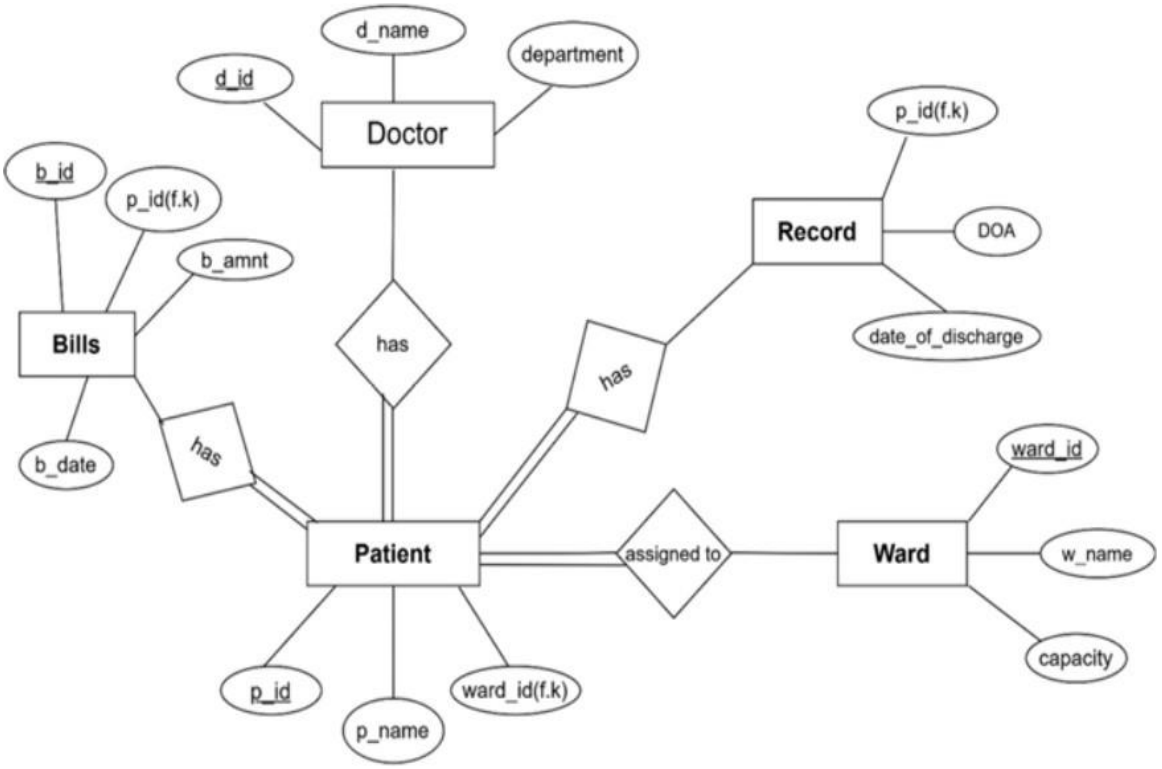
WEEK 1

• Create a Scenario based ER-Models with the entities. (Hospital Details like: Wards, Patients, Doctor, Bills etc)



• Convert this ER-model into table with all the entities. (Minimum five Entities).

CREATE TABLE Doctor(d_id int PRIMARY KEY, d_name varchar(25), dep varchar(12));

CREATE TABLE Warda(ward_id int PRIMARY KEY, w_name varchar(20), capacity int);

CREATE TABLE Patient(p_id int PRIMARY KEY, p_name varchar(20), ward_id int, FOREIGN KEY (ward_id) References Ward (ward_id));

CREATE TABLE Record(p_id int, doa Date, date_of_discharge Date, FOREIGN KEY (p_id) References Patient(p_id));

CREATE TABLE Bills(b_id int PRIMARY KEY, p_id int, b_amnt int, b_date Date, FOREIGN KEY (p_id) References Patient(p_id));

• Insert random data in each column of all the tables.

INSERT INTO Doctor VALUES (102, 'Dr Asta', 'cardiology');

INSERT INTO Ward VALUES (202, 'General', 17);

INSERT INTO Record VALUES (1, '01-SEP-2024', '06-SEP-2024');

INSERT INTO Bills VALUES (501, 1, 5600, '06-SEP-2024');

• **Update the table by applying some conditions.(For example: using alter command)**

ALTER TABLE Patient ADD Gender varchar(10);

UPDATE Patient SET Gender = 'Male' WHERE p_id= 1;

• **Apply the `DELETE` and `DROP` command, and then review the results.**

DELETE FROM Record WHERE doa = '01-SEP-2024';

DROP TABLE RECORD;

OUTPUT:

```
SQL> CREATE TABLE Doctor(d_id int PRIMARY KEY, d_name varchar(25), department varchar(12));
Table created.
SQL> CREATE TABLE Ward(ward_id int PRIMARY KEY, w_name varchar(12), capacity int);
Table created.
SQL>  CREATE TABLE Patient(p_id int PRIMARY KEY, p_name varchar(20), ward_id int, FOREIGN KEY(ward_id) References Ward(ward_id));
Table created.
SQL> CREATE TABLE Record(p_id int, doa Date, date_of_discharge Date, FOREIGN KEY(p_id) References Patient(p_id));
Table created.
SQL> CREATE TABLE Bills(b_id int PRIMARY KEY, p_id int, b_amnt int, b_date Date, FOREIGN KEY(p_id) References Patient(p_id));
Table created.
```

```
SQL> INSERT INTO Doctor VALUES (102, 'Dr Subhadra', 'cardiology');
1 row created.
SQL> INSERT INTO Ward VALUES (202, 'General', 16);
1 row created.
SQL> INSERT INTO Patient VALUES (1, 'Saksham', 202);
1 row created.
SQL> INSERT INTO Record VALUES(1, '01-SEP-2024', '06-SEP-2024');
1 row created.
SQL> INSERT INTO Bills VALUES (501, 1, 5600, '06-SEP-2024');
1 row created.
```

```
SQL> ALTER TABLE Patient ADD Gender varchar(10);
Table altered.
SQL> UPDATE Patient SET Gender = 'Male' WHERE p_id= 1;
1 row updated.
```

```
SQL> DELETE FROM Record WHERE doa = '01-SEP-2024';
1 row deleted.
SQL> DROP TABLE RECORD;
Table dropped.
```

WEEK 2

• Create a user and provide the GRANT privileges to the user on the database then REVOKE the given privileges.

