SQL Challenge Answer

Sample dataset - 1

==========

Q1. Query all columns for all American cities in the CITY table with populations larger than 100000.

=>

```
SELECT * FROM sample_dataset_1
WHERE
   population > 100000 AND country_code = 'USA'
   ;
```

==========

Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000

=>

```
SELECT name FROM sample_dataset_1
WHERE
   population > 120000 AND country_code = 'USA'
   ;
```

==========

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

=>

```
SELECT * FROM sample_dataset_1;
```

==========

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

No data exists with id = 1661

=>

```
SELECT * FROM sample_dataset_1
WHERE
   id = 1661
;
```

==========

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

=>

```
SELECT * FROM sample_dataset_1
WHERE
   country_code = 'JPN'
   ;
```

==========

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

SQL Challenge Answer

=>

```
SELECT name FROM sample_dataset_1
WHERE
   country_code = 'JPN'
   ;
```

==========

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

=>

```
SELECT city, state FROM sample_dataset_2;
```

==========

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 city FROM sample_dataset_2
WHERE
   id % 2 = 0
GROUP BY city
   ;
```

==========

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 Difference
FROM sample_dataset_2;
```

==========

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 city, LENGTH(city) as city_length from sample_dataset_2
WHERE
   LENGTH(city) = (select min(LENGTH(city)) FROM sample_dataset_2)
   ORDER BY city LIMIT 1)
UNION ALL
(SELECT city, LENGTH(city) as city_length from sample_dataset_2
WHERE
   LENGTH(city) = (select max(LENGTH(city)) FROM sample_dataset_2)
   ORDER BY city LIMIT 1)
;
```

==========

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 sample_dataset_2
WHERE
   city LIKE ("a%")
   OR city like ("e%")
   OR city like ("i%")
```

SQL Challenge Answer

```
   OR city like ("o%")
   OR city like ("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 sample_dataset_2
WHERE
   city LIKE ("%a")
   OR city like ("%e")
   OR city like ("%i")
   OR city like ("%o")
   OR city like ("%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 sample_dataset_2
WHERE
   city Not LIKE ("a%")
   and city not like ("e%")
   and city not like ("i%")
   and city not like ("o%")
```

```
    and city not like ("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 sample_dataset_2
WHERE
    city Not LIKE ("%a")
    and city not like ("%e")
    and city not like ("%i")
    and city not like ("%o")
    and city not like ("%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 sample_dataset_2
WHERE
    Left(city, 1) not in ("a", "i", "e", "o","u")
    OR
    right(city, 1) not in ("a", "i", "e", "o","u")
;
```

SQL Challenge Answer

==========

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 sample_dataset_2
WHERE
   Left(city, 1) not in ("a", "i", "e", "o","u")
   AND
   right(city, 1) not in ("a", "i", "e", "o","u")
;
```

==========

Product-sales Query:

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.

Return the result table in any order.

=>

```
SELECT * from product WHERE product_id in (SELECT product_id FROM sales
WHERE
   sale_date >= '2019-01-01' and 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 author_id FROM views
WHERE
   author_id = viewer_id
GROUP BY author_id
ORDER BY author_id;
```

==========

Q19. Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal

places.

=>

```
SELECT round(100 *
(SELECT COUNT(*) FROM delivery
   WHERE
      order_date = customer_pref_delivery_date
) / COUNT(*), 2)

FROM delivery;
```

==========

Q20. Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points.

Return the result table ordered by ctr in descending order and by ad_id in ascending order in case of a

tie

=>

```
SELECT ad_id,
```

SQL Challenge Answer

```
  round(
    (SUM(action='clicked')/
    (SUM(action='clicked') + SUM(action='viewed'))
    )*100, 2) as CTR
FROM ads GROUP BY ad_id
ORDER BY CTR desc, ad_id ASC;
```

==========

Q21. Write an SQL query to find the team size of each of the employees.

Return result table in any order

=>

```
SELECT employee_id,
COUNT(employee_id) over(partition by team_id) as total_count
from employee ORDER BY employee_id
;
```

==========

Q22. Write an SQL query to find the type of weather in each country for November 2019.

The type of weather is:

● Cold if the average weather_state is less than or equal 15,

● Hot if the average weather_state is greater than or equal to 25, and

● Warm otherwise.

Return result table in any order

=>

```
SELECT ct.country_id, ct.country_name, wt.new_state
FROM
countries as ct
```

```
RIGHT JOIN (SELECT country_id, CASE
when AVG(weather_state) <= 15 then "cold"
when AVG(weather_state) >= 25 then "hot"
else "warm"
end as new_state
FROM weather
WHERE day >= "2019-11-01" and day <= "2019-11-30"
GROUP BY country_id) as wt ON ct.country_id=wt.country_id
ORDER BY wt.new_state
;
```

==========

Q23. Write an SQL query to find the average selling price for each product. average_price should be rounded to 2 decimal places.

Return the result table in any order.

=>

```
with CTE as (SELECT  p.product_id, p.price, o.units FROM prices p  JOIN unit_sold o on o.product_id = p.product_id
WHERE
   o.purchase_date BETWEEN p.start_date AND p.end_date)
SELECT round(sum(price * units)/sum(units), 2) as average_selling_price FROM CTE
GROUP BY product_id
;
```

==========

Q24. Write an SQL query to report the first login date for each player.

Return the result table in any order.

=>

SQL Challenge Answer

```
select player_id, event_date as first_login_date FROM (SELECT player_id, event_date,

rank() over(partition by player_id ORDER BY event_date ASC) as first_login

FROM activity ORDER BY player_id) as tmp

WHERE

   first_login = 1

;
```

==========

Q25. Write an SQL query to report the device that is first logged in for each player.

Return the result table in any order.

=>

```
select player_id, device_id as first_login_device FROM (SELECT player_id, device_id, event_date,

rank() over(partition by player_id ORDER BY event_date ASC) as first_login

FROM activity ORDER BY player_id) as tmp

WHERE

   first_login = 1

;
```

==========

Q26. Write an SQL query to get the names of products that have at least 100 units ordered in February 2020

and their amount.

Return result table in any order.

=>

```
SELECT p.product_name, o.total_units FROM products as p INNER JOIN (SELECT product_id,
SUM(unit) as total_units FROM orders

WHERE
```

```
    order_date >= "2020-02-01" and order_date <= "2020-02-28"
GROUP BY product_id) as o on p.product_id = o.product_id
WHERE o.total_units >= 100
    ;
```

==========

Q27. Write an SQL query to find the users who have valid emails.

A valid e-mail has a prefix name and a domain where:

● The prefix name is a string that may contain letters (upper or lower case), digits, underscore '_', period '.', and/or dash '-'. The prefix name must start with a letter.

● The domain is '@leetcode.com'.

Return the result table in any order.

=>

```
SELECT user_id, name, mail FROM users
WHERE
    mail REGEXP "^[a-zA-Z]+[a-zA-Z0-9_.\-]*@leetcode.com$"
;
```

==========

Q28. Write an SQL query to report the customer_id and customer_name of customers who have spent at

least $100 in each month of June and July 2020.

Return the result table in any order.

=>

