

# Overview of SQL Interview Questions

The first step of analytics for most workflows involves quick slicing and dicing of data in SQL. That's why being able to write basic queries efficiently is a very important skill. Although many may think that SQL simply involves SELECTs and JOINs, there are many other operators and details involved for powerful SQL workflows. For example, utilizing subqueries is important and allows you to manipulate subsets of data by which later operations can be performed, while window functions allow you to cut data without combining rows explicitly using a GROUP BY. The questions asked within SQL are usually quite practical to the company at hand - a company like Facebook might ask about various user or app analytics question, whereas a company like Amazon will ask about products and transactions.



# Overview of Databases Design Questions

Although it isn't explicitly necessary to know the inner workings of databases (which is typically more data engineering oriented), it helps to have a high level understanding of basic concepts in Databases and Systems. Databases refers not to specific ones but more so how they operate at a high level and what design decisions and trade-offs are made during construction and querying. "Systems" is a broad term but refers to any set of frameworks or tools by which analysis of large volumes of data relies on. For example, a common interview topic is the MapReduce framework which is heavily utilized at many companies for parallel processing of large datasets.



# 20 SQL Data Science Interview Questions

1. **[Robinhood - Easy]** Assume you are given the below tables for trades and users. Write a query to list the top 3 cities which had the highest number of completed orders.

## trades

 column_name	 type
<u>order_id</u>	integer
<u>user_id</u>	integer
<u>symbol</u>	string ("NFLX", "FB", etc.)
<u>price</u>	float
<u>quantity</u>	integer
<u>side</u>	string ("buy", "sell")
<u>status</u>	string ("complete", "cancelled")
<u>timestamp</u>	datetime

## users

 column_name	 type
<u>user_id</u>	integer
<u>city</u>	string
<u>email</u>	string
<u>signup_date</u>	datetime

2. **[Facebook - Easy]** Assume you have the below events table on app analytics. Write a query to get the click-through rate per app in 2019.

## events

 column_name	 type
<u>app_id</u>	integer
<u>event_id</u>	string ("impression", "click")
<u>timestamp</u>	datetime

3. **[Uber - Easy]** Assume you are given the below table for spending activity by product type. Write a query to calculate the cumulative spend for each product over time in chronological order.

## total\_trans

<u>Aa</u> column_name	≡ type
<u>order_id</u>	integer
<u>user_id</u>	integer
<u>product_id</u>	string
<u>spend</u>	float
<u>date</u>	datetime

4. **[Snapchat - Easy]** Assume you have the below tables on sessions that users have, and a users table. Write a query to get the active user count of daily cohorts.

## sessions

<u>Aa</u> column_name	≡ type
<u>user_id</u>	integer
<u>session_id</u>	integer
<u>date</u>	datetime

## users + Add a view

<u>Aa</u> column_name	≡ type
<u>user_id</u>	integer
<u>email</u>	string
<u>date</u>	datetime

5. **[Facebook - Easy]** Assume you are given the below tables on users and user posts. Write a query to get the distribution of the number of posts per user.

## users



<u>Aa</u> column_name	≡ type
<u>user_id</u>	integer
<u>date</u>	datetime

## posts

<u>Aa</u> column_name	≡ type
<u>post_id</u>	integer
<u>user_id</u>	string
<u>body</u>	string
<u>date</u>	datetime



6. **[Amazon - Easy]** Assume you are given the below table on purchases from users. Write a query to get the number of people that purchased at least one product on multiple days.

## purchases

 column_name	 type
<u>purchase_id</u>	integer
<u>user_id</u>	integer
<u>product_id</u>	integer
<u>quantity</u>	integer
<u>price</u>	float
<u>purchase_time</u>	datetime



7. **[Opendoor - Easy]** Assume you are given the below table on house prices from various zip codes that have been listed. Write a query to get the top 5 zip codes by market share of house prices for any zip code with at least 10000 houses.

## housing

 column_name	 type
<u>house_id</u>	integer
<u>zip_code</u>	integer
<u>price</u>	float
<u>listing_date</u>	datetime


8. **[Etsy - Easy]** Assume you are given the below table on transactions from users for purchases. Write a query to get the list of customers where their earliest purchase was at least \$50.

### user\_transactions

 column_name	 type
<u>transaction_id</u>	integer
<u>product_id</u>	integer
<u>user_id</u>	integer
<u>spend</u>	float
<u>transaction_date</u>	datetime


9. **[Disney - Easy]** Assume you are given the below table on watch times (in minutes) for all users, where each user is based in a given city. Write a query to return all pairs of cities that have total watch times within 10000 minutes of one another.

