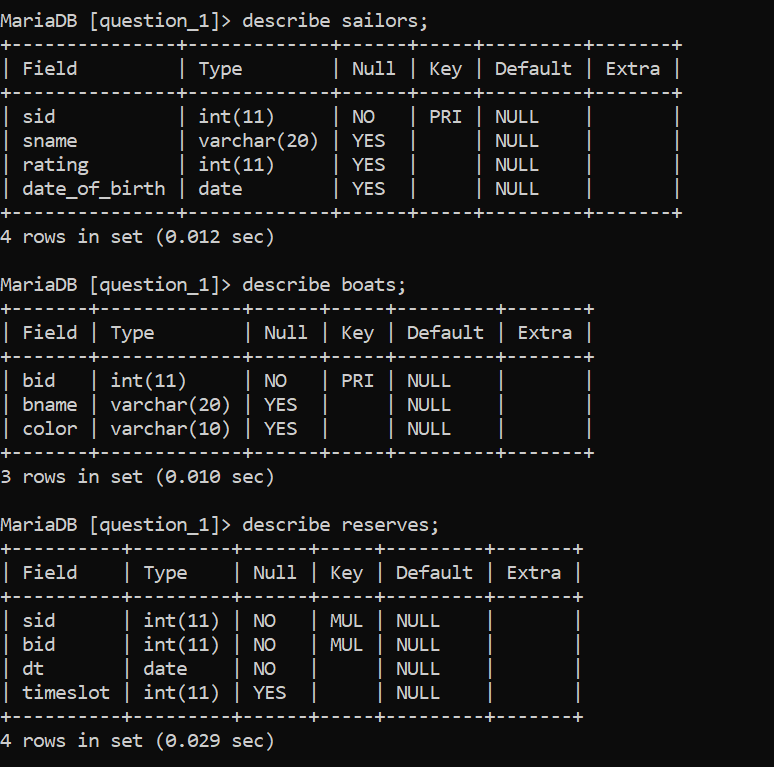
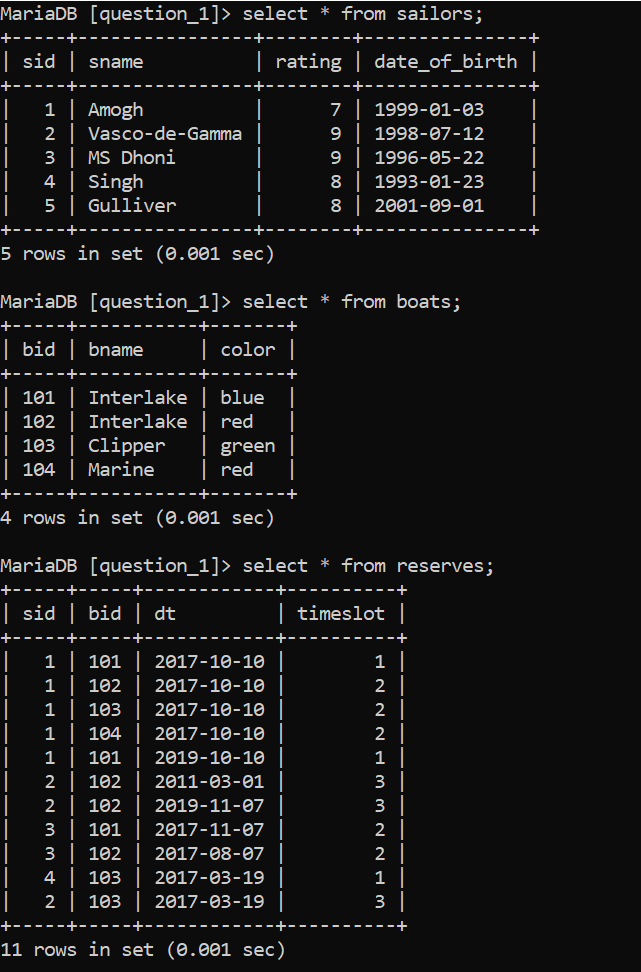
**DBMS ASSIGNMENTS**

