DBMS LAST ASSIGNMENT

Name -> Raunak Garhwal
Roll no. -> 28
MCA 1st sem

Question - 1

Q1 Draw the ER diagram and solve the queries pertaining to the schema below

Student (**Rollno, Name, Dob, Gender, Doa, Bcode**); Implement a check constraint for

- Gender
- Date of Admission

Branch (**Bcode, Bname, Dno**);
Department (**Dno, Dname**);
Course (**Ccode, Cname, Credits, Dno**);
Branch_Course (**Bcode, Ccode, Semester**);
Enrolls (**Rollno, Ccode, Sess, Grade**);
For Example,
SESS can take values **'APRIL 2013', 'NOV 2013'**
Implement a check constraint for grade Value Set **('S', 'A', 'B', 'C', 'D', 'E', 'U');**
Students are admitted to Branches and they are offered by Departments. A branch is offered by only one department.
Each branch has a set of Courses (Subjects). Each student must enroll during a semester. Courses are offered by Departments. A course is offered only by one department. If a student is unsuccessful in a course he/she must enroll for the course during next session. A student has successfully completed a course if the grade obtained by is from the list **(A, B, C, D, and E).**
A student is unsuccessful if he/she have grade **'U'** in a course.
Primary Keys are underlined.

1. Develop a SQL query to list details of Departments that offer more than 3 branches.
2. Develop a SQL query to list the details of Departments that offer more than 6 courses.
3. Develop a SQL query to list the details of courses that are common for more than 3 branches.
4. Develop a SQL query to list students who got 'S' in more than 2 courses during single enrollment.
5. Create a view that will keep track of the roll number, name and number of courses, a student has completed successfully.
6. **Retrieve the names of students who have enrolled in courses from the 'Computer Science' department:**
7. **Retrieve the names of students along with the courses they are enrolled in and their respective grades, only for courses from the 'Electrical Engineering' department:**
8. **Retrieve the names of students who have enrolled in more than one course:**
9. **Retrieve the names of students along with the courses they are enrolled in and their respective grades, ordered by student name and course name:**
10. **Retrieve the names of students who have enrolled in courses with more than 3 credits:**
11. **Retrieve the names of students who have not enrolled in any course:**

```
CREATE DATABASE question_1;

USE question_1;

CREATE TABLE Department(
```

```
    d_no INT(20) PRIMARY KEY,
    d_name VARCHAR(50)
);

CREATE TABLE Branch(
    b_code INT(20) PRIMARY KEY,
    b_name VARCHAR(50),
    d_no INT(20),
    FOREIGN KEY (d_no) REFERENCES department(d_no) ON DELETE
CASCADE ON UPDATE CASCADE
);

CREATE TABLE Course(
    c_code INT(20) PRIMARY KEY,
    c_name VARCHAR(40),
    Credits INT NOT NULL,
    d_no INT(20),
    FOREIGN KEY (d_no) REFERENCES department(d_no) ON DELETE
CASCADE ON UPDATE CASCADE
);

CREATE TABLE Branch_Course(
    b_code INT(20),
    c_code INT(20),
    semester VARCHAR(20) NOT NULL,
    PRIMARY KEY(b_code,c_code,semester),
    FOREIGN KEY (b_code) REFERENCES branch(b_code) ON DELETE
CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (c_code) REFERENCES course(c_code) ON DELETE
CASCADE ON UPDATE CASCADE
);

CREATE TABLE Student(
    roll_no INT(20) PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    dob DATE CHECK (TIMESTAMPDIFF(YEAR,dob,doa) BETWEEN 18 AND
40),
    gender VARCHAR(1) CHECK (gender IN ('M','F','O')),
    doa DATE CHECK ((doa BETWEEN '2013-02-01' AND '2013-03-31') OR
(doa BETWEEN '2013-09-01' AND '2013-10-31')),
    b_code INT(20),
    FOREIGN KEY (b_code) REFERENCES branch(b_code) ON DELETE
CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE Enrolls(
    roll_no INT(20),
    c_code INT(20),
    sess VARCHAR(20) CHECK (sess IN ('April 2013','Nov 2013')),
    grade VARCHAR(1) CHECK (grade IN
('S','A','B','C','D','E','U')),
    PRIMARY KEY(roll_no,c_code,sess),
    FOREIGN KEY (c_code) REFERENCES course(c_code) ON DELETE
CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (roll_no) REFERENCES student(roll_no) ON DELETE
CASCADE ON UPDATE CASCADE
);


INSERT INTO Department (d_no, d_name) VALUES
(1, 'Computer Science'),
(2, 'Electrical Engineering'),
(3, 'Mechanical Engineering');


INSERT INTO Branch (B_code, B_name, d_no) VALUES
(101, 'CS Branch A', 1),
(102, 'CS Branch B', 1),
(103, 'EE Branch X', 2),
(104, 'ME Branch Y', 3),
(105, 'CE Branch Z', 1),
(106, 'CHE Branch W', 2),
(107, 'BME Branch P', 2),
(108, 'AE Branch Q', 1),
(109, 'MS Branch R', 2),
(110, 'EnvE Branch S', 3);


INSERT INTO Course (C_code, C_name, Credits, d_no) VALUES
(501, 'Introduction to Programming', 3, 1),
(502, 'Database Management', 3, 1),
(503, 'Data Structures', 4, 1),
(504, 'Power Systems', 4, 1),
(505, 'Control Systems', 3, 1),
(506, 'Thermodynamics', 4, 3),
(507, 'Fluid Mechanics', 4, 2),
(508, 'Chemical Reaction Engineering', 3, 3),
(509, 'Biomechanics', 4, 1),
(510, 'Aerodynamics', 4, 2);
```

```
INSERT INTO Branch_Course (B_code, C_code, Semester) VALUES
(101, 501, 'APRIL 2013'),
(102, 501, 'NOV 2013'),
(103, 503, 'APRIL 2013'),
(104, 504, 'NOV 2013'),
(105, 505, 'APRIL 2013'),
(106, 501, 'NOV 2013'),
(107, 507, 'APRIL 2013'),
(108, 508, 'NOV 2013'),
(109, 501, 'APRIL 2013'),
(110, 510, 'NOV 2013');


INSERT INTO Student (Roll_no, Name, Dob, Gender, doa, B_code)
VALUES
(1001, 'John Doe', '1991-03-10', 'M', '2013-02-01', 101),
(1002, 'Jane Smith', '1993-07-25', 'F', '2013-09-02', 102),
(1003, 'Bob Johnson', '1993-12-05', 'M', '2013-02-03', 103),
(1004, 'Alice Brown', '1986-02-15', 'F', '2013-10-04', 104),
(1005, 'Charlie Wilson', '1989-09-20', 'M', '2013-09-05', 105),
(1006, 'Eva Davis', '1994-05-18', 'F', '2013-02-06', 106),
(1007, 'Frank Miller', '1982-11-30', 'M', '2013-09-07', 107),
(1008, 'Grace Taylor', '1990-08-12', 'F', '2013-03-08', 108),
(1009, 'Henry Lee', '1991-04-22', 'M', '2013-10-09', 109),
(1010, 'Ivy Chen', '1990-01-14', 'F', '2013-03-10', 110);

INSERT INTO Enrolls (Roll_no, C_code, Sess, Grade) VALUES
(1001, 501, 'NOV 2013', 'S'),
(1001, 502, 'NOV 2013', 'S'),
(1001, 503, 'NOV 2013', 'S'),
(1002, 502, 'APRIL 2013', 'A'),
(1003, 505, 'NOV 2013', 'S'),
(1004, 504, 'APRIL 2013', 'B'),
(1005, 503, 'NOV 2013', 'A'),
(1006, 504, 'APRIL 2013', 'B'),
(1007, 507, 'NOV 2013', 'A'),
(1008, 508, 'APRIL 2013', 'B'),
(1009, 509, 'NOV 2013', 'S');
```

