1. Consider the database schemas given below.
Write ER diagram and schema diagram. The primary keys are underlined and the data types are specified.
Create tables for the following schema listed below by properly specifying the primary keys and foreign keys.
Enter at least five tuples for each relation.
Sailors database
SAILORS (sid, sname, rating, age)
BOAT(bid, bname, color)
RSERVERS (sid, bid, date)
Queries, View and Trigger

i. **Find the colours of boats reserved by Albert**
ii. **Find all sailor id's of sailors who have a rating of at least 8 or reserved boat 103**
iii. **Find the names of sailors who have not reserved a boat whose name contains the string "storm". Order the names in ascending order.**
iv. **Find the names of sailors who have reserved all boats.**
v. **Find the name and age of the oldest sailor.**
vi. **For each boat which was reserved by at least 5 sailors with age >= 40, find the boat id and the average age of such sailors.**
vii. **Create a view that shows the names and colours of all the boats that have been reserved by a sailor with a specific rating.**
viii. **A trigger that prevents boats from being deleted If they have active reservations.**

```
-- Find the colours of the boats reserved by Albert
select color
from Sailors s, Boat b, reserves r
where s.sid=r.sid and b.bid=r.bid and s.sname="Albert";
```

```
-- Find all the sailor sids who have rating atleast 8 or reserved boat 103
(select sid
from Sailors
where Sailors.rating>=8)
UNION
(select sid
from reserves
where reserves.bid=103);
```

```
-- Find the names of the sailor who have not reserved a boat whose name contains the string "storm". Order the name in the ascending order
select s.sname
from Sailors s
where s.sid not in
(select s1.sid from Sailors s1, reserves r1 where r1.sid=s1.sid and s1.sname like "%storm%")
and s.sname like "%storm%"
order by s.sname ASC;
```

```
-- Find the name of the sailors who have reserved all boats
select sname from Sailors s where not exists
        (select * from Boat b where not exists
                (select * from reserves r where r.sid=s.sid and b.bid=r.bid));
```

```
-- Find the name and age of the oldest sailor
select sname, age
from Sailors where age in (select max(age) from Sailors);
```

```
-- For each boat which was reserved by atleast 5 sailors with age >= 40,
find the bid and average age of such sailors
select b.bid, avg(s.age) as average_age
from Sailors s, Boat b, reserves r
where r.sid=s.sid and r.bid=b.bid and s.age>=40
group by bid
having 5<=count(distinct r.sid);

-- Create a view that shows the names and colours of all the boats that
have been reserved by a sailor with a specific rating.
create view ReservedBoatsWithRatedSailor as
select distinct bname, color
from Sailors s, Boat b, reserves r
where s.sid=r.sid and b.bid=r.bid and s.rating=5;

select * from ReservedBoatsWithRatedSailor;

-- Trigger that prevents boats from being deleted if they have active
reservation
DELIMITER //
create or replace trigger CheckAndDelete
before delete on Boat
for each row
BEGIN
	IF EXISTS (select * from reserves where reserves.bid=old.bid)
THEN SIGNAL SQLSTATE '45000' SET message_text='Boat is
reserved and hence cannot be deleted';
	END IF;
END;//
DELIMITER ;

delete from Boat where bid=103; -- This gives error since boat 103 is reserved
```

2. Consider the database schemas given below.
Write ER diagram and schema diagram. The primary keys are underlined and the data types are specified.
Create tables for the following schema listed below by properly specifying the primary keys and foreign keys.
Enter at least five tuples for each relation.
Insurance database
PERSON (driver id#: string, name: string, address: string)
CAR (regno: string, model: string, year: int)
ACCIDENT (report_ number: int, acc_date: date, location: string)
OWNS (driver id#: string, regno: string)
PARTICIPATED(driver id#:string, regno:string, report_ number: int,damage_amount: int)

I. Find the total number of people who owned cars that were involved in accidents in 2021.
II. Find the number of accidents in which the cars belonging to “Smith” were involved.
III. Add a new accident to the database; assume any values for required attributes.
IV. Delete the Mazda belonging to “Smith”.
V. Update the damage amount for the car with license number “KA09MA1234” in the accident with report.
VI. A view that shows models and year of cars that are involved in accident.
VII. A trigger that prevents a driver from participating in more than 3 accidents in a given year.

```
-- Find the total number of people who owned a car that were involved in accidents in 2021
select COUNT(driver_id)
from participated p, accident a
where p.report_no=a.report_no and a.accident_date like "2021%";

-- Find the number of accident in which cars belonging to smith were involved
select COUNT(distinct a.report_no)
from accident a
where exists
(select * from person p, participated ptd where p.driver_id=ptd.driver_id
and p.driver_name="Smith" and a.report_no=ptd.report_no);

-- Add a new accident to the database
insert into accident values
(45562, "2024-04-05", "Mandya");

insert into participated values
("D222", "KA-21-BD-4728", 45562, 50000);

-- Delete the Mazda belonging to Smith
delete from car
where model="Mazda" and reg_no in
(select car.reg_no from person p, owns o where p.driver_id=o.driver_id
and o.reg_no=car.reg_no and p.driver_name="Smith");

-- Update the damage amount for the car with reg_no of KA-09-MA-1234 in the accident with report_no
update participated set damage_amount=10000 where report_no=8 and
reg_no="KA-09-MA-1234";
```

```
-- View that shows models and years of car that are involved in accident
create view CarsInAccident as
select distinct model, c_year
from car c, participated p
where c.reg_no=p.reg_no;

select * from CarsInAccident;

-- A trigger that prevents a driver from participating in more than 3
accidents in a given year.
DELIMITER //
create trigger PreventParticipation
before insert on participated
for each row
BEGIN
        IF 3<=(select count(*) from participated where
driver_id=new.driver_id) THEN
                signal sqlstate '45000' set message_text='Driver has
already participated in 3 accidents';
        END IF;
END;//
DELIMITER ;

INSERT INTO participated VALUES
("D222", "KA-20-AB-4223", 66666, 20000);
```