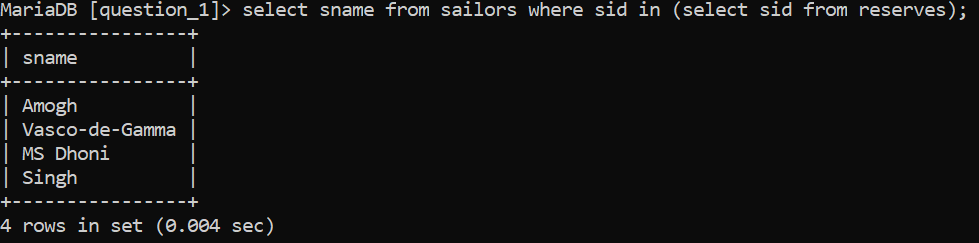
**NAME: Amogh Garg  
ROLL NUMBER: 2020UCO1688  
SECTION: COE-3**

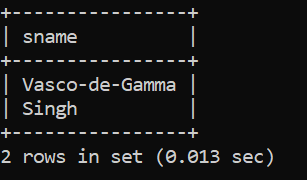
**Q 1: Consider the following relational schema SAILORS (sid, sname, rating, date\_of\_birth) BOATS (bid, bname, color) RESERVES (sid, bid, date, time slot)   
Write the following queries in SQL and relational algebra   
a) Find sailors who’ve reserved at least one boat   
b) Find names of sailors who’ve reserved a red or a green boat in the month of March.   
c) Find names of sailors who’ve reserved a red and a green boat   
d) Find sid of sailors who have not reserved a boat after Jan 2018.   
e) Find sailors whose rating is greater than that of all the sailors named “John”   
f) Find sailors who’ve reserved all boats   
g) Find name and age of the oldest sailor(s)   
h) Find the age of the youngest sailor for each rating with at least 2 such sailors**

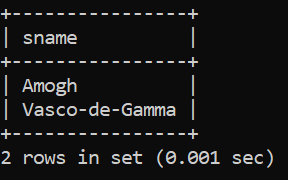
**CREATION OF THE TABLE:**create table sailors (sid int primary key, sname varchar (20), rating int, date\_of\_birth date);   
create table boats (bid int primary key, bname varchar (20), color varchar (10));   
create table reserves (sid int not null, bid int not null, dt date not null, timeslot int, foreign key (sid) references sailors(sid), foreign key (bid) references boats (bid));

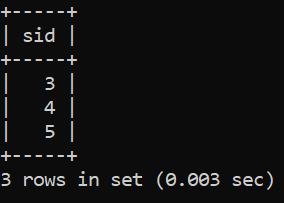


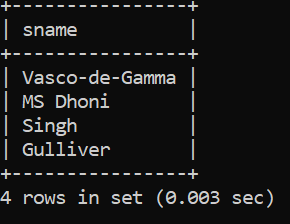
**INSERTION OF VALUES:**insert into sailors values (1, 'Amogh', 7, "1999-01-03");   
insert into sailors values (2, 'Vasco-de-Gamma', 9, "1998-07-12");  
insert into sailors values (3, 'MS Dhoni', 9, "1996-05-22");  
insert into sailors values (4, 'Singh', 8, "1993-01-23");   
insert into sailors values (5, 'Gulliver',8,"2001-09-01");   
insert into Boats values (101, 'Interlake', 'blue');   
insert into Boats values (102, 'Interlake', 'red');   
insert into Boats values (103, 'Clipper', 'green');   
insert into Boats values (104, 'Marine', 'red');   
insert into Reserves values (1, 101, '2017-10-10',1);   
insert into Reserves values (1, 102, '2017-10-10',2);   
insert into Reserves values (1, 103, '2017-10-10',2);   
insert into Reserves values (1, 104, '2017-10-10',2);  
insert into Reserves values (1, 101, '2019-10-10',1);   
insert into Reserves values (2, 102, '2011-03-01',3);   
insert into Reserves values (2, 102, '2019-11-07',3);   
insert into Reserves values (3, 101, '2017-11-07',2);   
insert into Reserves values (3, 102, '2017-08-07',2);   
insert into Reserves values (4, 103, '2017-03-19',1);   
insert into Reserves values (2, 103, '2017-03-19',3);  