Q.1 Develop a SQL query to list details of Dept.’s that offer more than 3 branches ?

```
SELECT d.d_no,d.d_name FROM department d RIGHT JOIN branch b ON
d.d_no = b.d_no GROUP BY d.d_no HAVING COUNT(b_code) > 3;
```

Q.2 Develop a SQL query to list the details of Departments that offer more than 5 courses ?

```
SELECT d.d_no,d.d_name FROM department d RIGHT JOIN course c ON
d.d_no = c.d_no GROUP BY d.d_no HAVING COUNT(c_code) > 5;
```

Q.3 Develop a SQL query to list the details of courses that are common for more than 3 branches ?

```
select c_code,c_name from course where c_code in (select c_code
from branch_course group by c_code having count(b_code) > 3);
```

Q.4 Develop a SQL query to list students who got ‘S’ in more than 2 courses during single enrollment ?

```
SELECT s.roll_no,s.name,s.dob,s.gender,e.grade FROM student s JOIN
enrolls e ON s.roll_no = e.roll_no WHERE grade = 'S' GROUP BY
e.roll_no,e.sess HAVING COUNT(c_code) > 2;
```

Q.5 Create a view that will keep track of the roll number, name and number of courses, a student has completed successfully ?

```
CREATE VIEW passed_students AS SELECT
s.Roll_no,s.Name,COUNT(c_code) AS Number_of_Courses FROM student s
JOIN enrolls e ON s.roll_no = e.roll_no WHERE e.grade IN
('A','B','C','D','E') GROUP BY s.roll_no;
```

Q.6 Retrieve the names of students who have enrolled in courses from the 'Computer Science' department ?

```
SELECT DISTINCT name from ((branch_course bc left join student s
on bc.b_code = s.b_code ) left join course c on bc.c_code =
c.c_code) where d_no = (select d_no from department where d_name =
'Computer Science' );
```

Q.7 Retrieve the names of students along with the courses they are enrolled in and their respective grades, only for courses from the 'Electrical Engineering' department ?

```
SELECT s.name,c.c_name,grade from (((branch_course bc left join
student s on bc.b_code = s.b_code ) left join course c on
bc.c_code = c.c_code ) right join enrolls e on e.c_code =
bc.c_code) where d_no = (select d_no from department where d_name
= 'Electrical Engineering' );
```

Q.8 Retrieve the names of students who have enrolled in more than one course ?

```
select name from student where roll_no in (select roll_no from
enrolls group by roll_no having count(c_code) > 1);
```

Q.9 Retrieve the names of students along with the courses they are enrolled in and their respective grades,ordered by student name and course name ?

```
SELECT s.name,c.c_name,e.grade from (( enrolls e left join student
s on s.roll_no = e.roll_no ) left join course c on e.c_code =
c.c_code) order by s.name,c.c_name;
```

Q.10 Retrieve the names of students who have enrolled in courses with more than 3 credits ?

```
SELECT name,credits from ((branch_course bc left join student s on
bc.b_code = s.b_code ) left join course c on bc.c_code = c.c_code)
where credits > 3;
```

Q.11 Retrieve the names of students who have not enrolled in any course ?

```
select name,c_code from student s left join enrolls e on s.roll_no
= e.roll_no where c_code is null;
```

Question - 2

Q2 Draw the ER diagram and solve the queries pertaining to the schema below

Customer (**Customerno varchar2 (5), Cname varchar2 (50)**);
Implement check constraints to check Customerno starts with **'C'**.
Cust_Order (**Orderno varchar2(5), Odate Date,** Customerno **references Customer, Ord_amt number(8)**);
Implement check constraints to check Orderno starts with '**O**'.
Ord_amt is derived attribute (default value is 0);
Item (**Itemno varchar2 (5), Item_name varchar2 (30), unit_price number (5)**);
Implement check constraint to check Itemno starts with **'I'**.
Order_item (**Orderno references Cust_order,** Itemno **references item, qty number (3)**);
Primary Key is underlined.

Questions

1. Develop SQL query to list the details of customers who have placed more than 3 orders.
2. Develop a SQL query to list details of items whose price is less than the average price of all items in each order.
3. Develop a SQL query to list the orderno and number of items in each order.
4. Develop a SQL query to list the details of items that are present in 25% of the orders.
5. Develop an update statement to update the value of Ord_amt.
6. Create a view that keeps track of detail of each customer and number of Order placed.
   Retrieve all customers and their orders:
7. **Retrieve total order amount for each customer:**
8. **Retrieve all items ordered along with their quantities:**
9. **Retrieve orders with their total amount and the customer's name:**
10. **Retrieve orders that have an order amount greater than a certain value:**
11. **Retrieve all customers who have not placed any orders:**
12. **Retrieve the top 5 customers based on their total order amount:**

```
create database question_2;

use question_2;

create table customer(
    customer_no varchar(25) primary key check (customer_no like
'C%'),
    customer_name varchar(40)
);

create table cust_order(
    order_no varchar(25) primary key check (order_no like 'O%'),
    order_date date,
    customer_no varchar(25),
    order_amt DECIMAL(10,2) default 0,
    foreign key (customer_no) references customer(customer_no)
```

```
);

create table item(
    item_no varchar(25) primary key check (item_no like 'I%'),
    item_name varchar(30),
    unit_price DECIMAL(10,2)
);

create table order_item(
    order_no varchar(25),
    item_no varchar(25),
    qty INT(100),
    primary key(order_no,item_no),
    foreign key (order_no) references cust_order(order_no),
    foreign key (item_no) references item(item_no));
INSERT INTO `customer`(`customer_no`, `customer_name`) VALUES
('C1', 'Rohan'),
('C2', 'Mukesh'),
('C3', 'John'),
('C4', 'Mohan'),
('C5', 'Rahul'),
('C6', 'Raunak'),
('C7', 'Nikesh'),
('C8','Virat');

INSERT INTO `cust_order`(`order_no`, `order_date`, `customer_no`,
`order_amt`) VALUES
('O1', '2022-01-01', 'C1', 1000),
('O2', '2022-02-15', 'C2', 15000),
('O3', '2022-03-20', 'C3', 1200),
('O4', '2022-04-05', 'C4', 800),
('O5', '2022-05-10', 'C5', 2000),
('O6', '2022-06-15', 'C6', 1300),
('O7', '2022-07-20', 'C7', 900),
('O8', '2022-08-25', 'C1', 1800),
('O9', '2022-09-30', 'C1', 700),
('O10', '2022-10-05', 'C1', 1600);

INSERT INTO `item`(`item_no`, `item_name`, `unit_price`) VALUES
('I1', 'Product A', 50),
('I2', 'Product B', 70),
('I3', 'Product C', 30),
('I4', 'Product D', 60),
('I5', 'Product E', 90),
```

```
('I6', 'Product F', 40),
('I7', 'Product G', 85),
('I8', 'Product H', 55),
('I9', 'Product I', 70),
('I10', 'Product J', 120);

INSERT INTO `order_item`(`order_no`, `item_no`, `qty`) VALUES
('O10', 'I1', 2),
('O1', 'I2', 3),
('O2', 'I1', 1),
('O3', 'I3', 5),
('O4', 'I4', 2),
('O5', 'I5', 4),
('O6', 'I6', 3),
('O7', 'I7', 2),
('O8', 'I8', 1),
('O9', 'I9', 4);
```

