Q1. Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.

select * from city where COUNTRYCODE = "USA" and POPULATION > 100000;

Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.

SELECT NAME FROM CITY WHERE COUNTRYCODE = "USA" AND POPULATION > 120000;

Q3. Query all columns (attributes) for every row in the CITY table.

SELECT * FROM CITY;

Q4. Query all columns for a city in CITY with the ID 1661.

SELECT * FROM CITY WHERE ID=1611;

Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

```
SELECT * FROM CITY WHERE COUNTRYCODE = 'JPN';
```

Q6. Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN

```
SELECT NAME FROM CITY WHERE COUNTRYCODE = 'JPN';
```

Q7. Query a list of CITY and STATE from the STATION table.

```
SELECT CITY, STATE FROM STATION;
```

Q8. Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.

```
SELECT DISTINCT CITY FROM STATION WHERE MOD(ID,2)=0 ORDER BY CITY ASC;
```

Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

```
SELECT COUNT(CITY) – COUNT(DISTINCT(CITY)) AS DISTINCT_CITY FROM STATION;
```

Q10. Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically

```
SELECT TOP 1 CITY, LEN(CITY) FROM STATION GROUP BY CITY ORDER BY LEN(CITY) ASC;

SELECT TOP 1 CITY, LEN(CITY) FROM STATION GROUP BY CITY ORDER BY LEN(CITY) DESC;
```

Q11. Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.

```
SELECT DISTINCT(CITY) FROM STATION WHERE LEFT(CITY,1) IN ('a','e','i','o','u');
```

Q12. Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

```
SELECT DISTINCT(CITY) FROM STATION WHERE RIGHT(CITY,1) IN ('a','e','i','o','u');
```

Q13. Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

```
SELECT DISTINCT(CITY) FROM STATION WHERE LEFT(CITY,1) NOT IN
('a','e','i','o','u');
```

Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.

```
SELECT DISTINCT(CITY) FROM STATION WHERE RIGHT(CITY,1) NOT IN
('a','e','i','o','u');
```

Q15. Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

```
SELECT DISTINCT(CITY) FROM STATION WHERE LEFT(CITY,1) NOT IN
('a','e','i','o','u') OR RIGHT(CITY,1) NOT IN ('a','e','i','o','u');
```

Q16. Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.

```
SELECT DISTINCT(CITY) FROM STATION WHERE LEFT(CITY,1) NOT IN
('a','e','i','o','u') AND RIGHT(CITY,1) NOT IN ('a','e','i','o','u');
```

Q17. Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.

```
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';
```

Q18. Write an SQL query to find all the authors that viewed at least one of their own articles. Return the result table sorted by id in ascending order.

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

Q19.

```
CREATE TABLE Delivery(
delivery_id int,
customer_id int,
order_date date,
customer_pref_delivery_date date
) ;


INSERT INTO Delivery VALUES(1, 1, "2019-08-01", "2019-08-02");

INSERT INTO Delivery VALUES(2, 5, "2019-08-02", "2019-08-02");
INSERT INTO Delivery VALUES(3, 1, "2019-08-11", "2019-08-11");
INSERT INTO Delivery VALUES(4, 3, "2019-08-24", "2019-08-26");
INSERT INTO Delivery VALUES(5, 4, "2019-08-21", "2019-08-22");
INSERT INTO Delivery VALUES(6, 2, "2019-08-11", "2019-08-13");


SELECT * FROM Delivery;

select ifnull(
round((select count(*) from Delivery where order_date =
customer_pref_delivery_date)/
        (count(delivery_id)) * 100, 2), 0) as immediate_percentage from Delivery;
```

Q20.

```
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 asc;
```

Q21.

```
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
ORDER BY e1.employee_id;
```

Q22.

```
CREATE TABLE Countries(country_id int,
country_name varchar(10));


INSERT INTO Countries Values(2, "USA");
INSERT INTO Countries Values(3, "Australia");
INSERT INTO Countries Values(7, "Peru");
INSERT INTO Countries Values(5, "China");
INSERT INTO Countries Values(8, "Morocco");
INSERT INTO Countries Values(9, "Spain");

CREATE TABLE Weather(
country_id int,
weather_state int,
day date
);

insert into Weather values(2, 15, "2019-11-01");
insert into Weather values(2, 12, "2019-10-28");
insert into Weather values(2, 12, "2019-10-27");
insert into Weather values(3, -2, "2019-11-10");
insert into Weather values(3, 0 , "2019-11-11");
insert into Weather values(3, 3 , "2019-11-12");
insert into Weather values(5, 16, "2019-11-07");
insert into Weather values(5, 18, "2019-11-09");
insert into Weather values(5, 21, "2019-11-23");
insert into Weather values(7, 25, "2019-11-28");
```

```
insert into Weather values(7, 22, "2019-12-01");
insert into Weather values(7, 20, "2019-12-02");
insert into Weather values(8, 25, "2019-11-05");
insert into Weather values(8, 27, "2019-11-15");
insert into Weather values(8, 31, "2019-11-25");
insert into Weather values(9, 7 , "2019-10-23");
insert into Weather values(9, 3 , "2019-12-23");

SELECT * FROM Weather;


SELECT c.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 w
INNER JOIN Countries c
ON c.country_id = w.country_id
WHERE LEFT(day,7) = "2019-11"
GROUP BY c.country_name
;
```

