

VINAYAKA PALLAKKI V

1.

Select * from CITY where populations > 100000 and CountryCode = 'USA';

2.

Select NAME from CITY where populations > 120000 and CountryCode = 'USA';

3.

Select * from CITY;

4.

Select * from CITY where ID='1661';

5.

Select * from CITY where populations > 100000 and CountryCode = 'JPN';

6.

Select NAME from CITY where CountryCode = 'JPN';

7.

Select CITY, STATE from Station;

8.

Select distinct (City) from STATION where id%2=0;

9.

Select sum(City) - sum(distinct (City)) from STATION;

10.

select CITY,LENGTH(CITY) from STATION order by Length(CITY) asc, CITY limit 1;

select CITY,LENGTH(CITY) from STATION order by Length(CITY) desc, CITY limit 1;

11.

select t.city from station t where lower(SUBSTR(city,1,1)) in ('a','e','i','o','u')

12.

SELECT DISTINCT CITY FROM STATION WHERE CITY REGEXP '[aeiouAEIOU]\$';

13.

SELECT DISTINCT CITY FROM STATION WHERE CITY NOT RLIKE '^([aeiouAEIOU]).*\$'

14.

SELECT DISTINCT CITY FROM STATION WHERE CITY REGEXP '[AEIOU]'

15.

SELECT DISTINCT city FROM station WHERE city RLIKE '^([aeiouAEIOU]).*\.^[^AEIOUaeiou]\$';

16.

SELECT DISTINCT CITY FROM STATION WHERE CITY NOT RLIKE '^[AEIOUaeiou].*\$';

17.

select s.product_id, p.product_name from sales s, product p where s.product_id = p.product_id group by s.product_id, p.product_name having min(s.sale_date) >= '2019-01-01' and max(s.sale_date) <= '2019-03-31'

18.

select distinct author_id as id from Views where author_id = viewer_id order by author_id;

19.

select round(100*d2.immediate_orders/count(d1.delivery_id), 2) as immediate_percentage
from Delivery d1,(select count(order_date) as immediate_orders from Delivery where (order_date =
customer_pref_delivery_date)) d2

20.

select ad_id, ifnull(round(sum(action ='Clicked')/sum(action !='ignored') *100,2),0) ctr
from ads group by ad_id order by ctr desc, ad_id

21.

select e1.employee_id, count(*) as team_size from Employee e1 left join Employee e2 on e1.team_id =
e2.team_id group by e1.employee_id;

22.

select Countries.country_name, (case when avg(weather_state) <= 15 then 'Cold' when
avg(weather_state) >= 25 then 'Hot' else 'Warm' end) as weather_type from Weather inner join Countries
on Weather.country_id = Countries.country_id where left(Weather.day,7) = '2019-11' group by
Countries.country_id ;

23.

```

select product_id, ifnull(round(sum(prices_sum) / sum(units), 2), 0) as average_price
from (
    select p.product_id as product_id, units, price * units as prices_sum
    from Prices p left join UnitsSold u
    on p.product_id = u.product_id and purchase_date between start_date and end_date
) as temp
group by product_id;

```

24.

```

SELECT a1.player_id, a1.event_date, SUM(a2.games_played) AS games_so_far
FROM activity a1
JOIN activity a2 ON a1.player_id = a2.player_id
AND a1.event_date >= a2.event_date
GROUP BY a1.player_id, a1.event_date
ORDER BY a1.player_id, a1.event_date

```

25.

```

SELECT player_id, device_id
FROM Activity WHERE (player_id, event_date) IN (SELECT player_id, MIN(event_date)
FROM Activity GROUP BY player_id)

```

26.

```

select p.product_name as product_name, o.sum_unit as unit from Products p join (select product_id,
sum(unit) as sum_unit from Orders where order_date >= '2020-02-01' and order_date < '2020-03-01'
group by product_id) o

```

27.

```

SELECT * FROM Users WHERE REGEXP_LIKE(mail, '^[a-zA-Z][a-zA-Z0-9\_\.\-]*@leetcode.com')

```

28.

```

select o.customer_id, name from Orders o join Product p on o.product_id = p.product_id join Customers c
on o.customer_id = c.customer_id group by 1, 2 having sum(case when date_format(order_date, '%Y-
%m')='2020-06' then price*quantity end) >= 100 and sum(case when date_format(order_date, '%Y-
%m')='2020-07' then price*quantity end) >= 100 ;

```

29.

```

select distinct title from (select title from Content cleft join TVProgram t on c.content_id = t.content_id
where Kids_content = 'Y' and content_type = 'Movies' and date_format(program_date, '%Y-%m') = '2020
06' ) x

```

30.

```
select q.id, q.year, ifnull(n.npv,0) as npvcfrom queries as q left join npv as n on (q.id, q.year) = (n.id, n.year)
```

31.

```
select Queries.id, Queries.year, ifnull(npv, 0) as npv
  from Queries left join NPV
    on Queries.id = NPV.id and Queries.year = NPV.year
 order by Queries.id;
```

32.

```
select unique_id, name from Employees left join EmployeeUNI on Employees.id = EmployeeUNI.id;
```

33.

```
select name, ifnull(sum(distance), 0) as travelled_distance
  from Users left join Rides
    on Users.id = Rides.user_id
 group by Users.id
 order by travelled_distance desc, name
```

34.

```
select p.product_name as product_name, o.sum_unit as unit from Products p
join
```

```
(select product_id, sum(unit) as sum_unit from Orders where order_date >= '2020-02-01' and order_date
< '2020-03-01'
group by product_id) o
```

```
on p.product_id = o.product_id
where o.sum_unit >= 100
```

35.

```
( SELECT u.name as results FROM MovieRating as m JOIN Users as u ON m.user_id = u.user_id
GROUP BY 1 ORDER BY COUNT(*) DESC, 1 LIMIT 1 ) UNION ( SELECT m.title FROM MovieRating as
mr JOIN Movies as m ON mr.movie_id = m.movie_id WHERE DATE_FORMAT(created_at, '%Y-%m') =
'2020-02' GROUP BY 1 ORDER BY AVG(rating) DESC, 1 LIMIT 1 )
```

36.

```
SELECT
  u.name,
  IFNULL(SUM(distance),0) as travelled_distance
FROM Users as u LEFT JOIN Rides as r
ON r.user_id = u.id
GROUP BY 1
```

ORDER BY 2 DESC, 1

37.

select unique_id, name from Employees left join EmployeeUNI on Employees.id = EmployeeUNI.id;

38.

select id, name from Students where department_id not in (select id from Departments);

39.

