2021-06-30T18:00:00+02:00
Zahin Rahman

Top 15 SQL Interview Questions in 2021

If you are on the lookout for a job in the field of data, you're almost guaranteed to face SQL interview questions. You might even be asked to complete some kind of SQL coding test. According to KDnuggets, irrespective of the specific role in data – data analyst, data engineer, data scientist, or business intelligence analyst – SQL lands in the top three skills found in postings across Indeed, SimplyHired, and Monster.

SQL's demand arises largely from its widespread application across a whole host of uses, including relational databases, data warehousing, NoSQL databases, and distributed data processing. SQL powers the majority of modern-day transaction processes, cloud storage solutions, search engines, analytics applications, and business intelligence operations across virtually all sectors.

This guide will help you brush up on your SQL skills, get a confidence boost, and be ready to ace your interview!

Below is a curated list of SQL interview questions along with some guidance on how to approach them. They are based on research into top tech companies' hiring practices and SQL job interview trends. That being said, be sure to put in your due diligence and craft the answers with your own knowledge and experience!

The best way to practice SQL is with our SQL Practice track. It has 600+ interactive exercises, and we keep adding more!

Can You Answer These 15 SQL Interview Questions?

1. What Is SQL?

SQL, or Structured Query Language, is the standard language used to access, update, and manipulate relational databases. It allows users to filter, sort, and extract data from a relational database.

A relational database stores data in tables made up of columns (attributes) and rows (values). Tables can be related to other tables based on row values. For more information on this, see SQL Terms Beginners Should Know and What Is the Benefit of Foreign Keys in SQL?.

SQL can be used in technologies other than relational databases and is considered a key skill for many data and software engineering jobs.

2. What Are the Basic Elements of a SQL Query?

The exact structure of each SQL query will obviously depend on the problem you are trying to solve. However, there are some basic elements that form most SQL queries. `SELECT`, `FROM`, `JOIN`, `WHERE`, `GROUP BY`, `HAVING`, and `ORDER BY` are common elements of a SQL query.

The `SELECT` statement is used to fetch data from a database; the `FROM` command is used to specify the table(s) within the database; `JOIN` is used to connect tables within the database using shared column values; `ORDER BY` is used to sort the data, and `GROUP BY` is used group the data. One of our prior articles discusses basic SQL query elements in detail.

3. What Is the SQL WHERE clause?

`WHERE` is applied after the `FROM` or `JOIN` statements. It retrieves those records that meet the specified condition. `WHERE` conditions can use comparison operators (`Age > 21`) or logical operators (`Name NOT 'Jones'`). You can specify complex `WHERE` conditions with `AND` or `OR` (`Name NOT 'Jones' OR 'Garcia'`).

4. How Does GROUP BY Work?

The `GROUP BY` clause groups records with the same values in a given column. For example, if you wrote `GROUP BY` department, all employees with the same value in the department column would be one group.

This is very useful when you need to compute aggregate statistics for that group (e.g. if you wanted to find average salaries by department).

`GROUP BY` follows the optional `WHERE` clause and is most often used when aggregate functions like `COUNT()`, `SUM()`, and `AVG()` are referenced in the `SELECT` statement. In one of our prior articles, we went over how GROUP BY works and some practical examples of SQL GROUP BY – feel free to check those out!

5. Name and Explain Some Common Aggregate Functions.

In SQL, an aggregate function uses the data from multiple rows to calculate a single result. Common aggregate functions include `COUNT` (returns the number of rows), `SUM` (returns the sum of all values in that column), `AVG` (returns the average value), `MIN` (returns the smallest value) and `MAX` (returns the largest value).

Aggregate functions are often used in conjunction with `GROUP BY` and `HAVING` to compute statistics for a data set. You can find more detailed information and examples in our article An Overview of SQL Aggregate Functions.

6. What Does the HAVING Clause Do?

The SQL HAVING clause is used for filtering the data created by the `GROUP BY` clause according to the specified condition. This clause filters the data on the grouped records rather than the individual records, so it must follow `GROUP BY`. It cannot be used without a `GROUP BY` clause.

7. What's the Difference Between HAVING and WHERE?

The difference between the WHERE and HAVING clause is in what each one filters. The `WHERE` clause filters individual records; it cannot be applied to grouped or aggregated records. The `HAVING` clause filters grouped or aggregated records.

A query can have both clauses; `WHERE` is applied on the individual records, followed by `GROUP BY` and then the `HAVING` clause.

8. Describe the Action of JOIN and the Different Join Types.

A `JOIN` merges data from two or more tables using matching values in specified columns; this is called the join condition and is listed after the `ON` keyword. For example, you could join the tables **`employee`** and **`role`** based on the matching values in the `employee.role_id` and `role.id` columns.