Q.1 Develop SQL query to list the details of customers who have placed more than 3 orders ?

```
select * from customer where customer_no in (select customer_no
from cust_order group by customer_no having count(order_no) > 3);
```

Q.2 Develop a SQL query to list details of items whose price is less than the average price of all items in each order ?

```
select * from item where unit_price < (select avg(unit_price) from
item);
```

Q.3 Develop a SQL query to list the orderno and number of items in each order ?

```
SELECT order_no,COUNT(item_no) AS number_of_items FROM order_item
GROUP BY order_no;
```

Q.4 Develop a SQL query to list the details of items that are present in 25% of the orders ?

```
SELECT I.* FROM (( Item I JOIN Order_item OI ON I.Item_no =
OI.Item_no ) JOIN Cust_Order CO ON OI.Order_no = CO.Order_no )
GROUP BY I.Item_no, I.Item_name, I.Unit_price HAVING
COUNT(DISTINCT OI.Order_no) >= 0.25 * COUNT(DISTINCT CO.Order_no);
```

Q.5 Develop an update statement to update the value of Ord_amt ?

```
UPDATE cust_order SET order_amt = order_amt*1.1;
```

Q.6 Create a view that keeps track of detail of each customer and number of Order placed ?

```
create view customer_order_view as select
c.customer_no,c.customer_name,count(co.order_no) as 'Number of
orders placed' from customer c right join cust_order co on
c.customer_no = co.customer_no group by c.customer_no;
```

Q.7 Retrieve all customers and their orders ?

```
SELECT c.*, co.* FROM customer c LEFT JOIN cust_order co ON
c.customer_no = co.customer_no;
```

Q.8 Retrieve total order amount for each customer ?

```
SELECT c.customer_no, c.customer_name, SUM(co.order_amt) AS
total_order_amount FROM customer c RIGHT JOIN cust_order co ON
c.customer_no = co.customer_no GROUP BY c.customer_no;
```

Q.9 Retrieve all items ordered along with their quantities ?

```
SELECT item_no, SUM(qty) as 'Quantity Ordered' FROM order_item
group by item_no;
```

Q.10 Retrieve orders with their total amount and the customer's name ?

```
SELECT cust_order.order_no, Customer.customer_name,
SUM(order_item.qty * item.unit_price) AS TotalAmount FROM
cust_order
JOIN Customer ON cust_order.Customer_no = Customer.Customer_no
JOIN order_item ON cust_order.order_no = Order_Item.order_no
JOIN item ON Order_Item.item_no = item.item_no
GROUP BY cust_order.order_no, Customer.customer_name;
```

Q.11 Retrieve orders that have an order amount greater than a certain value ?

```
SELECT * FROM cust_order WHERE order_amt > 1000;
```

Q.12 Retrieve all customers who have not placed any orders ?

```
SELECT * FROM customer WHERE customer_no NOT IN (SELECT
customer_no FROM cust_order);
```

Q.13 Retrieve the top 5 customers based on their total order amount ?

```
SELECT customer_no,SUM(order_amt) AS total_order_amount FROM
cust_order GROUP BY customer_no ORDER BY total_order_amount DESC
LIMIT 5;
```

Question - 3

Q3 Draw the ER diagram and solve the queries pertaining to the schema below

Staff (**Staffno number (5), Name varchar2 (30), Dob Date, Gender Char (2), Doj Date, Designation varchar2 (30), Basic_pay number (6), Deptno varchar2 (5)**);
Gender must take value **'M'** or **'F'.**
Dept (**Deptno varchar2 (5), Name varchar2 (30)**);
Skill (**Skill_code varchar2 (5), Description varchar2 (30), Charge_Outrage number (3)**); Staff_skill (**Staffno number (5), Skill_code varchar2 (5)**);
Project (**Projectno varchar2 (5), Pname varchar2 (5), Start_Date Date, End_Date Date, Project_Manager_Staffno number (5)**);
Project Number must start with **'P'.**
Works (**Staffno number (5), Projectno varchar2 (5), Date_Worked_On Date, Intime Timestamp, Outtime Timestamp**);
Primary Key is underlined.

Questions

1. Develop DDL to implement the above schema specifying appropriate data types for each attributes and enforcing primary key, check constraints and foreign key constraints.
2. Populate the database with rich data set.
3. Develop a SQL query to list the departmentno and number of staff in each department,
4. Develop a SQL query to list the details of staff who earn the AVG basic pay of all staff.
5. Develop a SQL query to list the details of staff who have more than 3 skills.
6. Develop a SQL query to list the details of staff who have skills with a charge greater than 60 per hour.
7. Create a view that will keep track of the department number, department name, the number of employees in the department and total basic pay expenditure for the department.
8. Develop a SQL query to list the details of Dept. which has more than 5 staff working in it.
9. Develop a SQL query to list the details of staff who have more than 3 skills.
10. Retrieve staff members who worked on a specific project on a given date:
11. Retrieve staff members and their associated skills:
12. **Retrieve staff members who have a basic pay greater than the average basic pay of all staff members:**
13. **Retrieve projects along with the total number of staff members working on each project:**
14. **Retrieve staff members who have worked on projects managed by staff members with a specific designation:**
15. **Retrieve projects with their managers and the number of staff members working on each project, sorted by the number of staff members in descending order:**
16. **Retrieve the top 5 staff members who have worked on the most number of projects:**
17. **Retrieve staff members along with their skills and the charge outrage associated with each skill:**
18. Retrieve staff members with their corresponding department names

```
create database question_3;

use question_3;

create table department(
dept_no varchar(20) PRIMARY KEY,
dept_name varchar(30)
```

```
);

create table staff(
staff_no int(20) PRIMARY KEY,
staff_name varchar(30),
dob date,
gender VARCHAR(1) CHECK (gender in ('M','F','O')),
doj date,
designation varchar(30),
basic_pay Decimal(10,2),
dept_no varchar(20), FOREIGN KEY (dept_no) REFERENCES
department(dept_no)
);

create table skill(
skill_code varchar(20) PRIMARY KEY,
description varchar(30),
charge_outrate DECIMAL(10,2)
);

create table staff_skill(
staff_no INT(10),
skill_code varchar(30),
PRIMARY KEY (staff_no,skill_code), FOREIGN KEY (staff_no)
REFERENCES staff(staff_no),
FOREIGN KEY (skill_code) REFERENCES skill(skill_code)
);

create table project(
project_no varchar(20) PRIMARY KEY CHECK (project_no LIKE 'P%'),
project_name varchar(30),
start_date date,
end_date date,
project_manager_staffno INT(10)
);

create table works(
staff_no INT(10),
project_no varchar(20),
date_worked_on date,
in_time timestamp not null default current_timestamp,
out_time timestamp not null default current_timestamp,
PRIMARY KEY (staff_no,project_no), FOREIGN KEY (project_no)
REFERENCES project(project_no)
```

```
);

INSERT INTO department(dept_no,dept_name)
VALUES ('D1', 'Human Resource'),
('D2', 'DevOps'),
('D3', 'Designing'),
('D4', 'AI & ML');

INSERT INTO Staff (staff_no, staff_name, dob, gender, doj,
designation, basic_pay, dept_no)
VALUES ('1', 'A', '1990-01-15', 'M', '2015-02-01', 'Manager',
75000, 'D1'),
('2', 'B', '1988-07-25', 'F', '2016-05-10', 'Developer', 45000,
'D2'),
('3', 'C', '1995-03-12', 'F', '2017-09-20', 'Analyst', 70000,
'D2'),
('4', 'D', '1992-11-05', 'F', '2018-04-15', 'Designer', 50000,
'D3'),
('5', 'E', '1997-05-23', 'M', '2014-12-27', 'Debugger', 40000,
'D2'),
('6', 'F', '1987-09-30', 'M', '2019-08-01', 'Tester', 45000,
'D2'),
('7', 'G', '1997-05-23', 'M', '2014-12-27', 'Debugger', 35000,
'D2'),
('8', 'H', '1987-09-30', 'M', '2019-08-01', 'Tester', 40000,
'D2');

INSERT INTO Skill (skill_code, description, charge_outrate)
VALUES ('S1', 'Programming', 50),
('S2', 'Designing', 90),
('S3', 'Testing', 70),
('S4', 'Debugging', 40);

INSERT INTO Staff_Skill (staff_no, skill_code)
VALUES (1, 'S1'),
(1, 'S2'),
(1, 'S3'),
(1, 'S4'),
(2, 'S3'),
(3, 'S1'),
(4, 'S2'),
(5, 'S4'),
(6, 'S3');
```

```
INSERT INTO Project (project_no, project_name, start_date,
end_date, project_manager_staffno)
VALUES ('P1', 'ProjectA', '2022-01-01', '2022-06-30', 1),
('P2', 'ProjectB', '2023-01-01', '2023-04-30', 2),
 ('P3', 'ProjectA', '2024-04-10', '2022-04-11', 1),
('P4', 'ProjectB', '2024-04-11', '2023-04-12', 2);

INSERT INTO Works (staff_no, project_no, date_worked_on)
VALUES (1, 'P1', '2024-04-10'),
(1, 'P2', '2024-06-25'),
(2, 'P1', '2024-04-10'),
(2, 'P3', '2024-11-03'),
(3, 'P2', '2024-04-11'),
(4, 'P1', '2024-04-11'),
(5, 'P2', '2024-04-12'),
(6, 'P3', '2024-04-10'),
(7, 'P3', '2024-04-11');
```

