

**51.)** name is the primary key column for this table.

Each row of this table gives information about the name of a country, the continent to which it

belongs, its area, the population, and its GDP value.

A country is big if:

● it has an area of at least three million (i.e., 3000000 km2), or

● it has a population of at least twenty-five million (i.e., 25000000).

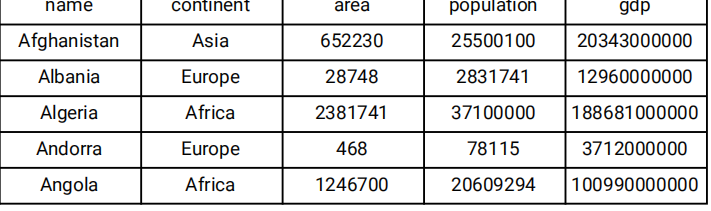
Write an SQL query to report the name, population, and area of the big countries.

Return the result table in any order.

The query result format is in the following example.

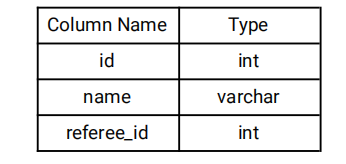
Input:

World table:



**QUERY:-**

select name,population,area from World where area >=3000000 or population >= 25000000;



**52.)** id is the primary key column for this table.

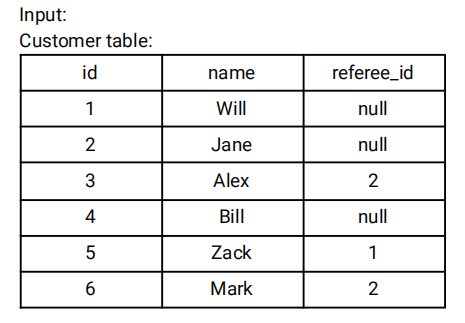
Each row of this table indicates the id of a customer, their name, and the id of the customer who

referred them.

Write an SQL query to report the names of the customer that are not referred by the customer with id = 2.

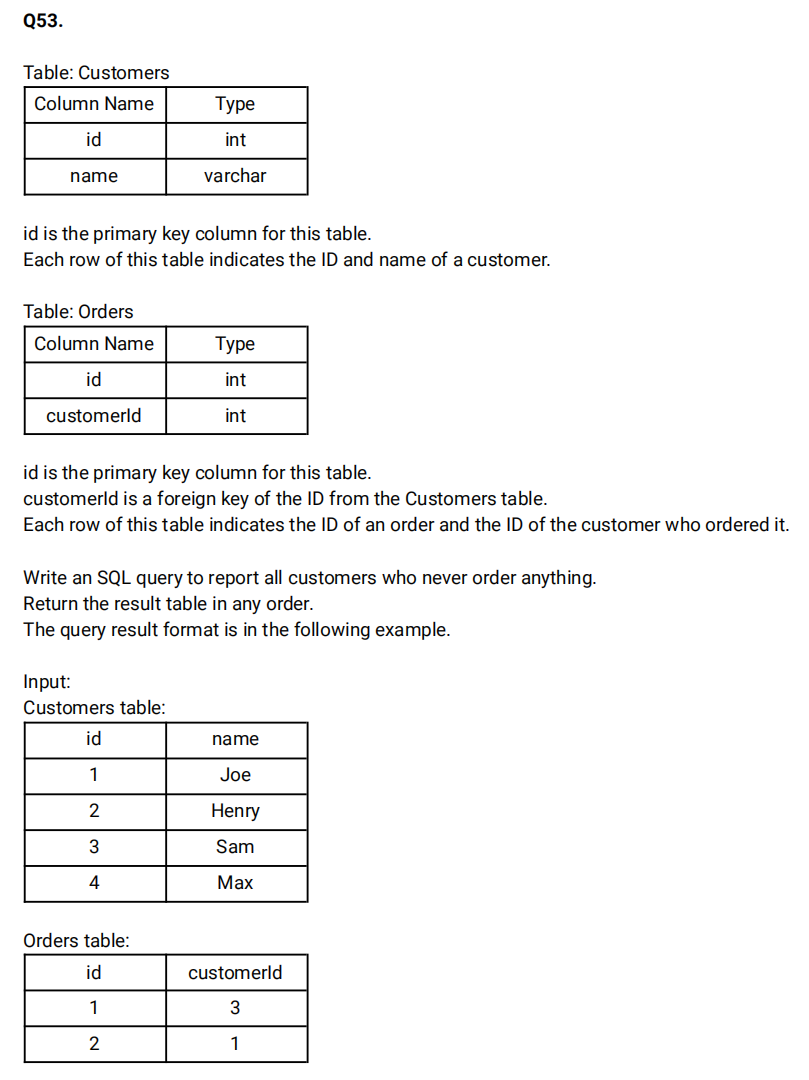
Return the result table in any order.

The query result format is in the following example.



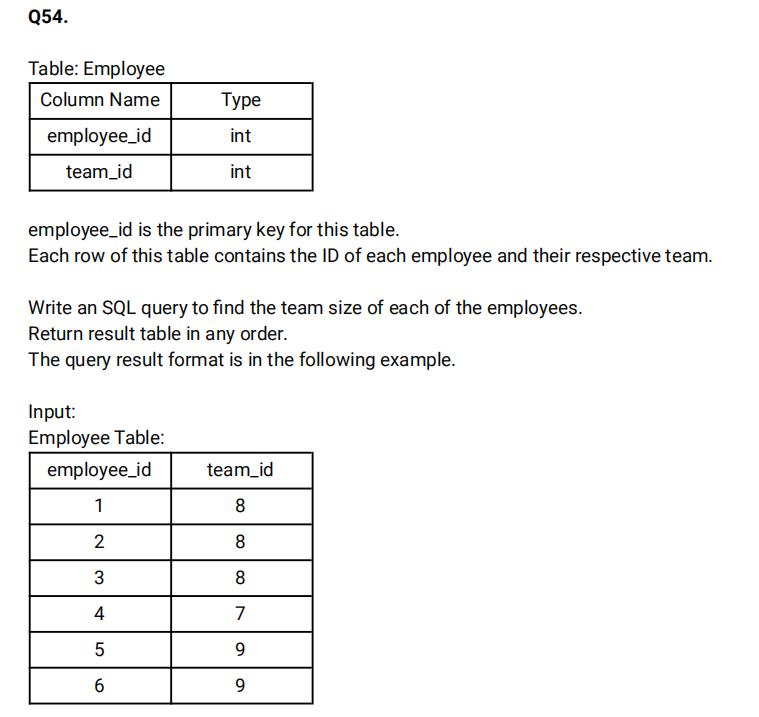
**QUERY:-**

select name from Customer where id<>2 OR id IS NULL;



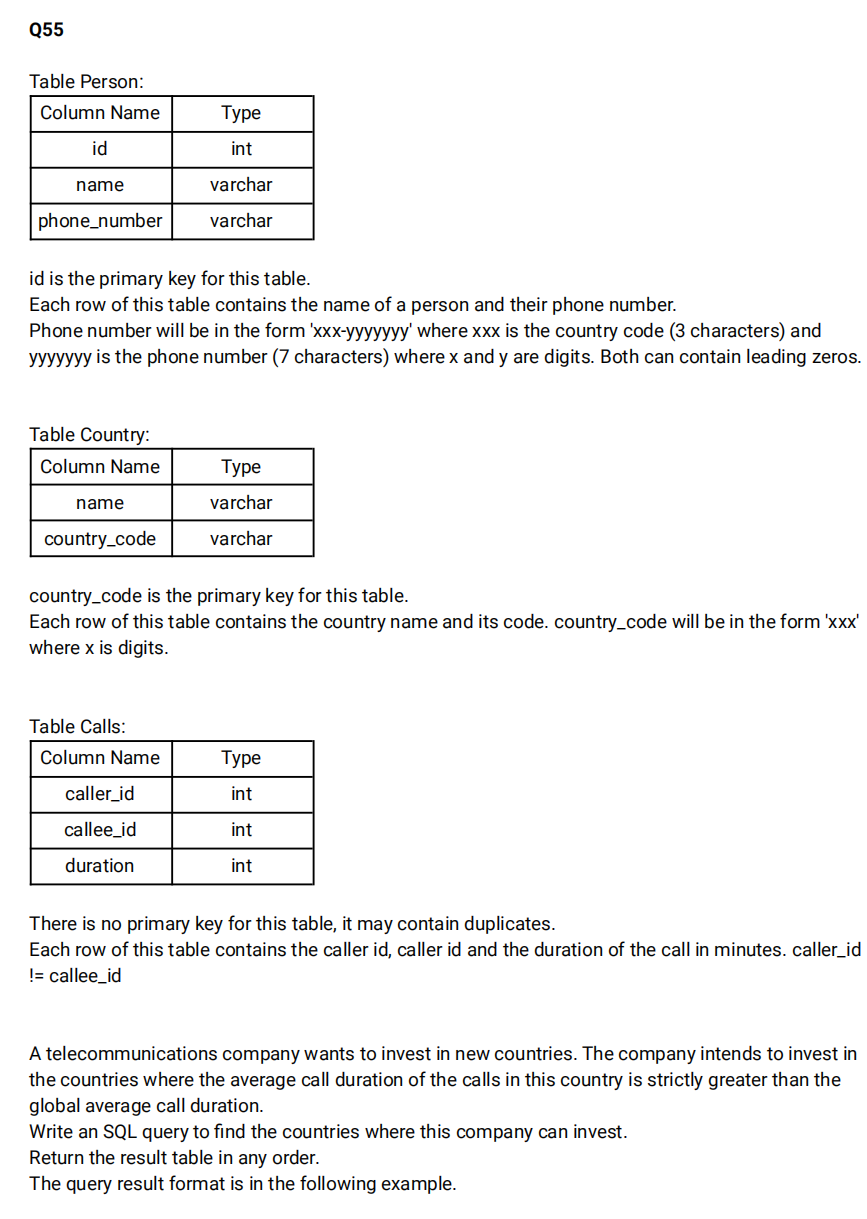
**QUERY:-**

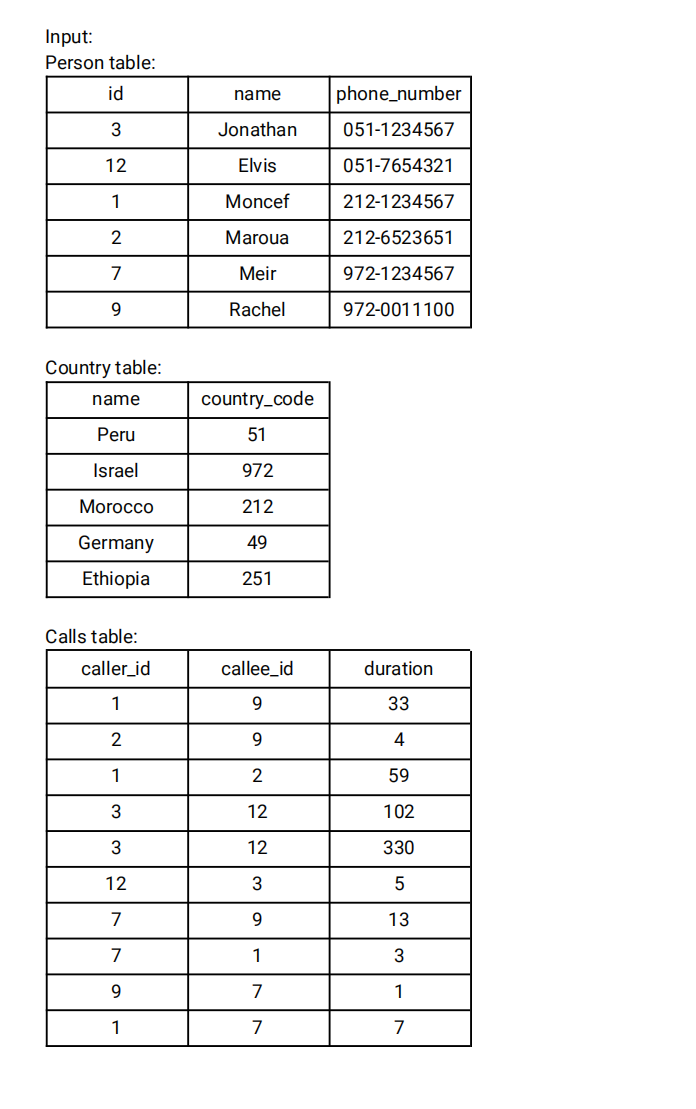
select distinct name from Customers C join orders o where C.id NOT IN(Select customer\_id from orders) ;