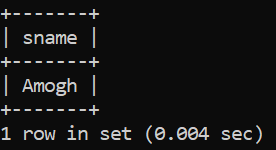
**QUERIES:** a) select sname from sailors where sid in (select sid from reserves);  


b) select sname from sailors where sid in (select r.sid from boats b, reserves r where r.bid = b.bid AND b.color = "red" and (select extract(month from r.dt)="03") union select r2.sid from boats b2, reserves r2 where r2.bid = b2.bid AND b2.color = "green" and (select extract(month from r2.dt)="03") );  


c) select distinct S1.sname from sailors S1, reserves R1, boats B1, reserves R2, boats B2 where S1.sid=R1.sid and R1.bid=B1.bid and S1.sid=R2.sid and R2.bid=B2.bid and B1.color="red" and B2.color="green";  


d) select sid from sailors where sid not in (select sid from reserves where dt>="2018-01-01");  


e) select sname from sailors where rating > all (select rating from sailors where sname="Amogh");  


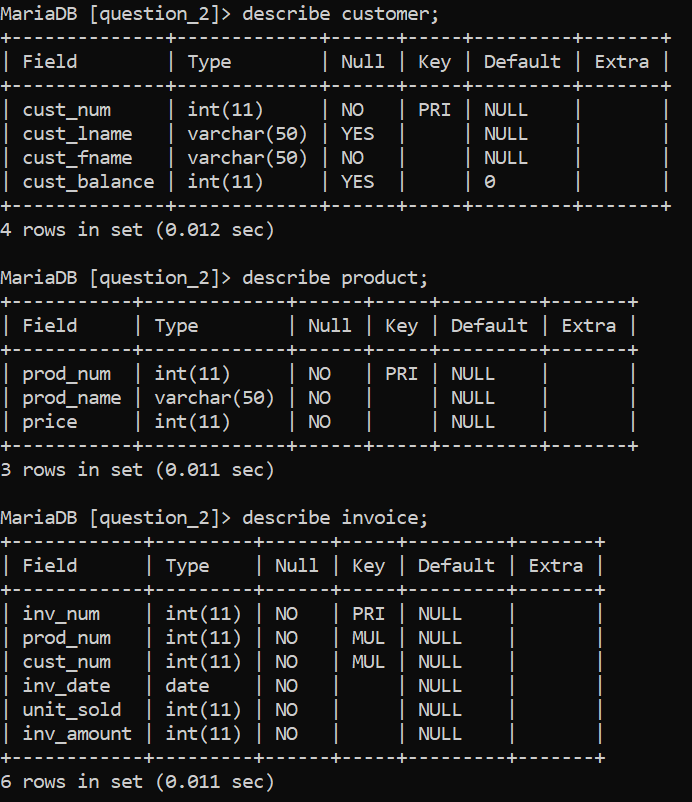
f) select sname from sailors s where not exists (select \* from boats b where not exists ( select \* from reserves r where r.sid=s.sid AND r.bid=b.bid));  


g) select sname,TIMESTAMPDIFF(YEAR,date\_of\_birth,"2021-11-16") as age from sailors where date\_of\_birth >=all(select date\_of\_birth from sailors);

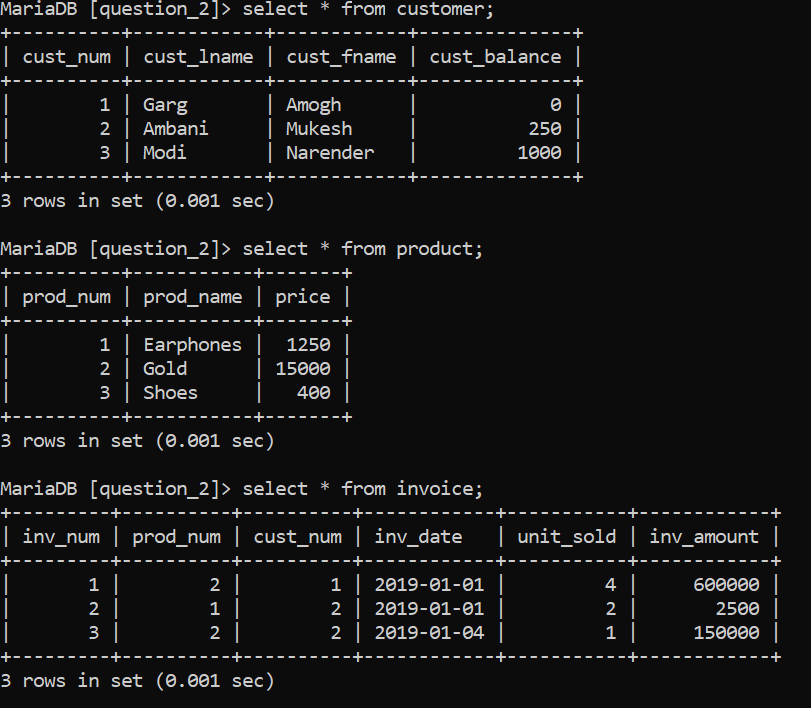
h) select rating,min(TIMESTAMPDIFF(YEAR,date\_of\_birth,"2021-11-16")) as minage from sailors group by rating having count(\*)>1;  
--------------------------------------------------------------------------------------------------------------------------------------