Q.3 Develop a SQL query to list the departmentno and number of staff in each department ?

```
SELECT d.dept_no,d.dept_name,COUNT(staff_no) as 'Number_of_staff'
FROM department d LEFT JOIN staff s ON d.dept_no = s.dept_no GROUP
BY d.dept_no;
```

Q.4 Develop a SQL query to list the details of staff who earn the AVG basic pay of all staff ?

```
SELECT staff_no,staff_name,basic_pay FROM staff WHERE basic_pay =
(SELECT AVG(basic_pay) FROM staff);
```

Q.5 Develop a SQL query to list the details of staff who have more than 3 skills ?

```
SELECT * FROM staff WHERE staff_no IN (SELECT staff_no FROM
staff_skill GROUP BY staff_no HAVING COUNT(skill_code) > 3);
```

Q.6 Develop a SQL query to list the details of staff who have skills with a charge greater than 60 per hour ?

```
SELECT * FROM staff WHERE staff_no IN (SELECT staff_no FROM
staff_skill WHERE skill_code IN (SELECT skill_code FROM skill
WHERE charge_outrate > 60));
```

Q.7 Create a view that will keep track of the department number, department name, the number of employees in the department and total basic pay expenditure for the department ?

```
CREATE VIEW Department_Summary_View AS SELECT d.dept_no AS
'Department_No.',d.dept_name AS 'Department_Name' ,COUNT(staff_no)
AS 'Number of Employees',SUM(basic_pay) AS 'Total Basic-pay
Expenditure' FROM department d LEFT JOIN staff s ON d.dept_no =
s.dept_no GROUP BY d.dept_no;
```

Q.8 Develop a SQL query to list the details of Department which has more than 5 staff working in it ?

```
SELECT d.* FROM department d LEFT JOIN staff s ON d.dept_no =
s.dept_no GROUP BY d.dept_no HAVING COUNT(s.staff_no) > 5;
```

Q.9 Develop a SQL query to list the details of staff who have more than 3 skills ?

```
SELECT * FROM staff WHERE staff_no IN (SELECT staff_no FROM
staff_skill GROUP BY staff_no HAVING COUNT(skill_code) > 3);
```

Q.10 Retrieve staff members who worked on a specific project on a given date ?

```
SELECT s.staff_no,s.staff_name FROM works w JOIN staff s ON
s.staff_no = w.staff_no JOIN project p ON p.project_no=
w.project_no WHERE p.project_name =  'ProjectA' AND
w.date_worked_on = 2024-04-11;
```

Q.11 Retrieve staff members and their associated skills ?

```
select s.staff_name,sk.description from staff s join staff_skill
ss on s.staff_no = sk.staff_no join skill sk on sk.skill_code =
sk.skill_code;
```

Q.12 Retrieve staff members who have a basic pay greater than the average basic pay of all staff members ?

```
SELECT s.staff_no,s.staff_name,s.basic_pay FROM staff s WHERE
s.basic_pay > (SELECT AVG(basic_pay) FROM staff);
```

Q.13 Retrieve projects along with the total number of staff members working on each project ?

```
SELECT p.project_no,p.project_name,COUNT(w.staff_no) AS
total_staff_members FROM project p LEFT JOIN works w ON
p.project_no = w.project_no GROUP BY p.project_no, p.project_name;
```

Q.14 Retrieve staff members who have worked on projects managed by staff members with a specific designation ?

```
SELECT DISTINCT s.staff_no, s.staff_name FROM staff s INNER JOIN
project p ON s.staff_no = p.project_manager_staffno INNER JOIN
works w ON p.project_no = w.project_no WHERE s.designation =
'Manager';
```

Q.15 Retrieve projects with their managers and the number of staff members working on each project, sorted by the number of staff members in descending order ?

```
SELECT p.project_no,p.project_name,s.staff_name AS
project_manager,COUNT(w.staff_no) AS num_staff_members
FROM project p INNER JOIN staff s ON p.project_manager_staffno =
s.staff_no LEFT JOIN works w ON p.project_no = w.project_no GROUP
BY p.project_no, p.project_name, s.staff_name ORDER BY
num_staff_members DESC;
```

Q.16 Retrieve the top 5 staff members who have worked on the most number of projects ?

```
SELECT w.staff_no,s.staff_name,COUNT(DISTINCT w.project_no) AS
num_projects_worked_on FROM works w INNER JOIN staff s ON
```

```
w.staff_no = s.staff_no GROUP BY w.staff_no ORDER BY
num_projects_worked_on DESC LIMIT 5;
```

Q.17 Retrieve staff members along with their skills and the charge outrage associated with each skill ?

```
SELECT s.staff_no,s.staff_name,sk.skill_code,sk.description AS
skill_description,sk.charge_outrate FROM staff s INNER JOIN
staff_skill ss ON s.staff_no = ss.staff_no INNER JOIN skill sk ON
ss.skill_code = sk.skill_code;
```

Q.18 Retrieve staff members with their corresponding department names ?

```
SELECT s.staff_no,s.staff_name,d.dept_name FROM staff s INNER JOIN
department d ON s.dept_no = d.dept_no;
```

