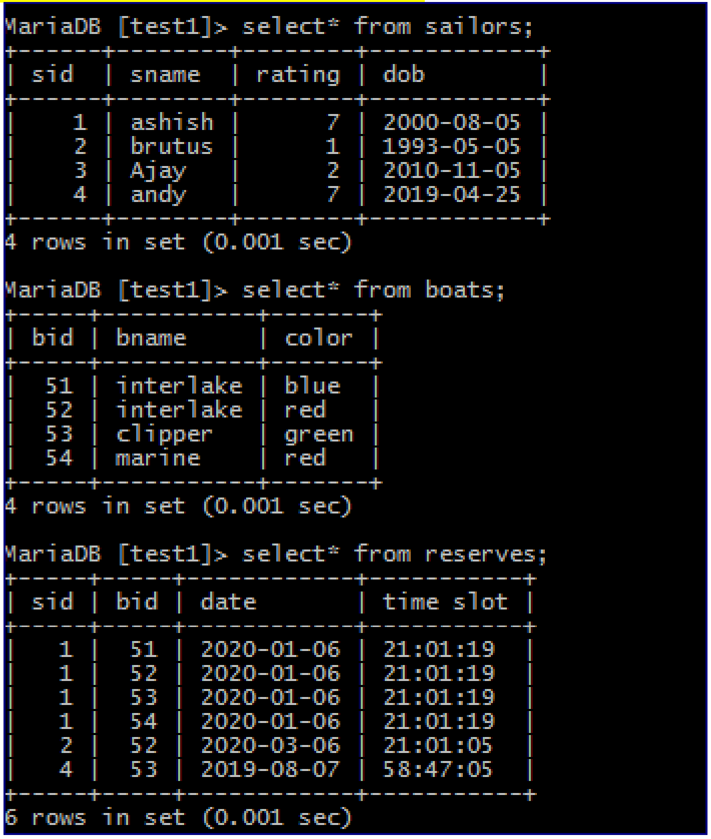
DBMS PRACTICAL
RELATIONAL ALGEBRA AND SQL

ASHISH KUMAR
2019UCO1518

QUES 1:

SAILORS (sid, sname, rating, date_of_birth)
BOATS (bid, bname, color)
RESERVES (sid, bid, date, time slot)

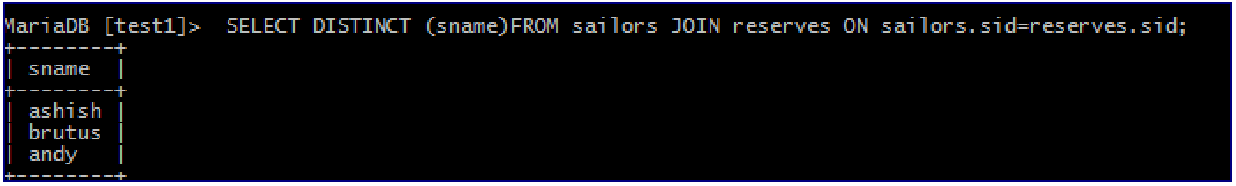


```
MariaDB [test1]> select* from sailors;
+------+--------+--------+------------+
| sid  | sname  | rating | dob        |
+------+--------+--------+------------+
|    1 | ashish |      7 | 2000-08-05 |
|    2 | brutus |      1 | 1993-05-05 |
|    3 | Ajay   |      2 | 2010-11-05 |
|    4 | andy   |      7 | 2019-04-25 |
+------+--------+--------+------------+
4 rows in set (0.001 sec)

MariaDB [test1]> select* from boats;
+-----+-----------+-------+
| bid | bname     | color |
+-----+-----------+-------+
|  51 | interlake | blue  |
|  52 | interlake | red   |
|  53 | clipper   | green |
|  54 | marine    | red   |
+-----+-----------+-------+
4 rows in set (0.001 sec)

MariaDB [test1]> select* from reserves;
+-----+-----+------------+-----------+
| sid | bid | date       | time slot |
+-----+-----+------------+-----------+
|   1 |  51 | 2020-01-06 | 21:01:19  |
|   1 |  52 | 2020-01-06 | 21:01:19  |
|   1 |  53 | 2020-01-06 | 21:01:19  |
|   1 |  54 | 2020-01-06 | 21:01:19  |
|   2 |  52 | 2020-03-06 | 21:01:05  |
|   4 |  53 | 2019-08-07 | 58:47:05  |
+-----+-----+------------+-----------+
6 rows in set (0.001 sec)
```

a) Find sailors who've reserved at least one boat

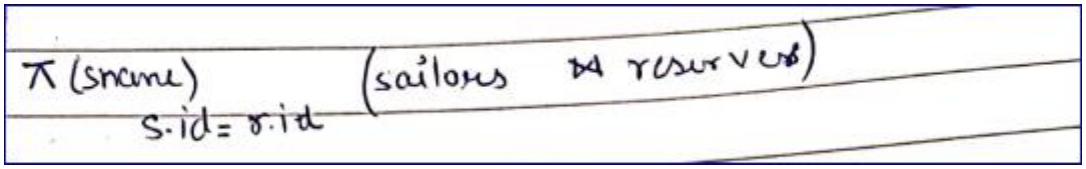
SQL QUERY:

SELECT DISTINCT (sname)FROM sailors JOIN reserves ON sailors.sid=reserves.sid;



```
MariaDB [test1]>  SELECT DISTINCT (sname)FROM sailors JOIN reserves ON sailors.sid=reserves.sid;
+--------+
| sname  |
+--------+
| ashish |
| brutus |
| andy   |
+--------+
```

RELATIONAL ALGEBRA:

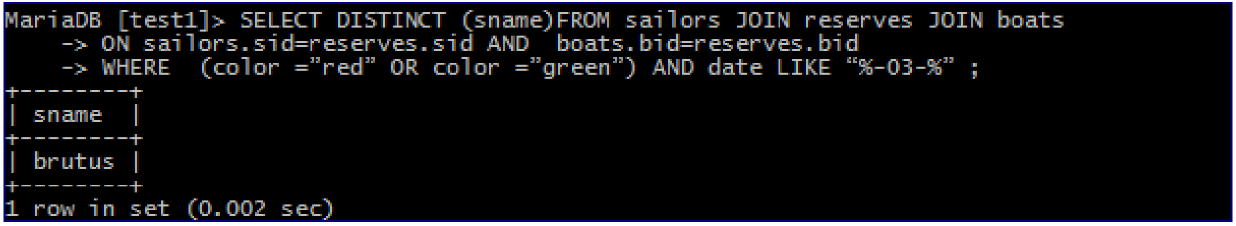


$\pi_{(sname)\; s.id = r.id} (sailors \bowtie resurvers)$

b) Find names of sailors who’ve reserved a red or a green boat in the month of March.

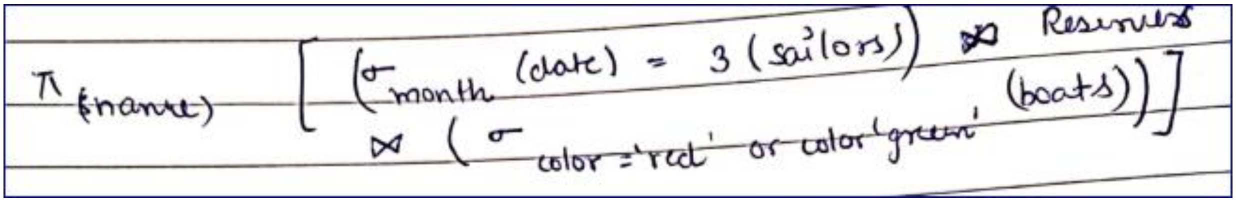
SQL QUERY:

SELECT DISTINCT (sname)FROM sailors JOIN reserves JOIN boats
ON sailors.sid=reserves.sid AND boats.bid=reserves.bid
WHERE (color =”red” OR color =”green”) AND date LIKE “%-03-%” ;



```
MariaDB [test1]> SELECT DISTINCT (sname)FROM sailors JOIN reserves JOIN boats
    -> ON sailors.sid=reserves.sid AND  boats.bid=reserves.bid
    -> WHERE  (color =”red” OR color =”green”) AND date LIKE “%-03-%” ;
+--------+
| sname  |
+--------+
| brutus |
+--------+
1 row in set (0.002 sec)
```

RELATIONAL ALGEBRA:

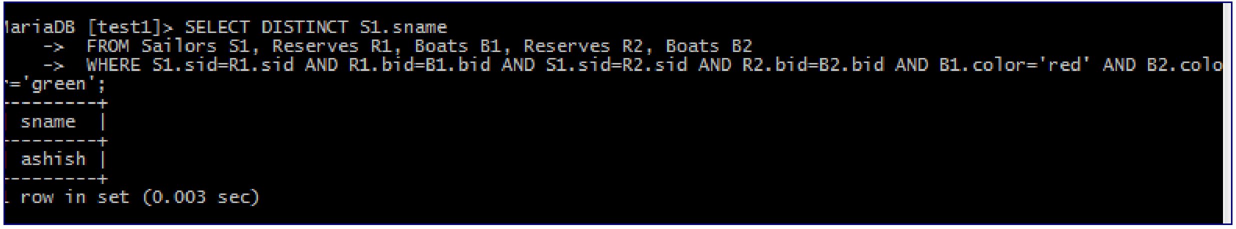


$\pi_{(sname)} \left[\left(\sigma_{month} (date) = 3 (sailors) \right) \bowtie Reserves \bowtie \left(\sigma_{color='red' \text{ or } color 'green'} (boats) \right) \right]$

c) Find names of sailors who’ve reserved a red and a green boat

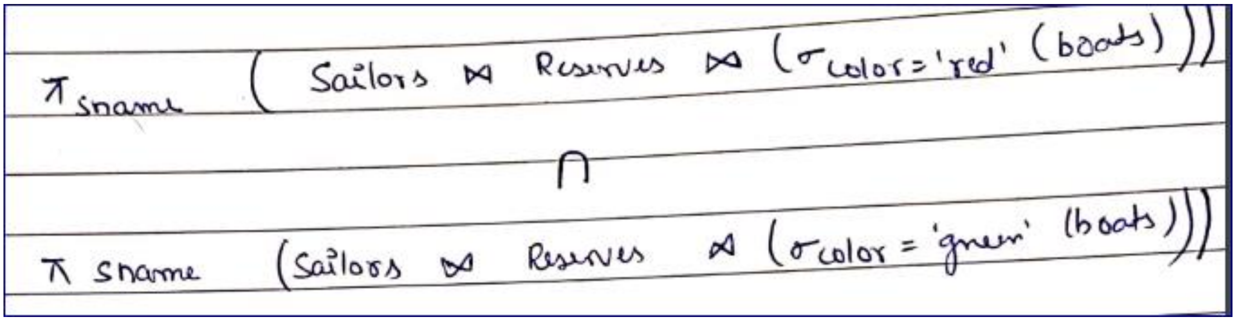
SQL QUERY:

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
.color='red' AND B2.color='green';



```
lariaDB [test1]> 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.colo
='green';
---------+
 sname  |
---------+
 ashish |
---------+
 row in set (0.003 sec)
```

RELATIONAL ALGEBRA:



$$\pi_{sname} \left(Sailors \bowtie Reserves \bowtie \left(\sigma_{color='red'} (boats) \right) \right)$$

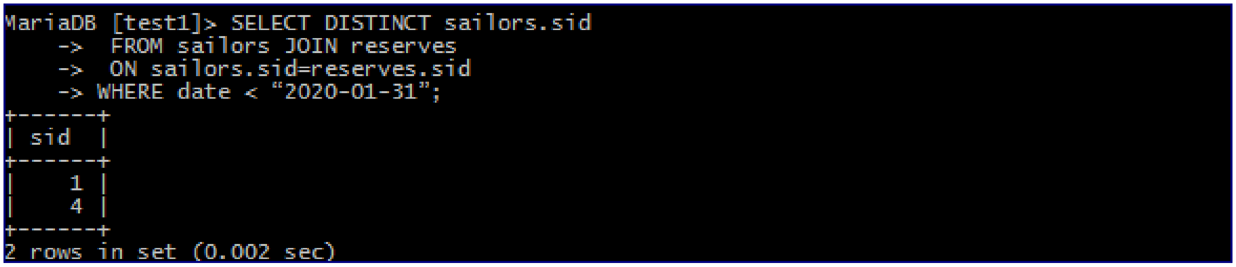
$$\cap$$

$$\pi_{sname} \left(Sailors \bowtie Reserves \bowtie \left(\sigma_{color='green'} (boats) \right) \right)$$

d) Find sid of sailors who have not reserved a boat after Jan 2020.

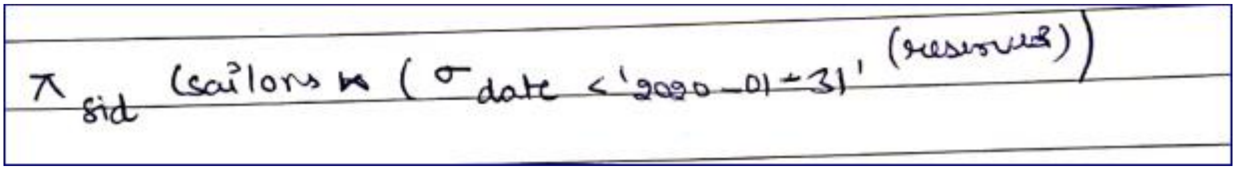
SQL QUERY:

```
SELECT DISTINCT sailors.sid
FROM sailors JOIN reserves
ON sailors.sid=reserves.sid
WHERE date < “2020-01-31”;
```



```
MariaDB [test1]> SELECT DISTINCT sailors.sid
    ->  FROM sailors JOIN reserves
    ->  ON sailors.sid=reserves.sid
    -> WHERE date < “2020-01-31”;
+-------+
| sid   |
+-------+
|     1 |
|     4 |
+-------+
2 rows in set (0.002 sec)
```

RELATIONAL ALGEBRA:

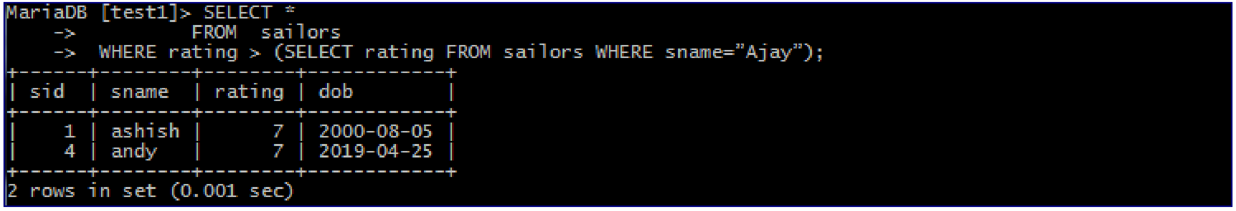


$$\pi_{sid} \left(sailors \bowtie \left(\sigma_{date < '2020-01-31'} (reserves) \right) \right)$$

e) Find sailors whose rating is greater than that of all the sailors named “Ajay”

SQL QUERY:

```
SELECT *
FROM  sailors
WHERE rating > (SELECT rating FROM sailors WHERE sname=”Ajay”);
```



```
MariaDB [test1]> SELECT *
    ->          FROM  sailors
    ->  WHERE rating > (SELECT rating FROM sailors WHERE sname=”Ajay”);
+------+--------+--------+------------+
| sid  | sname  | rating | dob        |
+------+--------+--------+------------+
|    1 | ashish |      7 | 2000-08-05 |
|    4 | andy   |      7 | 2019-04-25 |
+------+--------+--------+------------+
2 rows in set (0.001 sec)
```