**QUERY:-**

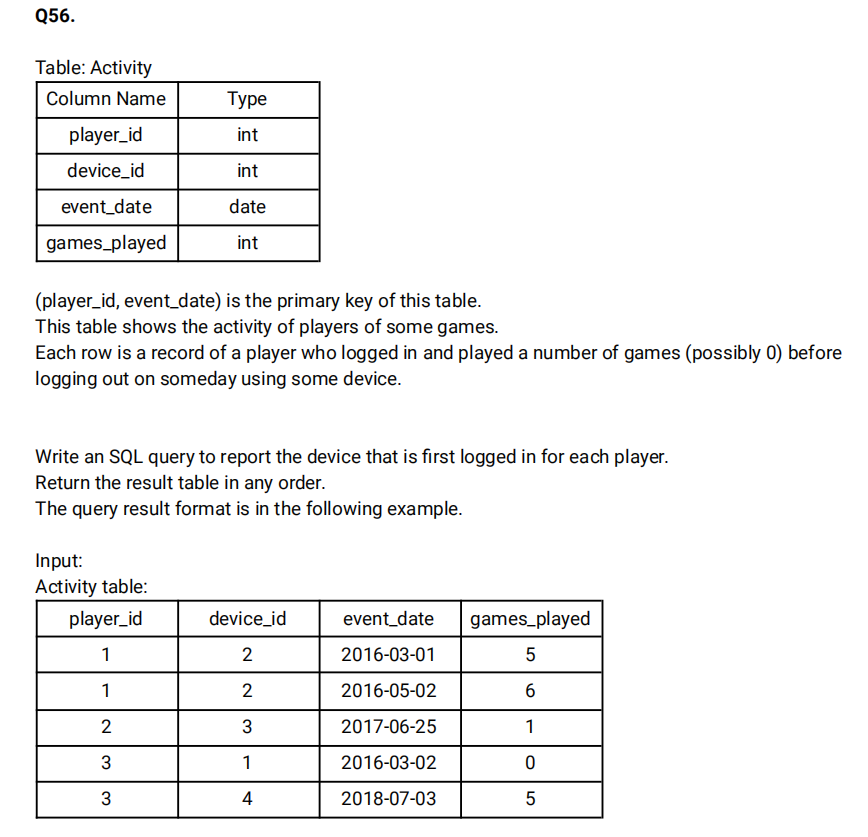
select employee\_id, COUNT(team\_id) OVER(PARTITION BY team\_id) as team\_size from employee order by employee\_id;





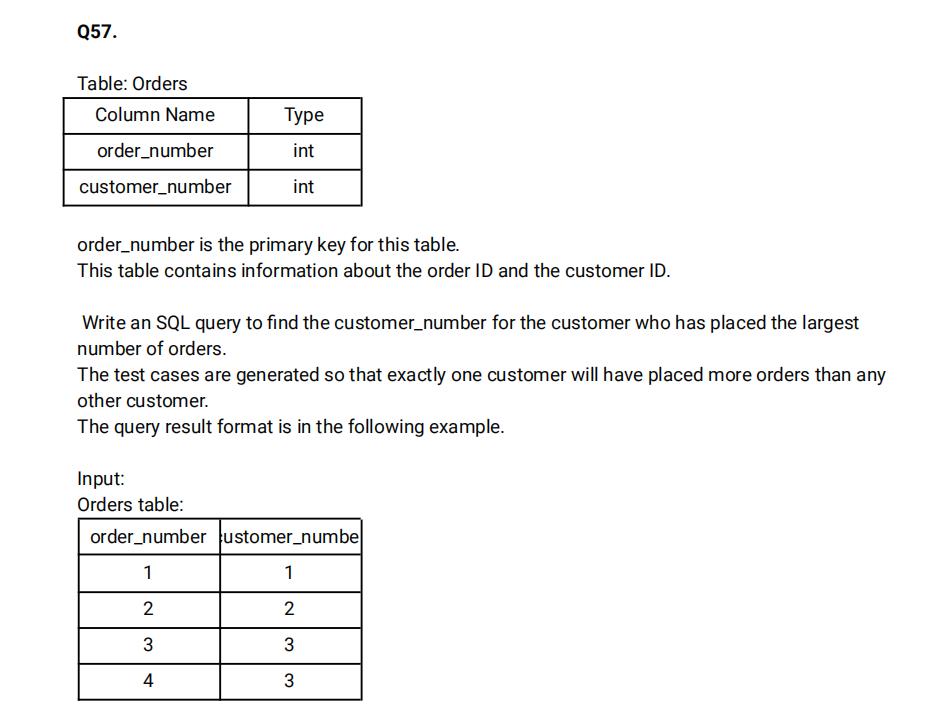
**QUERY:-**

 select c.name as country from person p join country c on SUBSTRING(phone\_number,1,3)=c.country\_code join calls ca on p.id IN (ca.caller\_id,ca.callee\_id) group by c.name HAVING AVG(duration)>(select AVG(duration) from calls);



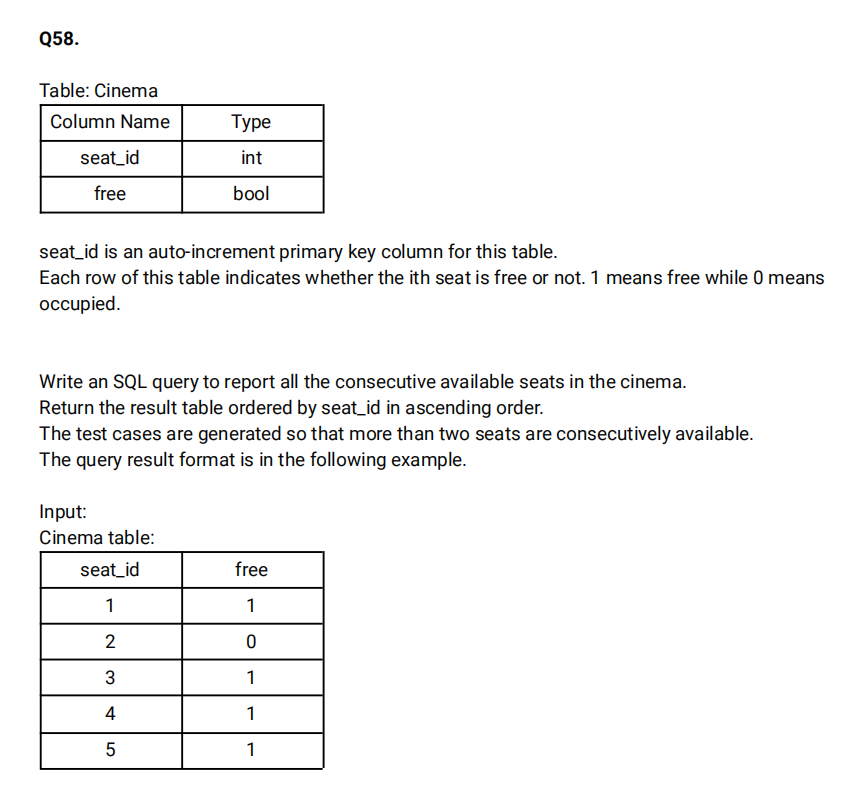
**QUERY:-**

select player\_id,device\_id FROM (select player\_id,device\_id, DENSE\_RANK() OVER (partition by player\_id order by event\_date ASC) as cnt from Activity)temp WHERE cnt=1;



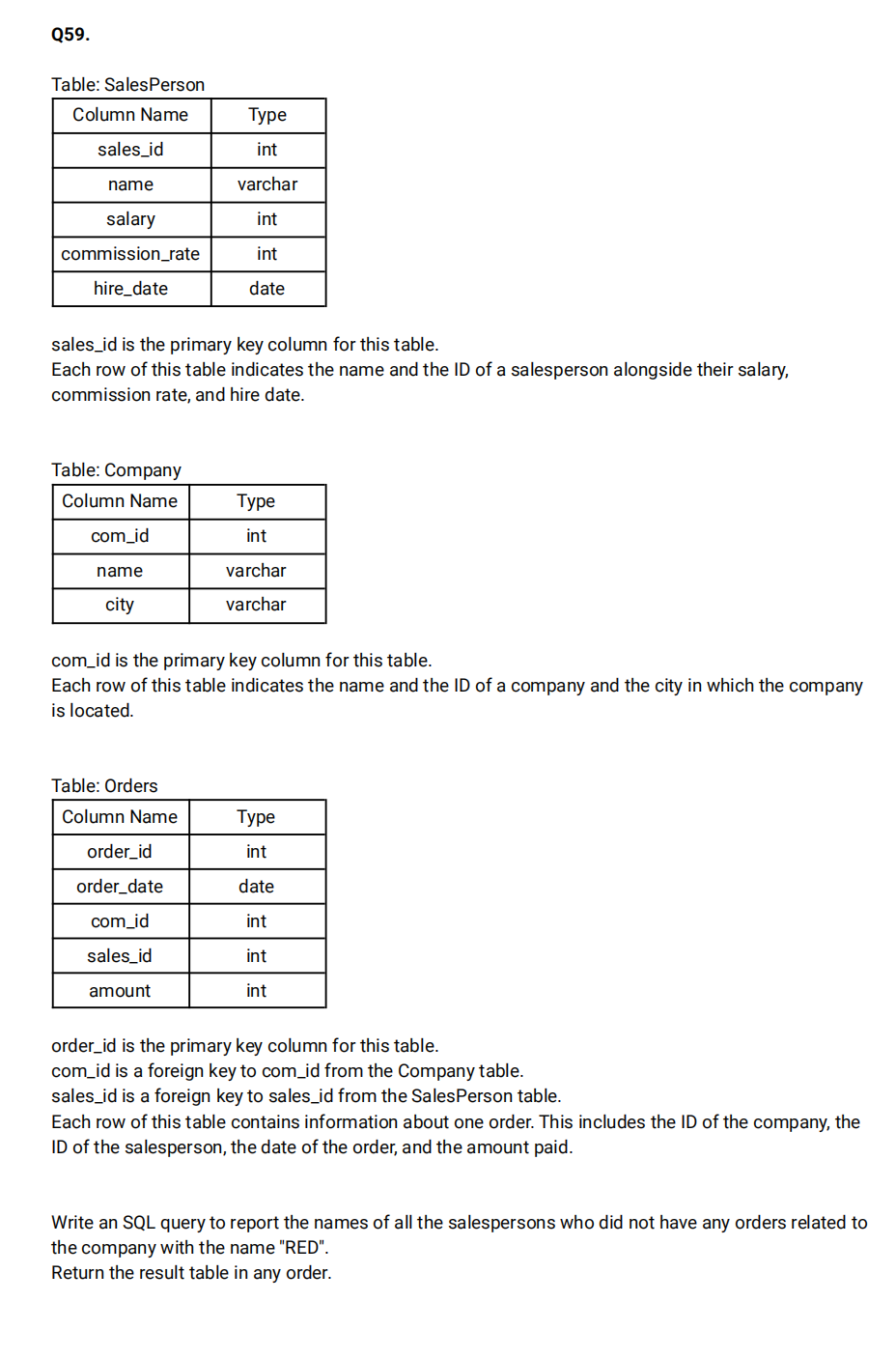
**QUERY:-**

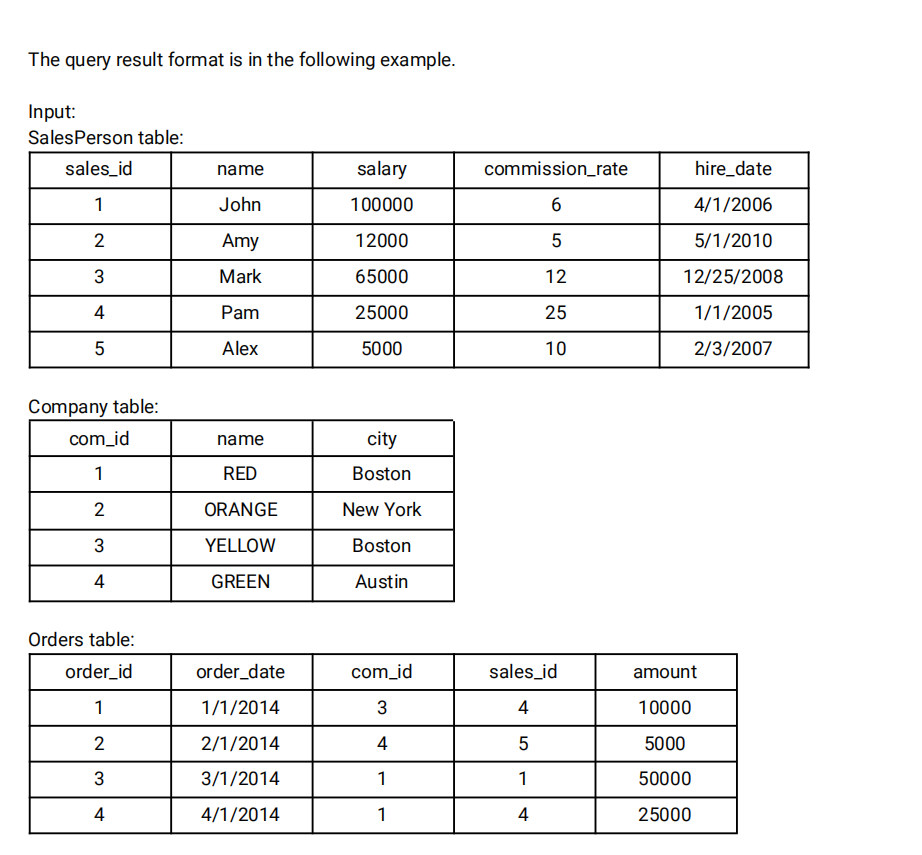
 select customer\_number from (select customer\_number,count(order\_number) as cnt from Orders group by customer\_number order by cnt desc limit 1)t;