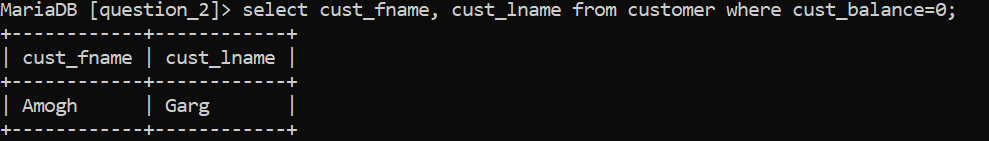
**Q2. Consider the following relational schema: CUSTOMER (cust\_num, cust\_lname , cust\_fname, cust\_balance); PRODUCT (prod\_num, prod\_name, price) INVOICE (inv\_num, prod\_num, cust\_num, inv\_date ,unit\_sold, inv\_amount);   
Write SQL queries and relational algebraic expression for the following   
a) Find the names of the customer who have purchased no item. Set default value of Cust\_balance as 0 for such customers.   
b) Write the trigger to update the CUST\_BALANCE in the CUSTOMER table when a new invoice record is entered for the customer.   
c) Find the customers who have purchased more than three units of a product on a day.  
d) Write a query to illustrate Left Outer, Right Outer and Full Outer Join.   
e) Count number of products sold on each date.   
f) As soon as customer balance becomes greater than Rs. 100,000, copy the customer\_num in new table called “GOLD\_CUSTOMER”   
g) Add a new attribute CUST\_DOB in customer table**

**CREATION OF TABLES:**create table customer (cust\_num int, cust\_lname varchar(50), cust\_fname varchar(50) not null, cust\_balance int default 0, primary key (cust\_num));  
create table product (prod\_num int, prod\_name varchar(50) not null, price int not null, primary key(prod\_num));  
create table invoice (inv\_num int, prod\_num int not null, cust\_num int not null, inv\_date date not null, unit\_sold int not null, inv\_amount int not null, primary key(inv\_num), foreign key(prod\_num) references product(prod\_num), foreign key(cust\_num) references customer(cust\_num), check(unit\_sold>0));