WITH caller as (

select from_id as person1, to_id as person2, duration

from Calls

UNION ALL

select to_id as person1, from_id as person2, duration

from Calls

),

unique_caller as (

select person1, person2, duration

from caller

where person1 < person2

)

select

person1, person2, count(*) as call_count, sum(duration) as total_duration

from unique_caller

group by person1, person2

40.

select p.product_id,

round(sum(p.price * u.units)/sum(u.units), 2) as average_price

from Prices p

left join UnitsSold u

on p.product_id = u.product_id and

datediff(u.purchase_date, p.start_date) >= 0 and

datediff(p.end_date, u.purchase_date) >= 0

group by p.product_id

41.

select warehouse_name, sum(volume) as volume from (

select w.name as warehouse_name, w.product_id, w.units * Width * Length * Height as volume

from Warehouse w left join Products p on w.product_id = p.product_id

```
) t group by warehouse_name;
```

42.

```
select date(sale_date) as sale_date,  
       sum(case when fruit = 'apples' then sold_num  
              when fruit = 'oranges' then -sold_num end) as diff  
from Sales  
group by 1  
order by 1  
;
```

43.

```
select round(  
  ifnull(  
    (  
      select count(distinct a.player_id)  
      from Activity as a join Activity as b  
      on a.player_id = b.player_id and datediff(b.event_date, a.event_date) = 1  
      where a.event_date = (  
        select min(event_date) from Activity where player_id = a.player_id  
      )  
    )  
    / -- divided by  
    ( select count(distinct player_id) from Activity ),  
  0),  
2)  
as fraction;
```

44.

```
SELECT e2.Name  
FROM Employee e1  
      JOIN Employee e2 ON e1.ManagerId = e2.Id  
GROUP BY e1.ManagerId  
HAVING count(e1.Id) >= 5;
```

45.

```
select  
  a.dept_name,  
  coalesce(count(student_id), 0) student_number  
from  
  department a
```

```
left join
  student b
on
  (a.dept_id = b.dept_id)
group by a.dept_name
order by student_number desc, a.dept_name asc;
```

46.

```
SELECT customer_id FROM customer GROUP BY customer_id HAVING COUNT( DISTINCT
product_key) = (SELECT COUNT(*) FROM product)
SELECT customer_id, COUNT( DISTINCT product_key) unique_product FROM customer GROUP BY
customer_id
```

47.

```
select project_id, employee_id
from Project
join Employee
using (employee_id)
where (project_id, experience_years) in (
  select project_id, max(experience_years)
  from Project
  join Employee
  using (employee_id)
  group by project_id)
```

48.

```
select Books.book_id, name from Books join Orders
  on Books.book_id = Orders.book_id
  where available_from < '2019-05-23'
  and dispatch_date between '2018-06-23' and '2019-06-23'
  group by Books.book_id
  having sum(quantity) < 10
union
select book_id, name from Books
  where available_from < '2019-05-23'
  and book_id not in (
    select distinct book_id from Orders where dispatch_date between '2018-06-23' and '2019-06-23'
  );
```

49.

```
select student_id, min(course_id) as course_id, grade from Enrollments
where (student_id, grade) in (
    select student_id, max(grade) from Enrollments group by student_id
)
group by student_id;
```

50.

```
select group_id, player_id from (
    select p.group_id, ps.player_id, sum(ps.score) as score
    from Players p,
        (
            select first_player as player_id, first_score as score
            from Matches
            union all
            select second_player, second_score
            from Matches
        ) ps
    where p.player_id = ps.player_id
    group by ps.player_id
    order by group_id, score desc, player_id
    -- limit 1 -- by default, groupby will pick the first one i.e. max score player here
) top_scores
group by group_id;
```

51.

```
SELECT
    name, population, area
FROM
    world
WHERE
    area >= 3000000 OR population >= 25000000
;
```

52.

```
SELECT
    name
FROM
    customer
WHERE
    referee_id <> 2 or referee_id IS NULL;
```


53.

```
select customers.name as 'Customers'           from
customers                                     where customers.id not in(           select
customerid from orders                       );
```

54.

```
select e1.employee_id, count(e2.employee_id) as team_size
from Employee e1
inner join Employee e2 on e1.team_id = e2.team_id
group by e1.employee_id, e2.team_id
```

55.

```
select c.name as country
from Person p
inner join Country c
on left (p.phone_number,3) = c.country_code
inner join (select caller_id as id, duration
            from Calls

            union all

            select callee_id as id, duration
            from Calls) phn
on p.id = phn.id
group by country
having avg(duration) > (select avg(duration) from Calls)
```

56.

```
[10:54 AM] U, Prashanth
SELECT
player_id, min(event_date) as first_login
FROM
Activity
group by player_id ;
```

57.

```
select
customer_number
from
```

```
(select customer_number, count(order_number) order_count
from orders group by customer_number) a
order by order_count desc limit 1
```

