Commonly asked SQL assessment questions:

Output the user and number of the days between each user's first and last post.

```
SELECT user_id,
(MAX(post_date::DATE) - MIN(post_date::DATE)) as days_between
FROM posts
WHERE DATE_PART('year', post_date::DATE) = 2021
GROUP BY user_id
HAVING COUNT(post_id)>1;
```

_____

Write a query to find the top 2 power users who sent the most messages on Microsoft Teams in August 2022. Display the IDs of these 2 users along with the total number of messages they sent. Output the results in descending count of the messages.

```
SELECT sender_id, count(distinct content) as message_count
FROM messages
where EXTRACT(month from sent_date) = '8'
  and EXTRACT(year from sent_date) = '2022'
group by sender_id
order by message_count desc LIMIT 2
```

_____

You are given the tables below containing information on Robinhood trades and users. Write a query to list the top three cities that have the most completed trade orders in descending order.
Output the city and number of orders.

```
SELECT city, COUNT(order_id) as total_orders
FROM users u inner join trades t on u.user_id=t.user_id
where status='Completed'
group by city
order by total_orders desc
limit 3;
```

_____

Given the reviews table, write a query to get the average stars for each product every month.
The output should include the month in numerical value, product id, and average star rating rounded to two decimal places. Sort the output based on month followed by the product id.

```
SELECT EXTRACT(month from submit_date) as mth,
product_id product,
round(avg(stars),2) avg_stars
FROM reviews
group by EXTRACT(month from submit_date), product
order by mth, product;
```

_____

Assume you have an events table on app analytics. Write a query to get the app's click-through rate (CTR %) in 2022. Output the results in percentages rounded to 2 decimal places.

```
SELECT app_id,
round(100.0 * sum(case when event_type = 'click' then 1 end) /
sum(case when event_type = 'impression' then 1 end),2) ctr
FROM events
where extract (year from timestamp) = '2022'
group by app_id;
```

Write a query that outputs the name of each credit card and the difference in issued amount between the month with the most cards issued, and the least cards issued. Order the results according to the biggest difference.

```
SELECT card_name,
MAX(issued_amount)-min(issued_amount) difference
FROM monthly_cards_issued
GROUP BY card_name
order by difference DESC;
```

You are trying to find the mean number of items bought per order on Alibaba, rounded to 1 decimal place. However, instead of doing analytics on all Alibaba orders, you have access to a summary table, which describes how many items were in an order (item_count), and the number of orders that had that many items (order_occurrences).

```
SELECT
ROUND(SUM(item_count::decimal*order_occurrences)/SUM(order_occurrences),1) mean
FROM items_per_order;
```

Write a query to display the ids of the users who did not confirm on the first day of sign-up, but confirmed on the second day.

```
SELECT user_id
FROM emails e inner JOIN texts t on e.email_id=t.email_id
where t.signup_action = 'Confirmed'
and t.action_date-e.signup_date = INTERVAL '1 day';
```

Write a query to find the top 3 most profitable drugs sold, and how much profit they made. Assume that there are no ties in the profits. Display the result from the highest to the lowest total profit.

```
SELECT drug, total_sales-cogs as total_profit FROM pharmacy_sales
order by total_profit DESC
limit 3;
```

Write a query to find out which manufacturer is associated with the drugs that were not profitable and how much money CVS lost on these drugs.
Output the manufacturer, number of drugs and total losses. Total losses should be in absolute value.
Display the results with the highest losses on top.

```
SELECT manufacturer, count(drug) as drug_count,
abs(sum(total_sales-cogs)) as total_loss
FROM pharmacy_sales
WHERE total_sales - cogs <= 0
group by manufacturer
order by total_loss DESC;
```

_____

Write a query to find the total sales of drugs for each manufacturer. Round your answer to the closest million, and report your results in descending order of total sales.
Because this data is being directly fed into a dashboard which is being seen by business stakeholders, format your result like this: "$36 million".

```
SELECT manufacturer,
concat('$',round(sum(total_sales)/1000000),' ','million') sale
FROM pharmacy_sales
group by manufacturer
order by sum(total_sales) desc;
```

