DATABASE MANAGEMENT SYSTEM
LAB ASSIGNMENT

BY

ARYAN TYAGI
ROLL NUMBER : 2020UCO1549

GROUP MEMBERS :

ARYAN TYAGI (2020UCO1549)
BHAVYA MITTAL (2020UCO1542)
VIRENDER NAYAAL (2020UCO1545)

QUESTION 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 orange boat in the month of March.
c) Find names of sailors who've reserved a black and a orange 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

SOLUTION

Creating the tables

```
    create table sailors(
    sid int primary key ,
    rating int,
    date_of_birth date,
    sname varchar(20)
);


create table boats(
    bid int primary key,
    bname varchar(30),
    color varchar(10)
);

create table reserves(
    sid int not null,
    bid int not null,
    dates date,
    timeslot int,
    primary key(sid , bid , dates , timeslot),
    foreign key (sid) references sailors(sid) on delete cascade,
    foreign key (bid) references boats (bid) on delete cascade
```

);

Inserting data

```
insert into sailors values (1, 8, "2001-11-03", 'Rohan');
insert into sailors values (2, 9, "1996-03-11", 'Rahul');
insert into sailors values (3, 7, "1991-11-23", 'Nishit');
insert into sailors values (4, 8, "2000-09-22", 'Ishit');
insert into sailors values (5, 6, "1995-10-10", 'Varun');
insert into sailors values (6, 8, "2002-04-05", 'Atharav');
insert into sailors values (7, 9, "2001-12-25", 'Aryan');
insert into sailors values (8, 8, "2001-07-24", 'Bhavya');


insert into Boats values (101, 'Shark', 'Black');
insert into Boats values (102, 'Whail', 'White');
insert into Boats values (103, 'Dark Horse', 'Black');
insert into Boats values (104, 'Roar', 'Brown');
insert into Boats values (105, 'Fastrack', 'Green');
insert into Boats values (106, 'Champion', 'Red');
insert into Boats values (107, 'Sea Horse', 'Blue');
insert into Boats values (108, 'Stormy', 'Grey');


insert into reserves value (1 , 102 , '2021-12-11' , 1);
insert into reserves value (2 , 103 , '2021-11-19' , 2);
insert into reserves value (3 , 107 , '2021-10-12' , 3);
insert into reserves value (4 , 101 , '2021-09-20' , 4);
insert into reserves value (4 , 108 , '2021-08-21' , 5);
insert into reserves value (7 , 109 , '2021-06-22' , 6);
insert into reserves value (8, 102 ,  '2021-07-02' , 7);
```

QUERIES

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

```
mysql>
mysql> -- a) Find sailors who've reserved at least one boat
mysql> select distinct sailors.sname , sailors.sid
    -> from sailors join reserves
    -> on sailors.sid = reserves.sid;
+--------+-----+
| sname  | sid |
+--------+-----+
| Ishit  |   4 |
| Rohan  |   1 |
| Bhavya |   8 |
| Rahul  |   2 |
| Nishit |   3 |
+--------+-----+
```

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

```
mysql> -- b) Find names of sailors who've reserved a Red or a orange boat in the month of March.
mysql> select sailors.sname from sailors
    -> where sailors.sid in
    -> (    select reserves.sid from reserves join boats
    ->  on boats.bid = reserves.bid where dates like '%-03-%' and (color = 'Red' or color = 'Orange')
    -> );
Empty set (0.00 sec)
```

c) **Find names of sailors who've reserved a black and an orange boat**

```
mysql> -- c) Find names of sailors who've reserved a black and a orange boat
mysql> select sailors.sname from sailors where sailors.sid in
    -> (
    -> select reserves.sid from reserves join boats
    -> on reserves.bid = boats.bid where boats.color = 'Black' and reserves.sid in
    -> (
    -> select reserves.sid from reserves join boats
    -> where boats.bid = reserves.bid and color = 'Orange'
    -> ));
Empty set (0.00 sec)
```

d) Find sid of sailors who have not reserved a boat after Jan 2018.

```
mysql>
mysql> -- d) Find sid of sailors who have not reserved a boat after Jan 2018.
mysql> select sid from sailors
    -> where sid not in
    -> (select distinct sid  from reserves where dates > '2018-01-31');
+-----+
| sid |
+-----+
|   5 |
|   6 |
|   7 |
+-----+
3 rows in set (0.00 sec)
```