Q23.

```
SELECT product_id, (round(sum(units_sum)/sum(units),2)) as average_price FROM
(SELECT p.product_id as product_id, units, price * units as units_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 tmp
GROUP BY product_id
;
```

Q24.

```
SELECT A.player_id, MIN(A.event_date) AS first_login FROM Activity A
GROUP BY A.player_id;
```

Q25.

```
Select player_id, device_id from activity where (player_id,  event_id) IN
(SELECT A.player_id, MIN(A.event_date) AS first_login FROM Activity A
GROUP BY A.player_id);
```

Q26.

```sql
SELECT p.product_id, sum(o.unit) FROM Products p
LEFT JOIN Orders o
ON p.product_id = o.product_id
WHERE o.order_date BETWEEN '2020-02-01' and '2020-02-29'
GROUP BY p.product_id
HAVING sum(unit) >=100;
```

Q27.

```sql
select * from Users
    where mail regexp '^[a-zA-Z]+[a-zA-Z0-9_\\./\\-]{0,}@leetcode.com$'
    order by user_id;
```

Q28.

```sql
SELECT o.customer_id, name from orders o
JOIN Customers c
ON o.customer_id = c.customer_id
JOIN Product p
ON o.product_id = p.product_id
Group by 1,2
Having sum(case when date_format(order_date, '%Y-%m') = "2020-06"
THEN price * quantity end) >=100
sum(case when date_format(order_date, '%Y-%m') = "2020-07"
THEN price * quantity end) >=100;
```

Q29.

```sql
SELECT a.title FROM Content a
LEFT JOIN TVProgram b
ON a.content_id = b.content_id
WHERE date_format(program_date, "%Y-%m") = "2020-06"
and Kids_content = "Y" and content_type = "Movies";
```

Q30.

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

Q31.

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

Q32.

```
select unique_id, name
from Employees
left join EmployeeUNI
on if (Employees.id = EmployeeUNI.id, Employees.id, null)
```

Q33.

```
SELECT a.name AS name, IFNULL(sum(b.distance),0) AS travelled_distance FROM rides
b
RIGHT JOIN users a
ON a.id = b.user_id
GROUP BY name
ORDER BY 2 DESC, 1 ASC;
```

Q34.

```
SELECT o.customer_id, name from orders o
JOIN Customers c
ON o.customer_id = c.customer_id
JOIN Product p
ON o.product_id = p.product_id
Group by 1,2
Having sum(case when date_format(order_date, '%Y-%m') = "2020-02"
THEN price * quantity end) >=100;
```

Q35.

```
(
    select name results
    from Movie_Rating natural join Users
    group by Users.user_id
    order by count(*) desc, name asc
    limit 1
)
union
(
    select Movies.title results
    from Movie_Rating natural join Movies
    where month(created_at)='2'
    group by Movies.movie_id
    order by avg(rating
```

```
)
desc, title asc;
```

Q36.

```
SELECT a.name AS name, IFNULL(sum(b.distance),0) AS travelled_distance FROM rides
b
RIGHT JOIN users a
ON a.id = b.user_id
GROUP BY name
ORDER BY 2 DESC, 1 ASC;
```

Q37.

```
select unique_id, name
from Employees
left join EmployeeUNI
on if (Employees.id = EmployeeUNI.id, Employees.id, null)
```

Q38.

```
SELECT id, name
FROM Students
WHERE department_id not in (SELECT id from Departments);
```

Q39.

```
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, sum(duration) as total_duration
from unique_caller
group by person1, person2;
```

Q40.

```
SELECT product_id, (round(sum(units_sum)/sum(units),2)) as average_price FROM
(SELECT p.product_id as product_id, units, price * units as units_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 tmp
GROUP BY product_id
;
```

Q41.

```
select name as warehouse_name, sum(units * vol) as volume
from Warehouse w
join (select product_id, Width*Length*Height as vol
     from Products) p
on w.product_id = p.product_id
group by name;
```

Q42.

```
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;
```

Q43.

```
WITH CTE AS (
SELECT
player_id, min(event_date) as event_start_date
from
Activity
group by player_id )

SELECT
round((count(distinct c.player_id) / (select count(distinct player_id) from
activity)),2)as fraction
FROM
CTE c
JOIN Activity a
on c.player_id = a.player_id
and datediff(c.event_start_date, a.event_date) = -1;
```