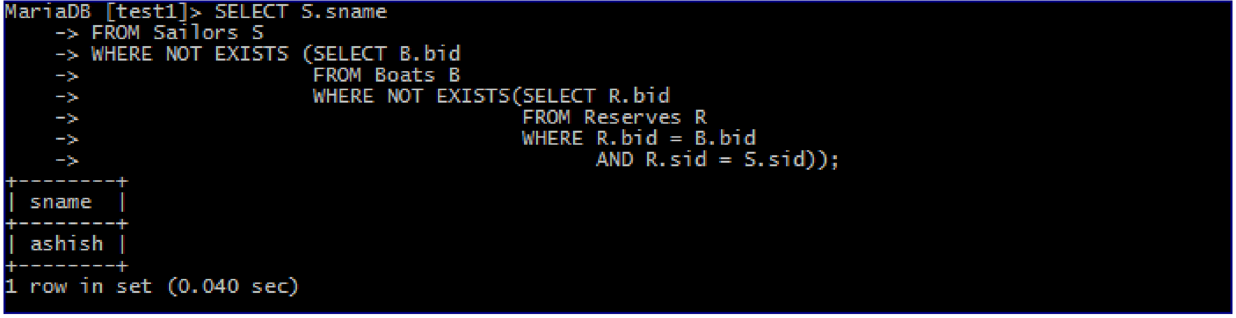
RELATIONAL ALGEBRA:

$$\pi_{sid}(\text{sailors}) - \pi_{S2.sid}\left(\sigma_{S_2.rating < S.rating}\left(\rho_S(\text{sailors}) \bowtie \rho_{S2}(\text{sailors})\right)\right)$$

f) Find sailors who’ve reserved all boats

SQL QUERY:

```
SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (SELECT B.bid
            FROM Boats B
            WHERE NOT EXISTS(SELECT R.bid
                        FROM Reserves R
                        WHERE R.bid = B.bid
                            AND R.sid = S.sid));
```



```
MariaDB [test1]> SELECT S.sname
    -> FROM Sailors S
    -> WHERE NOT EXISTS (SELECT B.bid
    ->                   FROM Boats B
    ->                   WHERE NOT EXISTS(SELECT R.bid
    ->                                    FROM Reserves R
    ->                                    WHERE R.bid = B.bid
    ->                                          AND R.sid = S.sid));
+--------+
| sname  |
+--------+
| ashish |
+--------+
1 row in set (0.040 sec)
```

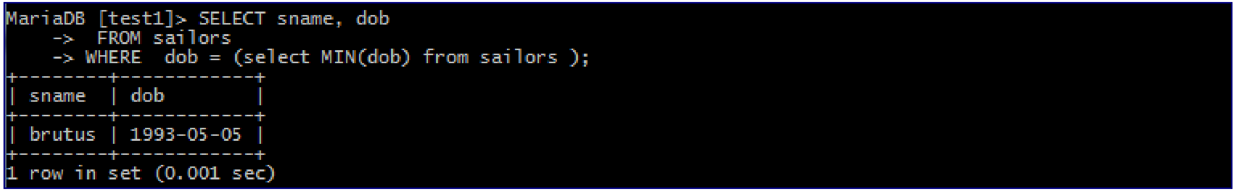
RELATIONAL ALGEBRA:

$$\pi_{sname}\left(\left(\pi_{sid,bid}(\text{reserves}) \div \pi_{bid}(\text{boat})\right) \bowtie \text{sailors}\right)$$

g) Find name and age of the oldest sailor(s)

SQL QUERY:

```
SELECT sname, dob
FROM sailors
WHERE  dob = (select MIN(dob) from sailors );
```



```
MariaDB [test1]> SELECT sname, dob
    ->  FROM sailors
    -> WHERE  dob = (select MIN(dob) from sailors );
+--------+------------+
| sname  | dob        |
+--------+------------+
| brutus | 1993-05-05 |
+--------+------------+
1 row in set (0.001 sec)
```

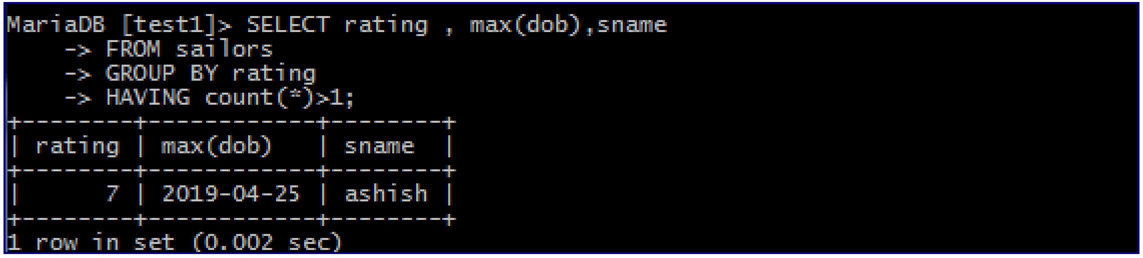
RELATIONAL ALGEBRA:

$$\pi_{sname, dob}((\pi_{sid}(seilors) - \pi_{S2.sid}(\sigma_{S_2.dob < S.dob}(\rho_{S_2}(sailors) \times \rho_{S1}(sailors))))) \bowtie sailors$$

h) Find the age of the youngest sailor for each rating with at least 2 such sailors

SQL QUERY:

```
SELECT rating , max(dob),sname
FROM sailors
GROUP BY rating
HAVING count(*)>1;
```



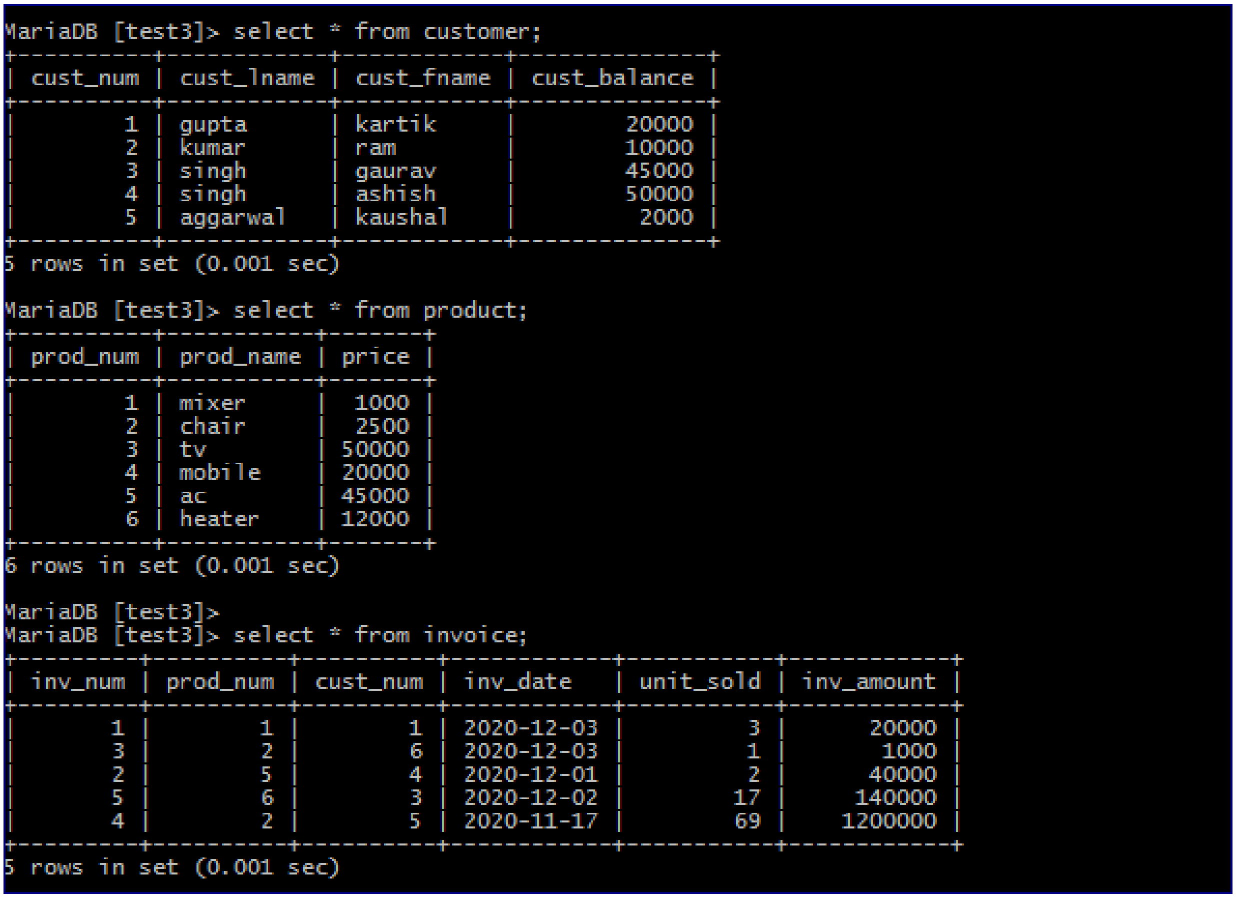
```
MariaDB [test1]> SELECT rating , max(dob),sname
    -> FROM sailors
    -> GROUP BY rating
    -> HAVING count(*)>1;
+--------+------------+--------+
| rating | max(dob)   | sname  |
+--------+------------+--------+
|      7 | 2019-04-25 | ashish |
+--------+------------+--------+
1 row in set (0.002 sec)
```

RELATIONAL ALGEBRA:

$$\pi_{rating, sname}(\sigma_{no\ of\ sailor > 1}(\rho_{r}(rating, no\ of\ sailors)\ {}_{rating}\mathcal{F}_{(count(sid), min(dob))}(sailor)))$$