58.

```
select seat_id
from (
    select
        current.seat_id,
        case when exists (
            select 1
            from cinema previous
            where previous.seat_id = current.seat_id - 1
            and previous.free = 1)
            and current.free = 1 then current.seat_id - 1 else null end previous,
        case when exists (
            select 1
            from cinema next
            where next.seat_id = current.seat_id + 1
            and next.free = 1)
            and current.free = 1 then current.seat_id + 1 else null end next
        from cinema current) seats
where seats.previous is not null or seats.next is not null
```

59.

```
select SalesPerson.name
from SalesPerson
where sales_id NOT IN
(
    select sales_id from orders
    left join
        company
        ON company.com_id = orders.com_id
    where company.name = 'RED'
);
```

60.

```
SELECT *, if(x+y>z and x+z>y and y+z>x, 'Yes', 'No') AS triangle FROM triangle;
```

61.

```
select min(abs(p2.x-p1.x)) as shortest
```

```
from point p1, point p2
```

```
where p1.x != p2.x
```

62.

```
select actor_id, director_id
```

```
from ActorDirector
```

```
group by actor_id, director_id
```

```
having count(*) >= 3
```

63.

```
select product_name, year, price from Sales left join Product on Sales.product_id = Product.product_id;
```

64.

```
select project_id, employee_id
```

```
from Project
```

```
join Employee
```

```
using (employee_id)
```

```
where (project_id, experience_years) in (
```

```
    select project_id, max(experience_years)
```

```
    from Project
```

```
    join Employee
```

```
    using (employee_id)
```

```
    group by project_id)
```

65.

```
select seller_id from Sales group by seller_id
```

```
    having sum(price) = (
```

```
        select sum(price) from sales group by seller_id order by sum(price) desc limit 1
```

```
    );
```

66.

```
WITH t1 AS ( SELECT s.buyer_id, s.product_id, p.product_name FROM sales as s JOIN product as p ON  
s.product_id = p.product_id ) SELECT DISTINCT buyer_id FROM sales WHERE buyer_id IN (SELECT  
buyer_id FROM t1 WHERE product_name = 'S8') AND buyer_id NOT IN (SELECT buyer_id FROM t1  
WHERE product_name = 'iPhone')
```

67.

```
select visits.visited_on as visited_on, sum(c.amount) as amount, round(sum(c.amount) / 7.0, 2) as  
average_amount
```

```
from (
```

```
    select distinct visited_on from Customer
```

```
    where datediff(visited_on, (select min(visited_on) from Customer)) >= 6
```

```
) visits left join Customer c
on datediff(visits.visited_on, c.visited_on) between 0 and 6
group by visits.visited_on
order by visited_on;
```

68.

```
select s.gender, s.day, (select sum(score_points) from Scores where gender = s.gender and day <=
s.day) as total
from Scores s
group by gender, day
order by gender, day;
```

69.

```
select log_start.log_id as START_ID, min(log_end.log_id) as END_ID from
(select log_id from logs where log_id - 1 not in (select * from Logs)) log_start,
(select log_id from logs where log_id + 1 not in (select * from Logs)) log_end
where log_start.log_id <= log_end.log_id
group by log_start.log_id;
```

70.

```
select Students.student_id, student_name, Subjects.subject_name, count(Examinations.student_id) as
attended_exams
from (Students join Subjects on 1=1) left join Examinations
on (Students.student_id, Subjects.subject_name) = (Examinations.student_id,
Examinations.subject_name)
group by Students.student_id, Students.student_name, Subjects.subject_name
order by Students.student_id, Subjects.subject_name;
```

71.

```
select employee_id as EMPLOYEE_ID from Employees where manager_id in
(select employee_id from Employees WHERE manager_id in
(select employee_id from Employees where manager_id =1))
and employee_id !=1
```

72.

```
select date_format(trans_date, '%Y-%m') as month, country, count(*) as trans_count,
sum(if(state = 'approved', 1, 0)) as approved_count, sum(amount) as trans_total_amount,
sum(if(state = 'approved', amount, 0)) as approved_total_amount
from Transactions
group by date_format(trans_date, '%Y-%m'), country;
```

73.

```
select round(avg(daily_count), 2) as average_daily_percent
```

```

from (select count(distinct b.post_id)/count(distinct a.post_id)*100 as daily_count
      from actions a
      left join removals b
      on a.post_id = b.post_id
      where extra = 'spam'
      group by action_date
     ) b

```

74.

```

select round(
  ifnull(
    (
      select count(distinct a.player_id)
      from Activity as a join Activity as b
      on a.player_id = b.player_id and datediff(b.event_date, a.event_date) = 1
      where a.event_date = (
        select min(event_date) from Activity where player_id = a.player_id
      )
    )
    / -- divided by
    ( select count(distinct player_id) from Activity ),
  0),
2)
as fraction;

```

75.

```

select round(
  ifnull(
    (
      select count(distinct a.player_id)
      from Activity as a join Activity as b
      on a.player_id = b.player_id and datediff(b.event_date, a.event_date) = 1
      where a.event_date = (
        select min(event_date) from Activity where player_id = a.player_id
      )
    )
    / -- divided by
    ( select count(distinct player_id) from Activity ),

```

```

0),
2)
as fraction;
76.
select company_id, employee_id, employee_name, round(salary - salary*tax, 0) as salary
from( select *,
case when max(salary) over(partition by company_id) < 1000 then 0    when max(salary) over(partition
by company_id) between 1000    and 10000 then 0.24    else 0.49 end as tax  from Salaries) x
77.
select e.left_operand, e.operator, e.right_operand,
case e.operator
when '>' then if(v1.value > v2.value, 'true', 'false')
when '<' 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 v1.name = e.left_operand
left join Variables v2 on v2.name = e.right_operand;
78.
with people_country as
(
select id, c.name country
from Person p left join Country c
on left(p.phone_number,3) = c.country_code
)

select country
from
(
select country, avg(duration) avgtime
from
(
select caller_id id, duration
from Calls
union all
select callee_id, duration
from Calls