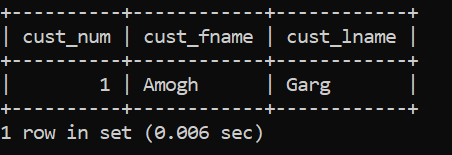


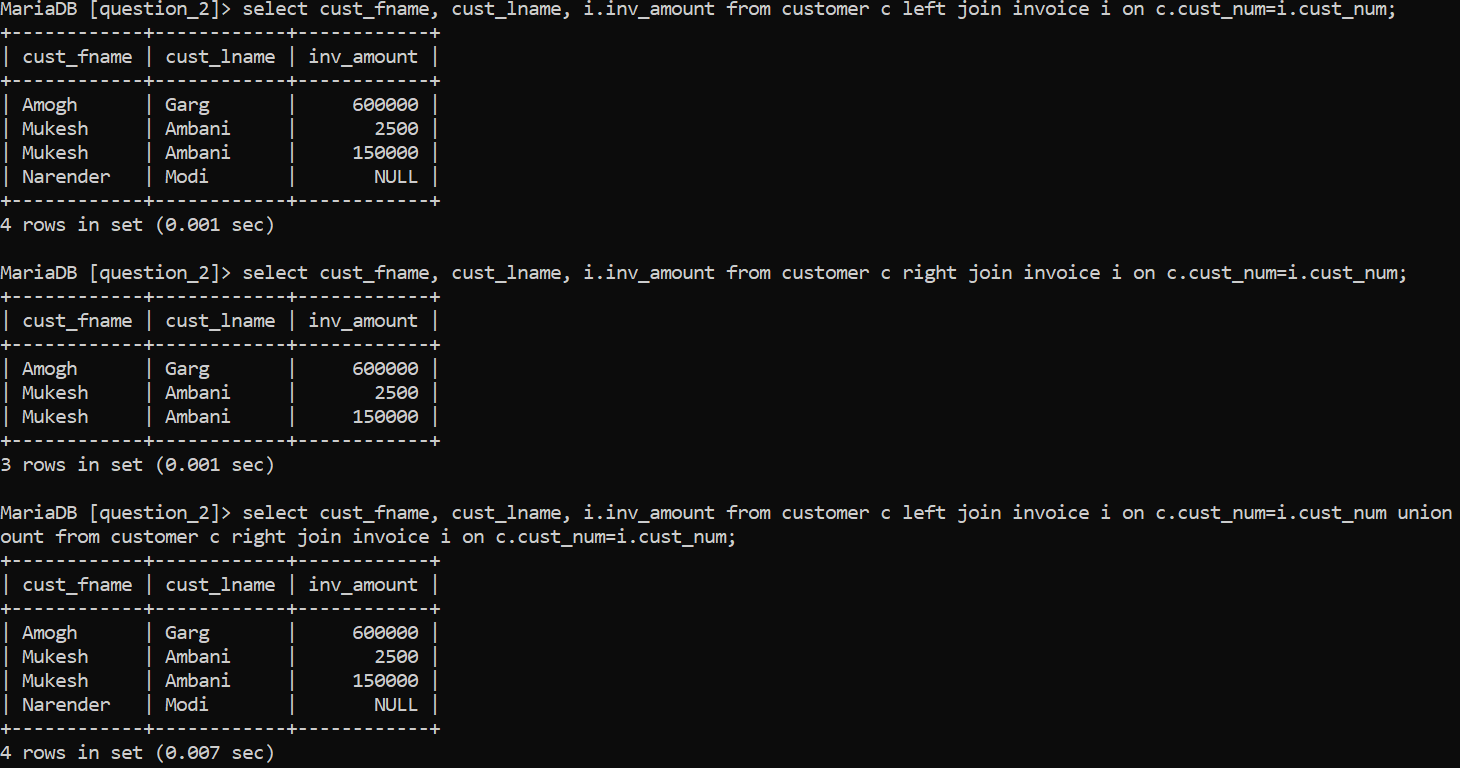
**INSERTION OF DATA:**

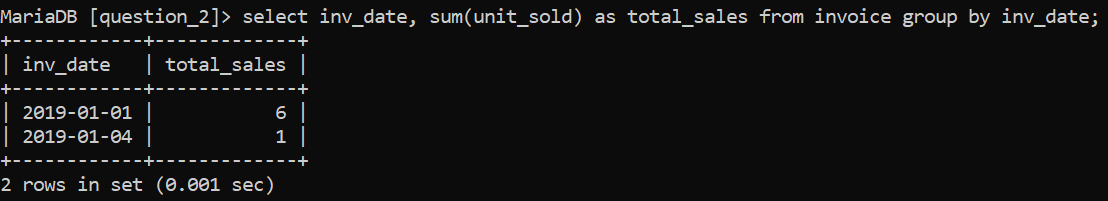
Insert into customer (cust\_num, cust\_lname, cust\_fname, cust\_balance) values (1, ‘Garg’, ‘Amogh’, 0), (2, ‘Ambani’, ‘Mukesh’, 250), (3, ‘Modi’, ‘Narender’, 1000);  
Insert into product (prod\_num,prod\_name,price) values (2, ‘Gold’, 15000), (1, ‘Earphones’, 1250), (3, ‘Shoes’, 400);  
Insert into invoice values (1,2,1,'2019-01-01',4,600000),(2,1,2,'2019-01-01',2,2500), (3,2,2,'2019-01-04',1,150000);  