**QUERY:-**

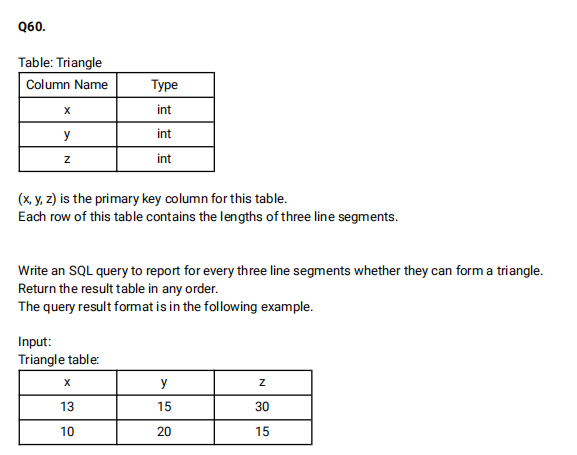
select  s1.seat\_id from cinema s1 join cinema s2 on s1.seat\_id=s2.seat\_id+1 and s1.free <> 0;





**QUERY:-**

 select s.name from salesperson s where s.sales\_id NOT IN (select o.sales\_id from Orders o join company c on o.com\_id=c.com\_id where c.name='Red');



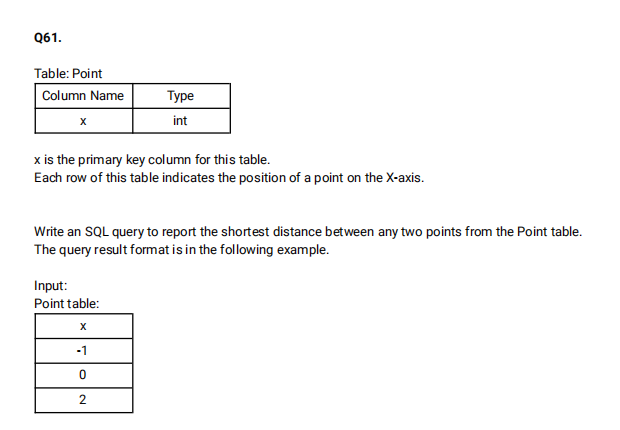
**QUERY:-**

 SELECT x,y,z,CASE

 when (x+y>z AND x+z>y AND y+z>x) then 'Yes'

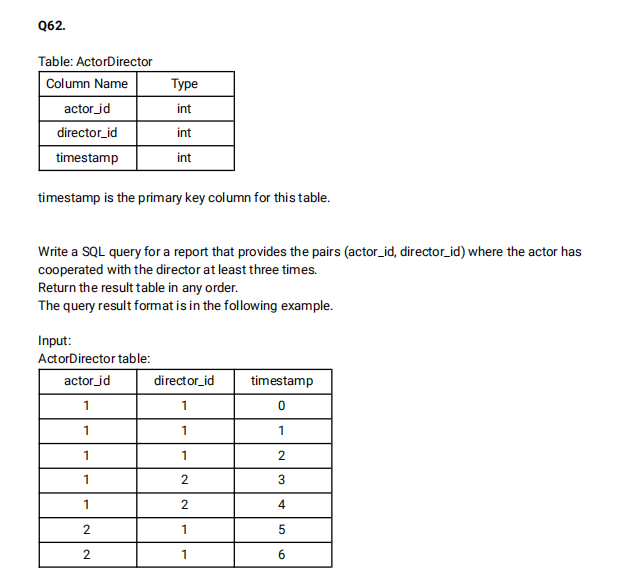
 else 'No'

 end as triangle FROM Triangle;



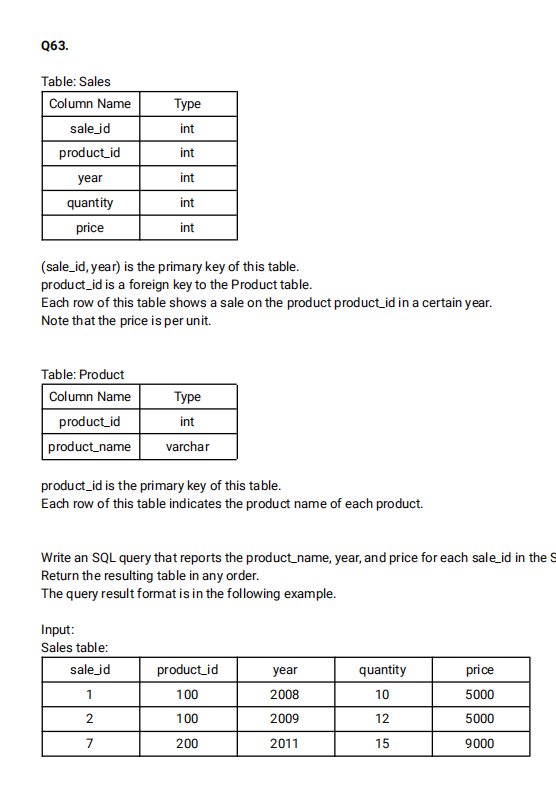
**QUERY:-**

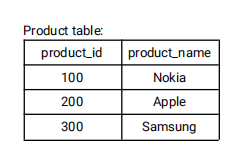
select MIN(shortest) as shortest from (select  LEAD(a.x) OVER(order by a.x) -b.x as shortest from point a join point b on a.x=b.x)c;



**QUERY:-**

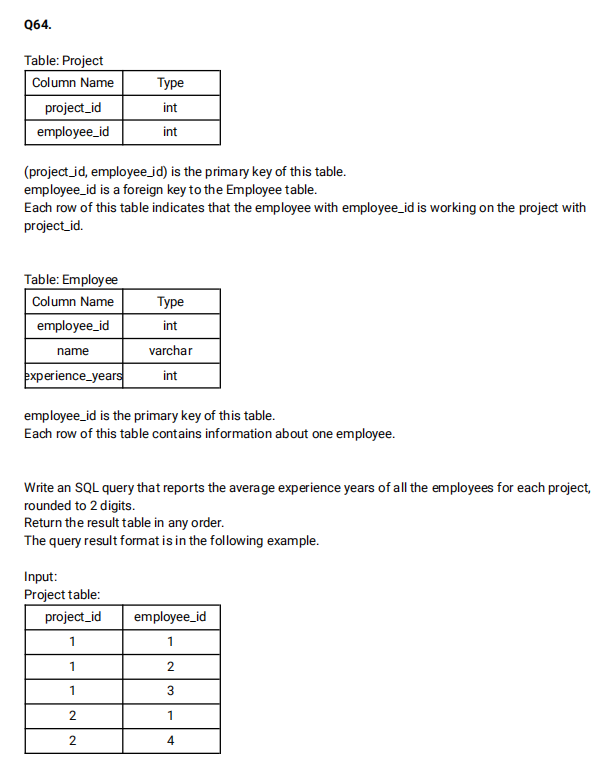
select actor\_id,director\_id from (select actor\_id,director\_id,COUNT(timestamp) as cnt from Actordirector group by actor\_id,director\_id) a where cnt >=3;

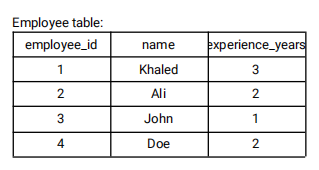




**QUERY:-**

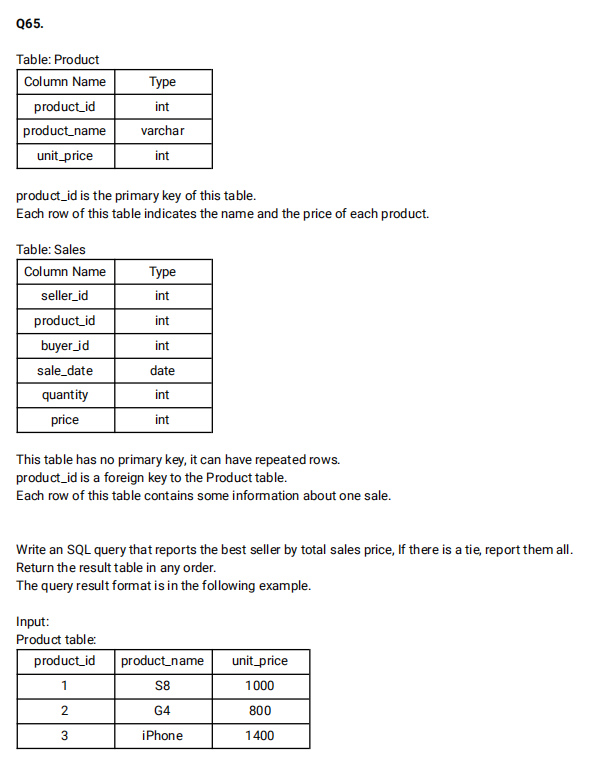
select product\_name,year,price from Sales s join product p on s.product\_id=p.product\_id;

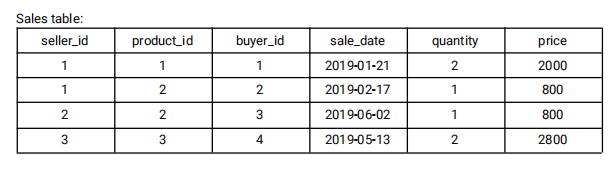




**QUERY:-**

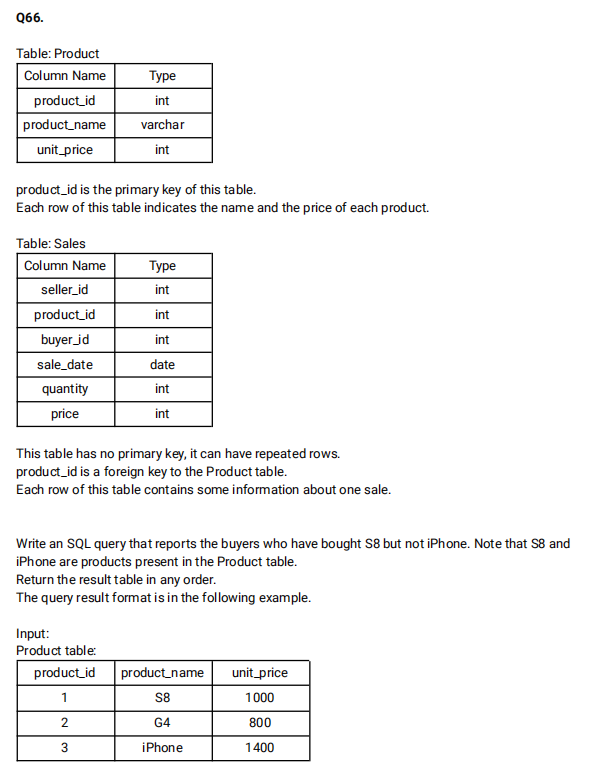
select project\_id,Round(AVG(experience\_years),2) as average\_years from Employee e join project p on e.employee\_id=p.employee\_id group by project\_id;