CREATE USER Anshul_14_L1 IDENTIFIED BY 1234;

GRANT ALL PRIVILEGES TO Anshul_14_L1;

REVOKE ALL PRIVILEGES FROM Anshul_14_L1;

• Insert any five records in the previous schema and apply the rollback. Also check the results.

INSERT ALL INTO Doctor VALUES (103, 'Dr Narendra', 'Orthoplogy')

INTO Doctor VALUES (104, 'Dr Rakshit', 'Neurology') INTO Doctor VALUES (105, 'Dr Nitin', 'Gyne') SELECT * FROM dual;

• Add default, check, unique and not null constraints to the schema.

ALTER TABLE DOCTOR MODIFY department VARCHAR2(20) DEFAULT ' UNKNOWN';

ALTER TABLE PATIENT ADD CONSTRAINT CHK2 CHECK (ward_id>200); ALTER TABLE PATIENT ADD CONSTRAINT UNIQUE_ UNIQUE(p_name); ALTER TABLE DOCTOR MODIFY department VARCHAR2(20) NOT NULL;

• Insert NULL values and check the results.

insert into doctor values (103, NULL, 'CSE');

select * from doctor;

• Add duplicate value and try to make a column as primary key, check what happen to the table.

insert into doctor values (106, 'taniya', 'ENT');

insert into doctor values (106, 'vaish', 'Cardiologist');

OUTPUT:

```
SQL> CREATE USER random IDENTIFIED BY 1111;

User created.

SQL> GRANT ALL PRIVILEGES TO random;

Grant succeeded.

SQL> REVOKE ALL PRIVILEGES FROM random;

Revoke succeeded.
```

```
SQL> INSERT ALL
  2  INTO Doctor VALUES (103, 'Dr Narendra', 'Orthoplogy')
  3  INTO Doctor VALUES (104, 'Dr Rakshit', 'Neurology')
  4  INTO Doctor VALUES (105, 'Dr Nitin', 'Gyne')
  5  SELECT * FROM dual;

3 rows created.

SQL> SELECT * FROM Doctor;

      D_ID D_NAME                    DEPARTMENT
---------- ------------------------- ------------
       102 Dr Subhadra               cardiology
       103 Dr Narendra               Orthoplogy
       104 Dr Rakshit                Neurology
       105 Dr Nitin                  Gyne

SQL> Rollback;

Rollback complete.

SQL> SELECT * FROM Doctor;

no rows selected
```

```
SQL> ALTER TABLE DOCTOR MODIFY department VARCHAR2(20) DEFAULT 'UNKNOWN';

Table altered.

SQL> ALTER TABLE PATIENT ADD CONSTRAINT CHK2 CHECK (ward_id>200);

Table altered.

SQL> ALTER TABLE PATIENT ADD CONSTRAINT UNIQUE_ UNIQUE(p_name);

Table altered.

SQL> ALTER TABLE DOCTOR MODIFY department VARCHAR2(20) NOT NULL;

Table altered.
```

```
SQL> insert into doctor values(103, NULL, 'CSE');

1 row created.

SQL> select * from doctor;

      D_ID D_NAME                    DEPARTMENT
---------- ------------------------- --------------------
       103                           CSE
SQL> show user
USER is "ANSHUL_14_L1"
SQL> insert into doctor values (106, 'taniya', 'ENT');

1 row created.

SQL> insert into doctor values (106, 'vaish', 'Cardiologist');
insert into doctor values (106, 'vaish', 'Cardiologist')
*
```

WEEK 3

• Create an Employee table with the following attributes and constraints: Employee Table - (Employee Id. (Primary key), Name, Department, Age (check >18), Salary, City).

create table employee(Emp_id int primary key, Name varchar2(20), Department varchar2(20), Age int CHECK(Age>18), Salary int, City Varchar(10));

insert all into employee values (101, 'radhika', 'Finance', 22, 23000, 'dehradun') into employee values (102, 'mansi', 'medical', 23, 45000, 'pune')

into employee values (103, 'manvi', 'finance', 34, 56000, 'pune')

into employee values (104, 'princy', 'engineering', 22, 70000, 'dehradun')

into employee values (105, 'anshul', 'finance', 27, 70000, 'pune')

select * from dual;

• Display the total number of employees.

select count(*) from employee;

• Retrieve all information of employees whose age is 22.

select * from employee where age =22;

• Fetch the employee id, name, and department, whose salary >= 50000.

select emp_id, name, department from employee where salary >=50000;

• Print the name of the employees and label the column as "Full Name" for those employees whose department name is 'Finance' and age is 22.

select name as "Full Name" from employee where department = 'Finance and age 22;

• Print the department names from the employee table without having the duplicates.

select distinct department from employee;

OUTPUT:

```
SQL> create table employee(Emp_id int primary key, Name varchar2(20), Department varchar2(20),
  2  Age int CHECK(Age>18), Salary int, City Varchar(10));

Table created.

SQL> insert all into employee values (101, 'radhika', 'Finance', 22, 23000, 'dehradun')
  2  into employee values (102, 'mansi', 'medical', 23, 45000, 'pune')
  3  into employee values (103, 'manvi', 'finance', 34, 56000, 'pune')
  4  into employee values (104, 'princy', 'engineering', 22, 70000, 'dehradun')
  5  into employee values (105, 'anshul', 'finance', 27, 70000, 'pune')
  6  select * from dual;

5 rows created.
```

```
SQL> select * from employee;

    EMP_ID NAME            DEPARTMENT                AGE     SALARY CITY
---------- --------------- -------------------- ---------- ---------- ----------
       101 radhika         Finance                    22      23000 dehradun
       102 mansi           medical                    23      45000 pune
       103 manvi           finance                    34      56000 pune
       104 princy          engineering                22      70000 dehradun
       105 anshul          finance                    27      70000 pune

SQL> select count(*) from employee;

  COUNT(*)
----------
         5
```

```
SQL> select * from employee where age =22;

    EMP_ID NAME            DEPARTMENT                AGE     SALARY CITY
---------- --------------- -------------------- ---------- ---------- ----------
       101 radhika         Finance                    22      23000 dehradun
       104 princy          engineering                22      70000 dehradun
```

```
SQL> select emp_id, name, department from employee where salary >=50000;

    EMP_ID NAME            DEPARTMENT
---------- --------------- --------------------
       103 manvi           finance
       104 princy          engineering
       105 anshul          finance
```

```
SQL> select name as "Full Name" from employee where department = 'Finance' and age= 22;

Full Name
---------------
radhika
```

```
SQL> select distinct department from employee;

DEPARTMENT
--------------------
medical
finance
engineering
Finance
```