Ques 2

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);



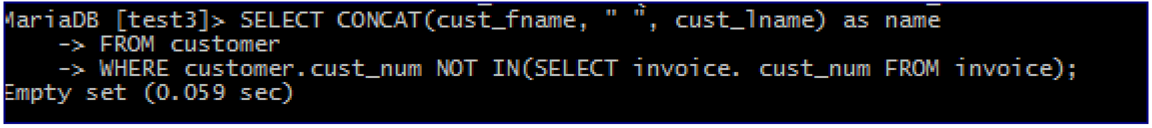
```
MariaDB [test3]> select * from customer;
+----------+------------+------------+--------------+
| cust_num | cust_lname | cust_fname | cust_balance |
+----------+------------+------------+--------------+
|        1 | gupta      | kartik     |        20000 |
|        2 | kumar      | ram        |        10000 |
|        3 | singh      | gaurav     |        45000 |
|        4 | singh      | ashish     |        50000 |
|        5 | aggarwal   | kaushal    |         2000 |
+----------+------------+------------+--------------+
5 rows in set (0.001 sec)

MariaDB [test3]> select * from product;
+----------+-----------+-------+
| prod_num | prod_name | price |
+----------+-----------+-------+
|        1 | mixer     |  1000 |
|        2 | chair     |  2500 |
|        3 | tv        | 50000 |
|        4 | mobile    | 20000 |
|        5 | ac        | 45000 |
|        6 | heater    | 12000 |
+----------+-----------+-------+
6 rows in set (0.001 sec)

MariaDB [test3]>
MariaDB [test3]> select * from invoice;
+---------+----------+----------+------------+-----------+------------+
| inv_num | prod_num | cust_num | inv_date   | unit_sold | inv_amount |
+---------+----------+----------+------------+-----------+------------+
|       1 |        1 |        1 | 2020-12-03 |         3 |      20000 |
|       3 |        2 |        6 | 2020-12-03 |         1 |       1000 |
|       2 |        5 |        4 | 2020-12-01 |         2 |      40000 |
|       5 |        6 |        3 | 2020-12-02 |        17 |     140000 |
|       4 |        2 |        5 | 2020-11-17 |        69 |    1200000 |
+---------+----------+----------+------------+-----------+------------+
5 rows in set (0.001 sec)
```

a) Find the names of the customer who have purchased no item. Set default value of Cust_balance as 0 for such customers.

SQL QUERY:

Find names:

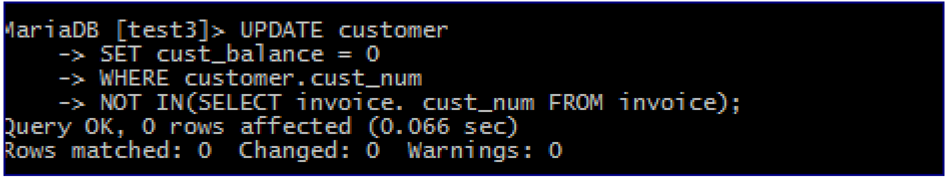
```
SELECT CONCAT(cust_fname, " ", cust_lname) as name
  -> FROM customer
   -> WHERE customer.cust_num NOT IN(SELECT invoice. cust_num FROM invoice);
```



```
MariaDB [test3]> SELECT CONCAT(cust_fname, " ", cust_lname) as name
    -> FROM customer
    -> WHERE customer.cust_num NOT IN(SELECT invoice. cust_num FROM invoice);
Empty set (0.059 sec)
```

Update:

```
UPDATE customer
   -> SET cust_balance = 0
   -> WHERE customer.cust_num NOT IN(SELECT invoice. cust_num FROM invoice);
```

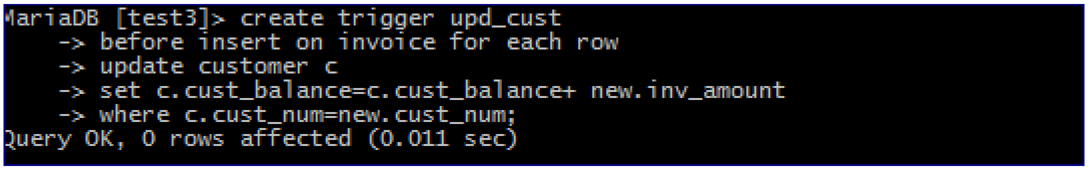


```
MariaDB [test3]> UPDATE customer
    -> SET cust_balance = 0
    -> WHERE customer.cust_num
    -> NOT IN(SELECT invoice. cust_num FROM invoice);
Query OK, 0 rows affected (0.066 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

RELATIONAL ALGEBRA:

$$\pi_{cust\_fname + " " + cust\_lname}\left(\sigma_{cust\_balance=0}(customer)\right)$$

b) Write the trigger to update the CUST_BALANCE in the CUSTOMER table when a new invoice record is entered for the customer.



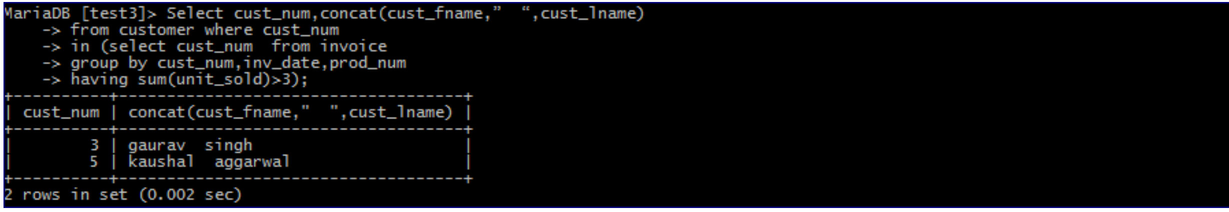
```
MariaDB [test3]> create trigger upd_cust
    -> before insert on invoice for each row
    -> update customer c
    -> set c.cust_balance=c.cust_balance+ new.inv_amount
    -> where c.cust_num=new.cust_num;
Query OK, 0 rows affected (0.011 sec)
```

Created a trigger that would update values in customer table

c) Find the customers who have purchased more than three units of a product on a day.

SQL QUERY:

```
Select cust_num,concat(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);
```



```
MariaDB [test3]> Select cust_num,concat(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);
+----------+------------------------------------+
| cust_num | concat(cust_fname,"  ",cust_lname) |
+----------+------------------------------------+
|        3 | gaurav  singh                      |
|        5 | kaushal  aggarwal                  |
+----------+------------------------------------+
2 rows in set (0.002 sec)
```

RELATIONAL ALGEBRA:

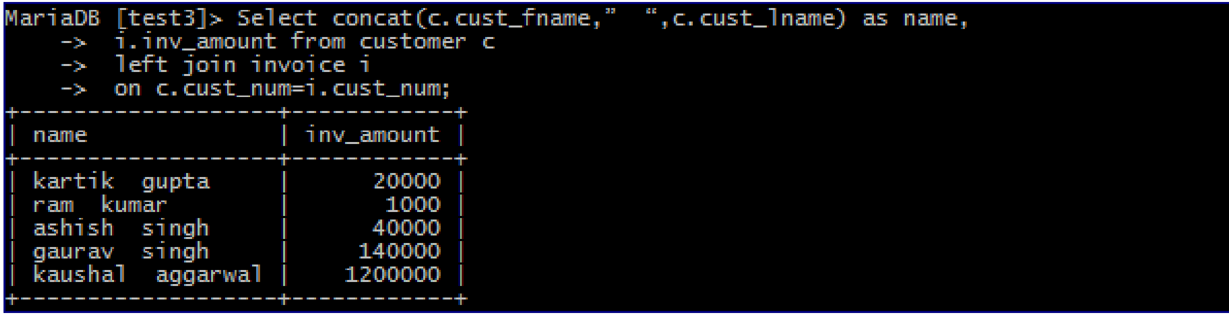
$$\pi_{cust\_num,\ cust\_fname + " " + cust\_lname}\left((customer) \bowtie \left(\sigma_{unit\ sold>3}\left({}_{cust\_num,\ \cancel{inv\_date},\ prod\_num}\mathcal{F}_{sum(unit\_sold)}(invoice)\right)\right)\right)$$

d) Write a query to illustrate Left Outer, Right Outer and Full Outer Join.

SQL QUERY:

Select concat(c.cust_fname,” “,c.cust_lname) as name,

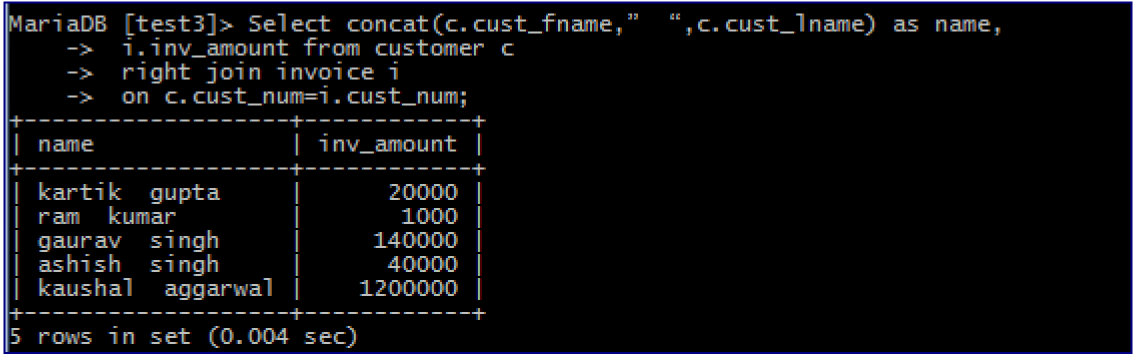
i.inv_amount from customer c
left join invoice i
on c.cust_num=i.cust_num;