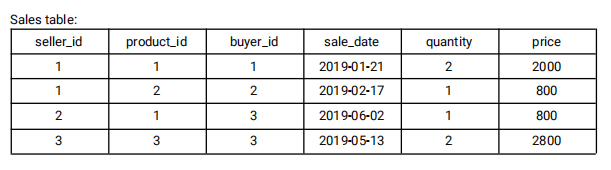




**QUERY:-**

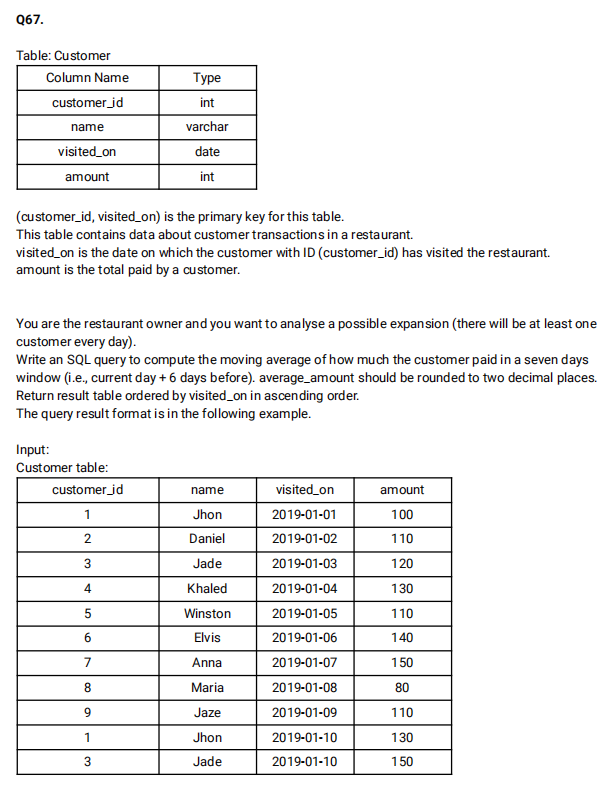
select seller\_id from (select seller\_id,sum,RANK() over( order by sum desc) as rnk from (select seller\_id, SUM(price) as sum from sales s join product p on s.product\_id=p.product\_id group by seller\_id)a)b where rnk=1 ;





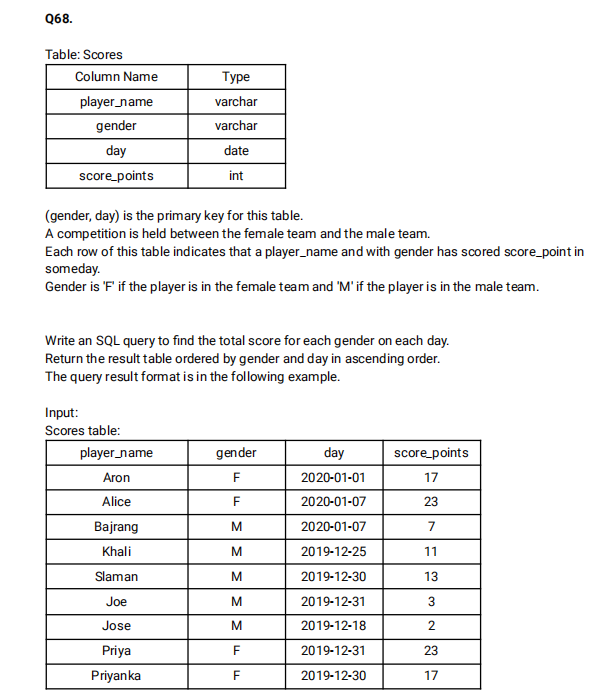
**QUERY:-**

select buyer\_id from sales s join product p on s.product\_id=p.product\_id where p.product\_name ='S8' AND p.product\_name <> 'iphone';



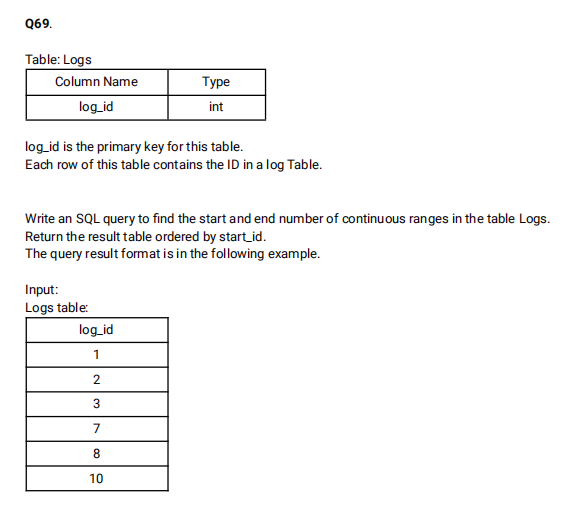
**QUERY:-**

select c1.visited\_on, SUM(c1.amount) as amount, round(AVG(c2.amount),2) as average\_amount from (select visited\_on,sum(amount) as amount from customer group by visited\_on)c1 join (select visited\_on,sum(amount) as amount from customer group by visited\_on)c2 on DATEDIFF(c1.visited\_on,c2.visited\_on) between 0 and 6 group by c1.visited\_on having count(c2.amount)=7 order by c1.visited\_on;



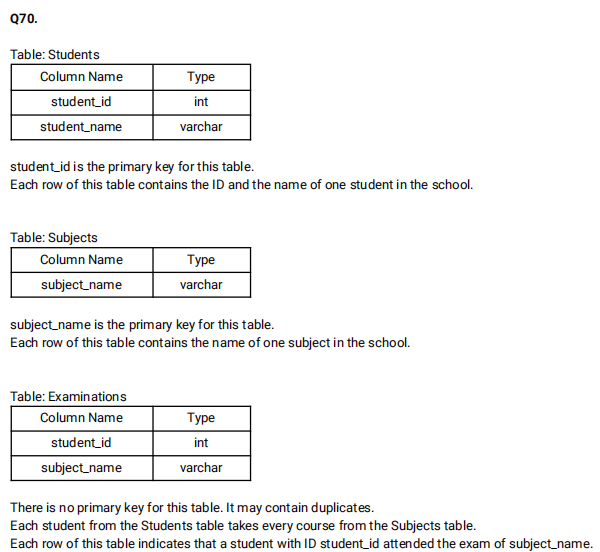
**QUERY:-**

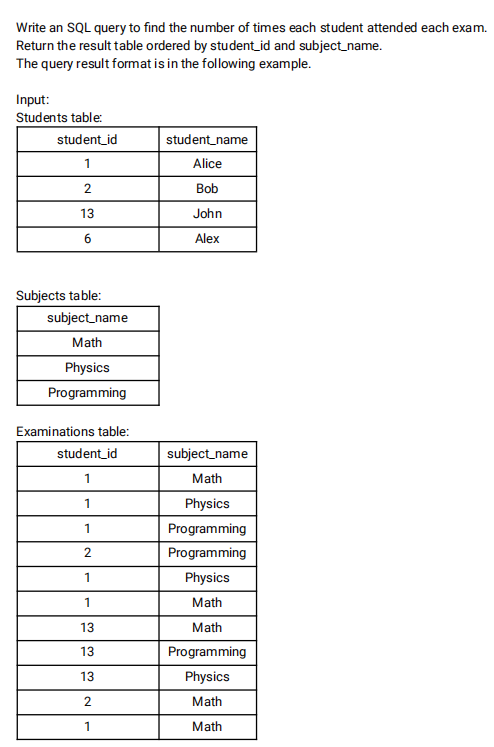
select gender,day,SUM(score\_points) over(partition by gender order by gender,day) as total from Scores;



**QUERY:-**

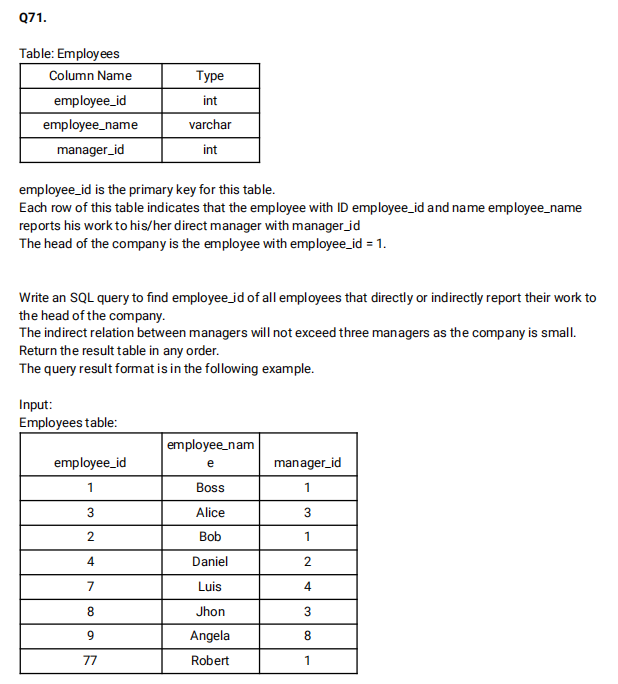
select min(log\_id) as start\_id,max(log\_id) as end\_id from (select log\_id,rn,log\_id-rn as diff from (select log\_id, ROW\_NUMBER() OVER() as rn from Logs)l1)l2 GROUP BY diff;





**QUERY:-**

select s.student\_id,student\_name,s1.subject\_name,count(e.student\_id) as attended\_exams from students s join subjects s1 left join examinations e on e.student\_id=s.student\_id and s1.subject\_name=e.subject\_name GROUP BY s.student\_id,student\_name,subject\_name order by s.student\_id,subject\_name;



**QUERY:-**

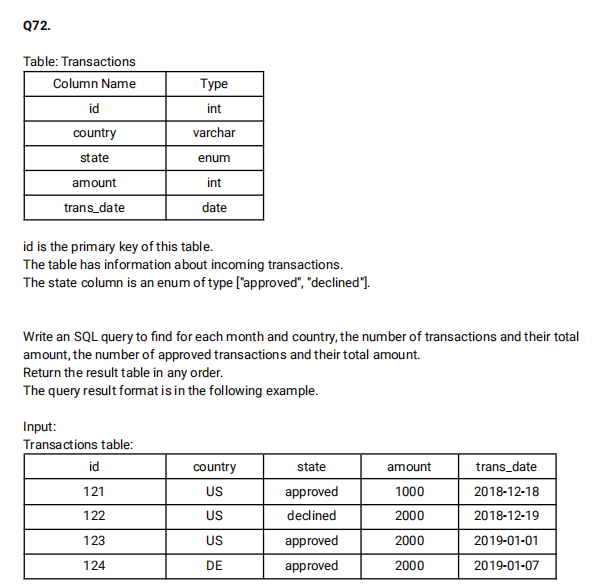
select e4.emp\_id  FROM

((select e2.employee\_id as e\_id,e1.manager\_id as m\_id from Employees e1 join Employees e2 where e2.manager\_id=e1.employee\_id and e2.employee\_id <>1)e3