Question - 4

Q1 Draw the ER diagram and solve the queries pertaining to the Student database

Table: Students :Columns: student_id (INT), student_name (VARCHAR), age (INT), gender (VARCHAR), department_id (INT)
Table: Departments :Columns: department_id (INT), department_name (VARCHAR), location (VARCHAR)
Table: Courses :Columns: course_id (INT), course_name (VARCHAR), department_id (INT)
Table: Enrollments :Columns: enrollment_id (INT), student_id (INT), course_id (INT), grade (VARCHAR)

QUERIES

1. Display the names of all students who are enrolled in the "Mathematics" course.
2. Display the names, ages, and genders of all students who are enrolled in the "Mathematics" course, have an age above the average age of all students, and have obtained a grade of 'A' in at least one course.
3. Display the names and ages of all students who are enrolled in the "Mathematics" course and are older than 20 years.
4. List the department names along with the count of students who are enrolled in at least one course, have an average grade above 'B', and have enrolled in courses offered by departments located in "New York".
5. List the department names and the count of students in each department, sorted by the count of students in descending order.
6. Retrieve the names of all students who have not enrolled in any course, are not from the "Engineering" department, and have not obtained any grade lower than 'C'.
7. List the department names along with the count of students who are enrolled in at least one course and have an average grade above 'B'.
8. Find the average age of male students who have enrolled in courses with a department located in "New York" and have obtained a grade of 'A' in at least two different courses.
9. Retrieve the names of all students who have not enrolled in any course.
10. Retrieve the names of all students who have not enrolled in any course and are not from the "Engineering" department.
11. Find the average age of male students who have enrolled in courses with a department located in "New York".
12. Display the names of courses along with the count of students enrolled in each course, where the count of students is greater than the count of students enrolled in the course with the lowest enrollment count.
13. Display the names of courses along with the count of students enrolled in each course.
14. Display the names of courses along with the count of students enrolled in each course, where the count of students is greater than the average count of students across all courses
15. List the department names along with the average age of students in each department, excluding departments with fewer than 10 students, and where the average age is below the overall average age of all students.

```
Create Database University;

Use University;

Create table Department(
dept_id int primary key,
dept_name varchar(40),
```

```
location varchar(40)
);

Create table Student(
student_id int primary key,
student_name varchar(40),
age int,
gender varchar(1) check (gender in ('M','F')),
dept_id int,
Foreign key (dept_id) references department(dept_id)
);

Create table Course(
course_id int primary key,
course_name varchar(40),
dept_id int,
Foreign key (dept_id) references department(dept_id)
);

Create table Enrollment(
enrollment_id int primary key,
student_id int,
course_id int,
grade varchar(1) check (grade in ('A','B','C','D','E')),
Foreign key (course_id) references course(course_id),
Foreign key (student_id) references student(student_id)
);

INSERT INTO Department VALUES
(1, 'Mathematics', 'New York'),
(2, 'Engineering', 'California');

INSERT INTO Student VALUES
(1, 'Alice', 22, 'F', 1),
(2, 'Bob', 20, 'M', 2),
(3, 'Charlie', 25, 'M', 1),
(4, 'David', 23, 'M', 1),
(5, 'Emma', 21, 'F', 2),
(6, 'Frank', 22, 'M', 1),
(7, 'Grace', 26, 'F', 1),
(8, 'Henry', 24, 'M', 1),
(9, 'Ivy', 22, 'F', 2),
(10, 'Jack', 22, 'M', 1),
(11, 'Kate', 23, 'F', 1),
```

```
(12, 'Liam', 21, 'M', 2),
(13, 'Mia', 20, 'F', 1),
(14, 'Noah', 22, 'M', 2),
(15, 'Olivia', 25, 'F', 1),
(16, 'Peter', 24, 'M', 2),
(17, 'Quinn', 23, 'M', 1),
(18, 'Rachel', 22, 'F', 1),
(19, 'Sam', 20, 'M', 1),
(20, 'Taylor', 22, 'F', 2),
(21, 'Uma', 21, 'F', 1),
(22, 'Victor', 24, 'M', 2),
(23, 'Wendy', 22, 'F', 1),
(24, 'Xavier', 23, 'M', 2),
(25, 'Yara', 22, 'F', 1),
(26, 'Zane', 21, 'M', 2);

INSERT INTO Course VALUES
(1, 'Mathematics', 1),
(2, 'Physics', 1),
(3, 'Chemistry', 2);

INSERT INTO Enrollment VALUES
(1, 1, 1, 'A'),
(2, 2, 1, 'B'),
(3, 3, 1, 'A'),
(4, 4, 1, 'A'),
(5, 5, 1, 'A'),
(6, 6, 1, 'B'),
(7, 7, 1, 'A'),
(8, 8, 1, 'A'),
(9, 9, 1, 'A'),
(10, 17, 1, 'A'),
(11, 17, 2, 'A'),
(12, 18, 1, 'A'),
(13, 23, 1, 'B'),
(14, 24, 1, 'A'),
(15, 25, 1, 'A'),
(16, 26, 1, 'A');
```

Q.1 Display the names of all students who are enrolled in the "Mathematics" course ?

```
select s.student_name,course_name from ((enrollment e left join
student s on s.student_id = e.student_id ) left join course c on
e.course_id = c.course_id ) where course_name = 'Mathematics';
```

Q.2 Display the names, ages, and genders of all students who are enrolled in the "Mathematics" course,have an age above the average age of all students, and have obtained a grade of 'A' in at least one course ?

```
select s.student_name,s.age,s.gender from ((enrollment e left join student s on s.student_id = e.student_id ) left join course c on e.course_id = c.course_id ) where course_name = 'Mathematics' and age > (select avg(age) from student) and grade = 'A';
```

Q.3 Display the names and ages of all students who are enrolled in the "Mathematics" course and are older than 20 years ?

```
select s.student_name,s.age from ((enrollment e left join student s on s.student_id = e.student_id ) left join course c on e.course_id = c.course_id ) where course_name = 'Mathematics' and age > 20;
```

Q.4 List the department names along with the count of students who are enrolled in at least one course,have an average grade above 'B', and have enrolled in courses offered by departments located in "New York" ?

```
select dept_name,count(*) as 'Count of Students' from ((student s right join department d on s.dept_id = d.dept_id ) right join enrollment e on e.student_id = s.student_id ) where grade = 'A' and d.location = 'New York' and course_id is not null group by d.dept_name;
```

Q.5 List the department names and the count of students in each department, sorted by the count of students in descending order ?

```
select d.dept_name,count(s.student_id) as 'count of students' from department d left join student s on s.dept_id = d.dept_id group by dept_id order by 'count of students' desc;
```

Q.6 Retrieve the names of all students who have not enrolled in any course, are not from the "Engineering" department, and have not obtained any grade lower than 'C' ?

```
SELECT student_name FROM Student WHERE student_id NOT IN (SELECT
student_id FROM Enrollment) AND dept_id <> (SELECT dept_id FROM
Department WHERE dept_name = 'Engineering') AND student_id NOT IN
(SELECT student_id FROM Enrollment WHERE grade IN ('D', 'E'));
```

Q.7 List the department names along with the count of students who are enrolled in at least one course and have an average grade above 'B' ?

```
select d.dept_name,count(s.student_id) as 'count of students' from
((student s left join department d on s.dept_id = d.dept_id ) left
join enrollment e on s.student_id = e.student_id ) group by
dept_id where course_id is not null and grade = 'A';
```

Q.8 Find the average age of male students who have enrolled in courses with a department located in "New York" and have obtained a grade of 'A' in at least two different courses ?

```
select avg(s.age) from ((student s left join department d on
s.dept_id = d.dept_id ) left join enrollment e on s.student_id =
e.student_id ) where location = 'New York' and grade = 'A' and
gender = 'M' group by s.student_id having count(course_id) >= 2;
```