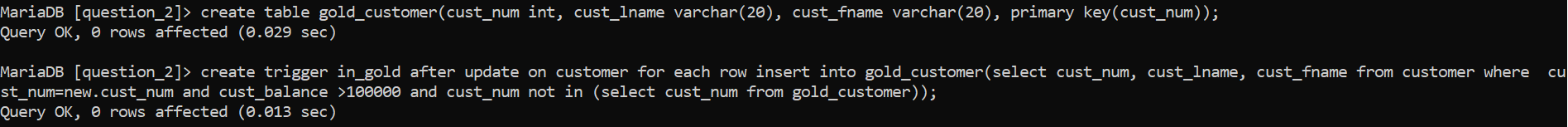
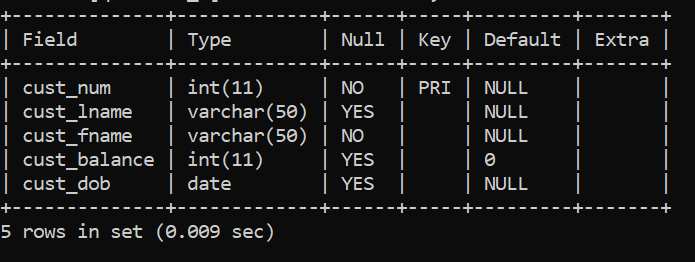
**QUERIES:** a) select cust\_fname, cust\_lname from customer where cust\_balance=0;  


b) create trigger upd\_cust before insert on invoice for each row update customer c set c.cust\_balance=c.cust\_balance+new.inv\_amount where c.cust\_num=new.cust\_num;

c) select cust\_num, cust\_fname, cust\_lname from customer where cust\_num in (select cust\_num from invoice group by cust\_num, inv\_date, prod\_num having sum(unit\_sold)>3);   


d) select cust\_fname, cust\_lname, i.inv\_amount from customer c left join invoice i on c.cust\_num=i.cust\_num;  
select cust\_fname, cust\_lname, i.inv\_amount from customer c right join invoice i on c.cust\_num=i.cust\_num;  
select cust\_fname, cust\_lname, i.inv\_amount from customer c left join invoice i on c.cust\_num=i.cust\_num union select cust\_fname, cust\_lname, i.inv\_amount from customer c right join invoice i on c.cust\_num=i.cust\_num;  


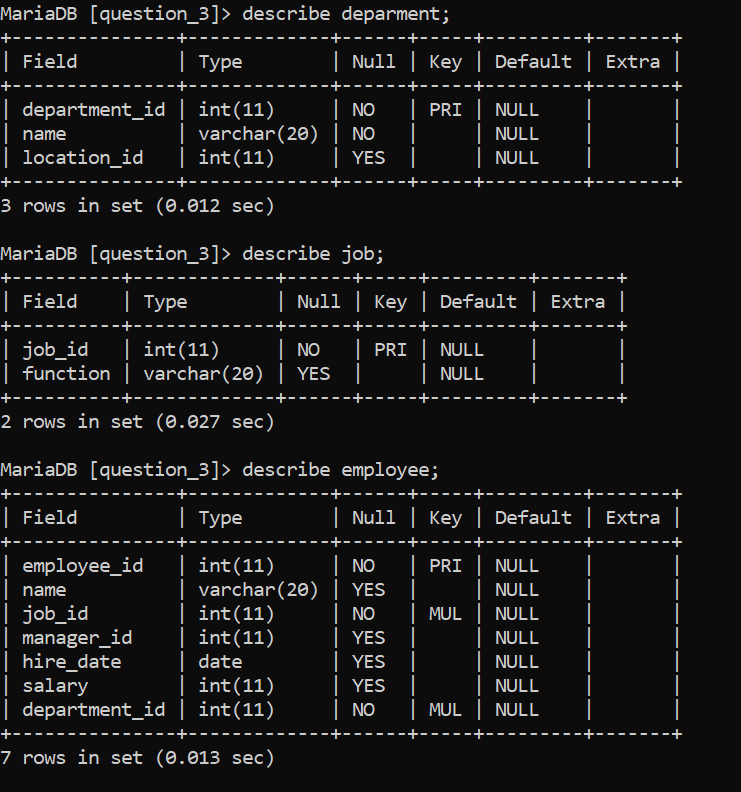
e) select inv\_date, sum(unit\_sold) as total\_sales from invoice group by inv\_date;  


f) create table gold\_customer(cust\_num int, cust\_lname varchar(20), cust\_fname varchar(20), primary key(cust\_num));  
create trigger in\_gold after update on customer for each row insert into gold\_customer(select cust\_num, cust\_lname, cust\_fname from customer where cust\_num=new.cust\_num and cust\_balance >100000 and cust\_num not in (select cust\_num from gold\_customer));  
  
g) alter table customer add column cust\_dob date;  