```

```

    ) t left join people_country
    using(id)
    group by country
) temp
where avgtime >
(
    select avg(duration) avgtime
    from
    (
        select caller_id, duration
        from Calls
        union all
        select callee_id, duration
        from Calls
    ) t

```

79.

```
SELECT NAME FROM EMPLOYEE ORDER BY NAME;
```

80.

```
WITH yearly_spend
```

```
AS (
```

```
    SELECT
```

```
        EXTRACT(YEAR FROM transaction_date) AS year,
```

```
        product_id,
```

```
        spend AS curr_year_spend
```

```
    FROM user_transactions
```

```
),
```

```
yearly_variance
```

```
AS (
```

```
    SELECT
```

```
        *,
```

```
        LAG(curr_year_spend, 1) OVER (
```

```
            PARTITION BY product_id
```

```
            ORDER BY product_id) AS prev_year_spend
```

```
    FROM yearly_spend
```

```
)
```

```
SELECT
```

```
year,  
product_id,  
curr_year_spend,  
prev_year_spend,  
ROUND(100 * (fill_in_column_1 - fill_in_column_2) / fill_in_column_2, 2) AS yoy_rate  
FROM yearly_variance;
```

81.

```
WITH summary AS (
```

```
),
```

```
prime_items AS (
```

```
),
```

```
non_prime_items AS (
```

```
)
```

```
SELECT
```

```
item_type,
```

```
fill_in_column_1 AS item_count
```

```
FROM prime_items
```

```
UNION ALL
```

```
SELECT
```

```
item_type,
```

```
fill_in_column_2 AS item_count
```

```
FROM non_prime_items;
```

82.

```
SELECT
```

```
EXTRACT(MONTH FROM curr_month.event_date) AS month,
```

```
fill_in_column_1
```

```
FROM user_actions AS curr_month
```

```
WHERE EXISTS (
```

```
SELECT last_month.user_id
```

```
FROM user_actions AS last_month
```

```
WHERE last_month.user_id = curr_month.user_id
```

```
AND EXTRACT(MONTH FROM last_month.event_date) = EXTRACT(MONTH FROM  
curr_month.event_date - interval '1 month')
```


83.

```
WITH searches_expanded AS (  
  SELECT searches  
  FROM search_frequency  
  GROUP BY searches,  
    GENERATE_SERIES(1, num_users)  
)
```

```
SELECT *  
FROM searches_expanded;
```

84.

85.

86.

```
SELECT  
  merchant_id,  
  credit_card_id,  
  amount,  
  EXTRACT(  
    EPOCH  
  FROM  
    transaction_timestamp - LAG(transaction_timestamp) OVER(  
      PARTITION BY merchant_id,  
        credit_card_id,  
        amount  
      ORDER BY  
        transaction_timestamp  
    )  
  )/60 AS minute_difference  --dividing by 60 to get the returned value in form of minutes  
FROM  
  transactions;
```

87.

88.

Write your MySQL query statement below

```
select s.gender, s.day, (select sum(score_points) from Scores where gender = s.gender and day <=  
s.day) as total  
from Scores s  
group by gender, day
```

```
order by gender, day;
89.
with people_country as
(
    select id, c.name country
    from Person p left join Country c
    on left(p.phone_number,3) = c.country_code
)
```

```
select country
from
(
    select country, avg(duration) avgtime
    from
    (
        select caller_id id, duration
        from Calls
        union all
        select callee_id, duration
        from Calls
    ) t left join people_country
    using(id)
    group by country
) temp
where avgtime >
(
    select avg(duration) avgtime
    from
    (
        select caller_id, duration
        from Calls
        union all
        select callee_id, duration
        from Calls
    ) t
)
```

90.

91.

```
select department_salary.pay_month, department_id,
       case
         when department_avg > company_avg then 'higher'
         when department_avg < company_avg then 'lower'
         else 'same'
       end as comparison
from (
  select department_id, avg(amount) as department_avg, date_format(pay_date, '%Y-%m') as
pay_month
    from salary join employee on salary.employee_id = employee.employee_id
    group by department_id, pay_month
  ) as department_salary
join (
  select avg(amount) as company_avg, date_format(pay_date, '%Y-%m') as pay_month
    from salary group by date_format(pay_date, '%Y-%m')
  ) as company_salary
on department_salary.pay_month = company_salary.pay_month;
```

92.

```
SELECT
player_id, min(event_date) as first_login
FROM
Activity
group by player_id ;
```

93.

```
select group_id, player_id from (
  select p.group_id, ps.player_id, sum(ps.score) as score
    from Players p,
      (
        select first_player as player_id, first_score as score
        from Matches
        union all
        select second_player, second_score
        from Matches
      ) ps
  where p.player_id = ps.player_id
  group by ps.player_id
```

```
        order by group_id, score desc, player_id
        -- limit 1 -- by default, groupby will pick the first one i.e. max score player here
    ) top_scores
group by group_id;
```

94.

```
select student_id, student_name
from
(
    select distinct student_id
    from Exam
    where student_id not in
    (
        select distinct student_id
        from Exam e left join
        (
            -- highest and lowest scores
            select exam_id, max(score) maxs, min(score) mins
            from Exam
            group by exam_id
        ) t
        using(exam_id)
        where score = maxs or score = mins
        -- student with the highest or lowest score
    )
) t
left join Student
using(student_id)
order by student_id
```