WEEK 4

• **Find out the maximum and minimum salary from the employee table.**

select max(salary), min (salary) from employee;

• **Show the total salary and average salary of all the employees**.

select sum(salary) as totalSalary, avg (salary) as AvgSalary from employee;

• **Show all the details of the employees who have the same salary.**

select el.* from employee el, employee e2 where el.salary = e2.salaryand el.emp_id <> e2.emp_id;

• **Display the employees name from lowest salary to the highest salary.**

select name from employee order by salary;

• **Display the employee name and salary (department-wise) for employees, whose salary is greater than or equal to 10,000 and age is greater than** 25.

select name, salary from employee where salary >=10000 and age>25 ORDER BY department;

OUTPUT:

```
SQL> select max(salary), min (salary) from employee;

MAX(SALARY) MIN(SALARY)
----------- -----------
      70000       23000
```

```
SQL> select sum(salary) as totalSalary, avg (salary) as AvgSalary from employee;

TOTALSALARY  AVGSALARY
----------- ----------
     264000      52800
```

```
SQL> select el.* from employee el, employee e2 where el.salary = e2.salary
  2  and el.emp_id <> e2.emp_id;

    EMP_ID NAME            DEPARTMENT                 AGE     SALARY CITY
---------- --------------- ------------------- ---------- ---------- ----------
       105 anshul          finance                    27      70000 pune
       104 princy          engineering                22      70000 dehradun
```

```
SQL> select name from employee order by salary;

NAME
---------------
radhika
mansi
manvi
princy
anshul
```

```
SQL> select name, salary from employee where salary >=10000 and age>25
  2  ORDER BY department;

NAME               SALARY
-------------- ----------
anshul              70000
manvi               56000
```

WEEK 5

• Fetch the information of employees who belong to the city "Delhi" or "Pune."

select * from employee where city = ‘delhi’ or city = ‘pune’;

• Print the name and department of employees whose ID is in the range from 102 to 106.

select name, department from employee where emp_id BETWEEN 102 and 106;

• Show the details of employees who belong to the same city (use the IN operator).

select * from employee where city in (select el.city from employee el, employee e2 where el.city =e2.city and el.emp_id<>e2.emp_id);

• Check whether the all employee is belongs to the same city or not. (use ALL operator).

select case when el.city= all(select e2.city from employee e2) then 'employees belong to same city' else 'employees does not belong to same city' end as result from employee el where rownum =1;

• Check whether the all employee is belongs to the same city or not. (use ANY operator).

select case when el.city <> any(select e2.city from employee e2) then 'employees does not belong to same city' else’ employees belong to same city' end as result from emmployee el where rownum =1;

• Check whether the all employee is belongs to the same city or not. (use Exists operator).

select case when exists(select city from employee e1 where e1.city<>e2.city) then 'employees does not belong to same city' else 'employees belong to same city' end as result from employee e2 where rownum=1;

OUTPUT:

```
SQL> select * from employee where city = 'delhi' or city = 'pune';

    EMP_ID NAME            DEPARTMENT                AGE     SALARY CITY
---------- --------------- ------------------- ---------- ---------- ----------
       102 mansi           medical                    23      45000 pune
       103 manvi           finance                    34      56000 pune
       105 anshul          finance                    27      70000 pune
```

```
SQL> select name, department from employee where emp_id BETWEEN 102 and 106;

NAME            DEPARTMENT
--------------- --------------------
mansi           medical
manvi           finance
princy          engineering
anshul          finance
```

```
SQL> select * from employee where city in (select el.city from employee el, employee e2
  2  where el.city =e2.city and el.emp_id<>e2.emp_id);

    EMP_ID NAME            DEPARTMENT                AGE     SALARY CITY
---------- --------------- ------------------- ---------- ---------- ----------
       101 radhika         Finance                    22      23000 dehradun
       102 mansi           medical                    23      45000 pune
       103 manvi           finance                    34      56000 pune
       104 princy          engineering                22      70000 dehradun
       105 anshul          finance                    27      70000 pune
```

```
SQL> select case when el.city= all(select e2.city from employee e2)
  2  then 'employees belong to same city' else
  3  'employees does not belong to same city'
  4  end as result from employee el where rownum =1;

RESULT
--------------------------------------
employees does not belong to same city
```

```
SQL> select case when el.city <> any(select e2.city from employee e2)
  2  then 'employees does not belong to same city' else
  3  'employees belong to same city' end as result from
  4  employee el where rownum =1;

RESULT
--------------------------------------
employees does not belong to same city
```

```
SQL> select case when exists(select city from employee e1 where
  2  e1.city<>e2.city) then 'employees does not belong to same city'
  3  else 'employees belong to same city' end as result
  4  from employee e2 where rownum=1;

RESULT
--------------------------------------
employees does not belong to same city
```

WEEK 6

• **Show the record of employees who are working in the 'CSE' department.**

select * from employee where department like ‘CSE’

• **Fetch the names of employees whose names start with the letters 'ay'.**

select name from employee where name like ‘ay%’;

• **Fetch the information of employees, including their names and departments, whose names end with the letters 'sh'.**

select name,department from employee where name like ‘%sh’;

• **Display the employee names and their departments of employees, whose city name starts with 'D' or ends with 'h'.**

select name, department from employee where city like ‘d%’ or city like ‘%h’;

• **Print all records of employees whose salary is greater than 15,000 and whose name starts with 'h'.**

select * from employee where salary>=15000 and name like ‘h%’;

• **Print the names of employees whose names consist of exactly three letters**.

select name from employee where name like ‘___’;