```
MariaDB [test3]> Select concat(c.cust_fname,”  “,c.cust_lname) as name,
    ->  i.inv_amount from customer c
    ->  left join invoice i
    ->  on c.cust_num=i.cust_num;
+--------------------+------------+
| name               | inv_amount |
+--------------------+------------+
| kartik  gupta      |      20000 |
| ram  kumar         |       1000 |
| ashish  singh      |      40000 |
| gaurav  singh      |     140000 |
| kaushal  aggarwal  |    1200000 |
+--------------------+------------+
```

SQL QUERY:

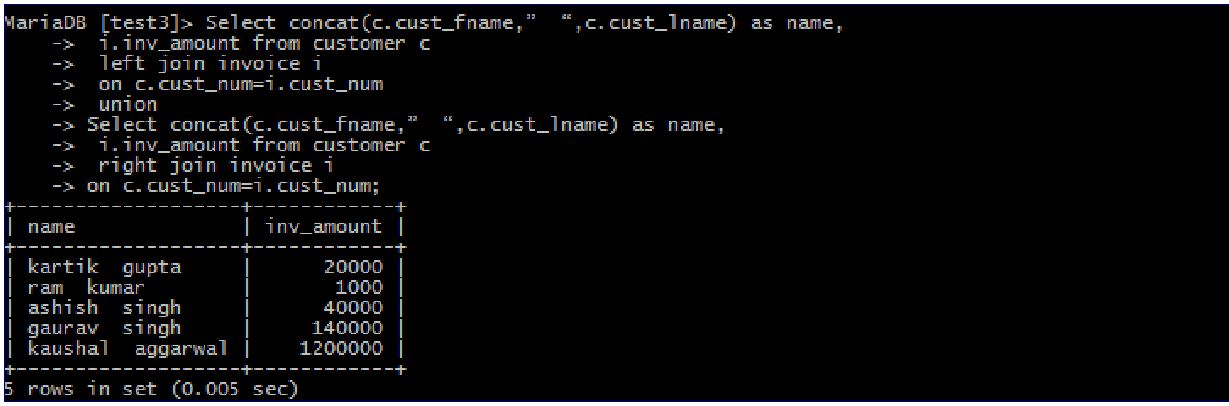
Select concat(c.cust_fname,” “,c.cust_lname) as name,
i.inv_amount from customer c
right join invoice i
on c.cust_num=i.cust_num;



```
MariaDB [test3]> Select concat(c.cust_fname,”  “,c.cust_lname) as name,
    ->  i.inv_amount from customer c
    ->  right join invoice i
    ->  on c.cust_num=i.cust_num;
+--------------------+------------+
| name               | inv_amount |
+--------------------+------------+
| kartik  gupta      |      20000 |
| ram  kumar         |       1000 |
| gaurav  singh      |     140000 |
| ashish  singh      |      40000 |
| kaushal  aggarwal  |    1200000 |
+--------------------+------------+
5 rows in set (0.004 sec)
```

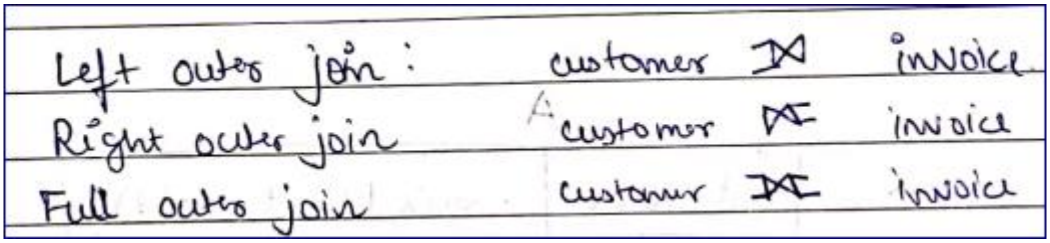
SQL QUERY:

Select concat(c.cust_fname,” “,c.cust_lname) as name,
i.inv_amount from customer c
left join invoice i
on c.cust_num=i.cust_num
union
Select concat(c.cust_fname,” “,c.cust_lname) as name,
i.inv_amount from customer c
right join invoice i
on c.cust_num=i.cust_num;



```
MariaDB [test3]> Select concat(c.cust_fname,"  “,c.cust_lname) as name,
    ->  i.inv_amount from customer c
    ->  left join invoice i
    ->  on c.cust_num=i.cust_num
    ->  union
    -> Select concat(c.cust_fname,"  “,c.cust_lname) as name,
    ->  i.inv_amount from customer c
    ->  right join invoice i
    -> on c.cust_num=i.cust_num;
+-------------------+------------+
| name              | inv_amount |
+-------------------+------------+
| kartik  gupta     |      20000 |
| ram  kumar        |       1000 |
| ashish  singh     |      40000 |
| gaurav  singh     |     140000 |
| kaushal  aggarwal |    1200000 |
+-------------------+------------+
5 rows in set (0.005 sec)
```

<u>RELATIONAL ALGEBRA:</u>



Left outer join : customer ⟕ invoice

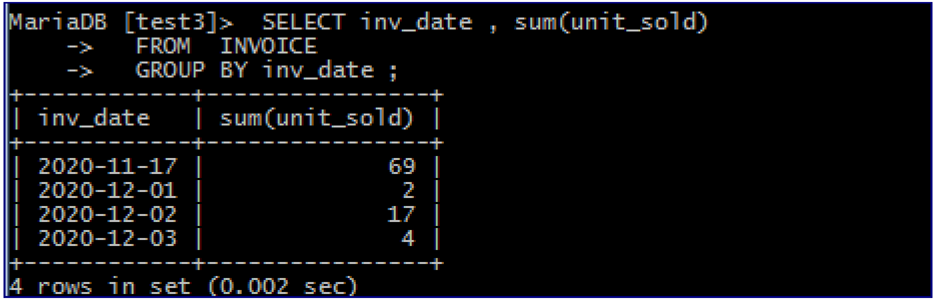
Right outer join customer ⟖ invoice

Full outer join customer ⟗ invoice

e) Count number of products sold on each date.

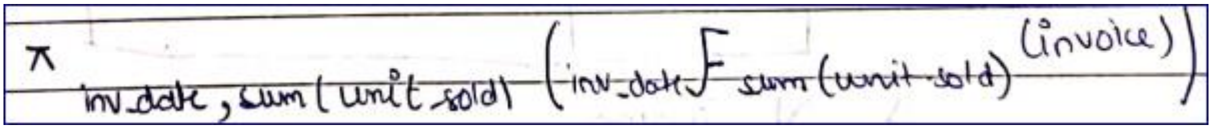
<u>SQL QUERY:</u>