_____

Write a query to find how many UHG members made 3 or more calls. case_id column uniquely identifies each call made.

```
select COUNT(*) member_count FROM
(
SELECT policy_holder_id
FROM callers
group by policy_holder_id
HAVING count(case_id)>2
) sq
```

_____

Calls to the Advocate4Me call centre are categorised, but sometimes they can't fit neatly into a category. These uncategorised calls are labelled “n/a”, or are just empty (when a support agent enters nothing into the category field).
Write a query to find the percentage of calls that cannot be categorised. Round your answer to 1 decimal place.

```
SELECT
  ROUND(100.0 *
    COUNT(case_id)/
      (SELECT COUNT(*) FROM callers),1) AS uncategorised_call_pct
FROM callers
WHERE call_category IS NULL
```

```
  OR call_category = 'n/a';
```

_____

Write a query to list the candidates who possess all of the required skills for the job. Sort the output by candidate ID in ascending order.

```
SELECT candidate_id
FROM candidates
WHERE skill IN ('Python', 'Tableau', 'PostgreSQL')
GROUP BY candidate_id
HAVING COUNT(skill) = 3
ORDER BY candidate_id;
```

_____

Output the total viewership for laptop and mobile devices in the format of "laptop_views" and "mobile_views".

```
SELECT
  SUM(CASE WHEN device_type = 'laptop' THEN 1 ELSE 0 END) AS laptop_views,
  SUM(CASE WHEN device_type IN ('tablet', 'phone') THEN 1 ELSE 0 END) AS mobile_views
FROM viewership;
```

_____

Repeated payments

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 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 minute_difference <= 10;
```

_____

Pizza toppings

Given a list of pizza toppings, consider all the possible 3-topping pizzas, and print out the total cost of those 3 toppings. Sort the results with the highest total cost on the top followed by pizza toppings in ascending order.

Break ties by listing the ingredients in alphabetical order, starting from the first ingredient, followed by the second and third.

```
SELECT
  CONCAT(p1.topping_name, ',', p2.topping_name, ',', p3.topping_name) AS pizza,
  p1.ingredient_cost + p2.ingredient_cost + p3.ingredient_cost AS total_cost
FROM pizza_toppings AS p1
INNER JOIN pizza_toppings AS p2
  ON p1.topping_name < p2.topping_name
INNER JOIN pizza_toppings AS p3
  ON p2.topping_name < p3.topping_name
ORDER BY total_cost DESC, pizza;
```

_____-

Active user retention

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).

```
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);
```

_____

Y-on-Y growth rate

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).

```
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;
```

_____

Median Google Search Frequency

Google's marketing team is making a Superbowl commercial and needs a simple statistic to put on their TV ad: the median number of searches a person made last year.

However, at Google scale, querying the 2 trillion searches is too costly. Luckily, you have access to the summary table which tells you the number of searches made last year and how many Google users fall into that bucket.

Write a query to report the median of searches made by a user. Round the median to one decimal point.

```
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;
```

_____

Advertiser status

Write a query to update the Facebook advertiser's status using the daily_pay table. Advertiser is a two-column table containing the user id and their payment status based on the last payment and daily_pay table has current information about their payment. Only advertisers who paid will show up in this table.

Output the user id and current payment status sorted by the user id.

```
WITH payment_status AS (
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;
```

_____

3 transaction of a user

Assume you are given the table below on Uber transactions made by users. Write a query to obtain the third transaction of every user. Output the user id, spend and transaction date.

```
SELECT
  user_id,
  spend,
  transaction_date
FROM (
  SELECT
    user_id,
    spend,
    transaction_date,
    ROW_NUMBER() OVER (
      PARTITION BY user_id ORDER BY transaction_date) AS row_num
  FROM transactions) AS trans_num
WHERE row_num = 3;
```

_____

Histogram of Users and Purchases

Output the user's most recent transaction date, user ID and the number of products sorted by the transaction date in chronological order.