## watch\_activity

<u>Aa</u> column_name	 type
<u>user_id</u>	integer
<u>session_id</u>	integer
<u>watch_time</u>	float
<u>city_name</u>	string
<u>date</u>	datetime

10. **[Twitter - Easy]** Assume you are given the below table on tweets by each user over a period of time. Calculate the 7-day rolling average of tweets by each user for every date.

## tweets Add a view

<u>Aa</u> column_name	 type
<u>tweet_id</u>	integer
<u>msg</u>	string
<u>user_id</u>	integer
<u>tweet_date</u>	datetime

11. **[Stitch Fix - Easy]** Assume you are given the below table on transactions from users. Write a query to get the number of users and total products bought per latest transaction date where each user is bucketed into their latest transaction date.



## user\_transactions

<u>Aa</u> column_name	≡ type
<u>transaction_id</u>	integer
<u>product_id</u>	integer
<u>user_id</u>	integer
<u>spend</u>	float
<u>transaction_date</u>	datetime

12. **[Amazon - Easy]** Assume you are given the below table on customer spend amounts on products in various categories. Calculate the top three most bought item within each category in 2020.

## product\_spend

<u>Aa</u> column_name	≡ type
<u>transaction_id</u>	integer
<u>category_id</u>	integer
<u>product_id</u>	integer
<u>user_id</u>	integer
<u>spend</u>	float
<u>transaction_date</u>	datetime

13. **[DoorDash - Easy]** Assume you are given the below table on transactions on delivery locations and times for meals - a start location, an end location, and timestamp for a given meal\_id. Certain locations are aggregation locations - where meals get sent to, and where meals then go to their final destination. Calculate the delivery time per meal to each final destination from a particular aggregation location, loc\_id = 4.

## delivery\_times

<u>Aa</u> column_name	≡ type
<u>meal_id</u>	integer
<u>start_loc_id</u>	integer
<u>end_loc_id</u>	integer
<u>cost</u>	float
<u>timestamp</u>	timestamp

14. **[Facebook - Medium]** Assume you have the below tables on user actions. Write a query to get the active user retention by month.

## user\_actions

<u>Aa</u> column_name	≡ type
<u>user_id</u>	integer
<u>event_id</u>	string ("sign-in", "like", "comment")
<u>timestamp</u>	datetime

15. **[Twitter - Medium]** Assume you are given the below tables for the session activity of users. Write a query to assign ranks to users by the total session duration for the different session types they have had between a start date (2020-01-01) and an end date (2020-02-01).

## sessions

<u>Aa</u> column_name	<u>≡</u> type
<u>session_id</u>	integer
<u>user_id</u>	integer
<u>session_type</u>	string
<u>duration</u>	integer
<u>start_time</u>	datetime

16. **[Snapchat - Medium]** Assume you are given the below tables on users and their time spent on sending and opening Snaps. Write a query to get the breakdown for each age breakdown of the percentage of time spent on sending versus opening snaps.

## activities

<u>Aa</u> column_name	≡ type
<u>activity_id</u>	integer
<u>user_id</u>	integer
<u>type</u>	string ('send', 'open')
<u>time_spent</u>	float
<u>activity_date</u>	datetime

## age\_breakdown

<u>Aa</u> column_name	≡ type
<u>user_id</u>	integer
<u>age_bucket</u>	string

17. **[Google - Medium]** Assume you are given the below table on sessions from users, with a given start and end time. A session is concurrent with another session if they overlap in the start and end times. Write a query to output the session that is concurrent with the largest number of other sessions.

## sessions

<u>Aa</u> column_name	≡ type
<u>session_id</u>	integer
<u>start_time</u>	datetime
<u>end_time</u>	datetime

18. **[Yelp - Medium]** Assume you are given the below table on reviews from users. Define a top-rated place as a business whose reviews only consist of 4 or 5 stars. Write a query to get the number and percentage of businesses that are top-rated places.

## reviews

<u>Aa</u> column_name	≡ type
<u>business_id</u>	integer
<u>user_id</u>	integer
<u>review_text</u>	string
<u>review_stars</u>	integer
<u>review_date</u>	datetime

19. **[Google - Medium]** Assume you are given the below table of measurement values from a sensor for several days. Each measurement can happen several times in a given day. Write a query to output the sum of values for every odd measurement and the sum of values for every even measurement by date.