95.

```
select student_id, student_name
from
(
    select distinct student_id
    from Exam
    where student_id not in
    (
        select distinct student_id
        from Exam e left join
```

```

(      -- highest and lowest scores
    select exam_id, max(score) maxs, min(score) mins
    from Exam
    group by exam_id
) t
using(exam_id)
where score = maxs or score = mins
-- student with the highest or lowest score
)
) t
left join Student
using(student_id)
order by student_id
96.
97.
WITH rate AS (
SELECT
    user_id,
    CASE WHEN texts.email_id IS NOT NULL THEN 1
    ELSE 0
    END AS activation_count
FROM emails
LEFT JOIN texts
    ON emails.email_id = texts.email_id
    AND signup_action = 'Confirmed'
)

SELECT
    SUM(activation_count)::DECIMAL
    / COUNT(user_id) AS activation_rate
FROM rate;
98.
AVG(tweet_num) OVER (
    PARTITION BY fill_in_column_1
    ORDER BY fill_in_column_2
    ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
)

```

99. WITH snaps_statistics

AS (

SELECT

age.age_bucket,

SUM(CASE WHEN activities.activity_type = 'send'

THEN activities.time_spent ELSE 0 END) AS send_timespent,

SUM(CASE WHEN activities.activity_type = 'open'

THEN activities.time_spent ELSE 0 END) AS open_timespent,

SUM(activities.time_spent) AS total_timespent

FROM activities

JOIN age_breakdown AS age

ON activities.user_id = age.user_id

WHERE activities.activity_type IN ('send', 'open')

GROUP BY age.age_bucket

)

SELECT

fill_in_column_1,

(send_timespent / total_timespent) AS send_perc,

(fill_in_column_2 / fill_in_column_3) AS open_perc

FROM snaps_statistics;

100.

with cte as (

select pf.name, pf.followers profile_followers, pf.profile_id, ec.company_id, cp.name company_name,

cp.followers as company_followers from personal_profiles pf

join employee_company ec

on pf.profile_id = ec.personal_profile_id

join company_pages cp

on ec.company_id = cp.company_id

)

select profile_id from cte

group by profile_id, profile_followers

having max(company_followers) < profile_followers

order by profile_id asc;

101.

select * from UserActivity where (username, startDate) in (

```

select u1.username, max(u1.startDate) from UserActivity u1
  where (u1.username, u1.startDate) not in (
    select u2.username, max(u2.startDate) from UserActivity u2
      group by u2.username
      having count(u2.username) > 1
    )
  group by u1.username
);

```

102.

```

select distinct username, activity, startDate, endDate
from
  (select u.*,
    rank() over (partition by username order by startDate desc) as rnk,
    count(activity) over (partition by username) as num
  from UserActivity u) t
where (num <> 1 and rnk = 2) or (num = 1 and rnk = 1)

```

103.

```

SELECT NAME FROM STUDENTS WHERE MARKS > 75 ORDER BY SUBSTR(NAME, -3), ID;

```

104.

```

SELECT name
FROM Employee
WHERE salary > 2000 AND months < 10
ORDER BY employee_id

```

105.

```

SELECT
CASE
  WHEN (A + B <= C) | (B + C <= A) | (A + C <= B) THEN 'Not A Triangle'
  WHEN (A = B) & (B = C) THEN 'Equilateral'
  WHEN ((A = B) & (A != C)) | ((B = C) & (B != A)) | ((A = C) & (A != B)) THEN 'Isosceles'
  WHEN (A != B) & (B != C) & (A != C) THEN 'Scalene'
END AS Triangle_Type
FROM
  TRIANGLES;

```

106.

```

select ceil(avg(salary) - avg(replace(salary, '0', ''))) from employees;

```

107.

```

select max(months * salary), count(months * salary)

```

```
from Employee where (months * salary)
= (select max(months * salary) from Employee);
108.
```

```
select concat(name, '(', substring(occupation, 1, 1), ')') as name
from occupations
order by name
select concat('There are a total of', ' ', count(occupation), ' ',
lower(occupation), 's.') as profession
from occupations
group by occupation
order by profession;
109.
```

```
SET @d = 0, @p = 0, @s = 0, @a = 0;
SELECT MIN (DOCTOR_NAMES), MIN (PROFESSOR_NAMES), MIN (SINGER_NAMES), MIN
(ACTOR_NAMES)
FROM
(
  SELECT
    CASE WHEN OCCUPATION = 'Doctor' THEN NAME END AS DOCTOR_NAMES,
    CASE WHEN OCCUPATION = 'Professor' THEN NAME END AS PROFESSOR_NAMES,
    CASE WHEN OCCUPATION = 'Singer' THEN NAME END AS SINGER_NAMES,
    CASE WHEN OCCUPATION = 'Actor' THEN NAME END AS ACTOR_NAMES,
    CASE
      WHEN OCCUPATION = 'Doctor' THEN (@d := @d + 1)
      WHEN OCCUPATION = 'Professor' THEN (@p := @p + 1)
      WHEN OCCUPATION = 'Singer' THEN (@s := @s + 1)
      WHEN OCCUPATION = 'Actor' THEN (@a := @a + 1)
    END AS ROW_NUM
  FROM OCCUPATIONS
  ORDER BY NAME
) AS TEMP
GROUP BY ROW_NUM;
```

```
110.
SELECT N,
CASE
  WHEN P IS NULL THEN 'Root'
  WHEN (SELECT COUNT(*) FROM BST WHERE B.N=P)>0 THEN 'Inner'
```



```

ELSE 'Leaf'
END AS PLACE
FROM BST B
ORDER BY N;
111.
SET sql_mode="";
SELECT DISTINCT
  C.company_code,
  C.founder,
  COUNT(DISTINCT L.lead_manager_code),
  COUNT(DISTINCT S.senior_manager_code),
  COUNT(DISTINCT M.manager_code),
  COUNT(DISTINCT E.employee_code)
FROM Company AS C
LEFT JOIN Lead_Manager AS L
  ON C.company_code = L.company_code
LEFT JOIN Senior_Manager AS S
  ON L.lead_manager_code = S.lead_manager_code
LEFT JOIN Manager AS M
  ON S.senior_manager_code = M.senior_manager_code
LEFT JOIN Employee AS E
  ON M.manager_code = E.manager_code
GROUP BY C.company_code
ORDER BY C.company_code
112.
SELECT CONCAT(2, '&', REPLACE(GROUP_CONCAT(T2.n), ',', '&'))
FROM
(
  SELECT T.n
  FROM (
    WITH recursive counter AS (
      SELECT 2 AS n
      UNION
      SELECT n + 1 FROM counter WHERE n < 1000
    )
    SELECT c1.n AS n, MOD(c1.n, c2.n) AS r
    FROM counter AS c1, counter AS c2