```
with CTE as (SELECT new_o.customer_id, new_o.month, sum(p.price * new_o.quantity) as
total_spent FROM products p
JOIN
    (
```

SQL Challenge Answer

```
    SELECT customer_id, product_id, quantity,
    CASE
      when order_date >= "2020-06-01" and order_date <= "2020-06-30" then "june"
      when order_date >= "2020-07-01" and order_date <= "2020-07-31" then "july"
      else "ignore"
    end as month
    FROM orders
    WHERE
      order_date >= "2020-06-01" and order_date <= "2020-07-30"
  ) as new_o on p.product_id = new_o.product_id
GROUP BY new_o.customer_id,  new_o.month
HAVING
  sum(p.price * new_o.quantity) >= 100)
SELECT customer_id, name FROM customers
WHERE
  customer_id = (SELECT customer_id FROM CTE
GROUP BY customer_id
HAVING count(customer_id) > 1)
;
```

==========

Q29. Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020.

Return the result table in any order.

=>

```
SELECT title FROM content
WHERE
  kids_content = "Y" and content_type = "Movies" and content_id in (SELECT content_id FROM
tv_program
```

```
WHERE
    program_date >= "2020-06-01 00:00" and program_date <= "2020-06-30 12:00")
ORDER BY title
;
```

==========

Q30. Write an SQL query to find the npv of each query of the Queries table.

Return the result table in any order

=>

```
SELECT q.id, q.year, IFNULL(npv_table.npv, 0) as npv_value FROM queries q
LEFT JOIN
(SELECT * FROM npv) as npv_table
ON
    npv_table.id = q.id and npv_table.year = q.year
;
```

==========

Q31. Write an SQL query to find the npv of each query of the Queries table.

Return the result table in any order.

=>

```
SELECT q.id, q.year, IFNULL(npv_table.npv, 0) as npv_value FROM queries q
LEFT JOIN
(SELECT * FROM npv) as npv_table
ON
    npv_table.id = q.id and npv_table.year = q.year
;
```

SQL Challenge Answer

==========

Q32. Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just

show null.

=>

```
SELECT IFNULL(e_uni.unique_id, NULL) as uniq_id, e.name FROM employee_uni as e_uni
RIGHT JOIN
   (
      SELECT * FROM employees
   ) as e
ON
   e.id = e_uni.id
   ;
```

==========

Q33. Write an SQL query to report the distance travelled by each user.

Return the result table ordered by travelled_distance in descending order, if two or more users

travelled the same distance, order them by their name in ascending order.

=>

```
SELECT u.name, IFNULL(r.total_distance, 0) FROM users u
LEFT JOIN
(
   SELECT user_id, SUM(distance) as total_distance FROM rides
   GROUP BY user_id
) as r on r.user_id = u.id
ORDER BY total_distance DESC, u.name ASC
;
```

==========

Q34. Write an SQL query to get the names of products that have at least 100 units ordered in February 2020

and their amount.

Return result table in any order

=>

```
SELECT p.product_name, o.total_units FROM products as p INNER JOIN (SELECT product_id,
SUM(unit) as total_units FROM orders
WHERE
   order_date >= "2020-02-01" and order_date <= "2020-02-28"
GROUP BY product_id) as o on p.product_id = o.product_id
WHERE o.total_units >= 100
   ;
```

==========

Q35. Write an SQL query to:

● Find the name of the user who has rated the greatest number of movies. In case of a tie,

return the lexicographically smaller user name.

● Find the movie name with the highest average rating in February 2020. In case of a tie, return

the lexicographically smaller movie name.

=>

```
(SELECT name FROM users
WHERE
   user_id IN
(SELECT user_id FROM movie_rating
GROUP BY user_id
HAVING
```

SQL Challenge Answer

```
   count(user_id) = (SELECT MAX(mvr.total_count) FROM (SELECT user_id, count(user_id) as
total_count FROM movie_rating

GROUP BY user_id) as mvr))

GROUP BY name

ORDER BY name ASC LIMIT 1)

UNION ALL

(SELECT title FROM movies

WHERE

   movie_id IN

(SELECT movie_id FROM movie_rating

GROUP BY movie_id

HAVING

   AVG(rating) = (SELECT MAX(mvr.total_rating) FROM (SELECT movie_id, AVG(rating) as total_rating
FROM movie_rating

WHERE

   created_at >= "2020-02-01" and created_at <= "2020-02-28"

GROUP BY movie_id) as mvr))

GROUP BY title

ORDER BY title ASC LIMIT 1

)

;
```

==========

Q36. Write an SQL query to report the distance travelled by each user.

Return the result table ordered by travelled_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.

=>

```
SELECT u.name, IFNULL(r.total_distance, 0) FROM users u

LEFT JOIN
```

```
(
   SELECT user_id, SUM(distance) as total_distance FROM rides
   GROUP BY user_id
) as r on r.user_id = u.id
ORDER BY total_distance DESC, u.name ASC
;
```

==========

Q37. Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just

show null.

=>

```
SELECT IFNULL(e_uni.unique_id, NULL) as uniq_id, e.name FROM employee_uni as e_uni
RIGHT JOIN
   (
      SELECT * FROM employees
   ) as e
ON
   e.id = e_uni.id
   ;
```

==========

Q38. Write an SQL query to find the id and the name of all students who are enrolled in departments that no

longer exist.

=>

```
        SELECT id, name FROM (SELECT name, id, department_id FROM students
HAVING
```

SQL Challenge Answer

```
    department_id not IN (SELECT id from departments)
) as tmp;
```

==========

Q39. Write an SQL query to report the number of calls and the total call duration between each pair of

distinct persons (person1, person2) where person1 < person2.

Return the result table in any order.

```
WITH CTE 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)
)

select person1, person2, count(*), sum(duration)
from CTE
where person1 < person2
GROUP BY person1, person2
;
```

==========

Q40. Write an SQL query to find the average selling price for each product. average_price should be rounded to 2 decimal places.

Return the result table in any order

=>

```
with CTE as (SELECT  p.product_id, p.price, o.units FROM prices p  JOIN unit_sold o on o.product_id = p.product_id
WHERE
   o.purchase_date BETWEEN p.start_date AND p.end_date)
SELECT round(sum(price * units)/sum(units), 2) as average_selling_price FROM CTE
GROUP BY product_id
;
```

==========

Q41. Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.

=>

```
SELECT w.name, sum((w.units * p.total_volume)) as total_cubic_feet FROM warehouse w
JOIN (SELECT product_id, (width * height * length) as total_volume FROM products) as p
ON
   p.product_id = w.product_id
GROUP BY w.name
;
```

==========

Q42. Write an SQL query to report the difference between the number of apples and oranges sold each day.

Return the result table ordered by sale_date.

=>