There are different types of JOINs, and each one gives slightly different results:

> INNER JOIN returns only rows that have matching values in both tables. This is the default join type.
>
> LEFT JOIN returns all rows from the left table (after `FROM`) and matching rows from the right table (after `JOIN`). If there is no matching row for the right table, NULLs will be returned for those values.
>
> RIGHT JOIN returns all rows from the right table and matching rows from the left table. If there is no matching row for the left table, NULLs will be returned for those values.
>
> FULL JOIN (or `FULL OUTER JOIN`) returns all records from both tables. It is the equivalent of using `UNION` on a `LEFT JOIN` and a `RIGHT JOIN`.

There are lots of applications for SQL JOINs, as well as some less common join types; see our SQL JOIN Cheat Sheet for more details.

9. What Does ORDER BY Do?

ORDER BY sorts records based on user-defined conditions, e.g. `ORDER BY Age` will return records sorted by age in ascending order. The optional `ASC` (ascending) and `DESC` (descending) keywords specify the sort order; sorts are in ascending order by default.

`ORDER BY` can be applied to one or more columns; if you use multiple columns, it will order them by the first column and then order any matching first-column values by the second column, etc.

10. What Is a NULL in SQL?

In SQL, NULLs represent values that are unknown or unavailable. A NULL is not the same as a blank space (a character) or a zero (a number); it's something that we don't yet know (e.g. an arrival date for an order that's just been placed) or that doesn't exist (like when someone doesn't have a middle name).

In database tables, some values cannot be NULL, like primary keys or mandatory fields.

Working with `NULL` can be quite tricky. You cannot use comparison operators (like =,<,>,or <>) to retrieve NULLs; instead, you have to use `IS NULL` and `IS NOT NULL`. You can find more details in How to Use Comparison Operators with NULLs in SQL.

11. What Does UNION Do? What's the Difference Between UNION and UNION ALL?

`UNION` is a set operator that merges the results of two queries into one table. (Note: the queries must be structurally compatible.) `UNION` does not include duplicate records; `UNION ALL` includes all records, even duplicates.

It may also be worthwhile to mention that UNION ALL requires less work from the server than `UNION`, which means the operation is generally faster.

12. What Do the INTERSECT and MINUS Clauses do?

INTERSECT and MINUS are also set operators; both work on two SELECT statements. `MINUS` subtracts the result of the first `SELECT` from the second `SELECT`. In other words, it returns records that are unique to the first `SELECT` result. `INTERSECT`, on the other hand, returns distinct rows that appear in both queries – i.e. what both results have in common. For more on SQL set operations, see this Introduction to Set Operations.

13. Describe the Role of the DISTINCT keyword.

`DISTINCT` is used with the `SELECT` statement to return only unique values. If the result set contains duplicates, then the `DISTINCT` statement can be used to return the unique, or distinct, values among duplicate records for selected columns.

You can read up further and see some examples in action in the article What Is the Role of Distinct in SQL.

14. What Is a Subquery in SQL?

A subquery (aka an inner or nested query) is a smaller query included in the body of a larger query (called the main or outer query). The results of the inner query are used in the outer query. They come in useful when you need a subset of the data to get the desired results, e.g. if you wanted to find which sales reps had above-average sales in their region, you'd first need a subquery to calculate the average sales for that region.

Here are some things to know about subquery syntax:

> The subquery goes inside
>
> In some database management systems (DBMSs), subqueries require an alias.
>
> If your subquery is in a `WHERE` or `HAVING` clause, its `SELECT` can only return the field being evaluated.

There are also a few different types of SQL subqueries, such as nested subqueries (a subquery within a subquery) scalar subqueries (which return exactly one value for a column), correlated subqueries (which depend on information in the outer query), and multi-row subqueries. You can brush up on those specifics in What are the Different Types of SQL Subqueries?

15. How Do You Modify Data in a Database Using SQL?

SQL has a subset called Data Manipulation Language, which include commands like `UPDATE`, `INSERT`, and `DELETE`. As the names of these

commands suggest, they manipulate the data inside the database by storing, modifying, deleting, and retrieving it. The use of these commands is relatively straightforward; you can read up on the subject at SQL INSERT, SQL UPDATE and SQL DELETE – Oh My!

If you want to practice SQL using hands-on exercises, try our SQL Practice track! Over 600 interactive SQL exercises.

Tips for a Successful SQL Interview

I hope this information is useful in your job interview prep. However, I must stress that nothing beats actual hands-on practice. If you want to dive deeper into SQL, check out our SQL Practice Set, which is designed to bring you up to speed on popular SQL concepts.

If you can reasonably do so, always try to link each question with a real-world scenario – that makes your answer resonate better with recruiters. This is another reason to practice SQL and put your knowledge to the test!

By being prepared and familiarizing yourself with these SQL interview questions and answers, you'll have a much better chance of impressing your interviewer and securing your next job. All the best!