## measurements

<u>Aa</u> column_name	≡ type
<u>measurement_id</u>	integer
<u>measurement_value</u>	float
<u>measurement_time</u>	datetime

20. [Etsy - Medium] Assume you are given the below table on transactions from various product search results from users on Etsy. For every given product keyword, there are multiple positions that being A/B tested, and user feedback is collected on the relevance of results (from 1-5). There are many displays for each position of every product, each of which is captured by a display\_id. Define a highly relevant display as one whereby the corresponding relevance score is at least 4. Write a query to get all products having at least one position with > 80% highly relevant displays.

## product\_searches

<u>Aa</u> column_name	≡ type
<u>product</u>	string
<u>position</u>	integer
<u>display_id</u>	integer
<u>relevance</u>	integer
<u>submit_time</u>	datetime

# 10 Database And Systems Design Interview Questions

21. **[MongoDB - Easy]** For each of the ACID properties, give a one-sentence description of each property and why are these properties important?
22. **[VMWare - Easy]** What are the three major steps of database design? Describe each step.
23. **[Microsoft - Easy]** What are the requirements for a primary key?
24. **[DataStax - Easy]** A B+ tree can contain a maximum of 5 pointers in a node. What is the minimum number of keys in leaves?
25. **[Databricks - Easy]** Describe MapReduce and the operations involved.
26. **[Microsoft - Easy]** Name one major similarity and difference between a WHERE clause and a HAVING clause in SQL.
27. **[Facebook - Easy]** How does a trigger allow you to build business logic into a database?
28. **[DataStax - Easy]** What are the six levels of database security and briefly explain what each one entails?
29. **[Databricks - Easy]** Say you have a shuffle operator, whereby the input is a dataset and the output is simply a randomly ordered version of that dataset. Describe the algorithm steps in English.
30. **[Rubrik - Medium]** Describe what a cache is and what a block is. Say you have a fixed amount of total data storage - what are the some trade-offs in varying block size?

## SQL And Database Interview Solutions

### Problem #4 Solution:

By definition, daily cohorts are active users from a particular day. First, we can use a subquery to get the sessions of new users by day using an inner

join with users. This is to filter for only active users by a particular join date for the cohort. Then we can get a distinct count to return the active user count:

```
WITH new_users_by_date AS (  
    SELECT sessions.*  
    FROM sessions  
    JOIN users on  
        sessions.user_id = users.user_id  
        sessions.date = users.date  
)  
SELECT date, COUNT(DISTINCT user_id) as active_user_count  
    FROM new_users_by_date  
    GROUP BY 1 ORDER BY 1 ASC
```

### Problem #8 Solution:

Although we could use a self join on `transaction_date = MIN(transaction_date)` for each user, we can also use the `RANK()` window function to get the ordering of purchase by customer, and then use that subquery to filter on customers where the first purchase (rank one) is at least 50 dollars. Note that this requires the subquery to include spend as well

```
WITH purchase_rank AS (  
    SELECT user_id, spend,  
        RANK() OVER  
            (PARTITION BY user_id ORDER BY transaction_date ASC)  
as rank  
    FROM user_transactions u  
)  
  
SELECT  
    user_id  
FROM  
    purchase_rank  
WHERE rank = 1 AND spend >= 50.00
```



### Problem #11 Solution:

First, we need to get the latest transaction date for each user, along with the number of products they have purchased. This can be done in a subquery where we GROUP BY user\_id and take a COUNT(DISTINCT product\_id) to get the number of products they have purchased, and a MAX(transaction\_date) to get the latest transaction date (while casting to a date). Then, using this subquery, we can simply do an aggregation by the transaction date column in the previous subquery, while doing a COUNT() on the number of users, and a SUM() on the number of products:

```
WITH latest_date AS (  
    SELECT user_id,  
           COUNT(DISTINCT product_id) AS num_products,  
           MAX(transaction_date::DATE) AS curr_date  
    FROM user_transactions  
    GROUP BY )  
  
SELECT curr_date,  
       COUNT(user_id) AS num_users,  
       SUM(num_products) AS total_products  
FROM latest_date  
GROUP BY 1
```