```
SELECT
  transaction_date,
  user_id,
```

```
  COUNT(product_id) AS purchase_count
FROM (
  SELECT
    transaction_date,
    user_id,
    product_id,
    RANK() OVER (
      PARTITION BY user_id
      ORDER BY transaction_date DESC) AS days_rank
  FROM user_transactions) AS latest_transaction
WHERE days_rank = 1
GROUP BY transaction_date, user_id
ORDER BY transaction_date;
```

_____

Duplicate job listings refer to two jobs at the same company with the same title and description.

```
SELECT COUNT(distinct company_id) as co_w_duplicate_jobs
from (
select company_id, title, description, COUNT(company_id)
from job_listings
group by company_id, title, description
having COUNT(company_id) > 1
) jobs
```

_____

Output the top 5 artist names in ascending order along with their song appearances ranking (not the number of song appearances, but the rank of who has the most appearances). The order of the rank should take precedence.

```
select * FROM
(
SELECT artist_name,
dense_rank() OVER(ORDER BY count(rank) desc) artist_rank
FROM artists a inner join songs s on a.artist_id=s.artist_id
inner join global_song_rank g on s.song_id=g.song_id
where rank < 11
GROUP BY artist_name
ORDER BY count(rank) DESC
) new
where artist_rank < 6;
```

_____

Assume you are given the table below on Uber transactions made by users. Write a query to obtain the third transaction of every user. Output the user id, spend and transaction date.

```
select user_id, spend, transaction_date FROM
(
SELECT user_id, spend, transaction_date,
```

```
row_number() OVER(PARTITION BY user_id ORDER BY transaction_date) rownum
FROM transactions
) tp
where CAST(rownum as int)=3;
```

_____

Calculate the 3-day rolling average of tweets published by each user for each date that a tweet was posted. Output the user id, tweet date, and rolling averages rounded to 2 decimal places.

```
SELECT
  user_id,
  tweet_date,
  ROUND(
    AVG(tweet_num) OVER (
      PARTITION BY user_id
      ORDER BY user_id, tweet_date
      ROWS BETWEEN 2 PRECEDING AND CURRENT ROW), 2)
  AS rolling_avg_3d
FROM (
  SELECT
    user_id,
    tweet_date,
    COUNT(DISTINCT tweet_id) AS tweet_num
  FROM tweets
  GROUP BY user_id, tweet_date) AS tweet_count;
```

_____

Write a query to obtain a breakdown of the time spent sending vs. opening snaps (as a percentage of **total time spent** on these activities) for each age group.
Output the age bucket and percentage of sending and opening snaps. Round the percentage to 2 decimal places.

```
SELECT
  age.age_bucket,
  ROUND(100.0 *
    SUM(activities.time_spent) FILTER (WHERE activities.activity_type = 'send')/
      SUM(activities.time_spent),2) AS send_perc,
  ROUND(100.0 *
    SUM(activities.time_spent) FILTER (WHERE activities.activity_type = 'open')/
      SUM(activities.time_spent),2) AS open_perc
FROM activities
INNER JOIN age_breakdown AS age
  ON activities.user_id = age.user_id
WHERE activities.activity_type IN ('send', 'open')
GROUP BY age.age_bucket;
```

_____

Diff between 2 latest entries

```
WITH evt AS
(
  select
    event_type,
    value,
    rank() over(partition by event_type order by time desc) rnk
  from events
)
select
    t1.event_type, t1.value - t2.value
from evt t1, evt t2
where t1.event_type = t2.event_type
and t1.rnk = 1
and t2.rnk = 2
```

_____

Soccer winning and losing team points

```
 select team_id, team_name,
     coalesce(sum(case when team_id = host_team then
                 (
                  case when host_goals > guest_goals then 3
                  when host_goals = guest_goals then 1
                  when host_goals < guest_goals then 0
                  end
                 )
                 when team_id = guest_team then
                 (
                 case when guest_goals > host_goals then 3
                 when guest_goals = host_goals then 1
                 when guest_goals < host_goals then 0
                 end
                 )
                 end), 0) as num_points
    from Teams
    left join Matches
    on
    Teams.team_id = Matches.host_team
    or Teams.team_id = Matches.guest_team
    group by team_id, team_name
    order by num_points desc, team_id;
```

_____