```
SELECT inv_date , sum(unit_sold)
FROM  INVOICE
GROUP BY inv_date ;
```



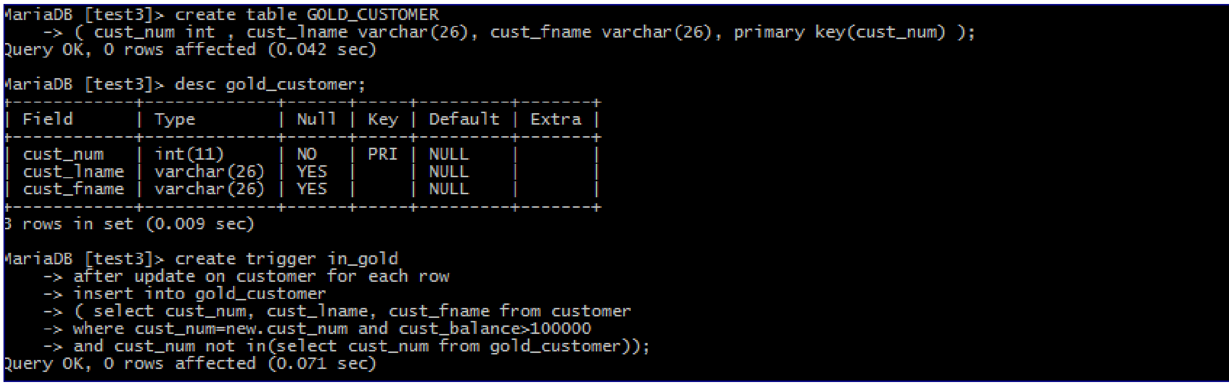
```
MariaDB [test3]>  SELECT inv_date , sum(unit_sold)
    ->   FROM  INVOICE
    ->   GROUP BY inv_date ;
+------------+----------------+
| inv_date   | sum(unit_sold) |
+------------+----------------+
| 2020-11-17 |             69 |
| 2020-12-01 |              2 |
| 2020-12-02 |             17 |
| 2020-12-03 |              4 |
+------------+----------------+
4 rows in set (0.002 sec)
```

<u>RELATIONAL ALGEBRA:</u>



$\pi_{inv\_date,\, sum(unit\_sold)}\left({}_{inv\_date}\mathcal{F}_{sum(unit\_sold)}(invoice)\right)$

f) As soon as customer balance becomes greater than Rs. 100,000, copy the customer_num in new table called ”GOLD_CUSTOMER” .

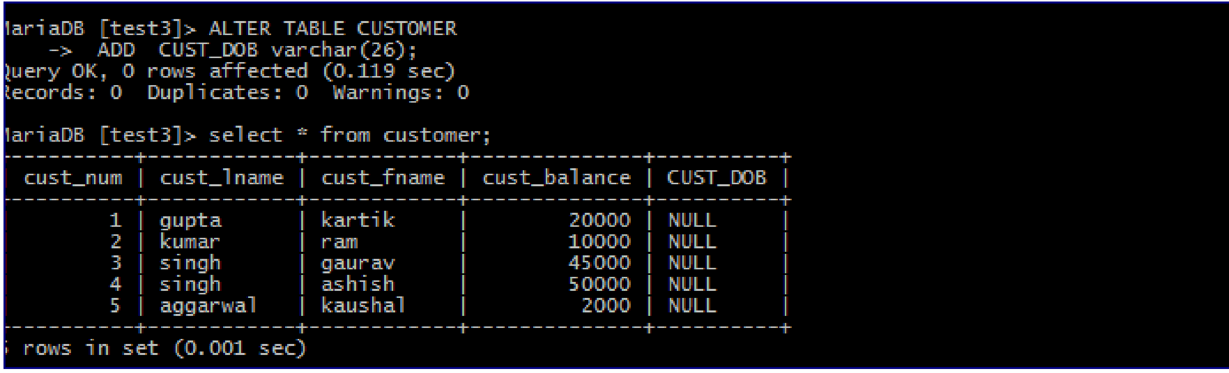


```
MariaDB [test3]> create table GOLD_CUSTOMER
    -> ( cust_num int , cust_lname varchar(26), cust_fname varchar(26), primary key(cust_num) );
Query OK, 0 rows affected (0.042 sec)

MariaDB [test3]> desc gold_customer;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| cust_num   | int(11)     | NO   | PRI | NULL    |       |
| cust_lname | varchar(26) | YES  |     | NULL    |       |
| cust_fname | varchar(26) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.009 sec)

MariaDB [test3]> 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));
Query OK, 0 rows affected (0.071 sec)
```

g) Add a new attribute CUST_DOB in customer table

SQL QUERY:

ALTER TABLE CUSTOMER
ADD CUST_DOB varchar(26);



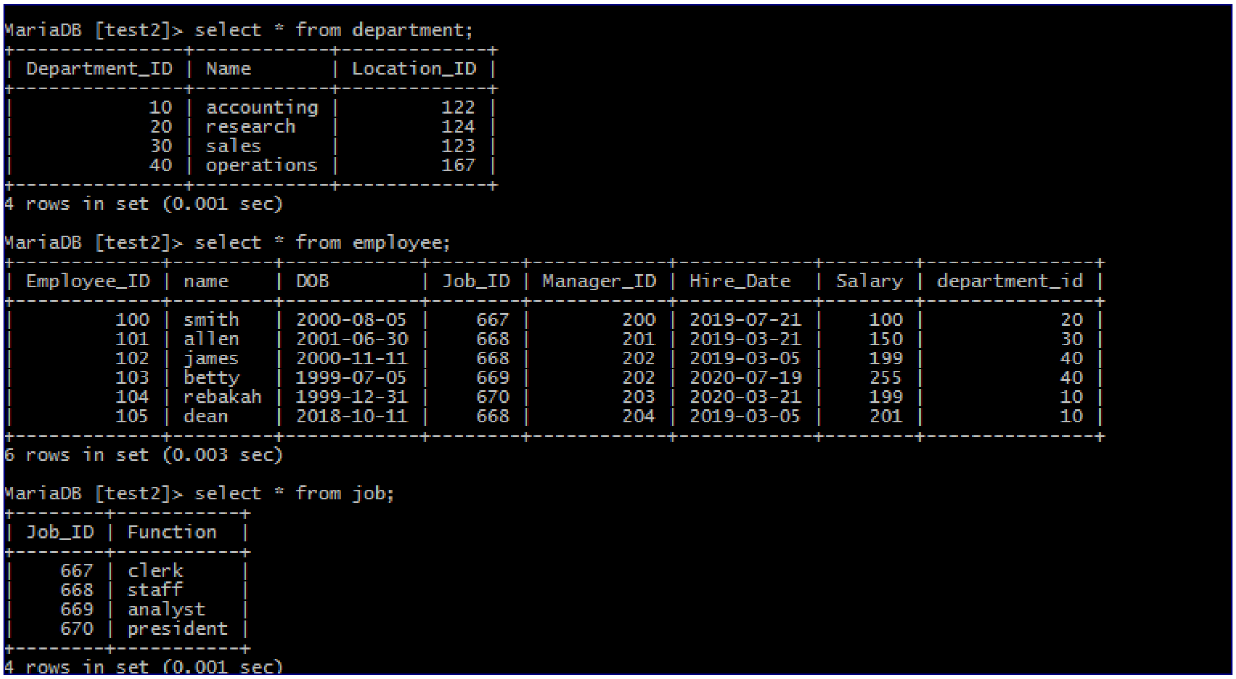
```
MariaDB [test3]> ALTER TABLE CUSTOMER
    ->  ADD  CUST_DOB varchar(26);
Query OK, 0 rows affected (0.119 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [test3]> select * from customer;
+----------+------------+------------+--------------+----------+
| cust_num | cust_lname | cust_fname | cust_balance | CUST_DOB |
+----------+------------+------------+--------------+----------+
|        1 | gupta      | kartik     |        20000 | NULL     |
|        2 | kumar      | ram        |        10000 | NULL     |
|        3 | singh      | gaurav     |        45000 | NULL     |
|        4 | singh      | ashish     |        50000 | NULL     |
|        5 | aggarwal   | kaushal    |         2000 | NULL     |
+----------+------------+------------+--------------+----------+
5 rows in set (0.001 sec)
```

Ques 3:

DEPARTMENT(Department_ID, Name, Location_ID)

JOB (Job_ID , Function)

EMPLOYEE (Employee_ID, name, DOB, Job_ID , Manager_ID, Hire_Date, Salary, department_id)

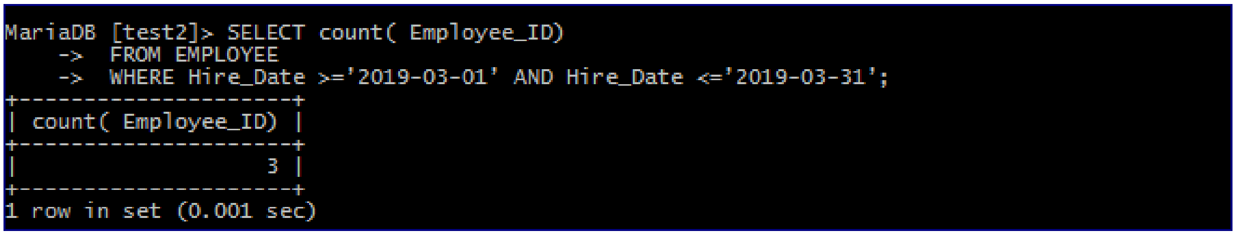