### Problem #16 Solution:

We can get the breakdown of total time spent on each activity by each user by filtering out for the activity\_type and taking the sum of time spent. In doing this, we want to do an outer join with the age bucket to get the total time by age bucket for both activity types. This results in the below two subqueries. Then, we can use these two subqueries to sum them by joining on the appropriate age bucket and take the proportion for send time and the proportion for open time per age bucket:

```

WITH send_timespent AS (
    SELECT age_breakdown.age_bucket, SUM(activities.time_spent)
AS send_timespent
    FROM age_breakdown
    LEFT JOIN activities on age_breakdown.user_id =
activities.user_id
    WHERE activities.type = 'send'
    GROUP BY 1
),
open_timespent AS (
    SELECT age_breakdown.age_bucket, SUM(activities.time_spent)
AS open_timespent
    FROM age_breakdown
    LEFT JOIN activities on age_breakdown.user_id =
activities.user_id
    WHERE activities.type = 'open'
    GROUP BY 1
),

SELECT a.age_bucket,
    s.send_timespent / (s.send_timespent + o.open_timespent) AS
pct_send,
    o.open_timespent / (s.send_timespent + o.open_timespent) AS
pct_open,
FROM age_breakdown a
LEFT JOIN send_timespent s ON a.age_bucket = s.age_bucket
LEFT JOIN open_timespent o ON a.age_bucket = o.age_bucket
GROUP BY 1

```

## Problem #18 Solution:

First, we need to get the places where the reviews are all 4 or 5 stars. We can do this using a HAVING clause, instead of a WHERE clause since the reviews need to all be 4 stars or above. For the HAVING condition, we can use a CASE statement that filters for 4 or 5 stars and then take a SUM over them. This can then be compared with the total row count of the particular business\_id reviews to ensure that the count of top reviews matches with the total review count. With the relevant businesses, we can then do an outer join with the original table on business\_id to get a COUNT of distinct

business\_id matches, and then the percentage by comparing the COUNT from the top places with the overall COUNT of business\_id:

```
WITH top_places AS (  
    SELECT business_id  
    FROM user_transactions  
    GROUP BY 1  
    HAVING  
        SUM(CASE WHEN rating >= 4 THEN 1 ELSE 0 END) = COUNT(*)  
    )  
  
SELECT  
    COUNT(DISTINCT t.business_id) AS top_places,  
    COUNT(DISTINCT t.business_id)/COUNT(DISTINCT r.business_id)  
AS top_places_pct  
FROM reviews r  
LEFT JOIN top_places t  
    ON r.business_id = t.business_id
```

### **Problem #21 Solution:**

ACID is a set of properties that ensures that even in the event of errors, power outages, and other unforeseen circumstances, a database will still hold up. It is an important framework for studying database systems.

A: Atomicity, meaning that an entire transaction happens as a whole or it doesn't happen at all (no partial transactions). This prevents partial updates which can be problematic. Therefore, transactions cannot be "in progress" to any user.

C: Consistency, meaning that there are integrity constraints such that the database is consistent before and after a given transaction. Essentially, if I search for what is in Row 3 and then do so again without any modifications to the database (no deletes or inserts), I should get the same result. Any

referential integrity is handled by appropriate checks for the primary and foreign keys.

I: Isolation, meaning that transactions happen in isolation and thus multiple transactions can occur independently without interference. This maintains concurrency properly.

D: Durability, meaning that once a transaction is complete, that information is now updated in the database even in the event of a system failure.

### **Problem #25 Solution:**

MapReduce is a framework that is heavily used in processing large datasets across a large number of clusters (many machines). Within the groups of machines, there are worker nodes (which carry out the computations) and master nodes (which delegate the tasks for each worker node). The three steps are generally as follows.

1. Map step: each worker node applies a specific map operations on input data (which the master node ensures will not be duplicated) and writes the output to a memory buffer.
2. Data is re-distributed based on the output keys from the prior step's map function, such that for any given key, it is located on the same worker node.
3. Each worker node processes each key in parallel using specific reduce operations to get the output result.

Since the mapping and reducing functions can all be done in parallel, the amount of processing done is only limited by the amount of compute and data available. Note that there are edge cases if there are failures from worker nodes - in those cases, the desired operations can be re-scheduled by the master node.

## **Problem #27 Solution:**

A trigger is like a CHECK condition, but every time there is an update to the database, the trigger condition will be checked to see if it has been violated. This allows you to implement some level of control and assurance that all your data entries meet a certain condition. For instance, a trigger that states that all ID values must be  $> 0$  will ensure that you get no null values or negative values. When someone tries to enter such a value, the entry will not go through.

That being said, there are reasons why to not include business logic within database triggers. For example: 1) introduction of side effects which lead to bugs or other unintended consequences, or 2) performance problems in which case there is a cascading effect on triggers that leads to locking and other issues.