Q.9 Retrieve the names of all students who have not enrolled in any course ?

```
select s.student_name from student s left join enrollment e on
e.student_id = s.student_id  where enrollment_id is null ;
SELECT student_name FROM Student WHERE student_id NOT IN (SELECT
student_id FROM Enrollment);
```

Q.10 Retrieve the names of all students who have not enrolled in any course and are not from the "Engineering" department ?

```
select s.student_name from student s left join enrollment e on
s.student_id = e.student_id where dept_id not in (select dept_id
from department where dept_name = 'Engineering') and enrollment_id
is null;
```

Q.11 Find the average age of male students who have enrolled in courses with a department located in "New York" ?

```
select avg(s.age) from ((student s left join department d on
s.dept_id = d.dept_id ) left join enrollment e on s.student_id =
e.student_id ) where location = 'New York' and gender = 'M' and
e.enrollment_id is not null;
```

Q.12 Display the names of courses along with the count of students enrolled in each course, where the count of students is greater than the count of students enrolled in the course with the lowest enrollment count ?

```
SELECT c.course_name, COUNT(DISTINCT e.student_id) AS
student_count FROM Course c LEFT JOIN Enrollment e ON c.course_id
= e.course_id GROUP BY c.course_name HAVING COUNT(DISTINCT
e.student_id) > (SELECT COUNT(DISTINCT e2.student_id) FROM Course
c2 LEFT JOIN Enrollment e2 ON c2.course_id = e2.course_id GROUP BY
c2.course_id ORDER BY COUNT(DISTINCT e2.student_id) ASC LIMIT 1);
```

Q.13 Display the names of courses along with the count of students enrolled in each course ?

```
select c.course_name,count(e.enrollment_id) as 'count of students'
from course c left join enrollment e on c.course_id = e.course_id
group by c.course_name;
```

Q.14 Display the names of courses along with the count of students enrolled in each course, where the count of students is greater than the average count of students across all courses ?

```
SELECT c.course_name, COUNT(DISTINCT e.student_id) AS
student_count FROM Course c LEFT JOIN Enrollment e ON c.course_id
= e.course_id
GROUP BY c.course_name HAVING COUNT(DISTINCT e.student_id) >
(SELECT AVG(student_count) FROM (SELECT COUNT(DISTINCT
e2.student_id) AS student_count FROM Course c2 LEFT JOIN
Enrollment e2 ON c2.course_id = e2.course_id GROUP BY
c2.course_name) AS avg_counts);
```

Q.15 List the department names along with the average age of students in each department, excluding departments with fewer than

10 students, and where the average age is below the overall average age of all students ?

```
select d.dept_name,avg(s.age) from department d left join student
s on d.dept_id = s.dept_id group by d.dept_id having
count(student_id) >= 10 and avg(age) < (select avg(age) from
student);
```

Question - 5

Q2 Draw the ER diagram and solve the queries pertaining to the Student database

Table: Students Columns: student_id (INT), student_name (VARCHAR), age (INT), gender (VARCHAR), department_id (INT)

Table: Departments Columns: department_id (INT), department_name (VARCHAR), location (VARCHAR)

Table: Courses Columns: course_id (INT), course_name (VARCHAR), department_id (INT)

Table: Enrollments Columns: enrollment_id (INT), student_id (INT), course_id (INT), grade (VARCHAR)

QUERIES

1. List the department names along with the average age of students in each department.
2. List the department names along with the average age of students in each department, excluding departments with fewer than 10 students.
3. Find the highest grade obtained by each student along with their names.
4. Find the highest grade obtained by each student along with their names, and only include students who have taken more than one course.
5. Find the highest grade obtained by each student along with their names, and only include students who have taken more than one course and have obtained at least one 'A' grade.
6. Display the course names along with the count of male and female students enrolled in each course, where the count of male students is less than the count of female students, and at least one student has obtained a grade higher than 'B'.
7. Find the student who has enrolled in the most number of courses, has obtained at least one 'A' grade, and is enrolled in courses offered by departments located in at least two different locations.
8. Retrieve the student names and their grades in the "Computer Science" course.
9. Retrieve the student names, their grades in the "Computer Science" course, and the number of courses they are enrolled in, excluding students who have not enrolled in any course.
10. Retrieve the student names and their grades in the "Computer Science" course, along with the number of courses they are enrolled in.
11. Find the student who has enrolled in the most number of courses.
12. Find the student who has enrolled in the most number of courses, and also display the count of courses they are enrolled in.
13. Display the course names along with the count of male and female students enrolled in each course.
14. Display the course names along with the count of female students enrolled in each course, and include courses where the count of female students is greater than the count of male students.
15. Display the course names along with the count of male students enrolled in each course, and include courses where the count of female students is greater than the count of male students.

```
Create Database University;

Use University;

Create table Department(
dept_id int primary key,
dept_name varchar(40),
location varchar(40)
);
```

```
Create table Student(
student_id int primary key,
student_name varchar(40),
age int,
gender varchar(1) check (gender in ('M','F')),
dept_id int,
Foreign key (dept_id) references department(dept_id)
);

Create table Course(
course_id int primary key,
course_name varchar(40),
dept_id int,
Foreign key (dept_id) references department(dept_id)
);

Create table Enrollment(
enrollment_id int primary key,
student_id int,
course_id int,
grade varchar(1) check (grade in ('A','B','C','D','E')),
Foreign key (course_id) references course(course_id),
Foreign key (student_id) references student(student_id)
);

INSERT INTO Department VALUES
(1, 'Mathematics', 'New York'),
(2, 'Engineering', 'California');

INSERT INTO Student VALUES
(1, 'Alice', 22, 'F', 1),
(2, 'Bob', 20, 'M', 2),
(3, 'Charlie', 25, 'M', 1),
(4, 'David', 23, 'M', 1),
(5, 'Emma', 21, 'F', 2),
(6, 'Frank', 22, 'M', 1),
(7, 'Grace', 26, 'F', 1),
(8, 'Henry', 24, 'M', 1),
(9, 'Ivy', 22, 'F', 2),
(10, 'Jack', 22, 'M', 1),
(11, 'Kate', 23, 'F', 1),
(12, 'Liam', 21, 'M', 2),
(13, 'Mia', 20, 'F', 1),
```

```
(14, 'Noah', 22, 'M', 2),
(15, 'Olivia', 25, 'F', 1),
(16, 'Peter', 24, 'M', 2),
(17, 'Quinn', 23, 'M', 1),
(18, 'Rachel', 22, 'F', 1),
(19, 'Sam', 20, 'M', 1),
(20, 'Taylor', 22, 'F', 2),
(21, 'Uma', 21, 'F', 1),
(22, 'Victor', 24, 'M', 2),
(23, 'Wendy', 22, 'F', 1),
(24, 'Xavier', 23, 'M', 2),
(25, 'Yara', 22, 'F', 1),
(26, 'Zane', 21, 'M', 2);

INSERT INTO Course VALUES
(1, 'Computer Science, 1),
(2, 'Mathematics', 1),
(3, 'Chemistry', 2);

INSERT INTO Enrollment VALUES
(1, 1, 1, 'A'),
(2, 2, 1, 'B'),
(3, 3, 1, 'A'),
(4, 4, 1, 'A'),
(5, 5, 1, 'A'),
(6, 6, 1, 'B'),
(7, 7, 1, 'A'),
(8, 8, 1, 'A'),
(9, 9, 1, 'A'),
(10, 17, 1, 'A'),
(11, 17, 2, 'A'),
(12, 18, 1, 'A'),
(13, 23, 1, 'B'),
(14, 24, 1, 'A'),
(15, 25, 1, 'A'),
(16, 26, 1, 'A');
```