e) Find sailors whose rating is greater than that of all the sailors named "John"

```
mysql> -- e) Find sailors whose rating is greater than that of all the sailors named "John" .
mysql> select sailors.sname , sailors.sid , sailors.rating from sailors
    -> where sailors.rating >
    -> (select sailors.rating from sailors where sailors.sname = 'Jhon Kutti');
Empty set (0.00 sec)
```

f) Find sailors who've reserved all boats

```
mysql>
mysql> -- f) Find sailors who've reserved atleast boats
mysql> select sailors.sname , sailors.sid , derived.counts
   -> from sailors join (select sid , count(distinct reserves.bid) as counts  from reserves group by reserves.sid) as derived
   -> on derived.sid = sailors.sid where counts = (select count(bid) from boats);
Empty set (0.01 sec)
```

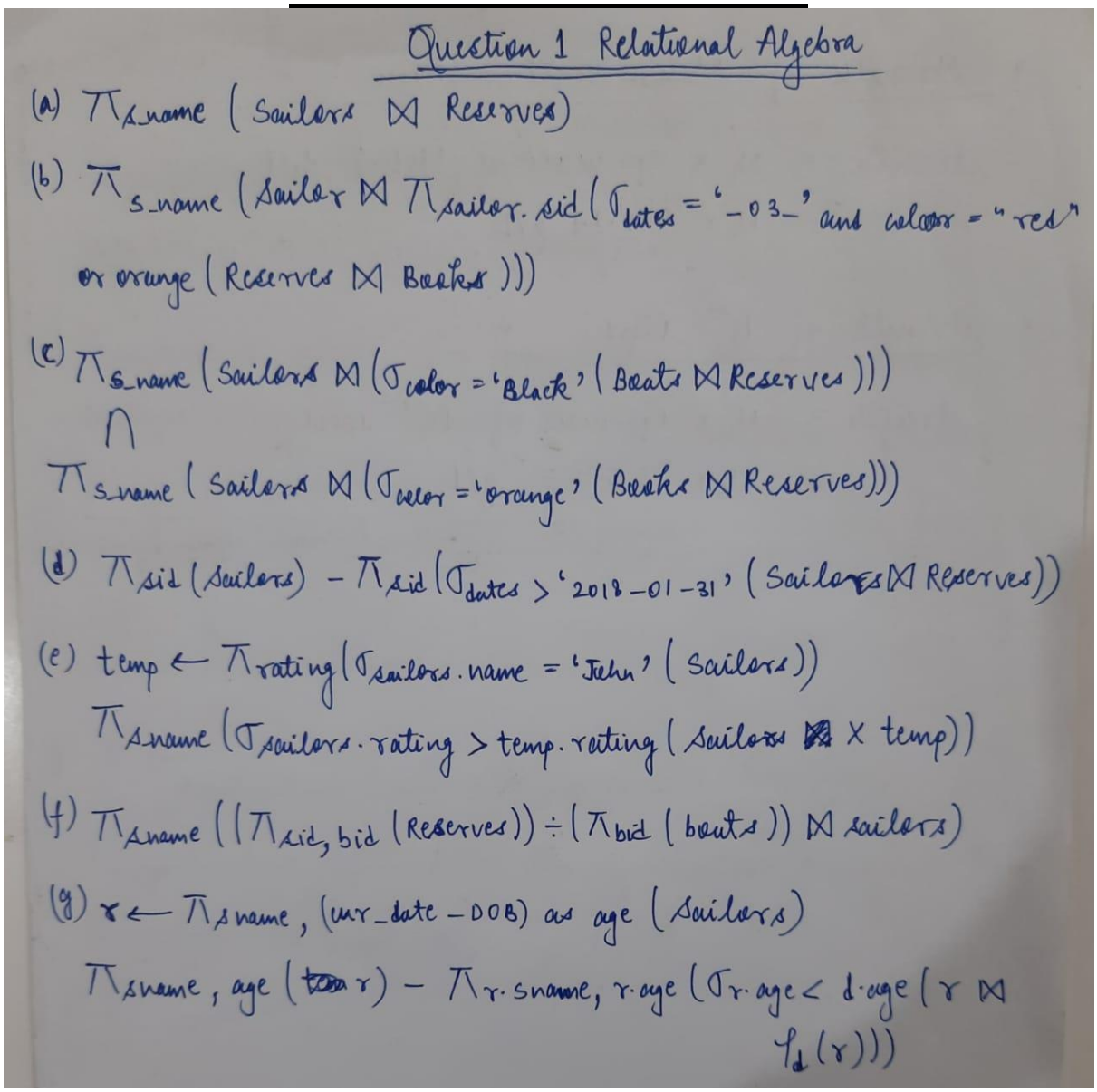
g) Find name and age of the oldest sailor(s)

```
mysql> -- g) Find name and age of the oldest sailor
mysql> select sname , floor(datediff(current_date() , date_of_birth)/365) as age
    -> from sailors
    -> where date_of_birth in
    -> (select min(date_of_birth) from sailors);
+--------+------+
| sname  | age  |
+--------+------+
| Nishit |   30 |
+--------+------+
1 row in set (0.00 sec)
```