--------------------------------------------------------------------------------------------------------------------------------------

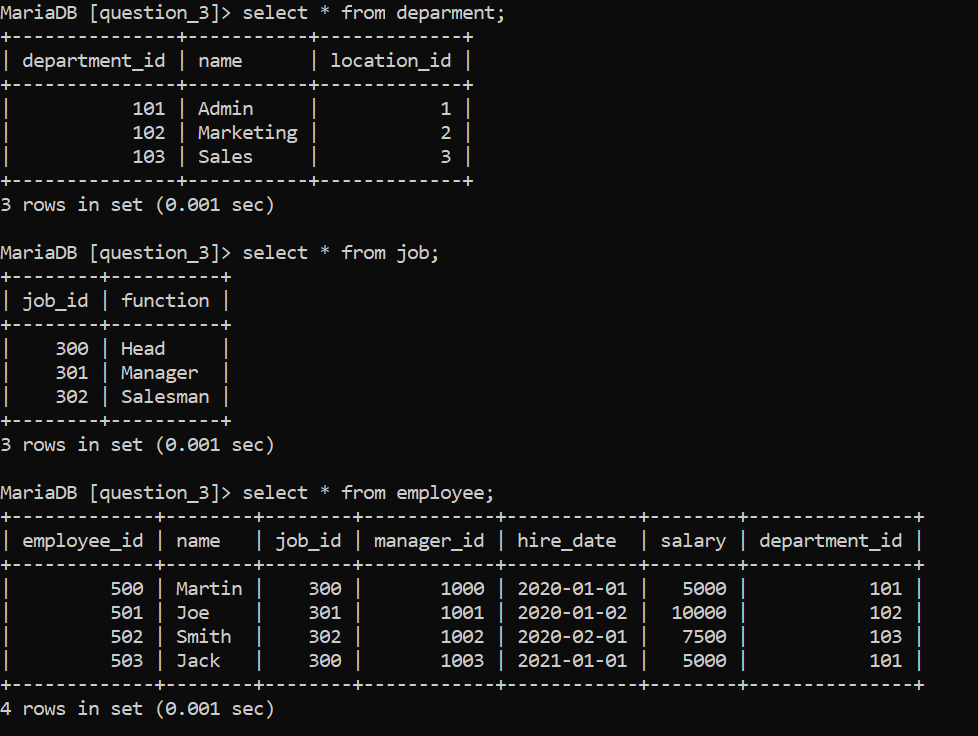
**Q.3 Consider the following relational schema: DEPARTMENT (Department\_ID, Name, Location\_ID) JOB (Job\_ID, Function) EMPLOYEE (Employee\_ID, name, DOB, Job\_ID, Manager\_ID, Hire\_Date, Salary, department\_id)  
 Answer the following queries using SQL and relational algebra:   
1) Write a query to count number of employees who joined in March 2015   
2) Display the Nth highest salary drawing employee details.   
3) Find the budget (total salary) of each department.   
4) Find the department with maximum budget.   
5) Create a view to show number of employees working in Delhi and update it automatically when the database is modified.   
6) Write a trigger to ensure that no employee of age less than 25 can be inserted in the database.**

**CREATION OF TABLE:**create table deparment (department\_id int, name varchar(20) not null, location\_id int, primary key (department\_id));  
create table job (job\_id int primary key, function varchar(20));  
create table employee (employee\_id int primary key, name varchar(20), job\_id int not null, manager\_id int, hire\_date date, salary int, department\_id int not null, foreign key (job\_id) references job (job\_id), foreign key (department\_id) references deparment (department\_id));