• **Print the names of employees along with their city for those whose names have at least five letters.**

select name, city from employee where name like ‘_____’;

OUTPUT:

```
SQL> select * from employee where department like 'CSE';

    EMP_ID NAME            DEPARTMENT                 AGE     SALARY CITY
---------- --------------- ------------------- ---------- ---------- ----------
       108 harsh           CSE                         23      34000 pune
       109 jai             CSE                         33      23000 delhi
```

```
SQL> select name from employee where name like 'ay%';

NAME
---------------
ayush
```

```
SQL> select name, department from employee where name like '%sh';

NAME            DEPARTMENT
--------------- --------------------
harsh           CSE
ayush           finance
```

```
SQL> select name, department from employee where city like 'd%' OR
  2  city like '%h';

NAME            DEPARTMENT
--------------- --------------------
radhika         Finance
princy          engineering
jai             CSE
ayush           finance
```

```
SQL> select * from employee where salary>=15000 and name like 'h%';

    EMP_ID NAME            DEPARTMENT                 AGE     SALARY CITY
---------- --------------- ------------------- ---------- ---------- ----------
       108 harsh           CSE                         23      34000 pune
```

```
SQL> select name from employee where name like '___';

NAME
---------------
jai
```

```
SQL> select name, city from employee where name like '%_____%';

NAME            CITY
--------------- ----------
radhika         dehradun
mansi           pune
manvi           pune
princy          dehradun
anshul          pune
harsh           pune
ayush           delhi

7 rows selected.
```

WEEK 7

• Create two tables named as employee and department with the given constraints and attributes: Employee table - (Employee Id.(Primary key), Department ID, Name, Age (check >18), Salary, City) Department table - (Department Id, and Department name)

create table Department(dept_id number primary key, dept_name varchar(30));

create table new_employee(emp_id number primary key, dept_id number, emp_name varchar(20), age number check(age>18), salary number, city varchar(20), FOREIGN KEY(dept_id) References Department(dept_id));

• Display the details of employees along with their corresponding department names.

select e1.*, d1.dept_name from new_employee e1 JOIN department d1 ON e1.dept_id = d1.dept_id;

• Print the names of employees who are not assigned to any department.

select emp_name from new_employee where dept_id is null;

• Print the employee names and department names for employees whose salary is greater than 25,000. (Using left join).

select emp_name, dept_name from new_employee e left join

department d on e.dept_id = d.dept_id where salary>=25000;

• Display the names of employees along with their department names for those who are not assigned to any department.

select emp_name, dept_name from new_employee e inner join

department d on e.dept_id =d.dept_id where age>30;

• Print the employee names and their corresponding department names for employees with a salary greater than 25,000. (Using right join).

select emp_name, dept_name from department d right join

new_employee e on e.dept_id = d.dept_id where salary>=25000;

• Display the names of departments along with the names of employees who are older than 30 years.

select emp_name, dept_name from new_employee e inner join

department d on e.dept_id =d.dept_id where age>30;

```
SQL> create table Department (dept_id number primary key, dept_name varchar(30));

Table created.

SQL> create table new_employee(emp_id number primary key, dept_id number, emp_name varchar(20),
  2  age number CHECK(age>18), salary number, city varchar(20), FOREIGN KEY (dept_id)
  3  References Department(dept_id));

Table created.
```

```
SQL> insert all into Department values (101, 'finance')
  2  into Department values(102,'cse')
  3  into Department values (103, 'accounting')
  4  into Department values(104, 'cse')
  5  into Department values (105, 'finance')
  6  into Department values(106, 'hr')
  7  select * from DUAL;

6 rows created.
```

```
SQL> insert all into new_employee values (1001, 101, 'vaish', 34,45000, 'pune')
  2  into new_employee values (1002, NULL, 'mansi', 35, 30000, 'pune')
  3  into new_employee values (1003, 102, 'harsh', 34, 35990, 'dehradun')
  4  into new_employee values (1004,103, 'princy', 30, 34989, 'dehradun')
  5  into new_employee values (1005, NULL, 'gaurav', 22, 34000, 'pune')
  6  into new_employee values (1006, 104, 'kshitij', 35, 20000,'jammu')
  7  into new_employee values (1007,105, 'jiya', 34, 36666, 'jammu')
  8  select * from dual;

7 rows created.
```

```
SQL> select e1.*, d1.dept_name from new_employee e1 JOIN
  2  department d1 ON e1.dept_id = d1.dept_id;

    EMP_ID    DEPT_ID EMP_NAME                   AGE     SALARY CITY                 DEPT_NAME
---------- ---------- -------------------- ---------- ---------- -------------------- ----------
      1001        101 vaish                        34      45000 pune                 finance
      1003        102 harsh                        34      35990 dehradun             cse
      1004        103 princy                       30      34989 dehradun             accounting
      1006        104 kshitij                      35      20000 jammu                cse
      1007        105 jiya                         34      36666 jammu                finance
```

```
SQL> select emp_name from new_employee where dept_id is null;

EMP_NAME
--------------------
mansi
gaurav
```

```
SQL> select emp_name, dept_name from new_employee e left join
  2  department d on e.dept_id = d.dept_id where salary>=25000;

EMP_NAME             DEPT_NAME
-------------------- ------------------------------
vaish                finance
mansi
harsh                cse
princy               accounting
gaurav
jiya                 finance

6 rows selected.
```

```
SQL> select emp_name from new_employee where dept_id is null;

EMP_NAME
--------------------
mansi
gaurav
```

```
SQL> select emp_name, dept_name from department d right join
  2  new_employee e on e.dept_id = d.dept_id where salary>=25000;

EMP_NAME             DEPT_NAME
-------------------- ------------------------------
vaish                finance
mansi
harsh                cse
princy               accounting
gaurav
jiya                 finance

6 rows selected.
```

```
SQL> select emp_name, dept_name from new_employee e inner join
  2  department d on e.dept_id =d.dept_id where age>30;

EMP_NAME             DEPT_NAME
-------------------- ------------------------------
vaish                finance
harsh                cse
kshitij              cse
jiya                 finance
```

WEEK 8

• Create the table to keep track of customer records and their order. Customer table - (Name as Not null, Customer_id as primary key, Age, Address) Order table - (Customer_id, order_id, date).