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

```
mysql> -- h) Find the age of the youngest sailor for each rating with at least 2 such sailors
mysql> select min(floor(datediff(current_date() , date_of_birth)/365)) as age , rating
    -> from sailors group by rating having count(*) > 1;
+------+--------+
| age  | rating |
+------+--------+
|   19 |      8 |
|   19 |      9 |
+------+--------+
2 rows in set (0.00 sec)
```

RELATIONAL ALGEBRA



Question 1 Relational Algebra

(a) Π_{s_name} (Sailors $\bowtie$ Reserves)

(b) Π_{s_name} (Sailor $\bowtie$ $\Pi_{sailor.sid}$ ($\sigma_{dates = '\_03\_' \text{ and } colour = \text{"red"}}$ or orange (Reserves $\bowtie$ Boats)))

(c) Π_{s_name} (Sailors $\bowtie$ ($\sigma_{color = 'Black'}$ (Boats $\bowtie$ Reserves)))

$\cap$

Π_{s_name} (Sailors $\bowtie$ ($\sigma_{color = 'orange'}$ (Boats $\bowtie$ Reserves)))

(d) Π_{sid} (Sailors) $-$ Π_{sid} ($\sigma_{dates > '2018-01-31'}$ (Sailors $\bowtie$ Reserves))

(e) temp $\leftarrow$ Π_{rating} ($\sigma_{sailors.name = 'John'}$ (Sailors))

$\Pi_{s.name}$ ($\sigma_{sailors.rating > temp.rating}$ (Sailors X temp))

(f) $\Pi_{s.name}$ (($\Pi_{sid, bid}$ (Reserves)) $\div$ (Π_{bid} (boats)) $\bowtie$ sailors)

(g) r $\leftarrow$ $\Pi_{s.name, (cur\_date - DOB) \text{ as } age}$ (Sailors)

$\Pi_{s.name, age}$ (r) $-$ $\Pi_{r.sname, r.age}$ ($\sigma_{r.age < d.age}$ (r $\bowtie$ ρ_d (r)))

QUESTION 2

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

SOLUTION

Creating the tables

```
create table customer (
cust_num int primary key,
cust_fname varchar(20),
cust_lname varchar(20),
cust_balance int
);

create table product (
prod_num int primary key,
prod_name varchar(30),
price int
);

create table invoice(
invoice_num int,
prod_num int ,
cust_num int not null,
```

```
invoice_date date,
unit_sold int,
invoice_amount int,
primary key (invoice_num ,prod_num , cust_num , invoice_date),
foreign key (cust_num) references customer(cust_num) on delete cascade,
foreign key (prod_num) references product(prod_num) on delete cascade
);
```

Inserting data

```
insert into customer value(101 , 'Aryan' , 'Tyagi' , 8000);
insert into customer value(102 , 'Atharav' , 'Mahajan' , 8000);
insert into customer value(103 , 'Bhavya' , 'Mittal' , 10000);
insert into customer value(104 , 'Tanishq' , 'Mehta' , 12000);
insert into customer value(105 , 'Akshat' , 'Jain' , 9000);
insert into customer value(106 , 'Junaid' , 'Ahmed' , 25000);
insert into customer value(107 , 'Tina' , 'Dabi' , 30000);
insert into customer value(108 , 'Apala' ,'Mishra' , 7000);
```

```
insert into product value(201, 'Laptop', 70000);
insert into product value(202, 'I Phone', 80000);
insert into product value(203, 'PC', 80000);
insert into product value(204, 'Keyboard', 1500);
insert into product value(205, 'Mouse', 3000);
insert into product value(206, 'I Pencil', 2000);
insert into product value(207, 'Android Phone', 30000);
insert into product value(208, 'AC', 30000);
```

```
insert into invoice value(1,'201' ,102 ,'2021-11-23' , 1 , 70000);
insert into invoice value(2,'204' ,103 ,'2021-12-13' ,2 ,3000);
insert into invoice value(3,'202' ,104 ,'2015-10-23' ,1 ,80000);
insert into invoice value(4,'205' ,105 ,'2011-01-23' ,2 ,6000);
insert into invoice value(5,'206' ,101 ,'2021-05-21' ,2 ,160000);
insert into invoice value(6,'208' ,107 ,'2010-12-13' ,1 ,30000);
insert into invoice value(7,'205' ,106 ,'2018-09-23' ,4 ,12000);
insert into invoice value(8,'208' ,108 ,'2019-10-22' ,2 ,60000);
insert into invoice value(9,'207' ,101 ,'2020-01-18' ,1 ,30000);
insert into invoice value(10,'204' ,102 ,'2021-11-23' ,3 ,4500);
```