  RIGHT join

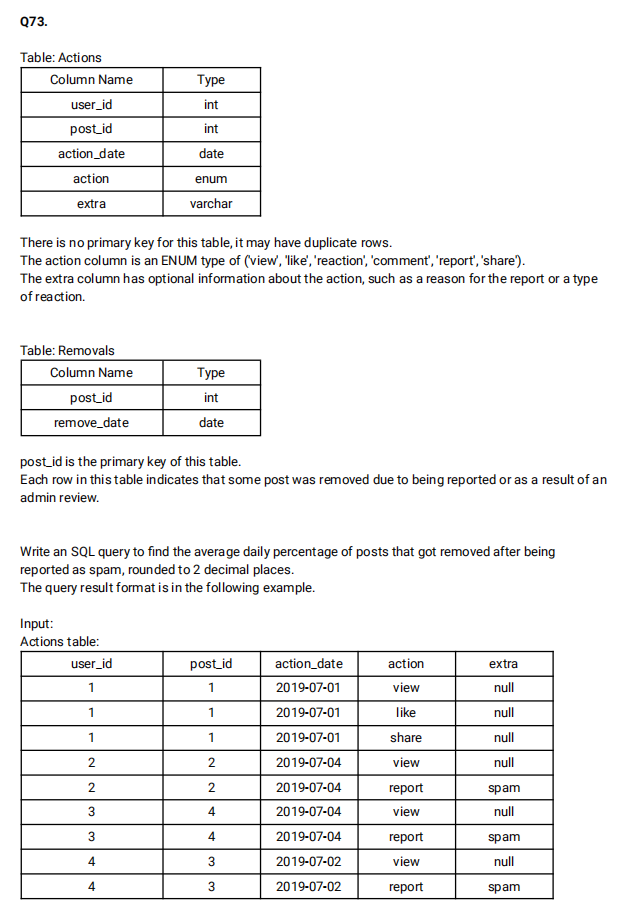
(select e2.employee\_id as emp\_id,e1.manager\_id as mng\_id from Employees e1 join Employees e2 where e2.manager\_id=e1.employee\_id and e2.employee\_id <>1)e4

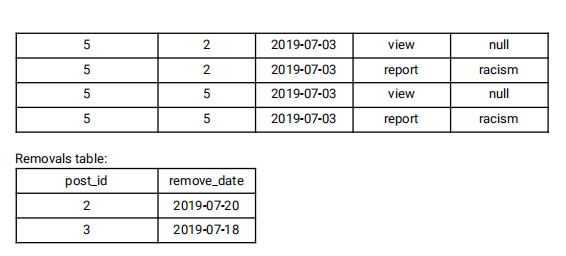
 ON e4.mng\_id=e3.e\_id) where m\_id is null or m\_id=1;



**QUERY:-**

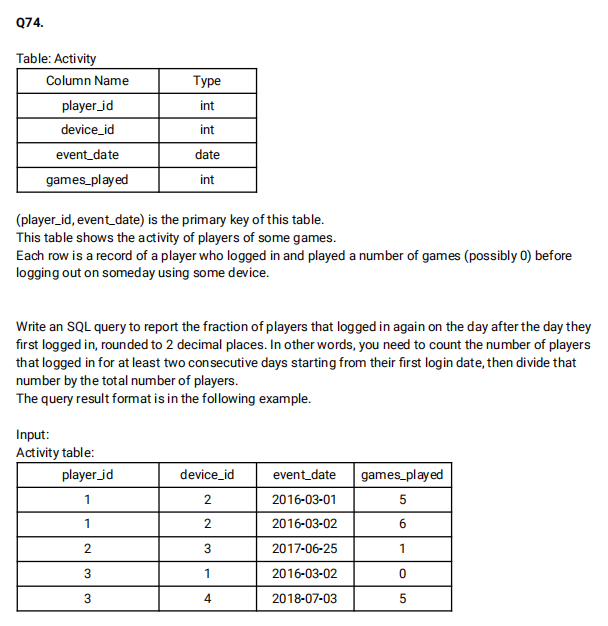
select EXTRACT(YEAR\_MONTH from trans\_date) as month,country,count(amount) as trans\_count,sum(case when state='approved' then 1 else 0 end) as approved\_count,sum(amount) as trans\_total\_amount, sum(case when state='approved' then amount else 0 end)as approved\_amount from Transactions GROUP BY month,country





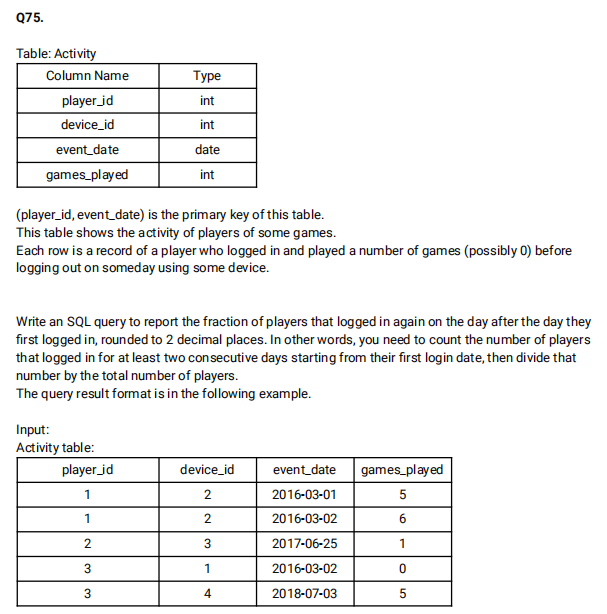
**QUERY:-**

select ROUND(AVG(cnt1),2) as avg\_daily\_percent from (select (count(b.post\_id)\*100)/(count(a.post\_id)) as cnt1 from Actions a left join removals b on a.post\_id=b.post\_id where extra='spam' group by action\_date)c;



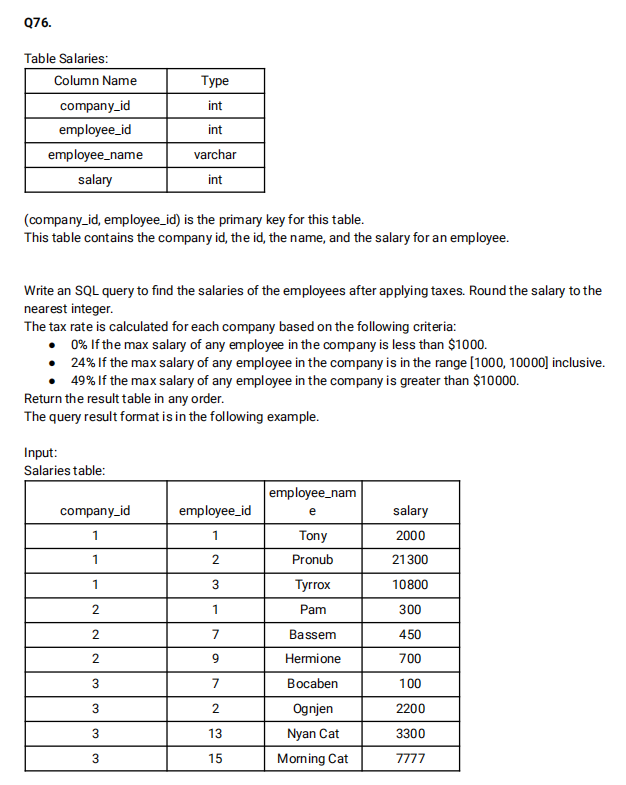
**QUERY:-**

select ROUND(count(distinct b.player\_id)/count(distinct a.player\_id),2) as fraction from Activity a left join Activity b on b.player\_id=a.player\_id and a.device\_id=b.device\_id and DATEDIFF(a.event\_date,b.event\_date)<>0



**QUERY:-**

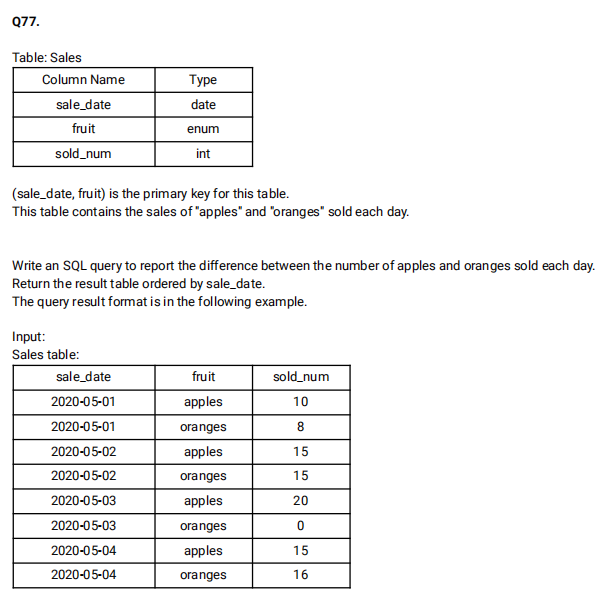
select ROUND(count(distinct b.player\_id)/count(distinct a.player\_id),2) as fraction from Activity a left join Activity b on b.player\_id=a.player\_id and a.device\_id=b.device\_id and DATEDIFF(a.event\_date,b.event\_date)<>0;



**QUERY:-**

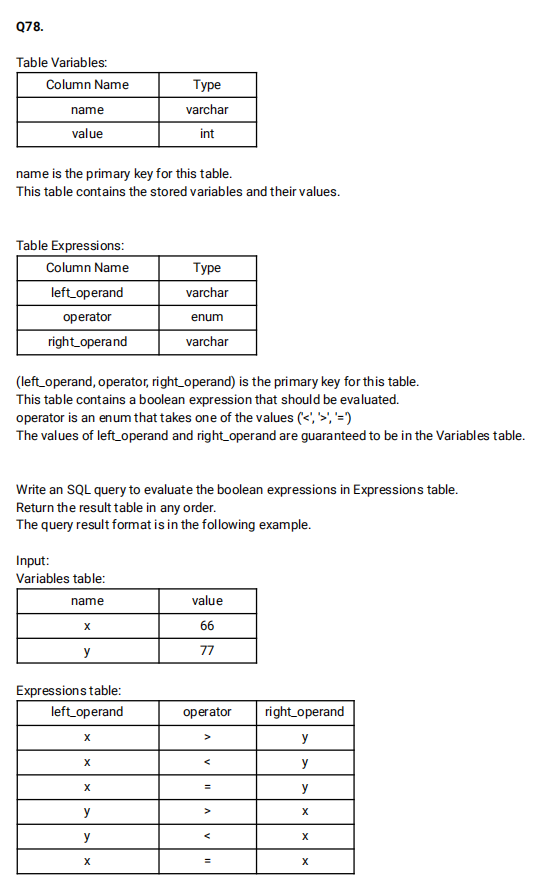
select s1.company\_id,s1.employee\_id,s1.employee\_name, ROUND(CASE when s2.max\_salary BETWEEN 1000 AND 10000 then 0.76\*salary when s2.max\_salary>10000 then 0.51\*salary

else salary end) as salary from salaries s1 join (select company\_id,MAX(salary) as max\_salary from salaries s2 group by company\_id)s2 on s1.company\_id=s2.company\_id order by company\_id;



**QUERY:-**

select sale\_date ,(sum(case when fruit='apples' then sold\_num else 0 end) - sum(case when fruit='oranges' then sold\_num else 0 end)) as diff from Sales GROUP BY sale\_date;



**QUERY:-**

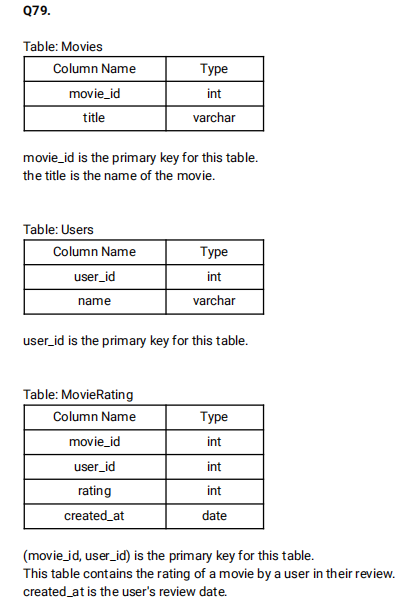
select left\_operand,operator,right\_operand,

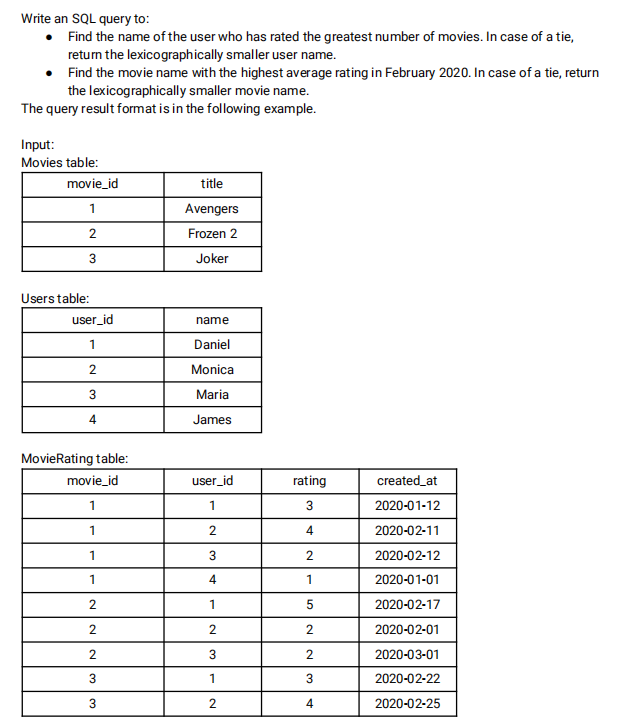
(case when operator='<' then if(v1.value<v2.value,'true','false')

when operator='>' then if(v1.value>v2.value,'true','false')

else if(v1.value=v2.value,'true','false') end) as value from Expressions e left join Variables v1 on e.left\_operand=v1.name

left join Variables v2 on e.right\_operand=v2.name;