```
SELECT sa.sale_date, (SUM(sa.sold_num) - so.total_oranges) as diff FROM sales as sa
JOIN (SELECT sale_date, SUM(sold_num) as total_oranges FROM sales
WHERE
```

SQL Challenge Answer

```
   fruit = "oranges"
GROUP BY sale_date) as so
ON
   so.sale_date = sa.sale_date
WHERE
   sa.fruit = "apples"
GROUP BY sale_date;
```

==========

Q43. Write an SQL query to report the fraction of players that logged in again on the day after the day they

first logged in, rounded to 2 decimal places. In other words, you need to count the number of players

that logged in for at least two consecutive days starting from their first login date, then divide that

number by the total number of players.

=>

```
with CTE as (SELECT player_id, event_date,
datediff(event_date, lag(event_date) over (partition by player_id ORDER BY event_date ASC)) as lag_date
FROM activity)
SELECT round(count(distinct(player_id)) / (select count(DISTINCT(player_id)) FROM activity), 2) as fraction
from CTE
WHERE
   lag_date = 1
;
```

==========

Q44. Write an SQL query to report the managers with at least five direct reports.

Return the result table in any order

=>

```
SELECT name FROM employee
WHERE
   id = (SELECT managerid FROM employee
GROUP BY managerid
HAVING
   COUNT(managerid) >= 5
   );
```

==========

Q45. Write an SQL query to report the respective department name and number of students majoring in

each department for all departments in the Department table (even ones with no current students).

Return the result table ordered by student_number in descending order. In case of a tie, order them by

dept_name alphabetically.

=>

```
SELECT dp.dept_name, IFNULL(s.total_students, 0) as total_students FROM department as dp
LEFT JOIN
(SELECT dept_id, count(dept_id) as total_students
FROM student
GROUP BY dept_id) as s
ON
   dp.dept_id = s.dept_id
;
```

SQL Challenge Answer

==========

Q46. Write an SQL query to report the customer ids from the Customer table that bought all the products in

the Product table.

=>

```
with CTE as (SELECT customer_id, COUNT(customer_id) as total_count FROM customer
WHERE
   product_key IN (SELECT * from product)
GROUP BY customer_id)
SELECT customer_id from CTE
WHERE
   total_count = (SELECT max(total_count) FROM CTE)
GROUP BY customer_id
;
```

==========

Q49. Write a SQL query to find the highest grade with its corresponding course for each student. In case of

a tie, you should find the course with the smallest course_id.

Return the result table ordered by student_id in ascending order

=>

```
SELECT mgt.student_id, e.course_id, e.grade FROM enrollments e
JOIN (SELECT student_id, max(grade) as max_grade FROM enrollments
GROUP BY student_id) as mgt
ON
   mgt.student_id = e.student_id and mgt.max_grade = e.grade
ORDER BY student_id;
```

==========

Q51. Write an SQL query to report the name, population, and area of the big countries.

Return the result table in any order.

=>

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

==========

Q52. Write an SQL query to report the names of the customer that are not referred by the customer with id = 2

=>

```
SELECT name FROM customer
WHERE
   referee_id != 2 or referee_id IS NULL
;
```

==========

Q53. Write an SQL query to report all customers who never order anything.

Return the result table in any order.

=>

```
SELECT name FROM customers
WHERE
   id not IN (SELECT customerid FROM orders)
```

SQL Challenge Answer

;

==========

Q54. Write an SQL query to find the team size of each of the employees.

Return result table in any order.

=>

```
SELECT employee_id,
COUNT(employee_id) over(partition by team_id) as total_count
from employee ORDER BY employee_id
;
```

==========

Q55. A telecommunications company wants to invest in new countries. The company intends to invest in

the countries where the average call duration of the calls in this country is strictly greater than the

global average call duration.

Write an SQL query to find the countries where this company can invest.

Return the result table in any order.

=>

```
with country_phone as (SELECT p.*, c.name as country_name FROM person p JOIN
(SELECT name,
CASE
   WHEN LENGTH(country_code) < 3 then CONCAT("0", country_code)
   else country_code
end as new_code
FROM country) as c
ON
```

```
    left(p.phone_number, 3) = c.new_code
)

SELECT country_name, sum(total_dur)/sum(total_count) as final FROM (SELECT cp.country_name, (2
* cal.duration) as total_dur,  (2 * count(cp.country_name)) as total_count FROM calls as cal
JOIN
    country_phone as cp
ON
    cal.caller_id = cp.id
GROUP BY cp.country_name, duration) as tmp
GROUP BY country_name ORDER BY final DESC LIMIT 1
```

==========

Q56. Write an SQL query to report the device that is first logged in for each player.

Return the result table in any order.

=>

```
select player_id, device_id as first_login_device FROM (SELECT player_id, device_id, event_date,
rank() over(partition by player_id ORDER BY event_date ASC) as first_login
FROM activity ORDER BY player_id) as tmp
WHERE
    first_login = 1
;
```

==========

Q57. Write an SQL query to find the customer_number for the customer who has placed the largest number of orders.

The test cases are generated so that exactly one customer will have placed more orders than any other customer.

SQL Challenge Answer

=>

```
SELECT customer_number FROM (SELECT customer_number, COUNT(customer_number) as
total_count FROM orders

GROUP BY customer_number

ORDER BY total_count DESC LIMIT 1) as tmp

;
```

==========

Q58. Write an SQL query to report all the consecutive available seats in the cinema.

Return the result table ordered by seat_id in ascending order.

The test cases are generated so that more than two seats are consecutively available.

=>

```
with CTE as (SELECT *,

lag(free) over(order by free) as new_val

FROM cinema)


SELECT seat_id FROM CTE

WHERE

   new_val = 1

GROUP BY seat_id

;
```

==========

Q59. Write an SQL query to report the names of all the salespersons who did not have any orders related to

the company with the name "RED".

=>

```
SELECT sales_id FROM orders
WHERE
   com_id = 1;


SELECT name FROM salesperson
WHERE
   sales_id NOT IN
   (
   SELECT sales_id FROM orders
   WHERE
      com_id = (
            SELECT com_id FROM company
            WHERE
               name = "red"
            )
   )
;
```

==========

Q60. Write an SQL query to report for every three line segments whether they can form a triangle.

Return the result table in any order.

=>

```
SELECT *,
CASE
   when (x + y) <= z or (y + z) <= x or (z + x) <= y then "NO"
   else "YES"
end as triangle_bool
FROM triangle
;
```

SQL Challenge Answer

==========

Q61. Write an SQL query to report the shortest distance between any two points from the Point table.

The query result format is in the following example.

=>

```
SELECT MIN(new_val) from (SELECT

IFNULL(ABS(lag(x) over(order by x ASC)), 0) as new_val

FROM point) as tmp

WHERE

   new_val != 0

;
```

==========

Q62. Write a SQL query for a report that provides the pairs (actor_id, director_id) where the actor has

cooperated with the director at least three times.

=>