QUERIES

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

```
mysql> -- a) Find the names of the customer who have purchased no item. Set default value of Cust_balance as 0 for such customers.
mysql> select cust_num from customer
    -> where cust_num not in
    -> (select distinct cust_num from invoice);
Empty set (0.01 sec)
```

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

```
mysql> delimiter $$
mysql> create trigger check_age before insert on invoice
    -> for each row
    -> Update customer c
    -> Set c.cust_balance = c.cust_balance + new.invoice_amount
    -> Where c.cust_num = new.cust_num;
    -> Select cust_fname ,cust_lname , cust_balance from customer;
```

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

```
mysql> select cust_num , cust_fname
    -> from customer where cust_num
    -> in( select cust_num from invoice
    -> group by cust_num,invoice_date,prod_num
    -> having sum(unit_sold)>3);
+----------+-------------+
| cust_num | cust_fname  |
+----------+-------------+
|      106 | Junaid      |
+----------+-------------+
1 row in set (0.00 sec)
```

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

Select MySQL 8.0 Command Line Client

```
mysql> -- d) Write a query to illustrate Left Outer, Right Outer and Full Outer Join.
mysql> select customer.cust_num ,cust_fname , cust_lname , invoice_amount ,invoice_date
    -> from customer left join invoice
    -> on customer.cust_num = invoice.cust_num;
+----------+------------+------------+----------------+--------------+
| cust_num | cust_fname | cust_lname | invoice_amount | invoice_date |
+----------+------------+------------+----------------+--------------+
|      101 | Aryan      | Tyagi      |         160000 | 2021-05-21   |
|      101 | Aryan      | Tyagi      |          30000 | 2020-01-18   |
|      102 | Atharav    | Mahajan    |          70000 | 2021-11-23   |
|      102 | Atharav    | Mahajan    |           4500 | 2021-11-23   |
|      103 | Bhavya     | Mittal     |           3000 | 2021-12-13   |
|      104 | Tanishq    | Mehta      |          80000 | 2015-10-23   |
|      105 | Akshat     | Jain       |           6000 | 2011-01-23   |
|      106 | Junaid     | Ahmed      |          12000 | 2018-09-23   |
|      107 | Tina       | Dabi       |          30000 | 2010-12-13   |
|      108 | Apala      | Mishra     |          60000 | 2019-10-22   |
+----------+------------+------------+----------------+--------------+
10 rows in set (0.00 sec)

mysql> select customer.cust_num ,cust_fname , cust_lname , invoice_amount ,invoice_date
    -> from customer right join invoice
    -> on customer.cust_num = invoice.cust_num;
+----------+------------+------------+----------------+--------------+
| cust_num | cust_fname | cust_lname | invoice_amount | invoice_date |
+----------+------------+------------+----------------+--------------+
|      102 | Atharav    | Mahajan    |          70000 | 2021-11-23   |
|      103 | Bhavya     | Mittal     |           3000 | 2021-12-13   |
|      104 | Tanishq    | Mehta      |          80000 | 2015-10-23   |
|      105 | Akshat     | Jain       |           6000 | 2011-01-23   |
|      101 | Aryan      | Tyagi      |         160000 | 2021-05-21   |
|      107 | Tina       | Dabi       |          30000 | 2010-12-13   |
|      106 | Junaid     | Ahmed      |          12000 | 2018-09-23   |
|      108 | Apala      | Mishra     |          60000 | 2019-10-22   |
|      101 | Aryan      | Tyagi      |          30000 | 2020-01-18   |
|      102 | Atharav    | Mahajan    |           4500 | 2021-11-23   |
+----------+------------+------------+----------------+--------------+
10 rows in set (0.00 sec)

mysql> (select customer.cust_num ,cust_fname , cust_lname , invoice_amount ,invoice_date
    -> from customer left join invoice
    -> on customer.cust_num = invoice.cust_num)
    -> union
    -> (select customer.cust_num ,cust_fname , cust_lname , invoice_amount ,invoice_date
    -> from customer right join invoice
    -> on customer.cust_num = invoice.cust_num);
+----------+------------+------------+----------------+--------------+
| cust_num | cust_fname | cust_lname | invoice_amount | invoice_date |
+----------+------------+------------+----------------+--------------+
```

```
+----------+------------+------------+----------------+--------------+
| cust_num | cust_fname | cust_lname | invoice_amount | invoice_date |
+----------+------------+------------+----------------+--------------+
|      101 | Aryan      | Tyagi      |         160000 | 2021-05-21   |
|      101 | Aryan      | Tyagi      |          30000 | 2020-01-18   |
|      102 | Atharav    | Mahajan    |          70000 | 2021-11-23   |
|      102 | Atharav    | Mahajan    |           4500 | 2021-11-23   |
|      103 | Bhavya     | Mittal     |           3000 | 2021-12-13   |
|      104 | Tanishq    | Mehta      |          80000 | 2015-10-23   |
|      105 | Akshat     | Jain       |           6000 | 2011-01-23   |
|      106 | Junaid     | Ahmed      |          12000 | 2018-09-23   |
|      107 | Tina       | Dabi       |          30000 | 2010-12-13   |
|      108 | Apala      | Mishra     |          60000 | 2019-10-22   |
+----------+------------+------------+----------------+--------------+
10 rows in set (0.00 sec)
```