CREATE TABLE Customer (Customer_ID INT PRIMARY KEY, Name VARCHAR(50) NOT NULL, Age INT, Address VARCHAR(100));

CREATE TABLE Orderas (Order_ID INT PRIMARY KEY, Customer_ID INT, Dates date, FOREIGN KEY (Customer_ID) REFERENCES Customer (Customer_ID));

• Apply the full join and the full outer join to the schema and review the results.

SELECT Customer.Customer_ID, Customer. Name, Customer.Age,
Customer. Address, Orderas. Order_ID, Orderas. Dates FROM Customer
FULL OUTER JOIN Orderas ON
Customer.Customer_ID = Orderas.Customer_ID;

• Display the name of the city as "destination" for customers who have placed orders.

SELECT Customer. Name, Customer. Address AS Destination
FROM Customer INNER JOIN Orderas ON
Customer. Customer_ID = Orderas.Customer_ID;

• Apply the cross join and check the results.

SELECT Customer.Customer_ID, Customer. Name, Orderas. Order_ID,
Orderas. Dates FROM Customer CROSS JOIN Orderas;

• Display the customer names and order IDs for customers who have placed orders from the same city.

SELECT c1. Name AS Customer_Name, 01.Order_ID, c1.Address AS City
FROM Customer c1 INNER JOIN Orderas 01 ON
c1.Customer_ID = 01. Customer_ID INNER JOIN Customer
c2 ON c1.Address = c2. Address AND c1.Customer_ID <>
c2.Customer_ID;

OUTPUT:

```
SQL> CREATE TABLE Customer (Customer_ID INT PRIMARY KEY,Name VARCHAR(50) NOT NULL,
  2  Age INT,Address VARCHAR(100));

Table created.

SQL> CREATE TABLE Orderas (Order_ID INT PRIMARY KEY, Customer_ID INT, Dates date,
  2   FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID));

Table created.
```

```
SQL> SELECT Customer.Customer_ID, Customer.Name, Customer.Age,
  2  Customer.Address, Orderas.Order_ID, Orderas.Dates FROM Customer
  3  FULL OUTER JOIN Orderas ON Customer.Customer_ID = Orderas.Customer_ID;

CUSTOMER_ID NAME                 AGE ADDRESS           ORDER_ID DATES
----------- -------------- ---------- -------------- ---------- ---------
        101 anshul                 18 clementtown           201 24-JUL-24
        102 radhika                19 marina                202 27-AUG-24
        103 tiya                   20 marina                203 22-JUL-24
```

```
SQL> SELECT Customer.Name, Customer.Address AS Destination
  2  FROM Customer INNER JOIN Orderas ON
  3  Customer.Customer_ID = Orderas.Customer_ID;

NAME            DESTINATION
--------------- ---------------
anshul          clementtown
radhika         marina
tiya            marina
```

```
SQL> SELECT Customer.Customer_ID, Customer.Name, Orderas.Order_ID,
  2  Orderas.Dates FROM Customer CROSS JOIN Orderas;

CUSTOMER_ID NAME              ORDER_ID DATES
----------- --------------- ---------- ---------
        101 anshul                 201 24-JUL-24
        102 radhika                201 24-JUL-24
        103 tiya                   201 24-JUL-24
        101 anshul                 202 27-AUG-24
        102 radhika                202 27-AUG-24
        103 tiya                   202 27-AUG-24
        101 anshul                 203 22-JUL-24
        102 radhika                203 22-JUL-24
        103 tiya                   203 22-JUL-24

9 rows selected.
```

```
SQL> SELECT c1.Name AS Customer_Name, o1.Order_ID, c1.Address AS City
  2  FROM Customer c1 INNER JOIN Orderas o1 ON
  3  c1.Customer_ID = o1.Customer_ID INNER JOIN Customer
  4  c2 ON c1.Address = c2.Address AND c1.Customer_ID <> c2.Customer_ID;

CUSTOMER_NAME     ORDER_ID CITY
--------------- ---------- ---------------
tiya                   203 marina
radhika                202 marina
```

WEEK 9

• Create the Student table, Register table and Program table. Student table - (Roll no. as primary key, Name as not null, city) Program table - (Program ID as primary key, Program Name as not null, Program Fee not less than 10000, Department) Register table - (Program ID and Roll no. as primary composite key)

CREATE TABLE Student (RollNo INT PRIMARY KEY, Name VARCHAR(15) NOT NULL, City VARCHAR(15));CREATE TABLE Program (p_id INT PRIMARY KEY, p_name VARCHAR(15) NOT NULL, 2 p_fee DECIMAL(10, 2) CHECK (p_fee >= 10000), Department VARCHAR(15));

CREATE TABLE Register(p_id INT, RollNo INT, PRIMARY KEY (p_id, RollNo), FOREIGN KEY (p_id) REFERENCES Program(p_id), FOREIGN KEY (RollNo) REFERENCES Student (RollNo));

• Display the details of students who are registered in the "MCA" program.

SELECT S. RollNo, S.Name, S.City FROM Student S JOIN Register RON S. RollNo = R. RollNo JOIN Program P ON R.p_id = P.P_id WHERE P.p_Name = 'MCA';

• Display the list of all students, who are registered in at least one program.

SELECT DISTINCT S. RollNo, S.Name, S.City FROM Student S JOIN Register R ON S. RollNo = R.RollNo;

• Display the details of programs that have fees greater than the average fee.

SELECT p_id, p_Name, p_fee, Department FROM Program WHERE p_fee > (SELECT AVG(p_fee) FROM Program);

• Display the names of students who are registered in a program having fees less than 30000.

SELECT DISTINCT S. Name FROM Student S JOIN Register R ON S. RollNo = R. RollNo JOIN Program P ON R.p_id = P.p_id WHERE P.p_fee < 30000;

• Display the details of students who have not registered in any course.