```
SELECT actor_id, director_id FROM actor_director

GROUP BY actor_id, director_id

HAVING

   count(CONCAT(actor_id,director_id)) >= 3 ;
```

==========

Q63. Write an SQL query that reports the product_name, year, and price for each sale_id in the Sales table.

Return the resulting table in any order

=>

```
SELECT p.product_name, s.year, s.price  FROM sales as s
JOIN (select * FROM product) as p ON p.product_id = s.product_id
;
```

==========

Q64. Write an SQL query that reports the average experience years of all the employees for each project,

rounded to 2 digits.

=>

```
SELECT p.project_id, round(avg(e.experience_years), 2) as average_exp
FROM project as p
JOIN (SELECT * from employee) as e ON
e.employee_id = p.employee_id
GROUP BY project_id;
```

==========

Q65. Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all.

Return the result table in any order.

=>

```
with CTE as (SELECT seller_id, sum(price) as total_price FROM sales
GROUP BY seller_id)
SELECT seller_id FROM CTE
WHERE
   total_price >= (SELECT max(total_price) FROM CTE)
;
```

SQL Challenge Answer

==========

Q66. Write an SQL query that reports the buyers who have bought S8 but not iPhone. Note that S8 and

iPhone are products present in the Product table

=>

```
SELECT buyer_id FROM sales
WHERE
   buyer_id NOT IN
   (SELECT buyer_id FROM sales
   WHERE
      product_id IN (SELECT product_id FROM product WHERE product_name != "S8")
   GROUP BY buyer_id
   )
;
```

==========

Q67. Write an SQL query to compute the moving average of how much the customer paid in a seven days

window (i.e., current day + 6 days before). average_amount should be rounded to two decimal places.

Return result table ordered by visited_on in ascending order.

=>

```
with CTE as
(SELECT visited_on,
SUM(total_amount) over (rows BETWEEN 6 preceding and current row) as sum_amount,
AVG(total_amount) over (rows BETWEEN 6 preceding and current row) as average_amount
FROM
```

```
    (
        SELECT visited_on, sum(amount) as total_amount FROM  customer
        GROUP BY visited_on
    ) as tmp
)
SELECT * FROM CTE ORDER BY visited_on ASC
;
```

==========

Q68. Write an SQL query to find the total score for each gender on each day.

Return the result table ordered by gender and day in ascending order.

=>

```
SELECT gender, day,
sum(score_points) over (partition by gender ORDER BY day rows between unbounded preceding and current row) as total_score
FROM scores;
```

==========

Q72. Write an SQL query to find for each month and country, the number of transactions and their total

amount, the number of approved transactions and their total amount.

Return the result table in any order

=>

```
SELECT trans_month, country, COUNT(trans_month), SUM(state="approved") as total_approved,
SUM(state="decline") as total_decline, SUM(amount)
FROM
(
    SELECT id, country, state, amount,
```

SQL Challenge Answer

```
    left(trans_date,7) as trans_month FROM transactions
) tmp
GROUP BY trans_month, country;
```

==========

Q73. Write an SQL query to find the average daily percentage of posts that got removed after being reported as spam, rounded to 2 decimal places

=>

```
with CTE as (SELECT post_id, action_date, SUM(extra="spam") as spam_count,
CASE
    when post_id IN (SELECT post_id FROM removals) then 1
    else 0
end as removed
FROM actions
GROUP BY action_date, post_id
HAVING
    sum(extra="spam") != 0)
SELECT round(sum(total_percent)/count(*), 0) as average_daily_percent
FROM
    (
        SELECT sum(removed)/sum(spam_count) * 100 as total_percent FROM CTE
        GROUP BY action_date
    ) tmp
;
```

==========

Q74. Write an SQL query to report the fraction of players that logged in again on the day after the day they

first logged in, rounded to 2 decimal places. In other words, you need to count the number of players

that logged in for at least two consecutive days starting from their first login date, then divide that

number by the total number of players.

=>

```
with CTE as (SELECT player_id, event_date,

datediff(event_date, lag(event_date) over (partition by player_id ORDER BY event_date ASC)) as lag_date

FROM activity)

SELECT round(count(distinct(player_id)) / (select count(DISTINCT(player_id)) FROM activity), 2) as fraction

from CTE

WHERE

   lag_date = 1

;
```

==========

Q75. Write an SQL query to report the fraction of players that logged in again on the day after the day they

first logged in, rounded to 2 decimal places. In other words, you need to count the number of players

that logged in for at least two consecutive days starting from their first login date, then divide that

number by the total number of players.

=>

```
with CTE as (SELECT player_id, event_date,

datediff(event_date, lag(event_date) over (partition by player_id ORDER BY event_date ASC)) as lag_date

FROM activity)

SELECT round(count(distinct(player_id)) / (select count(DISTINCT(player_id)) FROM activity), 2) as fraction

from CTE
```

SQL Challenge Answer

```
WHERE
   lag_date = 1
;
```

==========

Q76. Write an SQL query to find the salaries of the employees after applying taxes. Round the salary to the

nearest integer.

The tax rate is calculated for each company based on the following criteria:

● 0% If the max salary of any employee in the company is less than $1000.

● 24% If the max salary of any employee in the company is in the range [1000, 10000] inclusive.

● 49% If the max salary of any employee in the company is greater than $10000.

=>

```
with tax_table as (SELECT company_id,
case
   when max(salary) < 1000 then 0
   when max(salary) BETWEEN 1000 and 10000 then 24/100
   else 49/100
end as tax_percent
FROM salaries
GROUP BY company_id)

SELECT s.company_id, s.employee_id, s.employee_name,
round((s.salary - (s.salary * tax_table.tax_percent )), 0) as calculated_salary
FROM salaries as s
JOIN tax_table ON tax_table.company_id = s.company_id
;
```

==========

Q77. Write an SQL query to report the difference between the number of apples and oranges sold each day.

Return the result table ordered by sale_date.

=>

```
SELECT sa.sale_date, (SUM(sa.sold_num) - so.total_oranges) as diff FROM sales as sa
JOIN (SELECT sale_date, SUM(sold_num) as total_oranges FROM sales
WHERE
   fruit = "oranges"
GROUP BY sale_date) as so
ON
   so.sale_date = sa.sale_date
WHERE
   sa.fruit = "apples"
GROUP BY sale_date;
```

==========

Q78. Write an SQL query to report the difference between the number of apples and oranges sold each day.

Return the result table ordered by sale_date.

=>

```
SELECT sa.sale_date, (SUM(sa.sold_num) - so.total_oranges) as diff FROM sales as sa
JOIN (SELECT sale_date, SUM(sold_num) as total_oranges FROM sales
WHERE
   fruit = "oranges"
GROUP BY sale_date) as so
ON
   so.sale_date = sa.sale_date
```

SQL Challenge Answer

```
WHERE
   sa.fruit = "apples"
GROUP BY sale_date;
```

==========

Q79. Write an SQL query to evaluate the boolean expressions in Expressions table.

Return the result table in any order.

=>