e) Count number of products sold on each date.

```
mysql> -- e) Count number of products sold on each date.
mysql> select invoice_date , sum(unit_sold)
    -> from invoice group by invoice_date
    -> order by invoice_date asc;
+--------------+----------------+
| invoice_date | sum(unit_sold) |
+--------------+----------------+
| 2010-12-13   |              1 |
| 2011-01-23   |              2 |
| 2015-10-23   |              1 |
| 2018-09-23   |              4 |
| 2019-10-22   |              2 |
| 2020-01-18   |              1 |
| 2021-05-21   |              2 |
| 2021-11-23   |              4 |
| 2021-12-13   |              2 |
+--------------+----------------+
9 rows in set (0.00 sec)
```

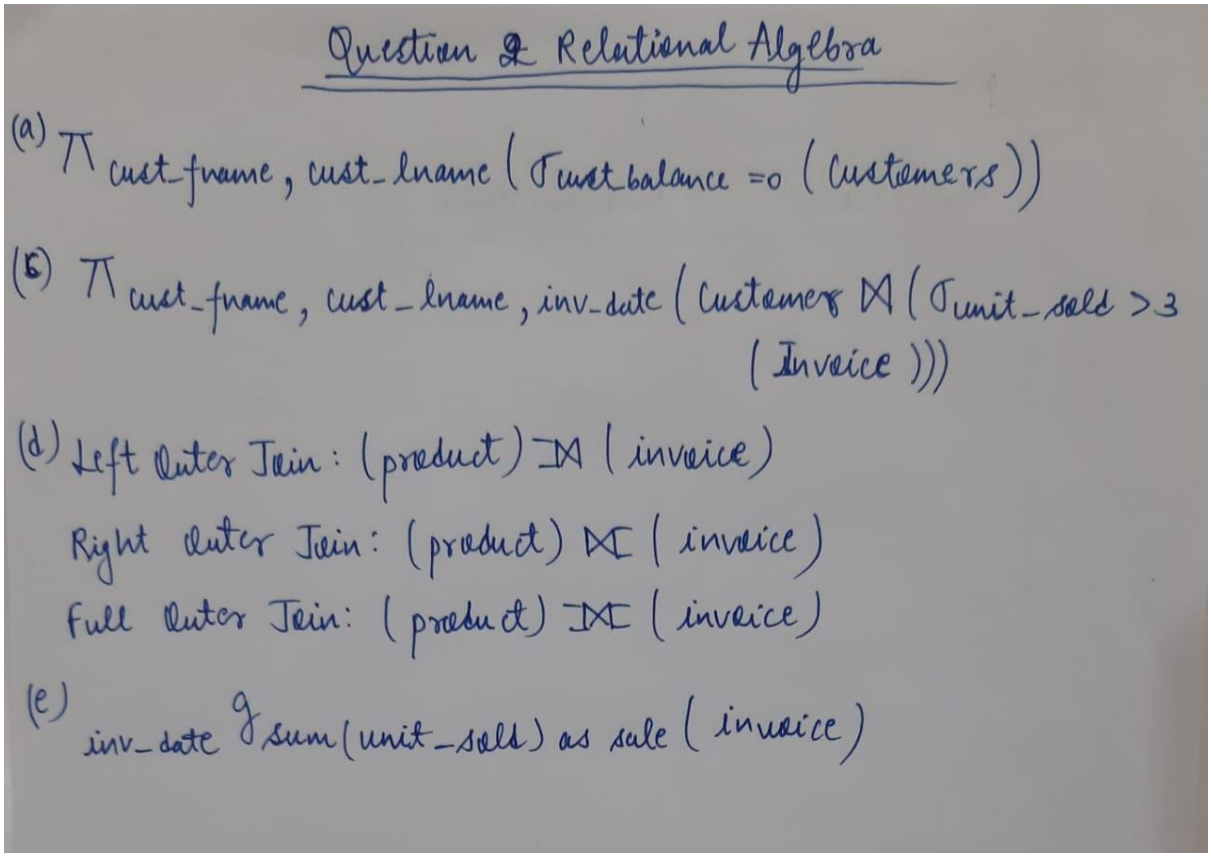
f) As soon as customer balance becomes greater than Rs. 100,000, copy the customer_num in new table called ”GOLD_CUSTOMER”

```
mysql> -- f) As soon as customer balance becomes greater than Rs. 100,000, copy the customer_num in new table called "GOLD_CUSTOMER"
mysql> select cust_num from
    -> (select * from customer where cust_balance > 10000)
    -> as gold_customer;
+----------+
| cust_num |
+----------+
|      104 |
|      106 |
|      107 |
+----------+
3 rows in set (0.00 sec)
```

g) Add a new attribute CUST_DOB in customer table

```
mysql> -- g) Add a new attribute CUST_DOB in customer table
mysql> alter table customer add cust_dob date;
Query OK, 0 rows affected (1.69 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

RELATIONAL ALGEBRA



Question 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)

a) Write a query to count number of employees who joined in March 2015
b) Display the Nth highest salary drawing employee details.
c) Find the budget (total salary) of each department.
