES VALUES (1004,'WARHORSE', 2011, 'ENGLISH', 63);

select \* from movies;



-------------------------MOVIE CAST ----------------------

create table movie\_cast(act\_id int, mov\_id int, role varchar(15),

foreign key (act\_id) references actor(actor\_id),foreign key (mov\_id) references movies(mov\_id), primary key(act\_id, mov\_id) );

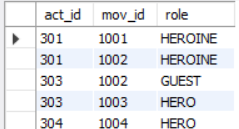
INSERT INTO MOVIE\_CAST VALUES (301, 1002, 'HEROINE');

INSERT INTO MOVIE\_CAST VALUES (301, 1001, 'HEROINE');

INSERT INTO MOVIE\_CAST VALUES (303, 1003, 'HERO');

INSERT INTO MOVIE\_CAST VALUES (303, 1002, 'GUEST');

INSERT INTO MOVIE\_CAST VALUES (304, 1004, 'HERO');



----------------------------- RATING -------------------

create table rating(mov\_id int, rev\_stars int, primary key(mov\_id),

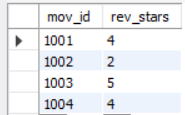
foreign key(mov\_id)references movies (mov\_id));

INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);



Write SQL queries to

1. List the titles of all movies directed by ‘Hitchcock’.

SELECT M.Mov\_Title

FROM MOVIES M

JOIN DIRECTOR D ON M.Dir\_id = D.Dir\_id

WHERE D.Dir\_Name = 'Hitchcock';

2. Find the movie names where one or more actors acted in two or more movies.

SELECT DISTINCT M.Mov\_Title

FROM MOVIES M

JOIN MOVIE\_CAST MC ON M.Mov\_id = MC.Mov\_id

WHERE MC.Act\_id IN (

SELECT Act\_id

FROM MOVIE\_CAST

GROUP BY Act\_id

HAVING COUNT(DISTINCT Mov\_id) >= 2

);

3. List all actors who acted in a movie before 2000 and also in a movie after

2015 (use JOIN operation).

SELECT DISTINCT A.Act\_Name

FROM ACTOR A

JOIN MOVIE\_CAST MC1 ON A.Act\_id = MC1.Act\_id

JOIN MOVIES M1 ON MC1.Mov\_id = M1.Mov\_id

JOIN MOVIE\_CAST MC2 ON A.Act\_id = MC2.Act\_id

JOIN MOVIES M2 ON MC2.Mov\_id = M2.Mov\_id

WHERE M1.Mov\_Year < 2000 AND M2.Mov\_Year > 2015;

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.

SELECT M.Mov\_Title, MAX(R.Rev\_Stars) AS Highest\_Stars

FROM MOVIES M

JOIN RATING R ON M.Mov\_id = R.Mov\_id

GROUP BY M.Mov\_Title

ORDER BY M.Mov\_Title;

5. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

UPDATE RATING

SET Rev\_Stars = 5

WHERE Mov\_id IN (

SELECT M.Mov\_id

FROM MOVIES M

JOIN DIRECTOR D ON M.Dir\_id = D.Dir\_id

WHERE D.Dir\_Name = 'Steven Spielberg'

);

3. Design ERD for the following schema and execute the following Queries on it: ANY 10

Students

create table students(stno int primary key, name varchar(30), addr varchar(30), city varchar(15), state varchar(10), zip varchar(10));

INSERT INTO STUDENTS VALUES

(1011, 'Edwards P. David', '10 Red Rd.', 'Newton', 'MA', '02159'),

(2415, 'Grogan A. Mary', '8 Walnut St.', 'Malden', 'MA', '02148'),

(2861, 'Mixon Leatha', '100 School St.', 'Brookline', 'MA', '02146'),

(2890, 'McLane Sandy', '30 Case Rd.', 'Boston', 'MA', '02122'),

(3442, 'Novak Roland', '42 Beacon St.', 'Nashua', 'NH', '03060'),

(3566, 'Pierce Richard', '70 Park St.', 'Brookline', 'MA', '02146'),

(4022, 'Prior Lorraine', '8 Beacon St.', 'Boston', 'MA', '02125'),

(5544, 'Rawlings Jerry', '15 Pleasant Dr.', 'Boston', 'MA', '02115'),

(5571, 'Lewis Jerry', '1 Main Rd.', 'Providence', 'RI', '02904');



create table instructors(empno int primary key, name varchar(30), rankk varchar(30), roomno int, telno int);

INSERT INTO INSTRUCTORS VALUES

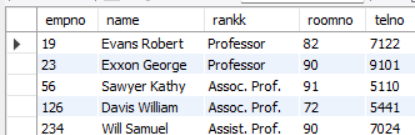
(19, 'Evans Robert', 'Professor', 82, 7122),

(23, 'Exxon George', 'Professor', 90, 9101),

(56, 'Sawyer Kathy', 'Assoc. Prof.', 91, 5110),

(126, 'Davis William', 'Assoc. Prof.', 72, 5441),

(234, 'Will Samuel', 'Assist. Prof.', 90, 7024);



Course

create table course(cno varchar(10) primary key, cname varchar(40), cr int, cap int);