```
with CTE as (SELECT * FROM expression)
SELECT *,
CASE
   when operator = "<" and (left_operand < right_operand) = 1 then "true"
   when operator = ">" and (left_operand > right_operand) = 1 then "true"
   when operator = "=" and (left_operand = right_operand) = 1 then "true"
   else "false"
end as new_val
FROM CTE;
```

==========

Q80. A telecommunications company wants to invest in new countries. The company intends to invest in

the countries where the average call duration of the calls in this country is strictly greater than the

global average call duration.

Write an SQL query to find the countries where this company can invest.

Return the result table in any order.

=>

```
with country_phone as (SELECT p.*, c.name as country_name FROM person p JOIN
```

```
(SELECT name,
CASE
   WHEN LENGTH(country_code) < 3 then CONCAT("0", country_code)
   else country_code
end as new_code
FROM country) as c
ON
   left(p.phone_number, 3) = c.new_code
)

SELECT country_name, sum(total_dur)/sum(total_count) as final FROM (SELECT cp.country_name, (2 * cal.duration) as total_dur,  (2 * count(cp.country_name)) as total_count FROM calls as cal
JOIN
   country_phone as cp
ON
   cal.caller_id = cp.id
GROUP BY cp.country_name, duration) as tmp
GROUP BY country_name ORDER BY final DESC LIMIT 1
```

==========

Q81 Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by

the last three characters of each name. If two or more students both have names ending in the same

last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID

=>

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

SQL Challenge Answer

==========

Q82. Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in

alphabetical order.

=>

```
SELECT name FROM employee
ORDER BY name ASC
;
```

==========

Q83. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in

Employee having a salary greater than $2000 per month who have been employees for less than 10

months. Sort your result by ascending employee_id.

=>

```
SELECT name FROM employee
WHERE
   salary > 2000 and months < 10
ORDER BY employee_id ASC
;
```

==========

Q84. Write a query identifying the type of each record in the TRIANGLES table using its three side lengths.

Output one of the following statements for each record in the table:

● Equilateral: It's a triangle with sides of equal length.

● Isosceles: It's a triangle with sides of equal length.

● Scalene: It's a triangle with sides of differing lengths.

● Not A Triangle: The given values of A, B, and C don't form a triangle

=>

```
SELECT *,
CASE
   when a = b and b = c then "equilateral"
   when a = b and b != c and a+b > c then "isoceles"
   when a + b < c or b + c < a or c + a < b then "not a triangle"
   when a != b and b != c and a != c then "scalene"
   else "normal triangle"
end as triangle_value
from triangle;
```

==========

Q85. Assume you are given the table below containing information on user transactions for particular

products. Write a query to obtain the year-on-year growth rate for the total spend of each product for

each year.

Output the year (in ascending order) partitioned by product id, current year's spend, previous year's

spend and year-on-year growth rate (percentage rounded to 2 decimal places).

=>

```
SELECT extract(year FROM transaction_date) as year_, product_id,
spend as curr_year_spend,
lag(spend) over() as prev_year_spend,
round((spend/lag(spend) over() * 100) - 100, 2) as yoy_rate
FROM transactions;
```

SQL Challenge Answer

==========

Q87. Assume you have the table below containing information on Facebook user actions. Write a query to obtain the active user retention in July 2022. Output the month (in numerical format 1, 2, 3) and the number of monthly active users (MAUs).

Hint: An active user is a user who has user action ("sign-in", "like", or "comment") in the current month and last month.

=>

```
SELECT extract(month FROM event_date) as month, COUNT(user_id) as MAU FROM user_actions
WHERE
   event_date BETWEEN "2022-06-01 00:00:00" and "2022-06-30 12:00:00" AND event_type = "sign-in"
   and user_id in
   (
      SELECT user_id FROM user_actions
      WHERE event_date BETWEEN "2022-06-01 00:00:00" and "2022-06-30 12:00:00" AND event_type != "sign-in"
   )
GROUP BY month
;
```

==========

Q90. Amazon Web Services (AWS) is powered by fleets of servers. Senior management has requested data-driven solutions to optimise server usage.

Write a query that calculates the total time that the fleet of servers was running. The output should be in units of full days.

=>

```
with CTE as (SELECT server_id, status_time,
lead(status_time) over(partition by server_id ORDER BY status_time ASC) as new_time
FROM server)
SELECT sum(DATEDIFF( new_time, status_time )) as total_uptime_days FROM CTE;
```

==========

Q91. Sometimes, payment transactions are repeated by accident; it could be due to user error, API failure or

a retry error that causes a credit card to be charged twice.

Using the transactions table, identify any payments made at the same merchant with the same credit

card for the same amount within 10 minutes of each other. Count such repeated payments.

=>

```
with CTE as (SELECT *,
timestampdiff(minute,transaction_timestamp,lag(transaction_timestamp) over())
as minutes_diff
FROM transactions)

SELECT count(minutes_diff) as payment_count FROM CTE
GROUP BY minutes_diff
HAVING
   abs(CTE.minutes_diff) <= 10
;
```

==========

Q93. Write an SQL query to find the total score for each gender on each day.

Return the result table ordered by gender and day in ascending order.

=>

```
SELECT gender, day,
```

SQL Challenge Answer

```
sum(score_points) over (partition by gender ORDER BY day rows between unbounded preceding and
current row) as total_score

FROM scores;
```

==========

Q94. A telecommunications company wants to invest in new countries. The company intends to invest in

the countries where the average call duration of the calls in this country is strictly greater than the

global average call duration.

Write an SQL query to find the countries where this company can invest.

Return the result table in any order.

=>

```
with country_phone as (SELECT p.*, c.name as country_name FROM person p JOIN

(SELECT name,

CASE

   WHEN LENGTH(country_code) < 3 then CONCAT("0", country_code)

   else country_code

end as new_code

FROM country) as c

ON

   left(p.phone_number, 3) = c.new_code

)


SELECT country_name, sum(total_dur)/sum(total_count) as final FROM (SELECT cp.country_name, (2
* cal.duration) as total_dur,  (2 * count(cp.country_name)) as total_count FROM calls as cal

JOIN

   country_phone as cp

ON

   cal.caller_id = cp.id
```

```
GROUP BY cp.country_name, duration) as tmp
GROUP BY country_name ORDER BY final DESC LIMIT 1
```

==========

Q96. Write an SQL query to report the comparison result (higher/lower/same) of the average salary of

employees in a department to the company's average salary.

=>

```
with CTE as
(
   SELECT distinct(full_table.pay_date), round(AVG(full_table.amount), 0) as avg_pay,
   full_table.department_id, avg_table.company_avg_pay
   FROM
   (
      SELECT s.*, e.department_id FROM salary s
      JOIN
         (SELECT * FROM employee) as e
      ON
         e.employee_id = s.employee_id
   ) full_table
   JOIN
      (SELECT pay_date, round(AVG(amount),0) as company_avg_pay FROM salary
      GROUP BY pay_date) as avg_table
   ON
      avg_table.pay_date = full_table.pay_date
   GROUP BY full_table.pay_date, full_table.department_id,
   avg_table.company_avg_pay
)