d) Find the department with maximum budget.
e) Create a view to show number of employees working in Delhi and update it automatically when the
database is modified.
f) Write a trigger to ensure that no employee of age less than 25 can be inserted in the database

SOLUTION

Creating the tables

```
create table department (
department_id int primary key,
department_name varchar(20),
location varchar(20)
);

create table job (
job_id int primary key,
functions varchar(30)
);

create table employee(
employee_id int primary key,
employee_name varchar(30) ,
date_of_birth date,
job_id int not null,
hire_date date,
manager_id int ,
salary int,
department_id int,
foreign key (job_id) references job(job_id) on delete cascade,
```

```
foreign key (department_id) references department(department_id) on delete set null,
foreign key (manager_id) references employee(employee_id) on delete set null
);
```

Inserting data

```
insert into department value(101 , 'Security' , 'Noida');
insert into department value(102 , 'Sales' , 'Delhi');
insert into department value(103 , 'Accounting' , 'Hyderabad');
insert into department value(104 , 'Marketing' , 'Pune');
insert into department value(105 , 'Purchasing' , 'Mumbai');
insert into department value(106 , 'Planning' , 'Chennai');
insert into department value(107 , 'HR' , 'Haryana');
insert into department value(108 , 'Testing' , 'Gurgaon');


insert into job value(201 , 'Developer');
insert into job value(202 , ' Researcher ');
insert into job value(203 , ' Purchaser ');
insert into job value(204 , 'Accountant');
insert into job value(205 , 'Data Analyst');
insert into job value(206 , 'Marketing Specialist');
insert into job value(207 , 'Production');
insert into job value(208 , ' Business Analyst');