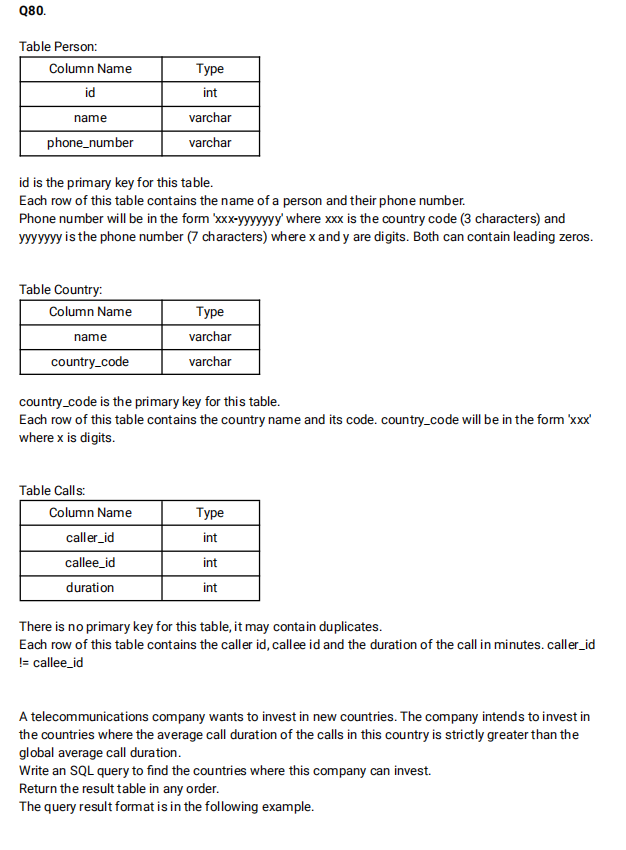


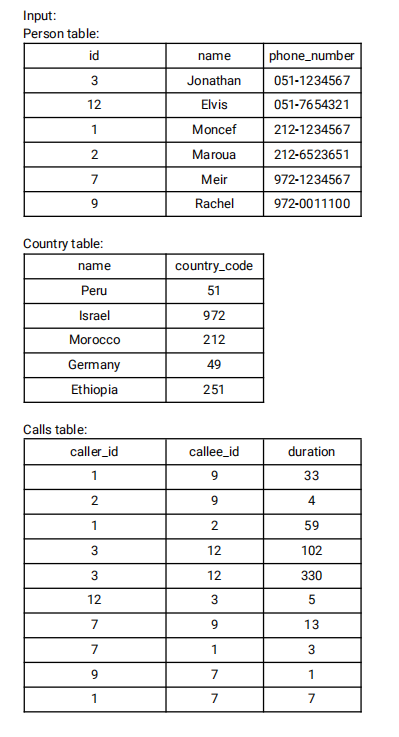
**QUERY:-**

select a.name as results from (select u.name,count(mr.rating) as cnt from Users u join movierating mr on mr.user\_id=u.user\_id group by u.name order by cnt desc,u.name limit 1)a

union ALL

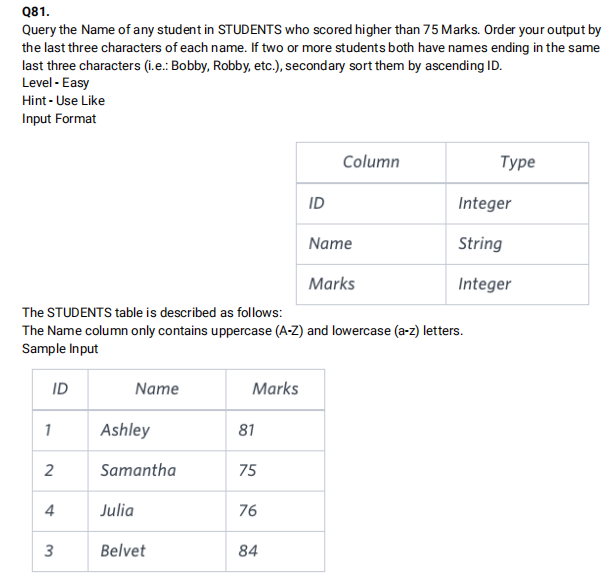
select b.title as results from (select title,AVG(mr.rating) as avg from movies m join movierating mr on mr.movie\_id=m.movie\_id where Extract(YEAR\_MONTH from created\_at)='202002' group by title order by avg desc,title limit 1)b;





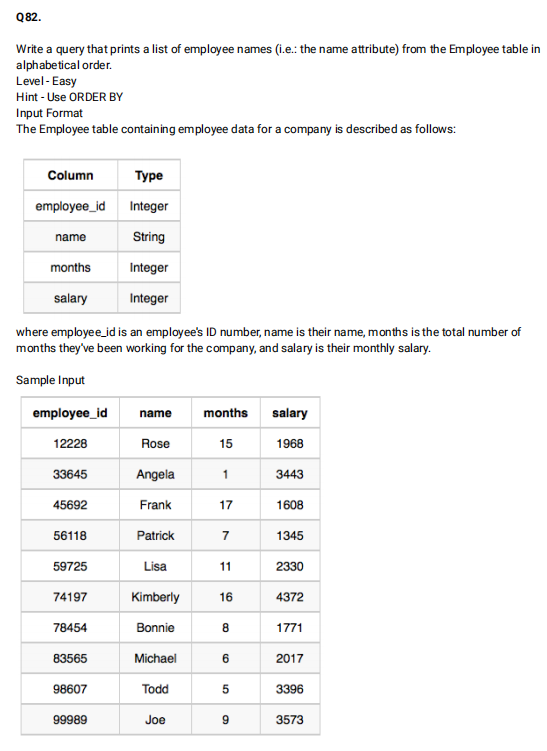
**QUERY:-**

 select c.name as country from person p join country c on SUBSTRING(phone\_number,1,3)=c.country\_code join calls ca on p.id IN (ca.caller\_id,ca.callee\_id) group by c.name HAVING AVG(duration)>(select AVG(duration) from calls);



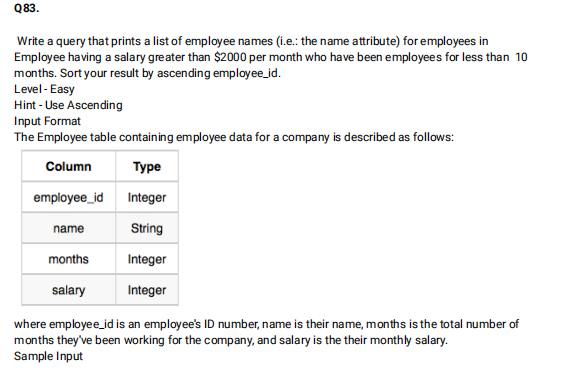
**QUERY:-**

select name from Students where marks>75 order by right(name,3),id;



**QUERY:-**

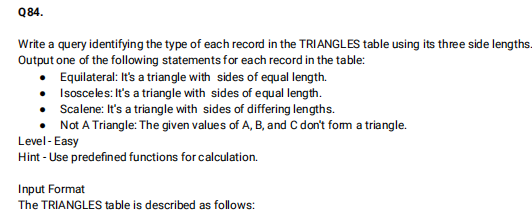
select name from Employees order by name;

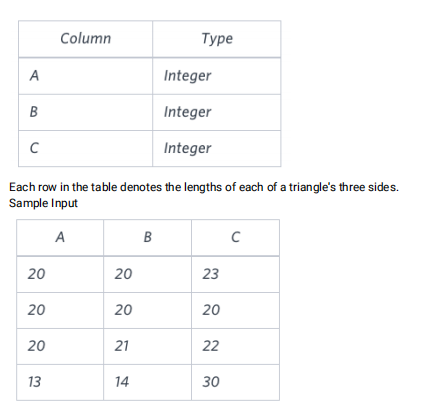




**QUERY:-**

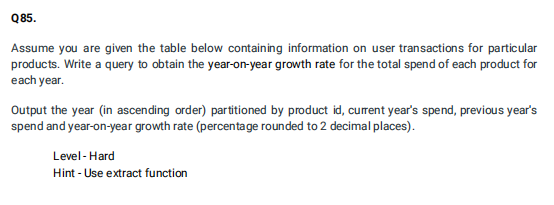
select name from Employees where months<10 and salary >2000;

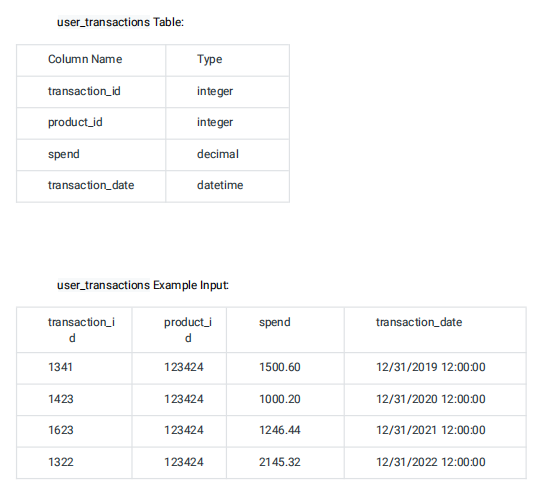




**QUERY:-**

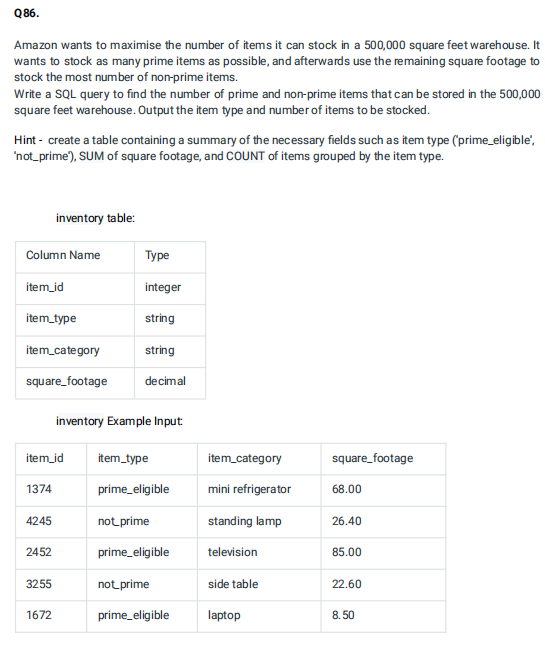
select IF((A+B<C or B+C<A or A+C<B),'Not a Triangle',(case when A=B=C then 'Equilateral' when A<>B and B<>C then 'Scalene' else 'Isoceles' end)) as Type from Triangle;





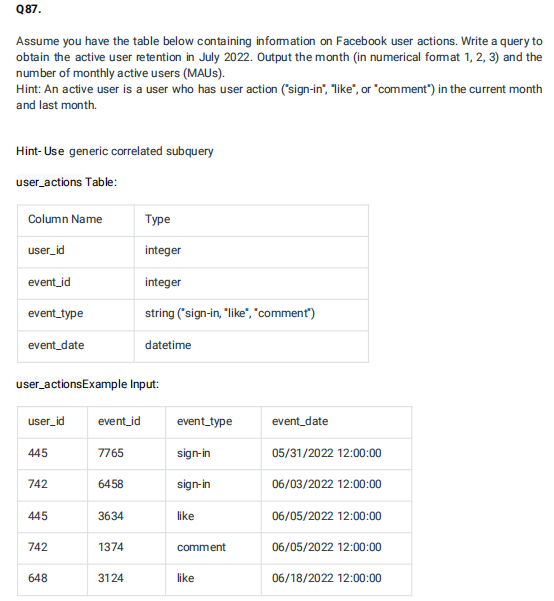
**QUERY:-**

select year,curr\_year\_spend,prev\_year\_spend,ROUND((((curr\_year\_spend-prev\_year\_spend)\*100)/prev\_year\_spend),2) as yoy\_rate from (select Extract(year from transaction\_date) as year,spend as curr\_year\_spend,lag(spend) over(order by EXTRACT(year from transaction\_date)) as prev\_year\_spend from user\_transactions order by year)a;