SELECT pay_date, department_id,
```

SQL Challenge Answer

```
CASE
   When avg_pay > company_avg_pay then "Higher"
   when avg_pay < company_avg_pay then "lower"
   when avg_pay = company_avg_pay then "same"
end as new_table_val
FROM CTE
GROUP BY pay_date, department_id, avg_pay
ORDER BY department_id, pay_date
;
```

==========

Q97.Write an SQL query to report for each install date, the number of players that installed the game on

that day, and the day one retention.

=>

```
SELECT first_log.*, IFNULL(round((game_table.game_play / first_log.installs),1),0) as retention
FROM
(
   select first_login, COUNT(first_login) as installs from
   (
      SELECT player_id, MIN(event_date) as first_login FROM activity
      GROUP BY player_id
   )
first_login_count
GROUP BY first_login) first_log
LEFT JOIN
(
   SELECT player_id,
   lag(event_date) over(partition by player_id) as game_date,
```

```
    event_date - lag(event_date) over(partition by player_id) as game_play
    FROM activity
) as game_table
ON
game_table.game_date = first_log.first_login and game_play = 1

;
```

==========

Q101. Write an SQL query to show the second most recent activity of each user.
If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.
=>

```
select distinct username, activity, startDate, endDate
from
    (select user.*,
        rank() over (partition by username order by startDate desc) as rnk,
        count(activity) over (partition by username) as num
    from user_activity user) new_table
    WHERE
       (num != 1 and rnk = 2) or (num = 1 and rnk = 1)
;
```

==========

Q102. Write an SQL query to show the second most recent activity of each user.
If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.
=>

SQL Challenge Answer

```
select distinct username, activity, startDate, endDate
from
   (select user.*,
        rank() over (partition by username order by startDate desc) as rnk,
        count(activity) over (partition by username) as num
   from user_activity user) new_table
   WHERE
      (num != 1 and rnk = 2) or (num = 1 and rnk = 1)
;
```

==========

Q103. Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by

the last three characters of each name. If two or more students both have names ending in the same

last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID

=>

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

==========

Q104. Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in

alphabetical order.

=>

```
SELECT name FROM employee
ORDER BY name ASC
;
```

==========

Q105. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee_id.

=>

```
SELECT name FROM employee
WHERE
   salary > 2000 and months < 10
ORDER BY employee_id ASC
;
```

==========

Q106. Write a query identifying the type of each record in the TRIANGLES table using its three side lengths.

Output one of the following statements for each record in the table:

● Equilateral: It's a triangle with sides of equal length.

● Isosceles: It's a triangle with sides of equal length.

● Scalene: It's a triangle with sides of differing lengths.

● Not A Triangle: The given values of A, B, and C don't form a triangle

=>

```
SELECT *,
CASE
   when a = b and b = c then "equilateral"
   when a = b and b != c and a+b > c then "isoceles"
```

SQL Challenge Answer

```
    when a + b < c or b + c < a or c + a < b then "not a triangle"
    when a != b and b != c and a != c then "scalene"
    else "normal triangle"
end as triangle_value
from triangle;
```

==========

Q107. Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realise her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.
Write a query calculating the amount of error (i.e.: actual - miscalculated average monthly salaries), and round it up to the next integer.

=>

```
SELECT
    round(
        AVG(salary) -
            (
                SELECT AVG(salary) FROM employees_incorrect
            ),
        0)
    as diff_salaries
FROM employees_correct;
```

==========

Q108. We define an employee's total earnings to be their monthly salary * months worked, and the maximum total earnings to be the maximum total earnings for any employee in the Employee table.

Write a query to find the maximum total earnings for all employees as well as the total number of employees who have maximum total earnings. Then print these values as 2 space-separated integers

=>

```
with CTE as
(
   SELECT *, (salary * months) as total_earnings FROM employees
)
SELECT
   concat(total_earnings, " ", count(total_earnings)) as output_table
FROM CTE
WHERE
   total_earnings = (SELECT max(total_earnings) FROM CTE)
GROUP BY total_earnings
;
```

==========

Q109. Generate the following two result sets:

1. Query an alphabetically ordered list of all names in OCCUPATIONS, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).

Query the number of occurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format:

=>

```
(SELECT CONCAT(name,"(",left(occupation, 1),")") FROM job
ORDER BY name)
UNION ALL
(SELECT
CONCAT("There are a total of ",COUNT(occupation), " ", occupation, "s")
from job
```

SQL Challenge Answer

```
GROUP BY occupation)
;
```

==========

Q 111. You are given a table, BST, containing two columns: N and P, where N represents the value of a node

in Binary Tree, and P is the parent of N

=>

```
SELECT n,
CASE
   when n not in (SELECT distinct(p) FROM nodes WHERE p is not NULL) then "Leaf"
   when p is NULL then "Root"
   else "Inner"
end as type_of_node
FROM nodes ORDER BY n;
```

==========

Q117. Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by

the last three characters of each name. If two or more students both have names ending in the same

last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID

=>

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

==========

Q118. Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in

alphabetical order.

=>

```
SELECT name FROM employee
ORDER BY name ASC
;
```

==========

Q119. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in

Employee having a salary greater than $2000 per month who have been employees for less than 10

months. Sort your result by ascending employee_id.

=>

```
SELECT name FROM employee
WHERE
   salary > 2000 and months < 10
ORDER BY employee_id ASC
;
```

==========

Q120. Write a query identifying the type of each record in the TRIANGLES table using its three side lengths.

Output one of the following statements for each record in the table:

● Equilateral: It's a triangle with sides of equal length.

● Isosceles: It's a triangle with sides of equal length.

SQL Challenge Answer

● Scalene: It's a triangle with sides of differing lengths.

● Not A Triangle: The given values of A, B, and C don't form a triangle

=>

```
SELECT *,
CASE
   when a = b and b = c then "equilateral"
   when a = b and b != c and a+b > c then "isoceles"
   when a + b < c or b + c < a or c + a < b then "not a triangle"
   when a != b and b != c and a != c then "scalene"
   else "normal triangle"
end as triangle_value
from triangle;
```

==========

Q121. Assume you are given the table below containing information on user transactions for particular

products. Write a query to obtain the year-on-year growth rate for the total spend of each product for

each year.

Output the year (in ascending order) partitioned by product id, current year's spend, previous year's

spend and year-on-year growth rate (percentage rounded to 2 decimal places).

=>

```
SELECT extract(year FROM transaction_date) as year_, product_id,
spend as curr_year_spend,
lag(spend) over() as prev_year_spend,
round((spend/lag(spend) over() * 100) - 100, 2) as yoy_rate
FROM transactions;
```

==========

Q 123. Assume you have the table below containing information on Facebook user actions. Write a query to obtain the active user retention in July 2022. Output the month (in numerical format 1, 2, 3) and the number of monthly active users (MAUs).

Hint: An active user is a user who has user action ("sign-in", "like", or "comment") in the current month and last month.

=>