Q.1 List the department names along with the average age of students in each department ?

```
select dept_name,avg(age) as 'Average age' from department d left
join student s on d.dept_id = s.dept_id group by d.dept_id;
```

Q.2 List the department names along with the average age of students in each department, excluding departments with fewer than 10 students ?

```
select dept_name,avg(age) as 'Average age' from department d left join student s on d.dept_id = s.dept_id group by d.dept_id having count(s.student_id) >= 10;
```

Q.3 Find the highest grade obtained by each student along with their names ?

```
Select student_name,max(e.grade) from student s right join enrollment e on s.student_id = e.student_id group by s.student_id;
```

Q.4 Find the highest grade obtained by each student along with their names, and only include students who have taken more than one course ?

```
Select student_name,max(e.grade) as 'Highest Grade' from student s right join enrollment e on s.student_id = e.student_id group by s.student_id having Count(e.course_id) > 1;
```

Q.5 Find the highest grade obtained by each student along with their names, and only include students who have taken more than one course and have obtained at least one 'A' grade ?

```
Select student_name,max(e.grade) as 'Highest Grade' from student s right join enrollment e on s.student_id = e.student_id where e.grade = 'A' group by s.student_id having Count(e.course_id) > 1;
```

Q.6 Display the course names along with the count of male and female students enrolled in each course, where the count of male students is less than the count of female students, and at least one student has obtained a grade higher than 'B'?

```
Select course_name,sum(case when s.gender = 'M' then 1 else 0 end) as Male_count,
Sum(case when s.gender = 'M' then 1 else 0 end) as Female_count
from enrollment e left join course c on c.course_id = e.course_id
left join student s on e.student_id = s.student_id where e.grade > 'B' group by c.course_id having Male_count < Female_count;
```

Q.7 Find the student who has enrolled in the most number of courses, has obtained at least one 'A' grade, and is enrolled in courses offered by departments located in at least two different locations ?

```
Select s.student_id from enrollment e right join student s on
s.student_id = e.student_id left join course c on e.course_id =
c.course_id join department d on d.dept_id = c.dept_id where
e.grade = 'A' group by s.student_id having count(e.course_id) =
(select count(e.course_id) from enrollment group by s.student_id )
and count(DISTINCT d.location) > 1;
```

Q.8 Retrieve the student names and their grades in the "Computer Science" course.

```
select s.student_name,e.grade from student s right join enrollment
e on s.student_id = e.student_id where e.course_id = (select
course_id from course where course_name = 'Computer Science');
```

Q.9 Retrieve the student names, their grades in the "Computer Science" course, and the number of courses they are enrolled in, excluding students who have not enrolled in any course ?

```
select s.student_name,grade,count(course_id) as
'Number_of_courses' from student s right join enrollment e on
s.student_id = e.student_id where course_id = (select course_id
from course where course_name = 'Computer Science') group by
s.student_id;
```

Q.10 Retrieve the student names and their grades in the "Computer Science" course, along with the number of courses they are enrolled in ?

```
select s.student_name,grade,count(course_id) as 'courses_enrolled'
from student s right join enrollment e on s.student_id =
e.student_id where course_id = (select course_id from course where
course_name = 'Computer Science') group by s.student_id;
```

Q.11 Find the student who has enrolled in the most number of courses ?

```
SELECT s.student_name FROM Student s JOIN Enrollment e ON
s.student_id = e.student_id GROUP BY s.student_id ORDER BY
COUNT(e.course_id) DESC LIMIT 1;
```

Q.12 Find the student who has enrolled in the most number of courses, and also display the count of courses they are enrolled in ?

```
SELECT s.student_name,COUNT(e.course_id) as 'Courses_enrolled'
FROM Student s JOIN Enrollment e ON s.student_id = e.student_id
GROUP BY s.student_id ORDER BY COUNT(e.course_id) DESC LIMIT 1;
```

Q.13 Display the course names along with the count of male and female students enrolled in each course ?

```
SELECT c.course_name,sum(case when s.gender = 'M' then 1 else 0
end) as Male_count,sum(case when s.gender = 'F' then 1 else 0 end)
as Female_count FROM Enrollment e RIGHT JOIN Course c ON
c.course_id = e.course_id left join student s on e.student_id =
s.student_id GROUP BY c.course_id;
```

Q.14 Display the course names along with the count of female students enrolled in each course, and include courses where the count of female students is greater than the count of male students ?

```
SELECT c.course_name,sum(case when s.gender = 'F' then 1 else 0
end) as Female_count FROM Enrollment e left JOIN Course c ON
c.course_id = e.course_id left join student s on e.student_id =
s.student_id GROUP BY c.course_id having Female_count > count(*) -
Female_count;
```

Q.15 Display the course names along with the count of male students enrolled in each course, and include courses where the count of male students is greater than the count of female students ?

```
SELECT c.course_name,sum(case when s.gender = 'M' then 1 else 0
end) as Male_count FROM Enrollment e left JOIN Course c ON
c.course_id = e.course_id left join student s on e.student_id =
s.student_id GROUP BY c.course_id having Male_count > count(*) -
Male_count;
```

Question - 6

Q3 Draw the ER diagram and solve the queries pertaining to the Medical database

Table: Patients Columns: patient_id (INT), patient_name (VARCHAR), age (INT), gender (VARCHAR), admission_date (DATE), discharge_date (DATE), doctor_id (INT)

Table: Doctors Columns: doctor_id (INT), doctor_name (VARCHAR), specialization (VARCHAR), experience_years (INT)

Table: Medications Columns: medication_id (INT), medication_name (VARCHAR), dosage (VARCHAR), manufacturer (VARCHAR), expiry_date (DATE)

Table: Prescriptions Columns: prescription_id (INT), patient_id (INT), medication_id (INT), prescription_date (DATE), quantity (INT)