S. RollNo, S.Name, S.City FROM Student S WHERE S. RollNo NOT IN (SELECT RollNo FROM Register);**• Display the names of programs in which a maximum number of students are registered.**

```
SELECT p_Name FROM Program P JOIN Register R ON P.p_id = R.p_id GROUP BY
P.p_id, P.p_Name HAVING
COUNT(R. RollNo) = (SELECT MAX(Student Count) FROM (
SELECT COUNT(R.RollNo) AS StudentCount FROM Register R
GROUP BY R.p_id));
```

• Display the names of programs in which a minimum number of students are registered.

```
SELECT p_Name FROM Program P JOIN
Register R ON P.p_id = R.p_id GROUP BY
P.p_id, P.p_Name HAVING COUNT(R.RollNo)=(
SELECT MIN(StudentCount) FROM (SELECT
COUNT(R. RollNo) AS StudentCount FROM Register R
GROUP BY R.p_id));
```

OUTPUT:

```
SQL> CREATE TABLE Student(RollNo INT PRIMARY KEY, Name VARCHAR(15) NOT NULL,
  2  City VARCHAR(15));

Table created.

SQL> CREATE TABLE Program(p_id INT PRIMARY KEY, p_name VARCHAR(15) NOT NULL,
  2  p_fee DECIMAL(10, 2) CHECK (p_fee >= 10000), Department VARCHAR(15));

Table created.

SQL> CREATE TABLE Register(p_id INT, RollNo INT,PRIMARY KEY (p_id, RollNo),
  2  FOREIGN KEY (p_id) REFERENCES Program(p_id),
  3  FOREIGN KEY (RollNo) REFERENCES Student(RollNo));

Table created.
```

```
SQL> SELECT S.RollNo, S.Name, S.City FROM Student S JOIN Register
  2  R ON S.RollNo = R.RollNo JOIN Program P ON
  3  R.p_id = P.P_id WHERE P.p_Name = 'MCA';

    ROLLNO NAME            CITY
---------- --------------- ---------------
         1 Alice           New York
         3 Charlie         Chicago
```

```
SQL> SELECT DISTINCT S.RollNo, S.Name, S.City FROM
  2  Student S JOIN Register R ON S.RollNo = R.RollNo;

    ROLLNO NAME            CITY
---------- --------------- ---------------
         1 Alice           New York
         2 Bob             Los Angeles
         3 Charlie         Chicago
         4 Diana           Houston
```

```
SQL> SELECT p_id, p_Name, p_fee, Department FROM
  2  Program WHERE p_fee > (SELECT AVG(p_fee) FROM Program);

      P_ID P_NAME              P_FEE DEPARTMENT
---------- --------------- ---------- ---------------
       102 MBA                  30000 Management
       104 M.Tech               35000 Engineering
```

```
SQL> SELECT DISTINCT S.Name FROM Student S JOIN
  2  Register R ON S.RollNo = R.RollNo JOIN
  3  Program P ON R.p_id = P.p_id
  4  WHERE P.p_fee < 30000;

NAME
---------------
Alice
Diana
Charlie
```

```
SQL> SELECT S.RollNo, S.Name, S.City FROM Student S
  2  WHERE S.RollNo NOT IN (SELECT RollNo FROM Register);

    ROLLNO NAME            CITY
---------- --------------- ---------------
         5 Eve             Phoenix
```

```
SQL> SELECT p_Name FROM Program P JOIN
  2  Register R ON P.p_id = R.p_id GROUP BY
  3  P.p_id, P.p_Name HAVING
  4  COUNT(R.RollNo) = (SELECT MAX(StudentCount)FROM (
  5  SELECT COUNT(R.RollNo) AS StudentCount FROM Register R
  6  GROUP BY R.p_id));

P_NAME
--------------
MCA
```

```
SQL> SELECT p_Name FROM Program P JOIN
  2  Register R ON P.p_id = R.p_id GROUP BY
  3  P.p_id, P.p_Name HAVING COUNT(R.RollNo)=(
  4  SELECT MIN(StudentCount) FROM (SELECT
  5  COUNT(R.RollNo) AS StudentCount FROM Register R
  6  GROUP BY R.p_id));

P_NAME
---------------
B.Tech
MBA
M.Tech
```

WEEK 10

• Find out the second minimum salary of an employee.

SELECT MIN(Salary) AS Second MinSalary FROM Employee

WHERE Salary > (SELECT MIN(Salary) FROM Employee);

• Find out the second minimum salary of an employee without using limit, dense range, and order by clause.

SELECT MIN(Salary) AS Second MinSalary FROM Employee

WHERE Salary > (SELECT MIN(Salary) FROM Employee);

• Find out the third maximum salary of an employee.

SELECT MAX(Salary) AS ThirdMaxSalary FROM Employee

WHERE Salary < (SELECT MAX(Salary) FROM Employee

WHERE Salary < (SELECT MAX(Salary) FROM Employee));

• Find out the third maximum salary of an employee without using limit, dense range, and order by clause.

SELECT MAX(Salary) AS ThirdMaxSalary FROM Employee

WHERE Salary < (SELECT MAX(Salary) FROM Employee

WHERE Salary < (SELECT MAX(Salary) FROM Employee));

• Display the names and salaries of employees who earn more than the average salary of their department.

SELECT E.Name, E.Salary, E. Department FROM Employee E JOIN (SELECT Department, AVG(Salary) AS AvgSalary FROM Employee GROUP BY Department) DeptAvg ON E. Department = DeptAvg. Department WHERE E. Salary > DeptAvg. AvgSalary;

• Fetch the list of the employee who belongs to the same department but earns less than the second employee.

SELECT e.emp_id, e.Name, e.Salary, e.Department FROM EMPLOYEE e WHERE e.Salary<(SELECT e2.Salary FROM(SELECT emp_id, Salary, Department, RANK() OVER (PARTITION BY Department ORDER BY Salary DESC) salary_rank FROM EMPLOYEE) e2 WHEREe2.salary_rank = 2 AND e2.Department = e.Department) ORDER BY e.Department, e.Salary;

• Display the names of employees who are older than their colleagues in the same department.

```
WITH DeptOldest AS (SELECT Department, MAX(Age) AS OldestAge

FROM Employee GROUP BY Department)SELECT E. Name, E.Age, E. Department FROM
Employee E

JOIN DeptOldest DO ON E. Department = DO. Department

WHERE E. Age = DO.OldestAge;
```