```
MariaDB [test2]> select * from department;
+---------------+------------+-------------+
| Department_ID | Name       | Location_ID |
+---------------+------------+-------------+
|            10 | accounting |         122 |
|            20 | research   |         124 |
|            30 | sales      |         123 |
|            40 | operations |         167 |
+---------------+------------+-------------+
4 rows in set (0.001 sec)

MariaDB [test2]> select * from employee;
+-------------+---------+------------+--------+------------+------------+--------+---------------+
| Employee_ID | name    | DOB        | Job_ID | Manager_ID | Hire_Date  | Salary | department_id |
+-------------+---------+------------+--------+------------+------------+--------+---------------+
|         100 | smith   | 2000-08-05 |    667 |        200 | 2019-07-21 |    100 |            20 |
|         101 | allen   | 2001-06-30 |    668 |        201 | 2019-03-21 |    150 |            30 |
|         102 | james   | 2000-11-11 |    668 |        202 | 2019-03-05 |    199 |            40 |
|         103 | betty   | 1999-07-05 |    669 |        202 | 2020-07-19 |    255 |            40 |
|         104 | rebakah | 1999-12-31 |    670 |        203 | 2020-03-21 |    199 |            10 |
|         105 | dean    | 2018-10-11 |    668 |        204 | 2019-03-05 |    201 |            10 |
+-------------+---------+------------+--------+------------+------------+--------+---------------+
6 rows in set (0.003 sec)

MariaDB [test2]> select * from job;
+--------+-----------+
| Job_ID | Function  |
+--------+-----------+
|    667 | clerk     |
|    668 | staff     |
|    669 | analyst   |
|    670 | president |
+--------+-----------+
4 rows in set (0.001 sec)
```

a) Write a query to count number of employees who joined in March 2019

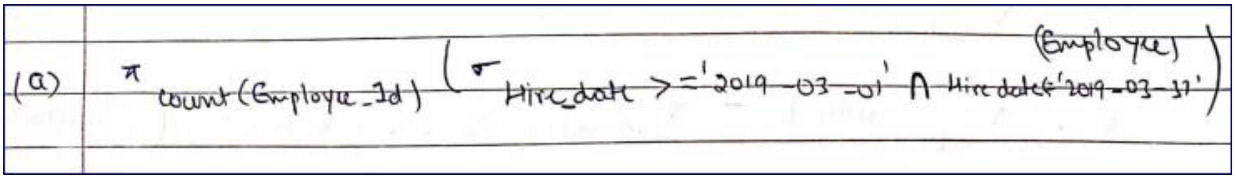
<u>SQL QUERY:</u>

```
SELECT count( Employee_ID)
FROM EMPLOYEE
WHERE Hire_Date >=’2019-03-01’ AND Hire_Date <=’2019-03-31’;
```



```
MariaDB [test2]> SELECT count( Employee_ID)
    ->  FROM EMPLOYEE
    ->  WHERE Hire_Date >='2019-03-01' AND Hire_Date <='2019-03-31';
+----------------------+
| count( Employee_ID) |
+----------------------+
|                    3 |
+----------------------+
1 row in set (0.001 sec)
```

<u>RELATIONAL ALGEBRA:</u>



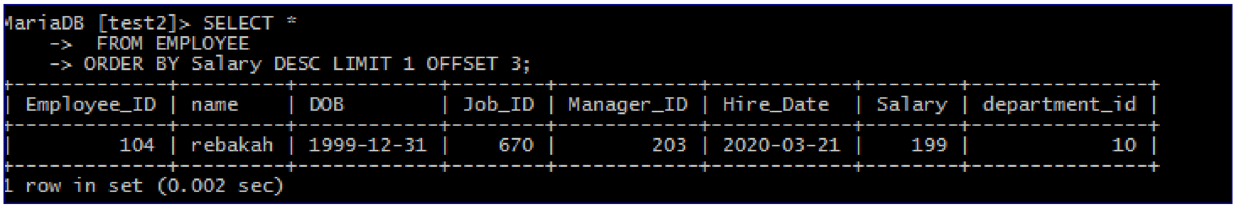
(a) $\pi_{count(Employee\_Id)}(\sigma_{Hire\_date >= '2019-03-01' \wedge Hire\_date \leq '2019-03-31'}(Employee))$

b) Display the Nth highest salary drawing employee details.

<u>SQL QUERY:</u>

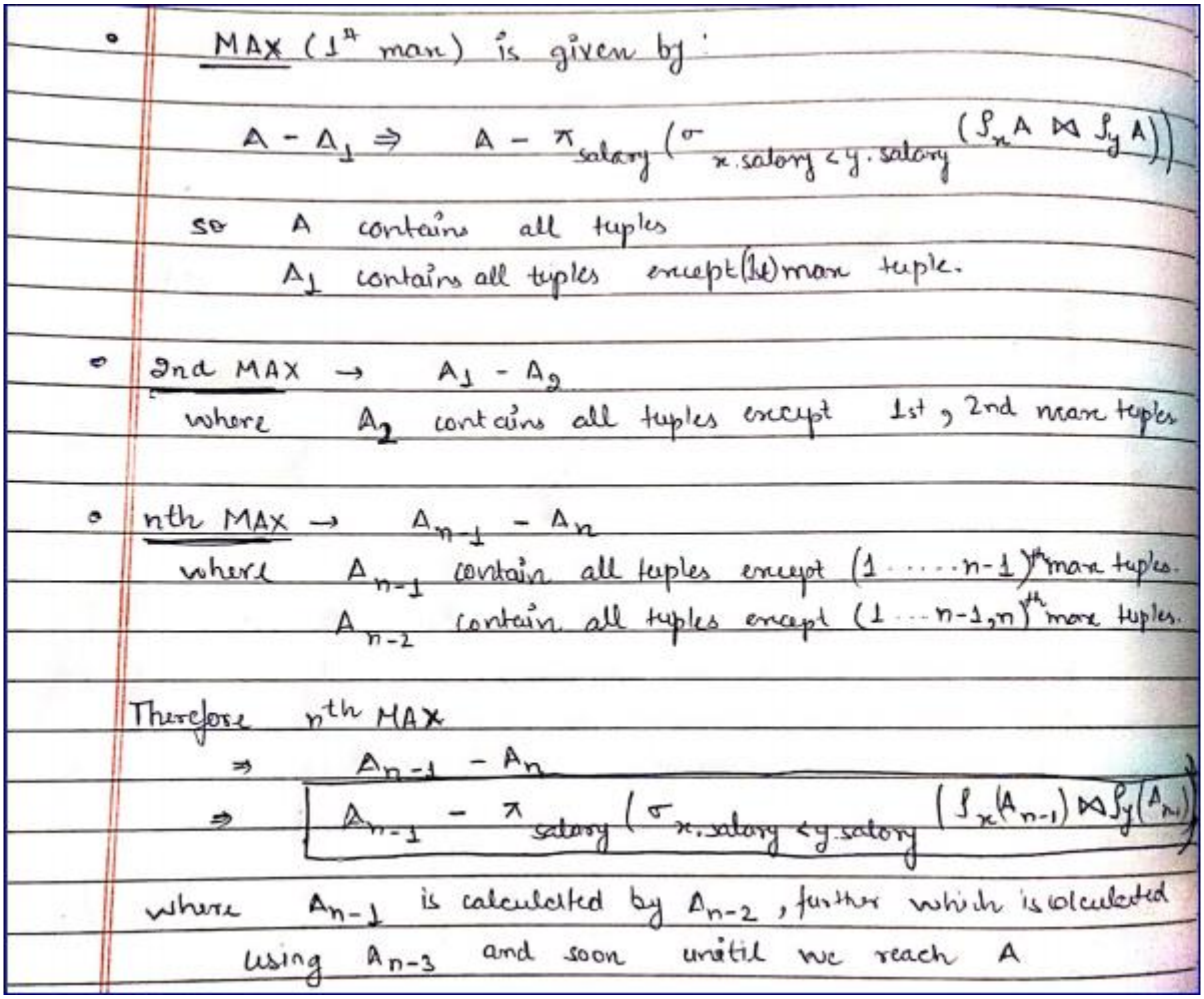
```
SELECT *
FROM EMPLOYEE
ORDER BY Salary DESC LIMIT 1 OFFSET N;
```

(taking n=3)