QUERIES

1. List patients who were admitted to the hospital between specific dates along with their doctors' names
2. Find the top 5 most prescribed medications along with their total quantities prescribed
3. Retrieve the average age of patients treated by each doctor, ordered by the average age in descending order
4. List patients who were prescribed expired medications:
5. Find doctors who have prescribed medications with a dosage higher than the average dosage:
6. Find the total number of prescriptions written by each doctor, along with their names and the total number of prescriptions, ordered by the total number of prescriptions in descending order:
7. Retrieve the names of medications that have not been prescribed to any patient:
8. List patients who have been discharged within 7 days of admission, along with their names and admission/discharge dates:
9. Find the doctors who have the highest average experience among doctors with at least 5 years of experience:
10. Retrieve the top 3 most common medications prescribed for male patients over the age of 50, along with the total quantity prescribed:
11. Find the total number of patients admitted to each department, ordered by the department with the highest number of admissions:
12. Retrieve the names of patients who have been prescribed medications with dosages greater than 100mg, along with the medication names and dosages:
13. List the names of doctors who have treated patients aged between 30 and 40, along with the total number of patients they treated within this age range:
14. Find the average age of patients admitted to each location, excluding locations with fewer than 10 admissions:
15. Retrieve the names of patients who have been prescribed medications with expiration dates within the next 3 months, along with the medication names and expiry dates:

```
Create database Medical_database;

Use Medical_database;

Create table Doctors(
```

```
doctor_id int primary key,
doctor_name varchar(40),
specialization varchar(40),
experience_years int
);

Create table Patients(
patient_id int primary key,
patient_name varchar(40),
age int,
gender varchar(1),
admission_date date,
discharge_date date,
doctor_id int,
Foreign key (doctor_id) references doctors(doctor_id)
);

Create table Medications(
medication_id int primary key,
medication_name varchar(40),
dosage varchar(40),
manufacturer varchar(40),
expiry_date date
);

Create table Prescriptions(
prescription_id int primary key,
patient_id int,
medication_id int,
prescription_date date,
quantity int,
Foreign key (medication_id) references medications(medication_id),
Foreign key (patient_id) references patients(patient_id)
);

INSERT INTO Doctors (doctor_id,doctor_name, specialization,
experience_years) VALUES
(1, 'Dr. Kumar', 'Cardiology', 10),
(2, 'Dr. Rahul', 'Orthopedics', 15),
(3, 'Dr. Pratap', 'Pediatrics', 8);

INSERT INTO Patients (patient_id,patient_name, age, gender,
admission_date, discharge_date, doctor_id) VALUES
(1, 'Rohit', 32, 'F', '2024-05-01', '2024-05-10', 1),
```

```
(2, 'Virat', 55, 'M', '2024-05-05', '2024-05-20', 2),
(3, 'Hardik', 40, 'F', '2024-05-10', '2024-05-15', 3),
(4, 'Kuldeep', 65, 'M', '2024-05-15', '2024-05-25', 1),
(5, 'Dhoni', 28, 'F', '2024-05-20', '2024-05-30', 2);

INSERT INTO Medications(medication_id,medication_name, dosage,
manufacturer, expiry_date) VALUES
(1, 'Paracetamol', '200mg', 'GHI Pharmaceuticals', '2024-08-01'),
(2, 'Combiflame', '10mg', 'DEF Pharmaceuticals', '2024-03-01'),
(3, 'Levocetrizine', '500mg', 'ABC Pharmaceuticals','2024-07-01'),
(4, 'Azithromycin', '200mg', 'XYZ Pharmaceuticals', '2024-02-01');

INSERT INTO Prescriptions (prescription_id,patient_id,
medication_id, prescription_date, quantity) VALUES
(1, 1, 1, '2024-05-03', 20),
(2, 2, 2, '2024-05-07', 15),
(3, 3, 3, '2024-05-12', 10),
(4, 4, 1, '2024-05-17', 25),
(5, 5, 2, '2024-05-22', 30);
```

Q.1 List patients who were admitted to the hospital between specific dates along with their doctors' names ?

```
select patient_name,patient_id,admission_date,doctor_name from
patients p left join doctors d on p.doctor_id = d.doctor_id;
```

Q.2 Find the top 5 most prescribed medications along with their total quantities prescribed ?

```
select m.medication_id,m.medication_name,sum(quantity) from
medications m left join prescriptions p on m.medication_id =
p.medication_id group by medication_id limit 5;
```

Q.3 Retrieve the average age of patients treated by each doctor, ordered by the average age in descending order ?

```
select p.doctor_id,doctor_name,avg(p.age) as 'average age' from
patients p right join doctors d on p.doctor_id = d.doctor_id group
by doctor_id order by 'average age' desc;
```

Q.4 List patients who were prescribed expired medications ?

```
SELECT p.patient_name,m.medication_name,pr.prescription_date FROM
Patients p INNER JOIN Prescriptions pr ON p.patient_id =
pr.patient_id INNER JOIN Medications m ON pr.medication_id =
m.medication_id WHERE m.expiry_date < CURRENT_DATE();
```

Q.5 Find doctors who have prescribed medications with a dosage higher than the average dosage ?

```
select doctor_id,medication_name,dosage from prescriptions ps
right join patients p on ps.patient_id = p.patient_id right join
medications m on ps.medication_id = m.medication_id where dosage >
(select avg(dosage) from medications) group by doctor_id ;
```

Q.6 Find the total number of prescriptions written by each doctor, along with their names and the total number of prescriptions, ordered by the total number of prescriptions in descending order ?

```
Select doctor_id,doctor_name,count(pr.prescription_id) as
'number_of_prescriptions' from patients p right join prescriptions
pr on p.patient_id = pr.patient_id right join doctors d on
d.doctor_id = p.doctor_id group by d.doctor_id order by
number_of_prescriptions desc;
```

Q.7 Retrieve the names of medications that have not been prescribed to any patient ?

```
Select medication_name from medications where medication_id not in
(select medication_id from prescriptions);
```

Q.8 List patients who have been discharged within 7 days of admission, along with their names and admission/discharge dates ?

```
SELECT patient_name,admission_date,discharge_date FROM Patients
WHERE DATEDIFF(discharge_date, admission_date) <= 7;
```

Q.9 Find the doctors who have the highest average experience among doctors with at least 5 years of experience ?

```
SELECT doctor_name,AVG(experience_years) AS average_experience
FROM Doctors GROUP BY doctor_id HAVING AVG(experience_years) >= 5
ORDER BY average_experience DESC LIMIT 1;
```

Q.10 Retrieve the top 3 most common medications prescribed for male patients over the age of 50,along with the total quantity prescribed ?

```
select m.medication_name,sum(quantity) as quantity_prescribed from
prescriptions ps right join patients p on ps.patient_id =
p.patient_id right join medications m on ps.medication_id =
m.medication_id where p.gender = 'M' and p.age > 50 group by
m.medication_id order by quantity_prescribed desc LIMIT 3;
```

Q.11 Find the total number of patients admitted to each department, ordered by the department with the highest number of admissions ?

```
SELECT d.specialization,COUNT(p.patient_id) AS total_admissions
FROM Doctors d INNER JOIN Patients p ON d.doctor_id = p.doctor_id
GROUP BY d.specialization ORDER BY total_admissions DESC;
```

Q.12 Retrieve the names of patients who have been prescribed medications with dosages greater than 100mg, along with the medication names and dosages ?

```
SELECT p.patient_name,m.medication_name,m.dosage FROM Patients p
INNER JOIN Prescriptions pr ON p.patient_id = pr.patient_id INNER
JOIN Medications m ON pr.medication_id = m.medication_id WHERE
CAST(REPLACE(m.dosage, 'mg', '') AS UNSIGNED) > 100;
```

Q.13 List the names of doctors who have treated patients aged between 30 and 40, along with the total number of patients they treated within this age range ?

```
SELECT d.doctor_name,COUNT(p.patient_id) AS total_patients FROM
Doctors d INNER JOIN Patients p ON d.doctor_id = p.doctor_id WHERE
p.age BETWEEN 30 AND 40 GROUP BY d.doctor_name;
```

Q.14 Find the average age of patients admitted to each location, excluding locations with fewer than 10 admissions ?

```
SELECT location,AVG(age) AS average_age FROM Patients GROUP BY
location HAVING COUNT(patient_id) >= 10;
```

Q.15 Retrieve the names of patients who have been prescribed medications with expiration dates within the next 3 months, along with the medication names and expiry dates ?

```
SELECT p.patient_name,m.medication_name,m.expiry_date FROM
Patients p INNER JOIN Prescriptions pr ON p.patient_id =
pr.patient_id INNER JOIN Medications m ON pr.medication_id =
m.medication_id WHERE m.expiry_date BETWEEN CURDATE() AND
DATE_ADD(CURDATE(), INTERVAL 3 MONTH);
```