OUTPUT:

```
SQL> SELECT MIN(Salary) AS SecondMinSalary FROM Employee
  2  WHERE Salary > (SELECT MIN(Salary) FROM Employee);

SECONDMINSALARY
---------------
          25000
```

```
SQL> SELECT MIN(Salary) AS SecondMinSalary FROM Employee
  2  WHERE Salary > (SELECT MIN(Salary) FROM Employee);

SECONDMINSALARY
---------------
          25000
```

```
SQL> SELECT MAX(Salary) AS ThirdMaxSalary FROM Employee
  2  WHERE Salary < (SELECT MAX(Salary) FROM Employee
  3  WHERE Salary < (SELECT MAX(Salary) FROM Employee));

THIRDMAXSALARY
--------------
         45000
```

```
SQL> SELECT MAX(Salary) AS ThirdMaxSalary FROM Employee
  2  WHERE Salary < (SELECT MAX(Salary) FROM Employee
  3  WHERE Salary < (SELECT MAX(Salary) FROM Employee));

THIRDMAXSALARY
--------------
         45000
```

```
SQL> SELECT E.Name, E.Salary, E.Department
  2  FROM Employee E JOIN (
  3  SELECT Department, AVG(Salary) AS AvgSalary
  4  FROM Employee GROUP BY Department)
  5  DeptAvg ON E.Department = DeptAvg.Department
  6  WHERE E.Salary > DeptAvg.AvgSalary;

NAME                 SALARY DEPARTMENT
--------------- ---------- --------------------
manvi                 56000 finance
anshul                70000 finance
harsh                 34000 CSE
```

```
SQL> SELECT e.emp_id, e.Name, e.Salary, e.Department
  2  FROM EMPLOYEE e WHERE e.Salary<( SELECT e2.Salary
  3  FROM( SELECT emp_id, Salary, Department,
  4  RANK() OVER (PARTITION BY Department ORDER BY Salary DESC)
  5  AS salary_rank FROM EMPLOYEE ) e2 WHERE
  6  e2.salary_rank = 2 AND
  7  e2.Department = e.Department) ORDER BY e.Department, e.Salary;

    EMP_ID NAME                SALARY DEPARTMENT
---------- --------------- ---------- --------------------
       109 jai                 23000 CSE
       101 radhika             23000 Finance
       110 ayush               25000 Finance
       103 manvi               56000 Finance
```

```
SQL> WITH DeptOldest AS(SELECT Department, MAX(Age)AS OldestAge
  2  FROM Employee GROUP BY Department)
  3  SELECT E.Name, E.Age, E.Department FROM Employee E
  4  JOIN DeptOldest DO ON E.Department = DO.Department
  5  WHERE E.Age = DO.OldestAge;

NAME                    AGE DEPARTMENT
--------------- ---------- --------------------
mansi                   23 medical
manvi                   34 Finance
princy                  22 engineering
jai                     33 CSE
```

WEEK 11

• Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the EMPLOYEE table. This trigger will display the salary difference between the old values and new values.

```
CREATE OR REPLACE TRIGGER trg_employee_changes
INSERT OR UPDATE ON EMPLOYEE
FOR EACH ROW
DECLARE
salary_diff DECIMAL(10, 2);
BEGIN
NEW. Name : UPPER(:NEW.Name);
IF UPDATING THEN
salary_diff : : NEW.Salary - : OLD.Salary;
DBMS_OUTPUT.PUT_LINE('Salary Difference for
emp_id || : OLD.emp_id || ' is: || salary_diff);
END IF;
END;
/
CREATE OR REPLACE TRIGGER trg_employee_delete
DELETE ON EMPLOYEE
EACH ROW
BEGIN
_OUTPUT.PUT_LINE('Employee with emp_id' ||
:OLD.emp_id || ' and Salary || : OLD.Salary || is being deleted.'); 1 1
7END;
/
```

• Add a new employee with the salary value inserted and check the result.

```
INSERT INTO EMPLOYEE VALUES(113, 'mohit', 'CSE', 21, 50000, 'haldwani');
```

• Try to update the existing employee salary and see what happens

```
UPDATE EMPLOYEE SET SALARY = 60000 WHERE NAME='anshul';
```

• Delete a record of employees and check what happens.

```
DELETE FROM EMPLOYEE WHERE NAME ='john';
```

• Convert employee name into uppercase whenever an employee record is inserted or updated.

INSERT INTO EMPLOYEE VALUES(113, 'mohit', 'CSE', 21, 50000, 'haldwani');

SELECT NAME FROM EMPLOYEE WHERE emp_id=113;

OUTPUT:

```
SQL> CREATE OR REPLACE TRIGGER trg_employee_changes
  2  BEFORE INSERT OR UPDATE ON EMPLOYEE
  3  FOR EACH ROW
  4  DECLARE
  5      salary_diff DECIMAL(10, 2);
  6  BEGIN
  7      :NEW.Name := UPPER(:NEW.Name);
  8
  9      IF UPDATING THEN
 10          salary_diff := :NEW.Salary - :OLD.Salary;
 11          DBMS_OUTPUT.PUT_LINE('Salary Difference for
 12  emp_id ' || :OLD.emp_id || ' is: ' || salary_diff);
 13      END IF;
 14  END;
 15  /

Trigger created.

SQL> CREATE OR REPLACE TRIGGER trg_employee_delete
  2  AFTER DELETE ON EMPLOYEE
  3  FOR EACH ROW
  4  BEGIN
  5      DBMS_OUTPUT.PUT_LINE('Employee with emp_id ' ||
  6  :OLD.emp_id || ' and Salary ' || :OLD.Salary || ' is being deleted.');
  7  END;
  8  /

Trigger created.
```

```
SQL> INSERT INTO EMPLOYEE VALUES(112, 'ram',
  2  'Finance', 24, 65000,'haridwar');
New Employee Added: Salary = 65000

1 row created.
```

```
SQL> UPDATE EMPLOYEE SET SALARY = 60000
  2  WHERE NAME='anshul';
Salary Difference for
emp_id 105 is: -10000
Employee Updated: Salary Difference = -10000

1 row updated.
```

```
SQL> DELETE FROM EMPLOYEE WHERE NAME ='john';
Employee with emp_id 111 and Salary 55000 is being deleted.
Employee Deleted: Old Salary = 55000

1 row deleted.
```

```
SQL> INSERT INTO EMPLOYEE VALUES(113, 'mohit', 'CSE', 21, 50000, 'haldwani');
New Employee Added: Salary = 50000

1 row created.

SQL> SELECT NAME FROM EMPLOYEE WHERE emp_id=113;

NAME
---------------
MOHIT
```