```

```

        WHERE c1.n > c2.n
    ) AS T
    GROUP BY T.n
    HAVING MIN(T.r) > 0
    ORDER BY T.n
) AS T2;
113.
DELIMITER $$
BEGIN
DECLARE CH VARCHAR(255);
DECLARE CT INT DEFAULT 20;
SET CH = ' *';
WHILE CT>0 DO
    SELECT REPEAT(CH,CT);
    SET CT=CT-1;
END WHILE
END $$
DELIMITER ;
114.
DECLARE @var int          -- Declare SELECT @var = 20          -- Initialization WHILE @var > 0
-- condition BEGIN          -- Begin PRINT replicate('* ', @var) -- Print SET @var = @var -
1          -- decrement END
115.
SELECT NAME
FROM STUDENTS
WHERE MARKS > 75
ORDER BY RIGHT(NAME,3) ASC, ID ASC;
116.
SELECT NAME FROM EMPLOYEE ORDER BY NAME;
117.
SELECT NAME
FROM EMPLOYEE
WHERE SALARY > 2000
AND MONTHS < 10
ORDER BY EMPLOYEE_ID ASC;
118.
SELECT

```

```
CASE
  WHEN A + B <= C or A + C <= B or B + C <= A THEN 'Not A Triangle'
  WHEN A = B and B = C THEN 'Equilateral'
  WHEN A = B or A = C or B = C THEN 'Isosceles'
  WHEN A <> B and B <> C THEN 'Scalene'
```

```
END tuple
```

```
FROM TRIANGLES;
```

```
119.
```

```
WITH yearly_spend
```

```
AS (
```

```
  SELECT
```

```
    EXTRACT(YEAR FROM transaction_date) AS year,
```

```
    product_id,
```

```
    spend AS curr_year_spend
```

```
  FROM user_transactions
```

```
),
```

```
yearly_variance
```

```
AS (
```

```
  SELECT
```

```
    *,
```

```
    LAG(curr_year_spend, 1) OVER (
```

```
      PARTITION BY product_id
```

```
      ORDER BY product_id) AS prev_year_spend
```

```
  FROM yearly_spend
```

```
)
```

```
SELECT
```

```
  year,
```

```
  product_id,
```

```
  curr_year_spend,
```

```
  prev_year_spend,
```

```
  ROUND(100 * (fill_in_column_1 - fill_in_column_2)/ fill_in_column_2, 2) AS yoy_rate
```

```
FROM yearly_variance;
```

```
120. WITH summary AS (
```

```
),
```

```
prime_items AS (
```

```
),  
non_prime_items AS (
```

```
)
```

```
SELECT  
    item_type,  
    fill_in_column_1 AS item_count  
FROM prime_items
```

```
UNION ALL
```

```
SELECT  
    item_type,  
    fill_in_column_2 AS item_count  
FROM non_prime_items;
```

```
121.
```

```
SELECT EXTRACT(MONTH FROM curr_month.event_date) AS month, COUNT(fill_in_column_1) AS  
monthly_active_users FROM user_actions AS curr_month WHERE EXISTS ( SELECT  
last_month.user_id FROM user_actions AS last_month WHERE last_month.user_id =  
curr_month.user_id AND EXTRACT(MONTH FROM last_month.event_date) = EXTRACT(MONTH  
FROM curr_month.event_date - interval '1 month') ) AND EXTRACT(MONTH FROM  
curr_month.event_date) = 7 AND EXTRACT(YEAR FROM curr_month.event_date) = 2022 GROUP BY  
fill_in_column_2;
```

```
122.
```

```
WITH searches_expanded AS  
    (SELECT searches  
    FROM search_frequency  
    GROUP BY searches,  
    GENERATE_SERIES(1, num_users)  
    )
```

```
SELECT  
    PERCENTILE_CONT(<percent in 0.0 format>)  
    WITHIN GROUP (  
        ORDER BY <column_name>) AS median  
FROM searches_expanded;
```

123.

```
WITH payment_status AS ( -- Insert query from Hint #3) SELECT user_id, CASE WHEN paid IS NULL  
THEN _____ WHEN status != 'CHURN' AND paid IS NOT NULL THEN 'EXISTING' WHEN status =  
'CHURN' AND paid IS NOT NULL THEN 'RESURRECT' WHEN status IS NULL THEN _____ END AS  
new_status FROM payment_status;
```

124.

```
WITH running_time AS (  
  SELECT  
    server_id,  
    session_status,  
    status_time AS start_time,  
    LEAD(status_time) OVER (  
      PARTITION BY server_id  
      ORDER BY status_time) AS stop_time  
  FROM server_utilization)
```

```
SELECT DATE_PART('days', JUSTIFY_HOURS(SUM(stop_time - start_time))) AS total_uptime_days  
FROM running_time  
WHERE condition 1 AND condition 2;
```

125.

```
WITH payments AS ( SELECT merchant_id, EXTRACT( EPOCH FROM transaction_timestamp -  
LAG(transaction_timestamp) OVER( PARTITION BY merchant_id, credit_card_id, amount ORDER BY  
transaction_timestamp ) )/60 AS minute_difference FROM transactions ) SELECT COUNT(merchant_id)  
AS payment_count FROM payments WHERE _____
```