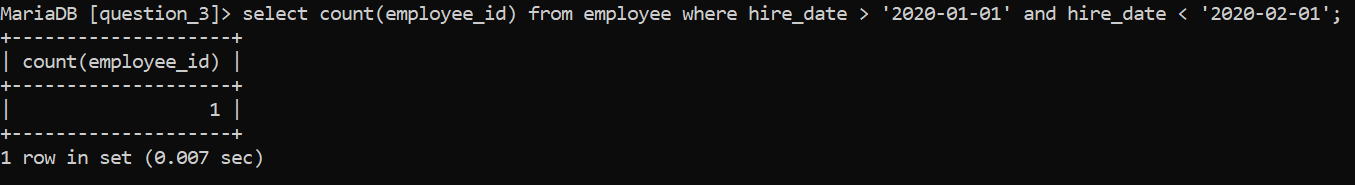


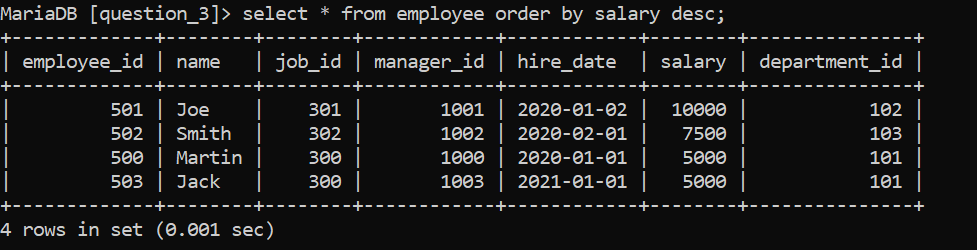
**INSERTION OF DATA:**

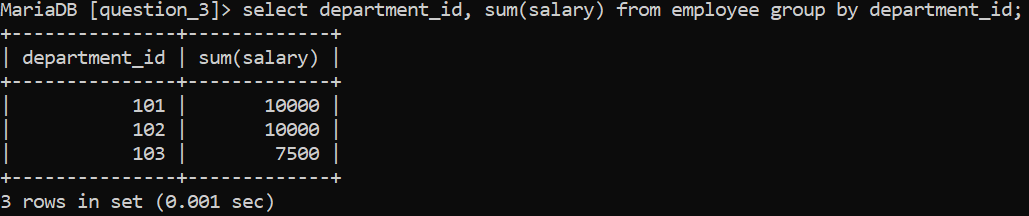
insert into deparment values (101, ‘Admin’, 1), (102, ‘Marketing’, 2), (103, ‘Sales’, 3);  
insert into job values (300, ‘Head’), (301, ‘Manager’), (302, ‘Salesman’);  
insert into employee values (500, ‘Martin’, 300, 1000, ‘2020-01-01’, 5000, 101), (501, ‘Joe’, 301, 1001, ‘2020-01-02’, 10000, 102), (502, ‘Smith’, 302, 1002, ‘2020-02-01’, 7500, 103), (503, ‘Jack’, 300, 1003, ‘2021-01-01’, 5000, 101);

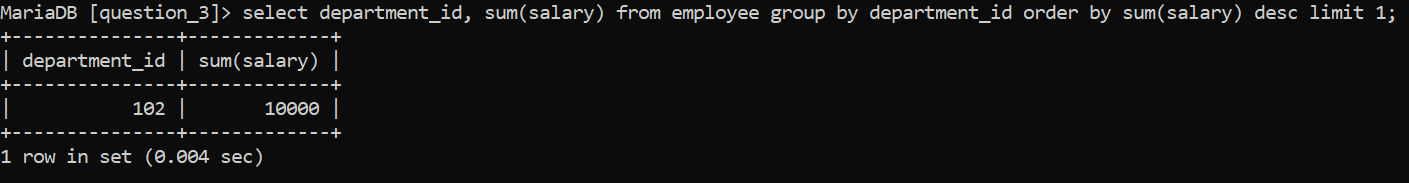


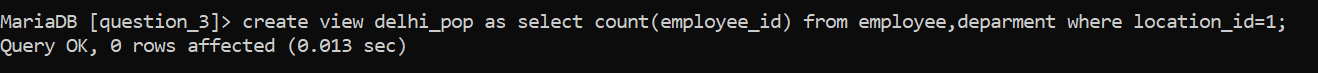
**QUERIES:**

1. select count(employee\_id) from employee where hire\_date > ‘2020-01-01’ and hire\_date < ‘2020-02-01’;  


2. select \* from employee order by salary desc;  


3.select department\_id, sum(salary) from employee group by department\_id;  


4. select department\_id, sum(salary) from employee group by department\_id order by sum(salary) desc limit 1;  


5. create view delhi\_pop as select count(employee\_id) from employee,deparment where deparment.location\_id=1;  


6. create trigger check\_age before insert on employee for each row begin if new.dob > 1993-01-01 then signal sqlstate ‘4500’ set message\_text = ‘Age must be atleast 25 years!’;