```
MariaDB [test2]> SELECT *
    ->  FROM EMPLOYEE
    -> ORDER BY Salary DESC LIMIT 1 OFFSET 3;
+-------------+---------+------------+--------+------------+------------+--------+---------------+
| Employee_ID | name    | DOB        | Job_ID | Manager_ID | Hire_Date  | Salary | department_id |
+-------------+---------+------------+--------+------------+------------+--------+---------------+
|         104 | rebakah | 1999-12-31 |    670 |        203 | 2020-03-21 |    199 |            10 |
+-------------+---------+------------+--------+------------+------------+--------+---------------+
1 row in set (0.002 sec)
```

RELATIONAL ALGEBRA:



- MAX (1^{st} max) is given by :

$$A - A_1 \Rightarrow A - \pi_{salary}(\sigma_{x.salary < y.salary}(\rho_x A \bowtie \rho_y A))$$

so A contains all tuples

A_1 contains all tuples except(1st) max tuple.

- 2nd MAX → $A_1 - A_2$

where A_2 contains all tuples except 1st , 2nd max tuples

- nth MAX → $A_{n-1} - A_n$

where A_{n-1} contain all tuples except $(1 \cdots\cdots n-1)^{th}$ max tuples.

A_{n-2} contain all tuples except $(1 \cdots n-1, n)^{th}$ max tuples.

Therefore n^{th} MAX

$$\Rightarrow A_{n-1} - A_n$$

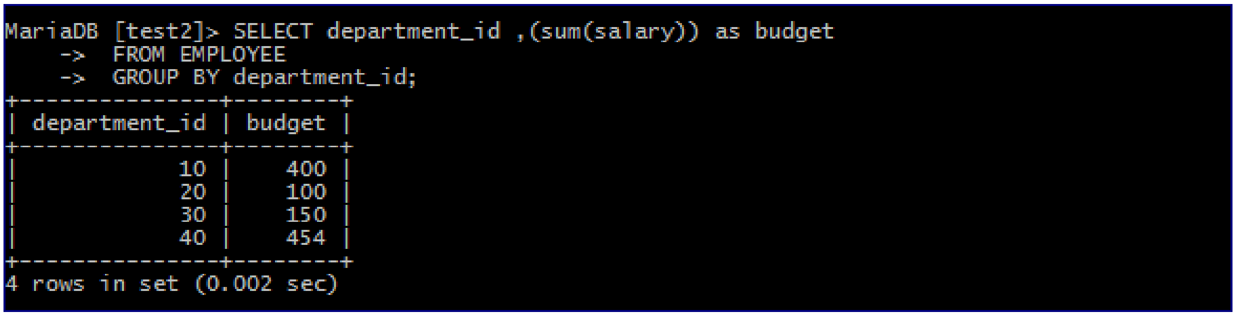
$$\Rightarrow \boxed{A_{n-1} - \pi_{salary}(\sigma_{x.salary < y.salary}(\rho_x(A_{n-1}) \bowtie \rho_y(A_{n-1})))}$$

where A_{n-1} is calculated by A_{n-2} , further which is calculated using A_{n-3} and soon until we reach A

c) Find the budget (total salary) of each department.

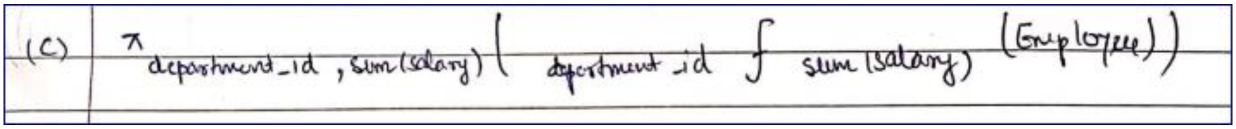
SQL QUERY:

```
SELECT department_id ,(sum(salary)) as budget
FROM EMPLOYEE
GROUP BY department_id;
```



```
MariaDB [test2]> SELECT department_id ,(sum(salary)) as budget
    ->  FROM EMPLOYEE
    ->  GROUP BY department_id;
+---------------+--------+
| department_id | budget |
+---------------+--------+
|            10 |    400 |
|            20 |    100 |
|            30 |    150 |
|            40 |    454 |
+---------------+--------+
4 rows in set (0.002 sec)
```

RELATIONAL ALGEBRA:

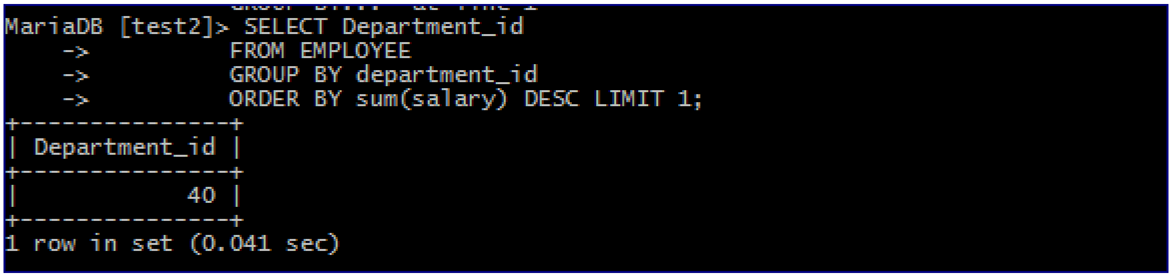


(c) $\pi_{department\_id\ ,\ sum(salary)}\left({}_{department\_id}\mathcal{F}_{sum(salary)}(Employee)\right)$

d) Find the department with maximum budget.

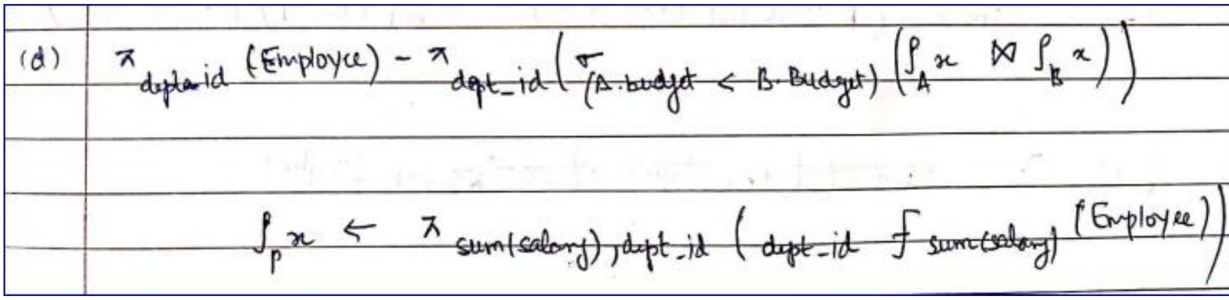
SQL QUERY:

```
SELECT Department_id
FROM EMPLOYEE
GROUP BY department_id
ORDER BY sum(salary) DESC LIMIT 1;
```



```
MariaDB [test2]> SELECT Department_id
    ->           FROM EMPLOYEE
    ->           GROUP BY department_id
    ->           ORDER BY sum(salary) DESC LIMIT 1;
+---------------+
| Department_id |
+---------------+
|            40 |
+---------------+
1 row in set (0.041 sec)
```

RELATIONAL ALGEBRA:



(d) $\pi_{dept\_id}(Employee) - \pi_{dpt\_id}\left(\sigma_{(A.budget < B.Budget)}\left(\rho_A x \bowtie \rho_B x\right)\right)$

$\rho_p x \leftarrow \pi_{sum(salary),\ dept\_id}\left({}_{dept\_id}\mathcal{F}_{sum(salary)}(Employee)\right)$

e) Create a view to show number of employees working in Delhi and update it automatically when the database is modified.

SQL QUERY:

```
CREATE VIEW DELHI_POPULATION ASSELECT COUNT ( Employee_ID )
FROM EMPLOYEE,DEPARTMENT
```

```
        WHERE Location_ID=10;
```

f) Write a trigger to ensure that no employee of age less than 25 can be inserted in the database

SQL QUERY:

```
        delimiter $$
        CREATE TRIGGER  Check_age  BEFORE INSERT ON Employee
        FOR EACH ROW
        BEGIN
        IF NEW.dob > 1995 -01-01 THEN
                SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'ERROR’:
                    AGE MUST BE ATLEAST 25 YEARS!';
        END IF;
        END; $$
        delimiter;
```