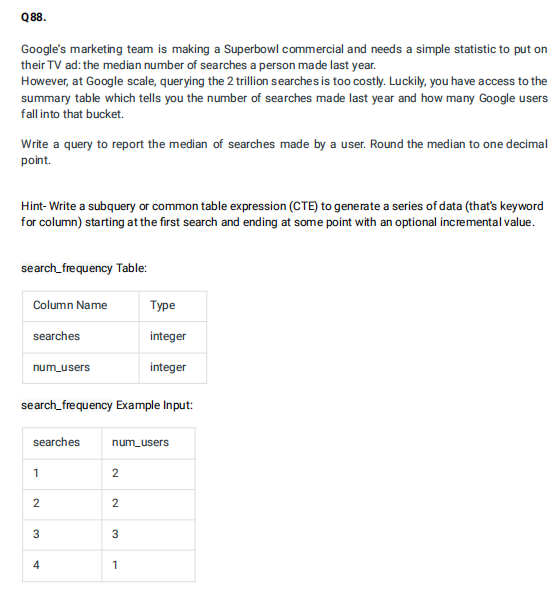
**QUERY:-**

select item\_type,count(item\_id) as item\_count from Inventory where square\_footage<500000 GROUP BY item\_type;



**QUERY:-**

select EXTRACT(MONTH from event\_date) as month,sum(case when event\_type='like' or event\_type='comment' or event\_type='sign-in' then 1 else 0 end) as monthly\_active\_users from user\_actions group by month;



**QUERY:-**

select AVG(searches) from (select searches, ROW\_NUMBER() OVER(order by searches) as rn\_asc,ROW\_NUMBER() OVER(order by searches desc) rn\_desc from search\_frequency order by searches)a where ABS(rn\_asc-rn\_desc)<=1;





**QUERY:-**

WITH pay\_status as

 ((select a.user\_id, a.status ,paid from advertiser a left join daily\_pay d on d.user\_id=a.user\_id)

UNION

(select d.user\_id,a.status,paid from daily\_pay d left join advertiser a on d.user\_id=a.user\_id))

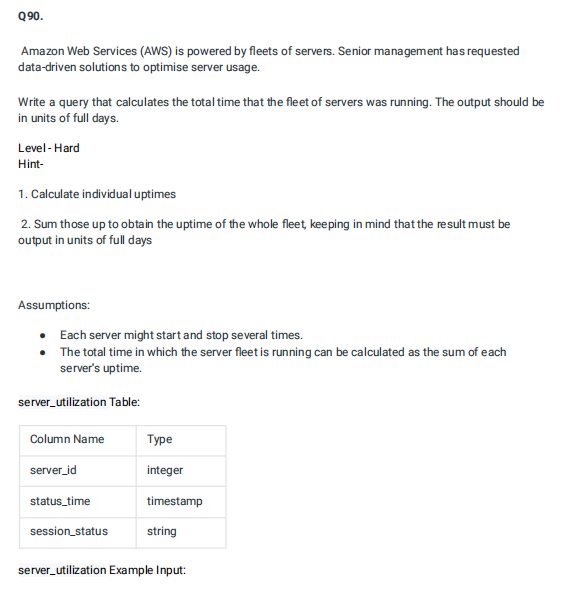
select user\_id, (case when paid is NULL then 'Churn'

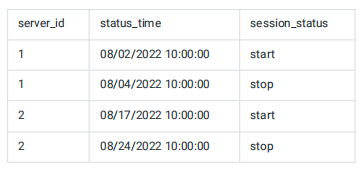
when STATUS ='Churn' and paid is not NULL then 'Resurrect'

when STATUS <> 'Churn' and paid is NOT NULL then 'Existing'

when STATUS is NULL then 'NEW' end) as new\_status

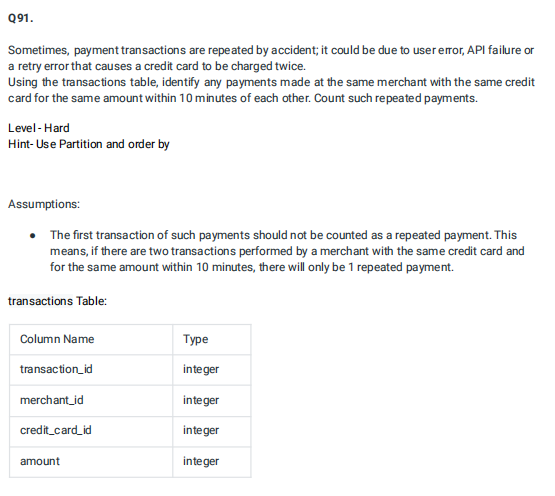
FROM pay\_status;

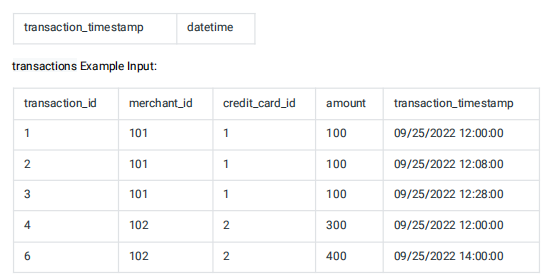




**QUERY:-**

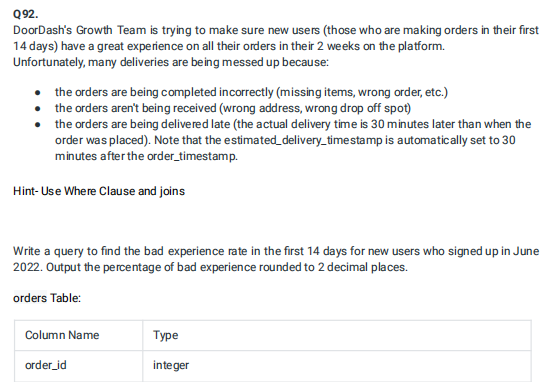
select sum(diff) as total\_uptime\_days from (select  s2.server\_id,DATEDIFF(s2.status\_time,s1.status\_time) as diff from server\_utilization s1 join server\_utilization s2 on s2.server\_id=s1.server\_id where s2.session\_status='stop')a;

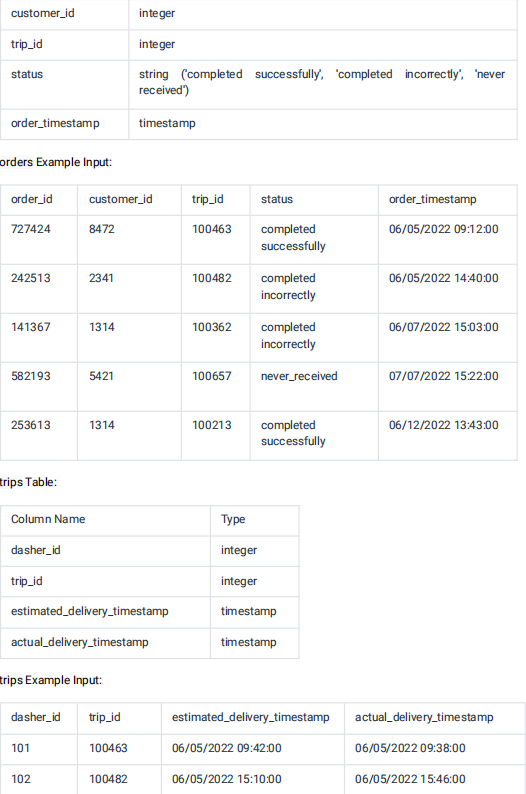


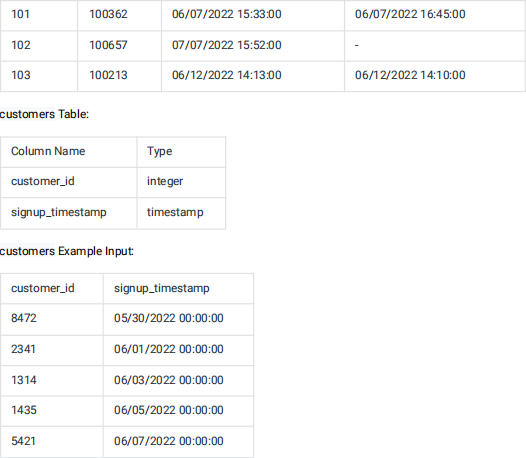


**QUERY:-**

select count(distinct amount) as payment\_count from (select  t2.transaction\_id,t2.amount,TIMESTAMPDIFF(MINUTE,t2.transaction\_timestamp,t1.transaction\_timestamp) as t from transactions t1 join transactions t2 WHERE t1.transaction\_id <> t2.transaction\_id AND t1.merchant\_id=t2.merchant\_id AND t1.credit\_card\_id=t2.credit\_card\_id AND t1.amount=t2.amount)a where abs(t)<=10;

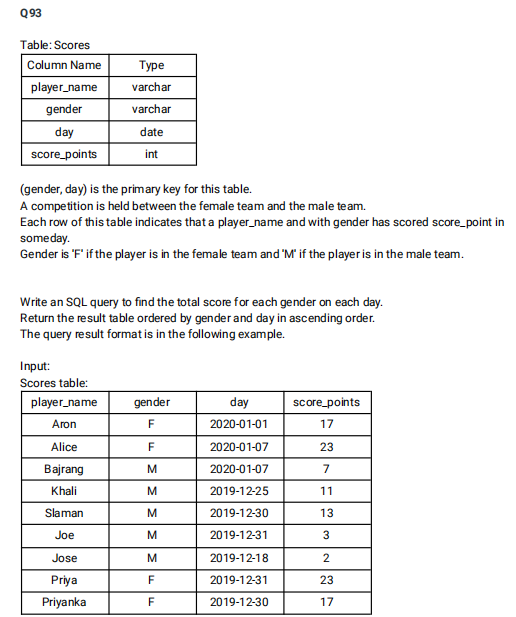






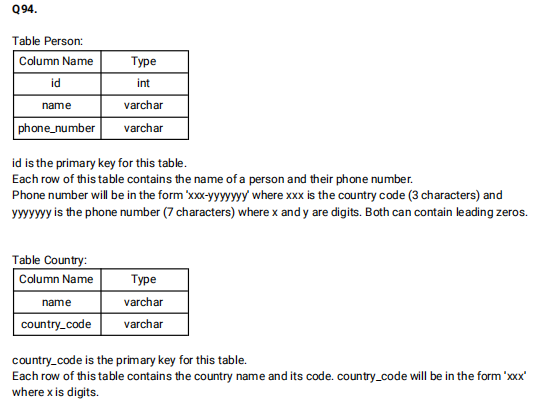
**QUERY:-**

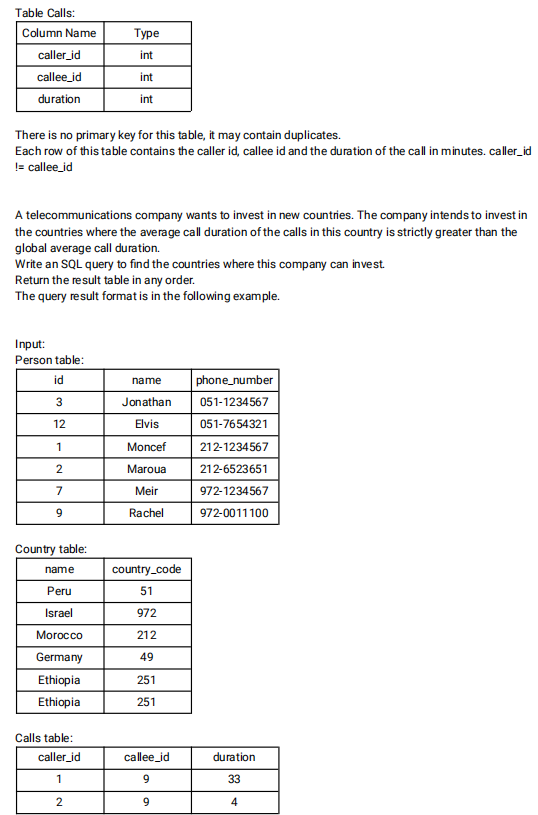
select (((cnt-good\_delivery)\*100)/cnt) as bad\_pct from (select COUNT(a.trip\_id) as cnt,sum(case when a.status='completed successfully' AND TIMESTAMPDIFF(MINUTE,estimated\_delivery\_timestamp,actual\_delivery\_timestamp)>=0 then 1 else 0 end) as good\_delivery from orders a left join trips b on b.trip\_id=a.trip\_id)a;