```
SELECT extract(month FROM event_date) as month, COUNT(user_id) as MAU FROM user_actions
WHERE
  event_date BETWEEN "2022-06-01 00:00:00" and "2022-06-30 12:00:00" AND event_type = "sign-in"
  and user_id in
  (
    SELECT user_id FROM user_actions
    WHERE event_date BETWEEN "2022-06-01 00:00:00" and "2022-06-30 12:00:00" AND event_type != "sign-in"
  )
GROUP BY month
;
```

==========

Q 126. Amazon Web Services (AWS) is powered by fleets of servers. Senior management has requested data-driven solutions to optimise server usage.

Write a query that calculates the total time that the fleet of servers was running. The output should be in units of full days.

=>

```
with CTE as (SELECT server_id, status_time,
```

SQL Challenge Answer

```
lead(status_time) over(partition by server_id ORDER BY status_time ASC) as new_time

FROM server)

SELECT sum(DATEDIFF( new_time, status_time )) as total_uptime_days FROM CTE;
```

==========

Q127. Sometimes, payment transactions are repeated by accident; it could be due to user error, API failure or

a retry error that causes a credit card to be charged twice.

Using the transactions table, identify any payments made at the same merchant with the same credit

card for the same amount within 10 minutes of each other. Count such repeated payments.

=>

```
with CTE as (SELECT *,

timestampdiff(minute,transaction_timestamp,lag(transaction_timestamp) over())

as minutes_diff

FROM transactions)

SELECT count(minutes_diff) as payment_count FROM CTE

GROUP BY minutes_diff

HAVING

   abs(CTE.minutes_diff) <= 10

;
```

==========

Q129. Write an SQL query to find the total score for each gender on each day.

Return the result table ordered by gender and day in ascending order.

=>

```
SELECT gender, day,
```

sum(score_points) over (partition by gender ORDER BY day rows between unbounded preceding and current row) as total_score

FROM scores;

==========

Q130. A telecommunications company wants to invest in new countries. The company intends to invest in

the countries where the average call duration of the calls in this country is strictly greater than the

global average call duration.

Write an SQL query to find the countries where this company can invest.

Return the result table in any order.

=>

```
with country_phone as (SELECT p.*, c.name as country_name FROM person p JOIN

(SELECT name,

CASE

   WHEN LENGTH(country_code) < 3 then CONCAT("0", country_code)

   else country_code

end as new_code

FROM country) as c

ON

   left(p.phone_number, 3) = c.new_code

)


SELECT country_name, sum(total_dur)/sum(total_count) as final FROM (SELECT cp.country_name, (2 * cal.duration) as total_dur,  (2 * count(cp.country_name)) as total_count FROM calls as cal

JOIN

   country_phone as cp

ON

   cal.caller_id = cp.id
```

SQL Challenge Answer

GROUP BY cp.country_name, duration) as tmp

GROUP BY country_name ORDER BY final DESC LIMIT 1

==========

Q132. Write an SQL query to report the comparison result (higher/lower/same) of the average salary of

employees in a department to the company's average salary.

=>

```
with CTE as
(
   SELECT distinct(full_table.pay_date), round(AVG(full_table.amount), 0) as avg_pay,
   full_table.department_id, avg_table.company_avg_pay
   FROM
   (
      SELECT s.*, e.department_id FROM salary s
      JOIN
         (SELECT * FROM employee) as e
      ON
         e.employee_id = s.employee_id
   ) full_table
   JOIN
      (SELECT pay_date, round(AVG(amount),0) as company_avg_pay FROM salary
      GROUP BY pay_date) as avg_table
   ON
      avg_table.pay_date = full_table.pay_date
   GROUP BY full_table.pay_date, full_table.department_id,
   avg_table.company_avg_pay
)
```

```
SELECT pay_date, department_id,
CASE
   When avg_pay > company_avg_pay then "Higher"
   when avg_pay < company_avg_pay then "lower"
   when avg_pay = company_avg_pay then "same"
end as new_table_val
FROM CTE
GROUP BY pay_date, department_id, avg_pay
ORDER BY department_id, pay_date
;
```

==========

Q 133. Assume you have the table below containing information on Facebook user actions. Write a query to obtain the active user retention in July 2022. Output the month (in numerical format 1, 2, 3) and the number of monthly active users (MAUs).

Hint: An active user is a user who has user action ("sign-in", "like", or "comment") in the current month and last month.

=>

```
SELECT extract(month FROM event_date) as month, COUNT(user_id) as MAU FROM user_actions
WHERE
   event_date BETWEEN "2022-06-01 00:00:00" and "2022-06-30 12:00:00" AND event_type = "sign-in"
   and user_id in
   (
      SELECT user_id FROM user_actions
      WHERE event_date BETWEEN "2022-06-01 00:00:00" and "2022-06-30 12:00:00" AND event_type != "sign-in"
   )
GROUP BY month
```

SQL Challenge Answer

;

==========

Q137. Write an SQL query to show the second most recent activity of each user.

If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.

=>

```
select distinct username, activity, startDate, endDate
from
   (select user.*,
        rank() over (partition by username order by startDate desc) as rnk,
        count(activity) over (partition by username) as num
   from user_activity user) new_table
   WHERE
      (num != 1 and rnk = 2) or (num = 1 and rnk = 1)
;
```

==========

Q138. Write an SQL query to show the second most recent activity of each user.

If the user only has one activity, return that one. A user cannot perform more than one activity at the same time.

=>

```
select distinct username, activity, startDate, endDate
from
   (select user.*,
        rank() over (partition by username order by startDate desc) as rnk,
        count(activity) over (partition by username) as num
```

```
   from user_activity user) new_table
   WHERE
      (num != 1 and rnk = 2) or (num = 1 and rnk = 1)
;
```

==========

Q139. Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by

the last three characters of each name. If two or more students both have names ending in the same

last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID

=>

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

==========

Q140. Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in

alphabetical order.

=>

```
SELECT name FROM employee
ORDER BY name ASC
;
```

==========

SQL Challenge Answer

Q141. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee_id.

=>

```
SELECT name FROM employee
WHERE
   salary > 2000 and months < 10
ORDER BY employee_id ASC
;
```

==========

Q142. Write a query identifying the type of each record in the TRIANGLES table using its three side lengths.

Output one of the following statements for each record in the table:

● Equilateral: It's a triangle with sides of equal length.

● Isosceles: It's a triangle with sides of equal length.

● Scalene: It's a triangle with sides of differing lengths.

● Not A Triangle: The given values of A, B, and C don't form a triangle

=>

```
SELECT *,
CASE
   when a = b and b = c then "equilateral"
   when a = b and b != c and a+b > c then "isoceles"
   when a + b < c or b + c < a or c + a < b then "not a triangle"
   when a != b and b != c and a != c then "scalene"
   else "normal triangle"
end as triangle_value
from triangle;
```

==========

Q143. Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realise her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.
Write a query calculating the amount of error (i.e.: actual - miscalculated average monthly salaries), and round it up to the next integer.
=>