Q44.

```
SELECT Name from Employee as t1
JOIN (select ManagerId from Employee group by ManagerId
having count(ManagerId)>=5) as t2
ON t1.Id = t2.ManagerId;
```

Q45.

```
SELECT a.dept_name, COUNT(student_id) as 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;
```

Q46.

```
select distinct customer_id
    from (
        select customer_id, count(distinct product_key) as product_count
            from Customer
            group by customer_id
    ) as customer_product_counts
    join (
        select count(distinct product_key) as product_count from Product
    ) as product_counts
    on customer_product_counts.product_count = product_counts.product_count;
```

Q47.

```
SELECT project_id, employee_id FROM (
    SELECT p.project_id, p.employee_id,
    dense_rank() over(partition by p.project_id order by e.experience_years desc)
as rnk
    FROM Project p JOIN Employee e
    ON p.employee_id = e.employee_id
) x
WHERE rnk = 1;
```

Q48.

```
Select b.book_id, b.name from books b
Left join (select book_id, sum(quantity) as nsold
From orders
Where dispatch_date between '2018-06-23' AND '2019-06-23'
Group by book_id) as o

Where
(o.nsold < 10 OR o.nsold IS NULL) and DATEDIFF('2019-06-23', b.available_from) > 30;
```

Q49.

```
SELECT
    student_id,
    course_id,
```

```
    grade
FROM (
SELECT
    student_id,
    course_id,
    grade,
    DENSE_RANK() OVER(PARTITION BY student_id ORDER BY grade DESC, course_id) as rnk
FROM Enrollments
    ) x
WHERE rnk=1
ORDER BY 1;
```

Q50.

```
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;
```

Q51.

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

Q52.

```
SELECT name FROM customer WHERE referee_Id <> 2;
```

Q53.

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

Q55.

```
SELECT
 co.name AS country
FROM
 person p
 JOIN
     country co
     ON SUBSTRING(phone_number,1,3) = country_code
 JOIN
     calls c
     ON p.id IN (c.caller_id, c.callee_id)
GROUP BY
 co.name
HAVING
 AVG(duration) > (SELECT AVG(duration) FROM calls);
```

Q57.

```
SELECT
    customer_number
FROM
    orders
GROUP BY customer_number
ORDER BY COUNT(*) DESC
LIMIT 1
;
```

Q58.

```
SELECT
     DISTINCT(a.seat_id)
FROM cinema a
INNER JOIN cinema b
ON abs(a.seat_id - b.seat_id) = 1
WHERE a.free = 1 and b.free = 1
ORDER BY a.seat_id;
```

Q59.

```
SELECT name
FROM salesperson
WHERE sales_id
```

```
NOT IN (
    SELECT s.sales_id FROM orders o
    INNER JOIN salesperson s ON o.sales_id = s.sales_id
    INNER JOIN company c ON o.com_id = c.com_id
    WHERE c.name = 'RED'
);
```

Q60.

```
SELECT
    x,
    y,
    z,
    IF(x + y > z AND y + z > x AND z + x > y, 'Yes', 'No') triangle
FROM
    triangle;
```

Q61.

```
SELECT MIN(abs(p2.x - p1.x)) shortest FROM point p1 JOIN point p2 ON p1.x !=
p2.x;
```

Q62.

```
SELECT actor_id, director_id
FROM ActorDirector
GROUP BY actor_id, director_id
HAVING COUNT(*) >= 3;
```

Q63.

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

Q64.

```
select project_id , round(avg(experience_years), 2) as average_years
from project as p
left join employee as e
on p.employee_id = e.employee_id
group by project_id;
```

Q65.

```
select
a.seller_id
              from
              (select seller_id, sum(price) as sum
              from Sales
              group by seller_id) a
              where a.sum = (select max(b.sum)from(select seller_id,
              sum(price) as sum
              from Sales
              group by seller_id)b );
```

Q66.

```
select distinct buyer_id from Sales s
join Product p
on p.product_id = s.product_id
where p.product_name = 'S8'
and buyer_id not in
(
select buyer_id from Sales s
    join Product p on p.product_id = s.product_id
    where p.product_name = 'iPhone';
)
```

Q67.

```
WITH result as (
SELECT
    visited_on,
    SUM(amount) as amount
FROM customer
GROUP BY visited_on
), result2 as (
SELECT
    visited_on,
    SUM(amount) OVER(ORDER BY visited_on ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) as
amount,
    ROUND(AVG(amount) OVER(ORDER BY visited_on ROWS BETWEEN 6 PRECEDING AND CURRENT
ROW),2) as average_amount,
    DENSE_RANK() OVER(ORDER BY visited_on) as rnk
FROM result
)
SELECT
    visited_on,
    amount,
    average_amount
FROM result2
WHERE rnk > 6;
```