126.

```
127. select s.gender, s.day, (select sum(score_points) from Scores where gender = s.gender and day <=  
s.day) as total  
  from Scores s  
  group by gender, day  
  order by gender, day;
```

128.

```
select c.name as country  
from Person p  
inner join Country c  
on left (p.phone_number,3) = c.country_code  
inner join (select caller_id as id, duration
```

```

        from Calls

        union all

        select callee_id as id, duration
        from Calls) phn
on p.id = phn.id
group by country
having avg(duration) > (select avg(duration) from Calls)
129.
130.
select
    pay_month,
    department_id,
    case when dept_avg > comp_avg then 'higher' when dept_avg < comp_avg then 'lower' else 'same'
end comparison
from (
    select date_format(b.pay_date, '%Y-%m') pay_month, a.department_id, avg(b.amount) dept_avg,
d.comp_avg
    from employee a
    inner join salary b
        on (a.employee_id = b.employee_id)
    inner join (select date_format(c.pay_date, '%Y-%m') pay_month, avg(c.amount) comp_avg
        from salary c
        group by date_format(c.pay_date, '%Y-%m')) d
        on (date_format(b.pay_date, '%Y-%m') = d.pay_month)
group by date_format(b.pay_date, '%Y-%m'), department_id, d.comp_avg) final
131.

```

```

SELECT a.event_date, ROUND((COUNT(DISTINCT CASE WHEN a.event_date = b.install_date THEN
a.player_id ELSE NULL END)/COUNT(DISTINCT CASE WHEN DATEDIFF(a.event_date, b.install_date)
= 1 THEN a.player_id ELSE NULL END)),2)FROM(SELECT a.player_id, a.event_date,
b.install_dateFROM activity aLEFT JOIN (SELECT player_id, MIN(event_date) install_dateFROM
activityGROUP BY player_id) b ON b.player_id = a.player_id) CWHERE a.event_date = b.install_date OR
DATEDIFF(a.event_date, b.install_date) = 1GROUP BY a.event_date

```

132.

```

select group_id, player_id
from (

```

```

select sc.group_id group_id, sc.player_id player_id,
       rank() over (partition by sc.group_id order by sum(sc.score) desc, sc.player_id asc) as rnk
from(
    select p.group_id group_id,
           p.player_id player_id ,
           sum(m.first_score) as score
    from players p
    inner join matches m
    on p.player_id = m.first_player
    group by p.group_id,p.player_id

    union all

    select p.group_id group_id,
           p.player_id player_id ,
           sum(second_score) as score
    from players p
    inner join matches m
    on p.player_id = m.second_player
    group by p.group_id,p.player_id
) sc
group by sc.group_id,sc.player_id
) A
where rnk = 1
133.
select distinct Student.*
from Student inner join Exam
on Student.student_id = Exam.student_id
where student.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

```

134.

```
WITH cte AS
(SELECT student_id,
       score,
       exam_id,
       (CASE WHEN score < MAX(score) OVER (PARTITION BY exam_id)
            AND score > MIN(score) OVER (PARTITION BY exam_id)
            THEN 'middle'
            ELSE 'highlow'
            END) AS category
FROM Exam
ORDER BY student_id),
```

```
cte1 AS (SELECT student_id
        FROM cte
        GROUP BY student_id
        HAVING SUM(CASE WHEN category = 'highlow'
            THEN 1 ELSE 0
            END) = 0
        )
```

```
SELECT cte1.student_id, s.student_name
FROM cte1 JOIN Student s
ON cte1.student_id = s.student_id
ORDER BY cte1.student_id
```

135.

```
select * from UserActivity where (username, startDate) in (
    select u1.username, max(u1.startDate) from UserActivity u1
    where (u1.username, u1.startDate) not in (
        select u2.username, max(u2.startDate) from UserActivity u2
        group by u2.username
        having count(u2.username) > 1
    )
    group by u1.username
);
```

136.

```
select distinct username, activity, startDate, endDate
```



```
from
  (select u.*,
    rank() over (partition by username order by startDate desc) as rnk,
    count(activity) over (partition by username) as num
  from UserActivity u) t
where (num <> 1 and rnk = 2) or (num = 1 and rnk = 1)
```

137.

```
select ceil(avg(salary) - avg(replace(salary, '0', ''))) from employees;
or
select cast(ceiling(avg(cast(salary as float)) - avg(cast(replace(salary, '0', '') as float)))) as int
from employees
```

138.

```
select max(months * salary), count(months * salary)
from Employee where (months * salary)
= (select max(months * salary) from Employee);
```

139.

```
select concat(name, '(', substring(occupation, 1, 1), ')') as namefrom occupationsorder by nameselect
concat('There are a total of', ' ', count(occupation), ' ', lower(occupation), 's.') as professionfrom
occupationsgroup by occupationorder by profession;
```

140.

```
select
  Doctor,
  Professor,
  Singer,
  Actor
from (
  select
    NameOrder,
    max(case Occupation when 'Doctor' then Name end) as Doctor,
    max(case Occupation when 'Professor' then Name end) as Professor,
    max(case Occupation when 'Singer' then Name end) as Singer,
    max(case Occupation when 'Actor' then Name end) as Actor
  from (
    select
      Occupation,
      Name,
```

```
row_number() over(partition by Occupation order by Name ASC) as NameOrder
from Occupations
) as NameLists
group by NameOrder
) as Names
```

141.

```
select c.company_code,
       c.founder,
       count(distinct lm.lead_manager_code),
       count(distinct sm.senior_manager_code),
       count(distinct m.manager_code),
       count(distinct e.employee_code)
from company c, lead_manager lm, senior_manager sm, manager m,
     employee e
where c.company_code = lm.company_code
and lm.lead_manager_code = sm.lead_manager_code
and sm.senior_manager_code = m.senior_manager_code
and m.manager_code = e.manager_code
group by c.company_code, c.founder
order by c.company_code
```