```
SELECT
   round(
      AVG(salary) -
         (
            SELECT AVG(salary) FROM employees_incorrect
         ),
      0)
   as diff_salaries
FROM employees_correct;
```

==========

Q144. We define an employee's total earnings to be their monthly salary * months worked, and the maximum total earnings to be the maximum total earnings for any employee in the Employee table. Write a query to find the maximum total earnings for all employees as well as the total number of employees who have maximum total earnings. Then print these values as 2 space-separated integers
=>

```
with CTE as
(
   SELECT *, (salary * months) as total_earnings FROM employees
```

SQL Challenge Answer

```
)
SELECT
   concat(total_earnings, " ", count(total_earnings)) as output_table
FROM CTE
WHERE
   total_earnings = (SELECT max(total_earnings) FROM CTE)
GROUP BY total_earnings
;
```

==========

Q145. Generate the following two result sets:

1. Query an alphabetically ordered list of all names in OCCUPATIONS, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).

Query the number of occurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format:

=>

```
(SELECT CONCAT(name,"(",left(occupation, 1),")") FROM job
ORDER BY name)
UNION ALL
(SELECT
CONCAT("There are a total of ",COUNT(occupation), " ", occupation, "s")
from job
GROUP BY occupation)
;
```

==========

Q 147. You are given a table, BST, containing two columns: N and P, where N represents the value of a node

in Binary Tree, and P is the parent of N

=>

```
SELECT n,
CASE
   when n not in (SELECT distinct(p) FROM nodes WHERE p is not NULL) then "Leaf"
   when p is NULL then "Root"
   else "Inner"
end as type_of_node
FROM nodes ORDER BY n;
```

==========

Q149. Two pairs (X1, Y1) and (X2, Y2) are said to be symmetric pairs if X1 = Y2 and X2 = Y1.

Write a query to output all such symmetric pairs in ascending order by the value of X. List the rows such that X1 ≤ Y1.

=>

```
SELECT x, y FROM val
   WHERE
      x in (SELECT y FROM val)
      AND
      y in (SELECT x FROM val)
      AND
      x <= y
LIMIT 1, 3;
```

==========

SQL Challenge Answer

Q153. In an effort to identify high-value customers, Amazon asked for your help to obtain data about users

who go on shopping sprees. A shopping spree occurs when a user makes purchases on 3 or more

consecutive days.

List the user IDs who have gone on at least 1 shopping spree in ascending order.

=>

```
with new_table as
(
   SELECT *,
   ifnull
   (
      datediff(transaction_date,
            lag(transaction_date) over(partition by user_id)),
   1) as lag_date
   FROM amazon
)

SELECT user_id FROM new_table
GROUP BY user_id, lag_date
having
   count(user_id) >= 3 and new_table.lag_date = 1
ORDER BY user_id ASC
;
```

==========

Q154. You are given a table of PayPal payments showing the payer, the recipient, and the amount paid. A

two-way unique relationship is established when two people send money back and forth. Write a

query to find the number of two-way unique relationships in this data.

=>

```
SELECT COUNT(person) as unique_relationships from
(
   SELECT person, COUNT(person) as person_count FROM
   (
      (SELECT concat(payer_id, recipient_id) as person FROM payments WHERE payer_id < recipient_id)
      UNION ALL
      (SELECT concat(recipient_id, payer_id) as person FROM payments where recipient_id < payer_id)
   ) nt
   GROUP BY person
) pt
WHERE
   person_count >= 2
;
```

==========

Q155. Assume you are given the table below containing information on Facebook user logins. Write a query

to obtain the number of reactivated users (which are dormant users who did not log in the previous

month, then logged in during the current month).

Output the current month (in numerical) and number of reactivated users.

=>

```
SELECT extract(month FROM login_date) as current_month, COUNT(user_id) as reactivations FROM
(
   SELECT user_id, login_date,
   datediff(login_date, lag(login_date) over(partition by user_id)) as lag_date
   FROM user_login
```

SQL Challenge Answer

```
) as react_table
WHERE
   lag_date >= 31
GROUP BY current_month
;
```

==========

Q156. Assume you are given the table below on user transactions. Write a query to obtain the list of customers whose first transaction was valued at $50 or more. Output the number of users.

=>

```
SELECT user_id FROM
(
   SELECT user_id, spend,
   lag(transaction_date) over(partition by user_id) as lag_num
   FROM transactions
) as nt
WHERE
   lag_num is NULL and spend >= 50;
```

==========

Q158. In an effort to identify high-value customers, Amazon asked for your help to obtain data about users

who go on shopping sprees. A shopping spree occurs when a user makes purchases on 3 or more consecutive days.

List the user IDs who have gone on at least 1 shopping spree in ascending order.

=>

```
with new_table as
```

```
(
   SELECT *,
   ifnull
   (
      datediff(transaction_date,
            lag(transaction_date) over(partition by user_id)),
   1) as lag_date
   FROM amazon
)

SELECT user_id FROM new_table
GROUP BY user_id, lag_date
having
   count(user_id) >= 3 and new_table.lag_date = 1
ORDER BY user_id ASC
;
```

==========

Q161. Your team at Accenture is helping a Fortune 500 client revamp their compensation and benefits

program. The first step in this analysis is to manually review employees who are potentially overpaid

or underpaid.

An employee is considered to be potentially overpaid if they earn more than 2 times the average salary

for people with the same title. Similarly, an employee might be underpaid if they earn less than half of

the average for their title. We'll refer to employees who are both underpaid and overpaid as

compensation outliers for the purposes of this problem.

Write a query that shows the following data for each compensation outlier: employee ID, salary, and

whether they are potentially overpaid or potentially underpaid (refer to Example Output below).

SQL Challenge Answer

=>

```
SELECT employee_id, salary, over_under FROM
(
   SELECT employee_id, salary, title,
   CASE
      when (AVG(salary) over(partition by title)/salary) > 2 then "Overpaid"
      when (AVG(salary) over(partition by title)/salary) < 0.5 then "Underpaid"
      else "Correct"
   end as over_under
   FROM accenture
) as nt
WHERE
   over_under != "Correct"
;
```

==========

Q162. You are given a table of PayPal payments showing the payer, the recipient, and the amount paid. A

two-way unique relationship is established when two people send money back and forth. Write a

query to find the number of two-way unique relationships in this data.

=>

```
SELECT COUNT(person) as unique_relationships from
(
   SELECT person, COUNT(person) as person_count FROM
   (
      (SELECT concat(payer_id, recipient_id) as person FROM payments WHERE payer_id < recipient_id)
      UNION ALL
```

```
      (SELECT concat(recipient_id, payer_id) as person FROM payments where recipient_id < payer_id
)
   ) nt
   GROUP BY person
) pt
WHERE
   person_count >= 2
;
```

==========
==========