insert into employee value(301 , 'Aryan Tyagi' ,'1996-12-15' , 201 , '2016-10-15' ,301 ,
100000 , 104);
insert into employee value(302 , 'Atharav' ,'1994-11-02' , 202 , '2017-11-15', 302 , 90000 ,
106);
insert into employee value(303 , 'Bhavya Mittal' ,'1994-11-01' , 204 , '2018-02-09' ,301 ,
80000 , 102);
insert into employee value(304 , 'Tanishq Mehta' ,'1989-10-10' , 201 , '2015-02-14',301 ,
90000 , 104);
insert into employee value(305 , 'Ajay Singh' ,'1992-12-31' , 207 , '2018-02-20' ,301 , 50000 ,
106);
insert into employee value(306 , 'Rahul Gupta' ,'1996-12-05' , 208 , '2015-12-05' ,302 ,
20000 , 105);
insert into employee value(307 , 'Sohum Gupta' ,'1992-07-18' , 204 , '2018-02-05' ,302 ,
80000 , 102);
insert into employee value(308 , 'Diya Aggarwal' ,'1995-03-11' , 203, '2016-05-08' ,302 ,
40000 , 101);
```

QUERIES

a) Write a query to count number of employees who joined in March 2015

```
mysql> -- a)Write a query to count number of employees who joined in March 2015
mysql> select count(employee_id) from
    -> (select employee_id from employee where hire_date like '%-03-%')
    -> as temp;
+--------------------+
| count(employee_id) |
+--------------------+
|                  0 |
+--------------------+
1 row in set (0.00 sec)
```

b) Display the Nth highest salary drawing employee details.

```
mysql> -- b)Display the Nth highest salary drawing employee details.
mysql> select * from employee
    -> where employee.salary not in
    -> (select employee.salary
    -> from employee cross join employee
    -> as temp on employee.salary < temp.salary
    -> );
+-------------+---------------+---------------+--------+------------+------------+--------+---------------+
| employee_id | employee_name | date_of_birth | job_id | hire_date  | manager_id | salary | department_id |
+-------------+---------------+---------------+--------+------------+------------+--------+---------------+
|         301 | Aryan Tyagi   | 1996-12-15    |    201 | 2016-10-15 |        301 | 100000 |           104 |
+-------------+---------------+---------------+--------+------------+------------+--------+---------------+
1 row in set (0.00 sec)

mysql>    ^C
```

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

```
mysql> -- c)Find the budget (total salary) of each department
mysql> select employee.department_id , sum(employee.salary)
    -> from employee group by department_id;
+---------------+----------------------+
| department_id | sum(employee.salary) |
+---------------+----------------------+
|           101 |                40000 |
|           102 |               160000 |
|           104 |               190000 |
|           105 |                20000 |
|           106 |               140000 |
+---------------+----------------------+
5 rows in set (0.00 sec)
```

d) Find the department with maximum budget.