3. Consider the database schemas given below.
Write ER diagram and schema diagram. The primary keys are underlined and the data types are specified.
Create tables for the following schema listed below by properly specifying the primary keys and foreign keys.
Enter at least five tuples for each relation.
Order processing database
Customer (Cust#:int, cname: string, city: string)
Order (order#:int, odate: date, cust#: int, order-amt: int)
Order-item (order#:int, Item#: int, qty: int)
Item (item#:int, unitprice: int)
Shipment (order#:int, warehouse#: int, ship-date: date)
Warehouse (warehouse#:int, city: string)

- i. **List the Order# and Ship_date for all orders shipped from Warehouse# "W2".**
- ii. **List the Warehouse information from which the Customer named "Kumar" was supplied his orders. Produce a listing of Order#, Warehouse#.**
- iii. **Produce a listing: Cname, #ofOrders, Avg_Order_Amt, where the middle column is the total number of orders by the customer and the last column is the average order amount for that customer. (Use aggregate functions)**
- iv. **Delete all orders for customer named "Kumar".**
- v. **Find the item with the maximum unit price.**
- vi. **A trigger that updates order_amout based on quantity and unitprice of order_item**
- vii. **Create a view to display orderID and shipment date of all orders shipped from a warehouse 5.**

```
-- List the Order# and Ship_date for all orders shipped from Warehouse# "W2".
select order_id,ship_date from Shipments where warehouse_id=W2;

-- List the Warehouse information from which the Customer named "Kumar" was supplied his orders. Produce a listing of Order#, Warehouse#
select order_id,warehouse_id from Warehouses natural join Shipments
where order_id in (select order_id from Orders where cust_id in (Select
cust_id from Customers where cname like "%Kumar%"));

-- Produce a listing: Cname, #ofOrders, Avg_Order_Amt, where the middle column is the total number of orders by the customer and the last column is the average order amount for that customer. (Use aggregate functions)
select cname, COUNT(*) as no_of_orders, AVG(order_amt) as
avg_order_amt
from Customers c, Orders o
where c.cust_id=o.cust_id
group by cname;

-- Delete all orders for customer named "Kumar".
delete from Orders where cust_id = (select cust_id from Customers where
cname like "%Kumar%");

-- Find the item with the maximum unit price.
select max(unitprice) from Items;
```

```
-- A tigger that updates order_amount based on quantity and unit price of
order_item
DELIMITER $$
create trigger UpdateOrderAmt
after insert on OrderItems
for each row
BEGIN
	update Orders set order_amt=(new.qty*(select distinct unitprice
from Items NATURAL JOIN OrderItems where item_id=new.item_id))
where Orders.order_id=new.order_id;
END; $$
DELIMITER ;

INSERT INTO Orders VALUES
(006, "2020-12-23", 0004, 1200);

INSERT INTO OrderItems VALUES
(006, 0001, 5); -- This will automatically update the Orders Table also

select * from Orders;

-- Create a view to display orderID and shipment date of all orders
shipped from a warehouse 5.
create view ShipmentDatesFromWarehouse5 as
select order_id, ship_date
from Shipments
where warehouse_id=W5;

select * from ShipmentDatesFromWarehouse5;
```

4. Consider the database schemas given below.
Write ER diagram and schema diagram. The primary keys are underlined and the data types are specified.
Create tables for the following schema listed below by properly specifying the primary keys and foreign keys.
Enter at least five tuples for each relation.
Student enrollment in courses and books adopted for each course
STUDENT (regno: string, name: string, major: string, bdate: date)
COURSE (course#:int, cname: string, dept: string)
ENROLL(regno:string, course#: int,sem: int,marks: int)
BOOK-ADOPTION (course#:int, sem: int, book-ISBN: int)
TEXT (book-ISBN: int, book-title: string, publisher: string,author: string)