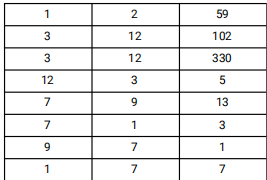


**QUERY:-**

 select gender,day,SUM(score\_points) over(partition by gender order by gender,day) as total from Scores;

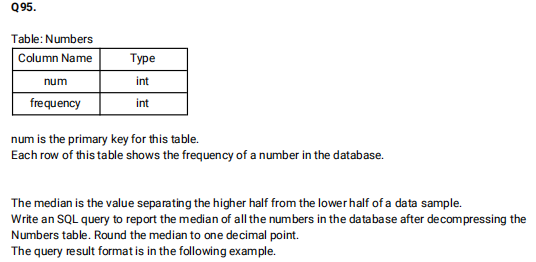


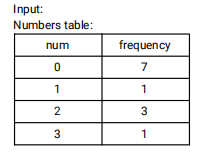




**QUERY:-**

 select c.name as country from person p join country c on SUBSTRING(phone\_number,1,3)=c.country\_code join calls ca on p.id IN (ca.caller\_id,ca.callee\_id) group by c.name HAVING AVG(duration)>(select AVG(duration) from calls)





**QUERY:-**

with cte as(

  select \* from numbers

  union all

  select num, freq - 1

  from cte where freq > 1

), cte2 as(

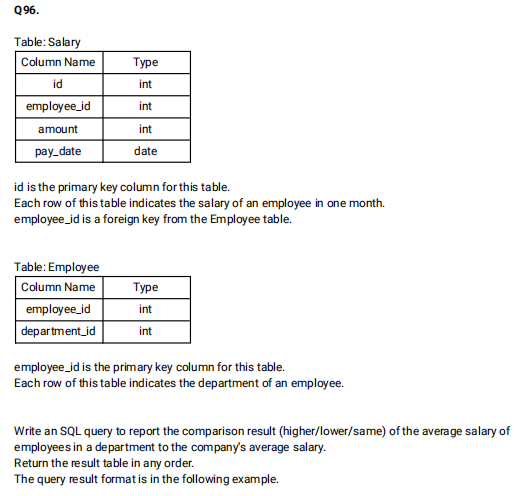
  select \*, rn = row\_number() over(order by num), cn = count(1) over()

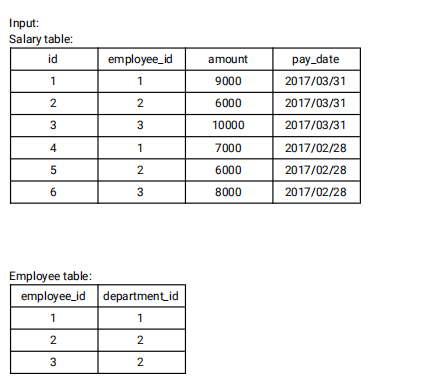
  from cte

)

select avg(num) from cte2

where rn between cn\*1.0/2 and cn\*1.0/2 + 1

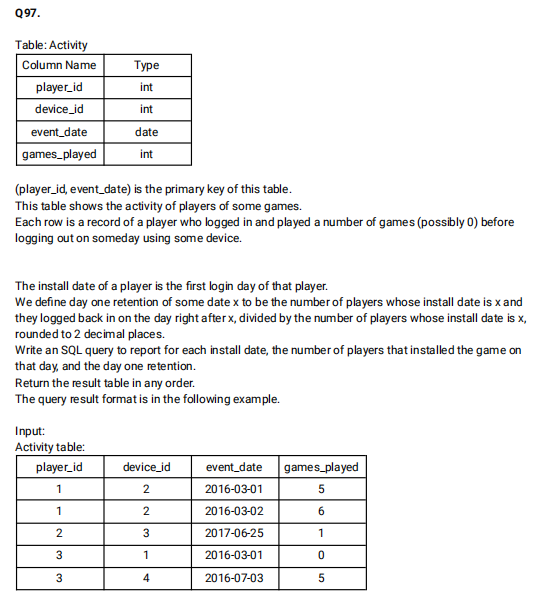




**QUERY:-**

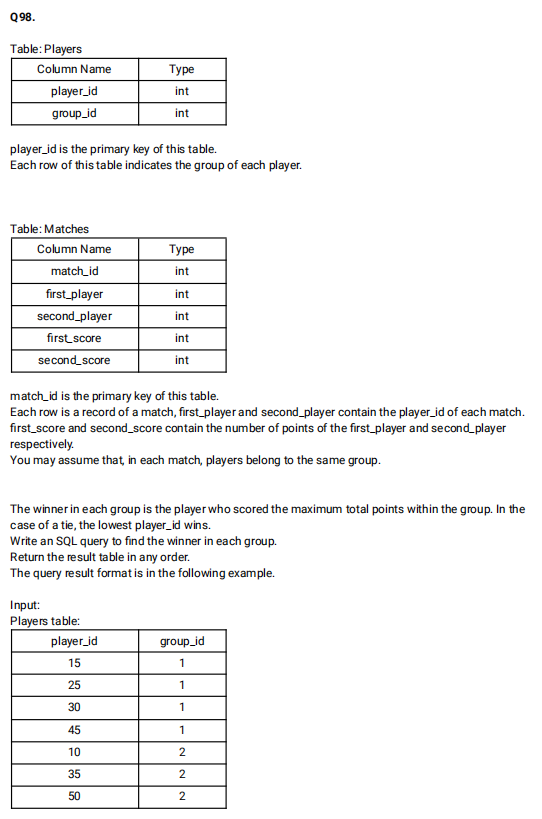
select dept\_salary.pay\_month,dept\_salary.department\_id,(case when dept\_avg>comp\_avg then 'Higher' when dept\_avg<comp\_avg then 'Lower' else 'Same' end) as comparison  from ((select e.department\_id,EXTRACT(YEAR\_MONTH from pay\_date) as pay\_month,AVG(amount) as dept\_avg from Salary s join Employee e on e.employee\_id=s.employee\_id group by e.department\_id,pay\_month) as dept\_salary join

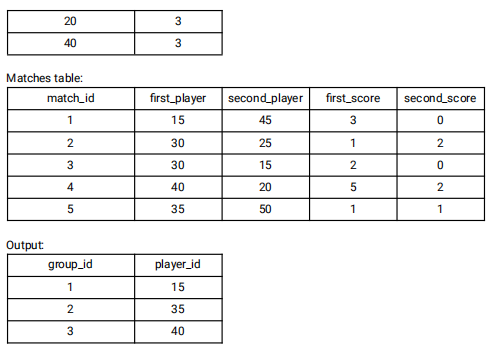
(select AVG(amount) as comp\_avg,EXTRACT(year\_month from pay\_date) as pay\_month from Salary GROUP BY pay\_month) as comp\_salary on dept\_salary.pay\_month=comp\_salary.pay\_month) order by dept\_salary.department\_id;



**QUERY:-**

select a.install\_date,count(a.install\_date) as installs,ROUND(count(b.event\_date)/count(\*),2) from (select player\_id,min(event\_date) as install\_date from Activity GROUP BY player\_id)a left join Activity b on DATE\_ADD(a.install\_date, INTERVAL 1 DAY)=b.event\_date AND a.player\_id=b.player\_id group by a.install\_date order by a.install\_date;





**QUERY:-**

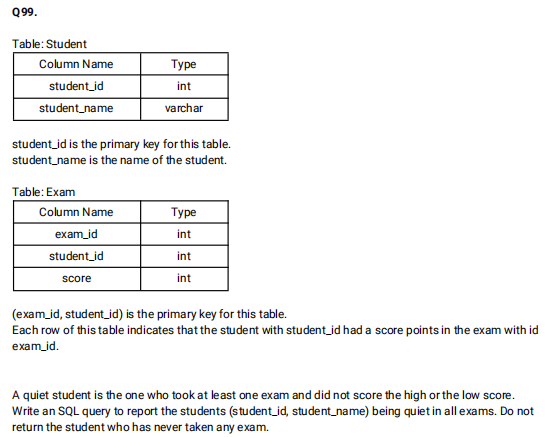
SELECT group\_id,player\_id

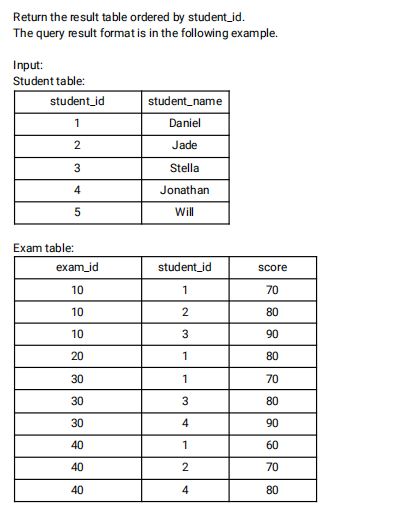
FROM   (SELECT p.group\_id,ps.player\_id,  Sum(ps.score) AS score   FROM players p INNER JOIN (SELECT first\_player AS player\_id,  first\_score  AS score  FROM   matches

  UNION ALL

SELECT second\_player AS player\_id, second\_score  AS score FROM   matches) ps  ON  p.player\_id = ps.player\_id   GROUP  BY ps.player\_id  ORDER  BY group\_id, score DESC, player\_id) top\_scores

GROUP  BY group\_id;





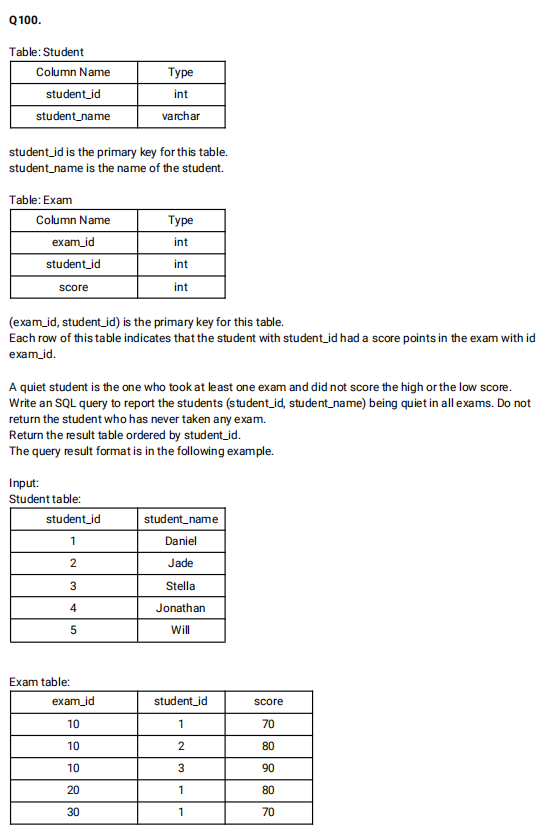
**QUERY:-**

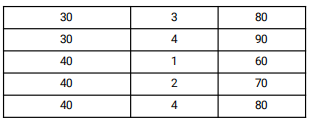
select distinct s.student\_id,s.student\_name

from Student s inner join Exam e

on s.student\_id = e.student\_id

where s.student\_id not in (select e1.student\_id from Exam as e1 inner join (select exam\_id, min(score) as min\_score, max(score) as max\_score from Exam group by exam\_id) as e2 on e1.exam\_id = e2.exam\_id where e1.score = e2.min\_score or e1.score = e2.max\_score) order by student\_id





**QUERY:-**

select distinct s.student\_id,s.student\_name

from Student s inner join Exam e

on s.student\_id = e.student\_id

where s.student\_id not in (select e1.student\_id from Exam as e1 inner join (select exam\_id, min(score) as min\_score, max(score) as max\_score from Exam group by exam\_id) as e2 on e1.exam\_id = e2.exam\_id where e1.score = e2.min\_score or e1.score = e2.max\_score) order by student\_id;