;

142.

```
select N,
if(P is null, 'Root', if((select count(*) from BST where P = B.N)> 0, 'Inner', 'Leaf'))
from BST as B
order by N;
```

143.

```
SELECT X, Y FROM (
SELECT X, Y FROM Functions WHERE X=Y GROUP BY X, Y HAVING COUNT(*)=2
UNION
SELECT f1.X, f1.Y FROM Functions f1, Functions f2
WHERE f1.X < f1.Y
AND f1.X=f2.Y
AND f2.X=f1.Y
)t
ORDER BY X, Y;
```

144.

```

SELECT t.Name
FROM (
    SELECT s1.ID, s1.Name, p1.Salary, f.Friend_ID, s2.name as friend_name, p2.Salary as friend_salary
    FROM Students s1
    JOIN Packages p1 ON s1.ID = p1.ID
    JOIN Friends f ON s1.ID = f.ID
    JOIN Students s2 ON f.Friend_ID = s2.ID
    JOIN Packages p2 ON f.Friend_ID = p2.ID
) t
WHERE t.friend_salary > t.Salary
ORDER BY friend_salary;

```

145.

```

SELECT h.hacker_id, h.name
FROM submissions s
INNER JOIN challenges c
ON s.challenge_id = c.challenge_id
INNER JOIN difficulty d
ON c.difficulty_level = d.difficulty_level
INNER JOIN hackers h
ON s.hacker_id = h.hacker_id
WHERE s.score = d.score AND c.difficulty_level = d.difficulty_level
GROUP BY h.hacker_id, h.name
HAVING COUNT(s.hacker_id) > 1
ORDER BY COUNT(s.hacker_id) DESC, s.hacker_id ASC

```

146.

```

SELECT Start_Date, min(End_Date)
FROM
    (SELECT Start_Date FROM Projects WHERE Start_Date NOT IN (SELECT End_Date FROM Projects))
a ,
    (SELECT End_Date FROM Projects WHERE End_Date NOT IN (SELECT Start_Date FROM Projects))
b
WHERE Start_Date < End_Date
GROUP BY Start_Date
ORDER BY DATEDIFF(min(End_Date), Start_Date) ASC, Start_Date ASC;

```

147 and 151



```

WITH cte AS (
  SELECT
    *,
    LEAD(date_diff, 1) OVER(
      PARTITION BY user_id
      ORDER BY
        transaction_date
    ) - LEAD(date_diff, 0) OVER(
      PARTITION BY user_id
      ORDER BY
        transaction_date
    ) AS 'diff'
  FROM
    (
      SELECT
        *,
        DATEDIFF(
          LEAD(transaction_date, 1) OVER(
            PARTITION BY user_id
            ORDER BY
              transaction_date
          ),
          LEAD(transaction_date, 0) OVER(
            PARTITION BY user_id
            ORDER BY
              transaction_date
          )
        ) AS 'date_diff'
      FROM
        mytable
    ) sub
),
cte2 AS (
  SELECT
    user_id,
    date_diff,
    diff,
    COUNT(date_diff),
    COUNT(diff)
  FROM
    cte
  GROUP BY
    user_id,
    date_diff,
    diff
  HAVING
    COUNT(date_diff) >= 2
    AND date_diff = 1

```

148.

WITH T1 AS

(SELECT

PAYER_ID,

RECIPIENT_ID

FROM PAYMENTS

INTERSECT

SELECT

RECIPIENT_ID,

PAYER_ID

FROM PAYMENTS)

SELECT

COUNT(PAYER_ID)/2 AS UNIQUE_RELATIONSHIPS

FROM

T1;

149.

150.

152.

153.

154.

```

WITH payout AS (
SELECT
    employee_id,
    salary,
    title,
    (AVG(salary) OVER (PARTITION BY title)) * 2 AS double_average,
    (AVG(salary) OVER (PARTITION BY title)) / 2 AS half_average
FROM employee_pay)

SELECT
    employee_id,
    salary,
    CASE WHEN salary > double_average THEN 'Overpaid'
         WHEN salary < half_average THEN 'Underpaid'
    END AS outlier_status
FROM payout
WHERE salary > double_average OR salary < half_average;

```

155.

```

SELECT
    COUNT(PAYER_ID)/2 AS UNIQUE_RELATIONSHIPS
FROM
    T1;
WITH T1 AS
    (SELECT
        PAYER_ID,
        RECIPIENT_ID
    FROM PAYMENTS
    INTERSECT
    SELECT
        RECIPIENT_ID,
        PAYER_ID
    FROM PAYMENTS)

```

156.

157

158.

```

WITH product_category_spend AS (
SELECT
    category,
    product,
    SUM(spend) AS total_spend
FROM product_spend
WHERE transaction_date >= '2022-01-01'
    AND transaction_date <= '2022-12-31'
GROUP BY category, product
),
top_spend AS (
SELECT *,
    RANK() OVER (
        PARTITION BY category
        ORDER BY total_spend DESC) AS ranking
FROM product_category_spend)

SELECT ____, ____, ____
FROM top_spend
WHERE ranking <= ____
ORDER BY ____, ____;

```

WITH product_category_spend AS (SELECT category, product, SUM(spend) AS total_spend FROM product_spend WHERE transaction_date >= '2022-01-01' AND transaction_date <= '2022-12-31' GROUP BY category, product), top_spend AS (SELECT *, RANK() OVER (PARTITION BY category ORDER BY total_spend DESC) AS ranking FROM product_category_spend) SELECT ____, ____, ____ FROM top_spend WHERE ranking <= ____ ORDER BY ____, ____;

159.