```
mysql> -- d)Find the department with maximum budget.
mysql> select department_id ,sum(salary)
    -> from employee group by department_id
    -> order by sum(salary) desc limit 1;
+---------------+-------------+
| department_id | sum(salary) |
+---------------+-------------+
|           104 |      190000 |
+---------------+-------------+
1 row in set (0.02 sec)
```

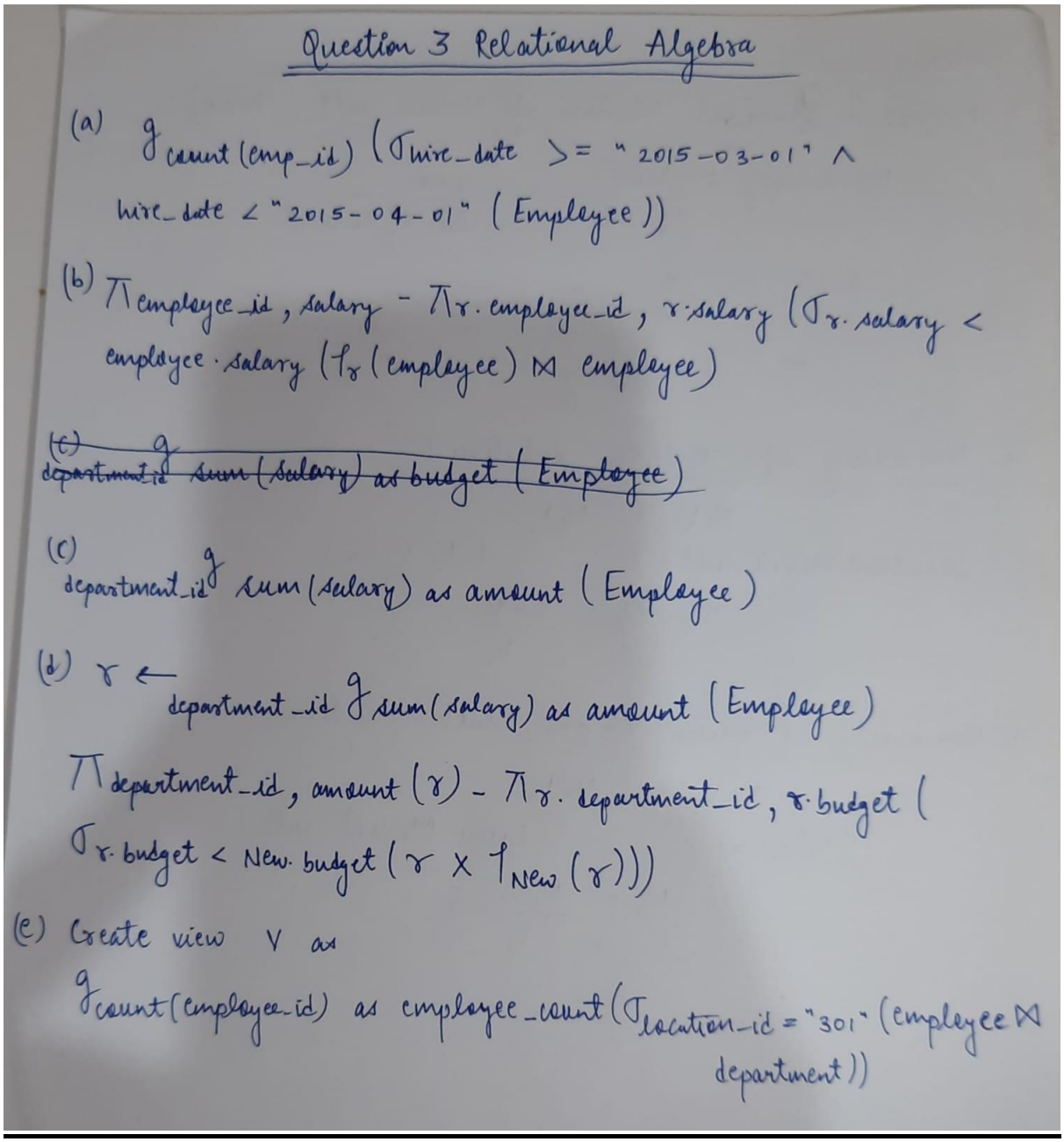
e) Create a view to show number of employees working in Delhi and update it automatically when the database is modified.

```
mysql> -- e)Create a view to show number of employees working in Delhi and update it automatically when the database is modified
mysql> create view delhi_employee as select count(distinct employee_id) from
    -> (select * from employee natural join department
    -> where location = 'Delhi') as derived;
Query OK, 0 rows affected (0.60 sec)
```

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

```
mysql> -- f)Write a trigger to ensure that no employee of age less than 25 can be inserted in the database
mysql> delimiter $$
mysql> create trigger check_age before insert on employee
    -> for each row
    -> begin
    -> if new.date_of_birth > '1993-01-01' then
    -> signal sqlstate '45000'
    -> set message_text = 'error:
    '> age muste be atleast 25 years!';
    -> end if;
    -> end;
```

RELATIONAL ALGEBRA



Question 3 Relational Algebra

(a) $\gamma_{count(emp\_id)}(\sigma_{hire\_date >= "2015-03-01" \wedge hire\_date < "2015-04-01"}(Employee))$

(b) $\Pi_{employee\_id, salary} - \Pi_{r.employee\_id, r.salary}(\sigma_{r.salary < employee.salary}(\rho_r(employee) \bowtie employee))$

~~(c) $\gamma_{department\_id, sum(salary) as budget}(Employee)$~~

(c) $\gamma_{department\_id, sum(salary) as amount}(Employee)$

(d) $r \leftarrow \gamma_{department\_id, sum(salary) as amount}(Employee)$

$\Pi_{department\_id, amount}(r) - \Pi_{r.department\_id, r.budget}(\sigma_{r.budget < New.budget}(r \times \rho_{New}(r)))$

(e) Create view V as

$\gamma_{count(employee\_id) as employee\_count}(\sigma_{location\_id = "301"}(employee \bowtie department))$