- **I. Demonstrate how you add a new text book to the database and make this book be adopted by some department.**
- **II. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.**
- **III. List any department that has all its adopted books published by a specific publisher.**
- **IV. List the students who have scored maximum marks in 'DBMS' course.**
- **V. Create a view to display all the courses opted by a student along with marks obtained.**
- **VI. Create a trigger that prevents a student from enrolling in a course if the marks prerequisite is less than 40.**

```
-- Demonstrate how you add a new text book to the database and make this book be adopted by some department.
insert into TextBook values
(123456, "Chandan The Autobiography", "Pearson", "Chandan");

insert into BookAdoption values
(001, 5, 123456);

-- Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
SELECT c.course,t.bookIsbn,t.book_title
    FROM Course c,BookAdoption ba,TextBook t
    WHERE c.course=ba.course
    AND ba.bookIsbn=t.bookIsbn
    AND c.dept='CS'
    AND 2<(
    SELECT COUNT(bookIsbn)
    FROM BookAdoption b
    WHERE c.course=b.course)
    ORDER BY t.book_title;
```

```
-- List any department that has all its adopted books published by a
specific publisher.
SELECT DISTINCT c.dept
    FROM Course c
    WHERE c.dept IN
    ( SELECT c.dept
    FROM Course c,BookAdoption b,TextBook t
    WHERE c.course=b.course
    AND t.bookIsbn=b.bookIsbn
    AND t.publisher='PEARSON')
    AND c.dept NOT IN
    ( SELECT c.dept
    FROM Course c, BookAdoption b, TextBook t
    WHERE c.course=b.course
    AND t.bookIsbn=b.bookIsbn
    AND t.publisher!='PEARSON');
```

```
-- List the students who have scored maximum marks in ‘DBMS’ course.
select name from Student s, Enroll e, Course c
where s.regno=e.regno and e.course=c.course and c.cname="DBMS" and
e.marks in (select max(marks) from Enroll e1, Course c1 where
c1.cname="DBMS" and c1.course=e1.course);
```

```
-- Create a view to display all the courses opted by a student along with
marks obtained.
create view CoursesOptedByStudent as
select c.cname, e.marks from Course c, Enroll e
where e.course=c.course and e.regno="01HF235";

select * from CoursesOptedByStudent;
```

```
-- Create a trigger that prevents a student from enrolling in a course if
the marks pre_requisit is less than 40
DELIMITER //
create or replace trigger PreventEnrollment
before insert on Enroll
for each row
BEGIN
        IF (new.marks<40) THEN
                signal sqlstate '45000' set message_text='Marks below
threshold';
        END IF;
END;//
DELIMITER ;

INSERT INTO Enroll VALUES
("01HF235", 002, 5, 5); -- Gives error since marks is less than 40
```

5. Consider the database schemas given below.
Write ER diagram and schema diagram. The primary keys are underlined and the data types are specified.
Create tables for the following schema listed below by properly specifying the primary keys and foreign keys.
Enter at least five tuples for each relation.
Company Database:
EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo)
DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate)
DLOCATION (DNo,DLoc)
PROJECT (PNo, PName, PLocation, DNo)
WORKS_ON (SSN, PNo, Hours)

i. **Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.**
ii. **Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise.**
iii. **Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department**
iv. **Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).**
v. **For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.**
vi. **Create a view that shows name, dept name and location of all employees.**
vii. **Create a trigger that prevents a project from being deleted if it is currently being worked by any employee.**

-- Make a list of all project numbers for projects that involve an employee whose last name is 'Scott', either as a worker or as a manager of the department that controls the project.
```
select p_no,p_name,name from Project p, Employee e where
p.d_no=e.d_no and e.name like "%Krishna";
```

-- Show the resulting salaries if every employee working on the 'IoT' project is given a 10 percent raise
```
select w.ssn,name,salary as old_salary,salary*1.1 as new_salary from
WorksOn w join Employee e where w.ssn=e.ssn and w.p_no=(select
p_no from Project where p_name="IOT") ;
```

-- Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department
```
select sum(salary) as sal_sum, max(salary) as sal_max,min(salary) as
sal_min,avg(salary) as sal_avg from Employee e join Department d on
e.d_no=d.d_no where d.dname="Accounts";
```

-- Retrieve the name of each employee who works on all the projects controlled by department number 1 (use NOT EXISTS operator).
```
select Employee.ssn,name,d_no from Employee where not exists
   (select p_no from Project p where p.d_no=1 and p_no not in
         (select p_no from WorksOn w where w.ssn=Employee.ssn));
```

-- *For each department that has more than one employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.*

```
select d.d_no, count(*) from Department d join Employee e on
e.d_no=d.d_no where salary>600000 group by d.d_no having count(*) >1;
```

-- *Create a view that shows name, dept name and location of all employees*

```
create view emp_details as
select name,dname,d_loc from Employee e join Department d on
e.d_no=d.d_no join DLocation dl on d.d_no=dl.d_no;

select * from emp_details;
```

-- *Create a trigger that prevents a project from being deleted if it is currently being worked by any employee.*

```
DELIMITER //
create trigger PreventDelete
before delete on Project
for each row
BEGIN
        IF EXISTS (select * from WorksOn where p_no=old.p_no)
THEN
                signal sqlstate '45000' set message_text='This project has
an employee assigned';
        END IF;
END; //
DELIMITER ;

delete from Project where p_no=241563; -- Will give error
```