WEEK 12

Case study 1: (General Hospital)

A hospital relies on a database to manage its operations effectively. This database helps keep track of various aspects, including different wards like the General Ward, Emergency Ward, and Specific Ward. Each ward contains patients who are admitted based on their General Practitioner's (GP) recommendation and the approval of a consultant from the hospital. When a patient is admitted, the hospital records essential personal details such as their name, age, gender, address, and contact information. This information is crucial for medical and administrative purposes. Additionally, the hospital maintains a separate register to record all medical tests and treatments for each patient, ensuring that their medical history is thoroughly documented. Patients may undergo multiple tests during their stay, and the database is designed to link each patient with these test records. Each patient is assigned a leading consultant who oversees their treatment, but they may also be examined by other doctors if needed. The database also tracks the connections between patients, consultants, and doctors. Consultants and doctors might specialize in different medical fields and can treat patients from various wards, adding flexibility to the care provided. Overall, this database ensures that patient information, medical records, and hospital operations are managed efficiently. It supports the hospital in delivering high-quality care, streamlining administrative tasks, and addressing the specialized needs of patients and medical staff. Based on the details provided in the case study, address the following requirements:

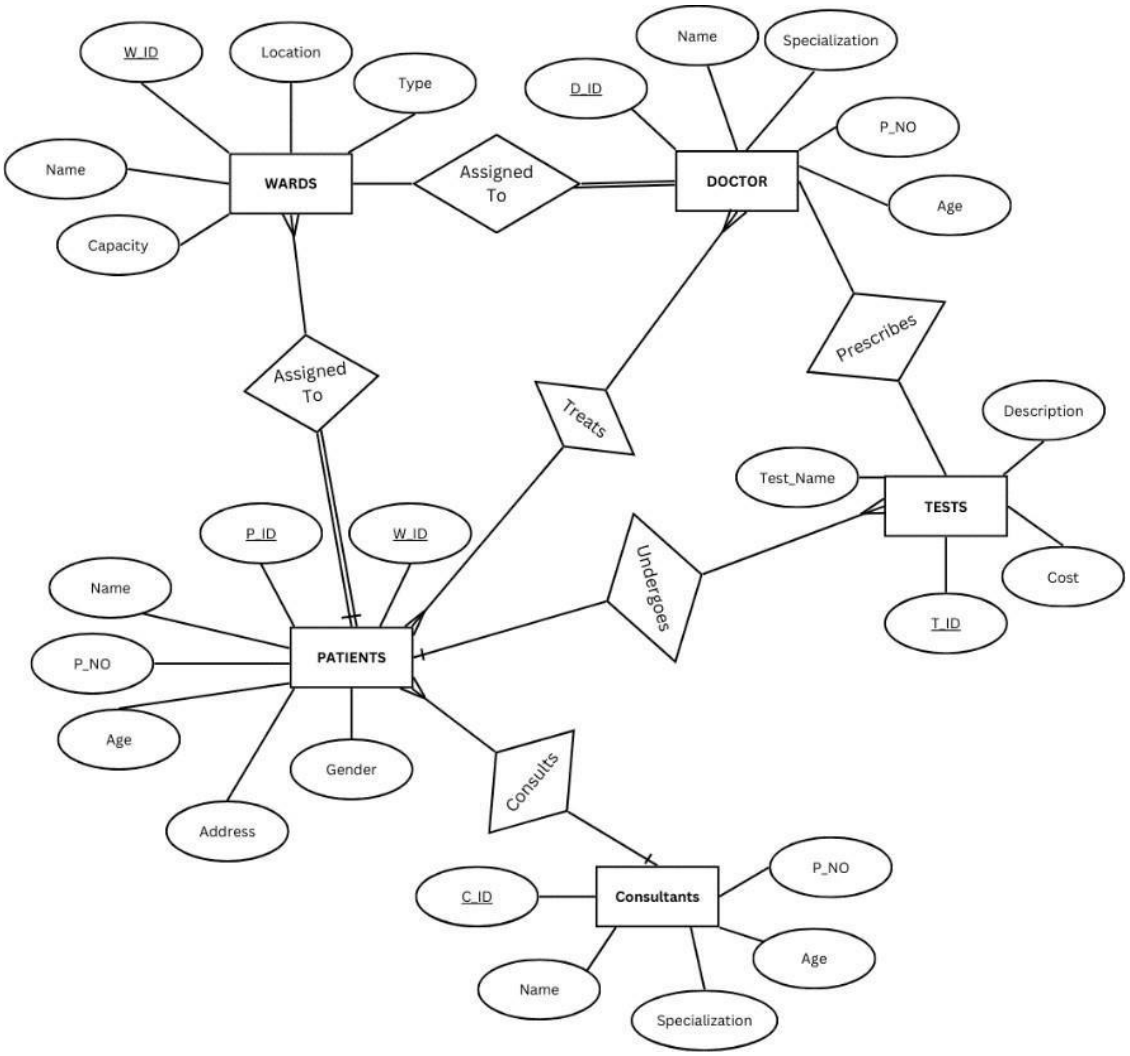
Create an ER diagram based on the hospital's database system case study. You will need to specify two things:

a. Specify all attributes and keys for each entity. Clearly define relationships, such as patients being associated with wards, consultants, and doctors, and include connections between patients and their medical

tests.

b. Define all relationships and constraints, including primary keys, cardinality, and participation constraints. Show how a patient can undergo multiple tests and be treated by various doctors.

Note: Model most constraints from the description. If some constraints can't be represented, provide comments explaining the limitations.


W_ID
Location
Type
Name
WARDS
Capacity
Assigned To
Name
Specialization
D_ID
P_NO
DOCTOR
Age
Assigned To
Treats
Prescribes
Description
Test_Name
TESTS
Cost
T_ID
P_ID
W_ID
Name
P_NO
PATIENTS
Age
Gender
Address
Undergoes
Consults
C_ID
Consultants
P_NO
Age
Name
Specialization