Q68.

```
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;
```

Q69.

```
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;
```

Q70.

```
select a.student_id, a.student_name, b.subject_name, count(c.subject_name) as
attended_exams
from Students as a
join Subjects as b
left join Examinations as c
on a.student_id = c.student_id and b.subject_name = c.subject_name
group by a.student_id, b.subject_name;
```

Q101.

```
select distinct username, activity, startDate, endDate
FROM (SELECT u.*, DENSE_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);
```

Q103.

```
SELEC name FROM students
WHERE marks > 75
ORDER BY RIGHT(name, 3) ASC, id ASC;
```

Q104.

```
SELECT Name FROM Employee;
```

Q105.

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

Q106.

```
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 B = C OR A = C THEN 'Isosceles'
        ELSE 'Scalene'
        END
        FROM TRIANGLES;
```

Q107.

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

Q108.

```
select months*salary, count(*) from employee
group by months*salary
order by months*salary desc
limit 1;
```

Q109.

```
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
;
```

Q110.

```
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;
```

Q111.

```
select N,
       CASE
            when N not in (select distinct P from tree where P is not null) then 'LEAF'
            when P is null then 'ROOT'
            else 'INNER'
       END as node_type
from tree;
```

Q113.

```
select listagg(Prime_Number,'&') within group(order by Prime_Number)
from (select L Prime_Number from
(select Level L
from Dual
connect by Level <= 1000),
(select Level M
from Dual
connect by Level <= 1000)
where M <= L
group by L
having count(case when L/M = trunc(L/M) then 'Y' end) = 2
order by L);
```

Q114.

```
Declare @Val INT = 1
while @Val <= 20
BEGIN
Print Replicate("* ",@Val)
SET @Val = @Val + 1
END
```

Q115.

```
Declare @Val INT = 20
while @Val > 0
```

```
BEGIN
Print Replicate("* ",@Val)
SET @Val = @Val - 1
END
```

Q116.

```
SELECT f1.X, f1.Y FROM Functions AS f1
WHERE f1.X = f1.Y AND
(SELECT COUNT(*) FROM Functions WHERE X = f1.X AND Y = f1.Y) > 1
UNION
SELECT f1.X, f1.Y from Functions AS f1
WHERE EXISTS(SELECT X, Y FROM Functions WHERE f1.X = Y AND f1.Y = X AND f1.X < X)
ORDER BY X;
```

Q121.

```
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, year) AS prev_year_spend FROM yearly_spend)

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

Q121.

```
WITH summary AS (
  SELECT
    item_type,
    SUM(square_footage) AS total_sqft,
    COUNT(*) AS item_count
  FROM inventory
  GROUP BY item_type
),
```

```
prime_items AS (
  SELECT
    DISTINCT item_type,
    total_sqft,
    TRUNC(500000/total_sqft,0) AS prime_item_combo,
    (TRUNC(500000/total_sqft,0) * item_count) AS prime_item_count
  FROM summary
  WHERE item_type = 'prime_eligible'
),
non_prime_items AS (
  SELECT
    DISTINCT item_type,
    total_sqft,
    TRUNC(
      (500000 - (SELECT prime_item_combo * total_sqft FROM prime_items))
      / total_sqft, 0) * item_count AS non_prime_item_count
  FROM summary
  WHERE item_type = 'not_prime')

SELECT
  item_type,
  prime_item_count AS item_count
FROM prime_items
UNION ALL
SELECT
  item_type,
  non_prime_item_count AS item_count
FROM non_prime_items;
```

Q122.

```
SELECT
  EXTRACT(MONTH FROM curr_month.event_date) AS mth,
  COUNT(DISTINCT curr_month.user_id) 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 EXTRACT(MONTH FROM curr_month.event_date);
```

Q124.

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

SELECT
  ROUND(PERCENTILE_CONT(0.50) WITHIN GROUP (
    ORDER BY searches)::DECIMAL, 1) AS  median
FROM searches_expanded;
```

Q124.

```
SELECT
  advertiser.user_id,
  advertiser.status,
  payment.paid
FROM advertiser
LEFT JOIN daily_pay AS payment
  ON advertiser.user_id = payment.user_id

UNION

SELECT
  payment.user_id,
  advertiser.status,
  payment.paid
FROM daily_pay AS payment
LEFT JOIN advertiser
  ON advertiser.user_id = payment.user_id
)

SELECT
  user_id,
  CASE WHEN paid IS NULL THEN 'CHURN'
       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 'NEW'
```

```
  END AS new_status
FROM payment_status
ORDER BY user_id;
```