INSERT INTO COURSE VALUES

('cs110', 'Introduction to Computing', 4, 120),

('cs210', 'Computer Programming', 4, 100),

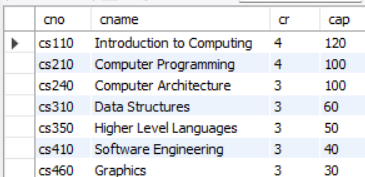
('cs240', 'Computer Architecture', 3, 100),

('cs310', 'Data Structures', 3, 60),

('cs350', 'Higher Level Languages', 3, 50),

('cs410', 'Software Engineering', 3, 40),

('cs460', 'Graphics', 3, 30);



INSERT INTO GRADES (stno, empno, cno, sem, year, grade)

VALUES

(1011, 19, 'cs110', 'Fall', 2001, 40),

(2661, 19, 'cs110', 'Fall', 2001, 80),

(3566, 19, 'cs110', 'Fall', 2001, 95),

(5544, 19, 'cs110', 'Fall', 2001, 100),

(1011, 23, 'cs110', 'Spring', 2002, 75),

(4022, 23, 'cs110', 'Spring', 2002, 60),

(3566, 19, 'cs240', 'Spring', 2002, 100),

(5571, 19, 'cs240', 'Spring', 2002, 50),

(2415, 19, 'cs240', 'Spring', 2002, 100),

(3442, 234, 'cs240', 'Spring', 2002, 60),

(5571, 234, 'cs240', 'Spring', 2002, 80),

(1011, 19, 'cs210', 'Fall', 2002, 90),

(2661, 19, 'cs210', 'Fall', 2002, 70),

(3566, 19, 'cs210', 'Fall', 2002, 90),

(5571, 19, 'cs210', 'Spring', 2003, 85),

(4022, 19, 'cs210', 'Spring', 2003, 70),

(5544, 56, 'cs240', 'Spring', 2003, 70),

(1011, 56, 'cs240', 'Spring', 2003, 90),

(4022, 56, 'cs240', 'Spring', 2003, 80),

(2661, 234, 'cs310', 'Spring', 2003, 100),

(4022, 234, 'cs310', 'Spring', 2003, 75);

INSERT INTO ADVISING (stno, empno)

VALUES

(1011, 19),

(2415, 19),

(2661, 23),

(2890, 23),

(3442, 56),

(3566, 126),

(4022, 234),

(5544, 23),

(5571, 234);

1. Find the names of students who took some four-credit courses.

SELECT DISTINCT S.name

FROM STUDENTS S

JOIN GRADES G ON S.stno = G.stno

JOIN COURSES C ON G.cno = C.cno

WHERE C.cr = 4;

2. Find the names of students who took every four-credit course.

SELECT S.name

FROM STUDENTS S

WHERE NOT EXISTS (

SELECT C.cno

FROM COURSES C

WHERE C.cr = 4

AND NOT EXISTS (

SELECT G.stno

FROM GRADES G

WHERE G.stno = S.stno AND G.cno = C.cno

)

);

3. Find the names of students who took a course with an instructor who is also their advisor.

SELECT DISTINCT S.name

FROM STUDENTS S

JOIN ADVISING A ON S.stno = A.stno

JOIN GRADES G ON S.stno = G.stno

JOIN COURSES C ON G.cno = C.cno

JOIN INSTRUCTORS I ON G.empno = I.empno

WHERE A.empno = G.empno;

4. Find the names of students who took cs210 and cs310.

SELECT S.name

FROM STUDENTS S

WHERE S.stno IN (

SELECT G.stno

FROM GRADES G

WHERE G.cno = 'cs210'

)

AND S.stno IN (

SELECT G.stno

FROM GRADES G

WHERE G.cno = 'cs310'

);

5. Find the names of all students whose advisor is not a full professor.

SELECT DISTINCT S.name

FROM STUDENTS S

JOIN ADVISING A ON S.stno = A.stno

JOIN INSTRUCTORS I ON A.empno = I.empno

WHERE I.rank <> 'Professor';

6. Find instructors who taught students who are advised by another instructor who shares the same room.

SELECT DISTINCT I1.name

FROM INSTRUCTORS I1

JOIN GRADES G ON I1.empno = G.empno

JOIN ADVISING A ON G.stno = A.stno

JOIN INSTRUCTORS I2 ON A.empno = I2.empno

WHERE I1.roomno = I2.roomno AND I1.empno <> I2.empno;

7. Find course numbers for courses that enroll exactly two students;

SELECT G.cno

FROM GRADES G

GROUP BY G.cno

HAVING COUNT(DISTINCT G.stno) = 2;

8. Find the names of all students for whom no other student lives in the same city.

SELECT S1.name

FROM STUDENTS S1

WHERE NOT EXISTS (

SELECT S2.stno

FROM STUDENTS S2

WHERE S1.city = S2.city AND S1.stno <> S2.stno

);

9. Find course numbers of courses taken by students who live in Boston and which are taught by an associate professor.

SELECT DISTINCT G.cno

FROM GRADES G

JOIN STUDENTS S ON G.stno = S.stno

JOIN INSTRUCTORS I ON G.empno = I.empno

WHERE S.city = 'Boston' AND I.rank = 'Assoc. Prof.';

10. Find the telephone numbers of instructors who teach a course taken by any student who lives in Boston.

SELECT DISTINCT I.telno

FROM INSTRUCTORS I

JOIN GRADES G ON I.empno = G.empno

JOIN STUDENTS S ON G.stno = S.stno

WHERE S.city = 'Boston';