Top 5 SQL Interview Questions With Implementation

DATA ENGINEERING DATA WAREHOUSE DATABASE INTERMEDIATE INTERVIEW QUESTIONS SQL

Introduction

In today's world, technology has increased tremendously, and many people are using the internet. This results in the generation of so much data daily. This generated data is stored in the database and will maintain it. SQL is a structured query language used to read and write these databases. In simple words, SQL is used to communicate with databases. SQL allows you to perform any database-related task. It is accessible and economical for the majority of organizations. To ace an interview, learning SQL is very much helpful. In this article, we will learn some interesting topics of SQL which will help you in interviews.

Source: DigitalOcean

Learning Objectives

In this article, we will learn:

1. About Common Table Expressions and how we can implement them.
2. Different ways to replace null values with default values in SQL.
3. About auto increment, normalization, and denormalization.
4. Different rank functions and how they are different from each other.

This article was published as a part of the Data Science Blogathon.

Table of Contents

Q1. What is Common Table Expression in SQL?

A Common Table Expression (CTE) is a query's result set that lives temporarily within the execution scope of a statement like SELECT, INSERT, UPDATE, DELETE, or MERGE. The output of a CTE is not kept and only exists while the query is being executed. Making complex queries more readable and simple makes it easier for users to create and maintain them.

The following demonstrates the typical SQL Server syntax for a CTE:

```
WITH expression_name[(columns [,...])] AS (CTE_definition) SQL_statement;
```

- Use WITH clause to define a common table expression.
- In the columns, specify the names of all the columns that need to be retrieved in the CTE
- Define CTE after AS that retrieves a table with specified columns.
- Finally, write an SQL query using a common table expression.

Source: CoderEarth

Let's see an example to define a Common Table Expression as students_data with id, name, and roll_no columns. And then, a query to return the names of students that starts with the letter A among them.

```
WITH students_data[(id,name,roll_no)] AS ( SELECT id, name,roll_no FROM students ) SELECT name FROM students_data WHERE name LIKE 'A%';
```

Q2. How to Replace Null Values with Default Values in MYSQL?

Sometimes, while using MySQL, you don't want NULL values to be returned as NULL. But sometimes, you want NULL values to return a different default value. There are some ways in MYSQL to replace these null values with default values.

There are four ways to replace it-

1. Using IFNULL() function

2. Using

COALESCE() function

3. Combination of IF() function and IS NULL operator

4. Combination of CASE expression and IS NULL operator

Let's see them one by one.

1. Using IFNULL() function:

The IFNULL() function takes two expressions and returns the first arguments if the first expression is not null. The second parameter is returned if the first expression returns null.

Let's see the syntax.

```
IFNULL(expression, alternate_value) #Example SELECT IFNULL(Name,'N/A') FROM students
```

The above example returns names from table students. If the entry is not null, the name will be returned, and if the entry is null, the default N/A will be returned.

2. Using COALESCE() function:

The COALESCE() method returns the first non-null arguments from the given list of expressions. The function gives a null result if the expression is empty. Moreover, a specified default value can be used to replace null entries in a table.

Simply, it returns the first non-null argument in the given list of expressions. If there are no non-null values, then NULL is returned.

Let's see some examples to understand.

```
SELECT COALESCE('one', 'two', 'three') AS result #result #one SELECT COALESCE(NULL, 'one', 'two', 'three') AS result #result #one SELECT COALESCE(NULL, NULL, 'two', 'three') AS result #result #two SELECT COALESCE('A', NULL, 'B', NULL) AS result #result #A SELECT COALESCE(NULL, NULL, 'P', NULL, 'Q') AS result #result #P SELECT COALESCE(NULL, NULL, NULL, NULL, NULL) AS result #result #NULL
```

3. Combination of IF() function and IS NULL operator:

We can also use IF() function and IS NULL operator to replace null values with default values. It works like if the value is null, then replace it with a default value; else, return the original expression. Let's see how it works with some examples.

To replace null values with 'N/A' in the names column of a students_data table.

```
SELECT IF(names IS NULL, 'N/A', names ) AS result FROM students_data
```

4. Combination of CASE expression and IS NULL operator:

This is almost similar to the previous one. Here we use the CASE operator instead of the IF() function. So first, we will take cases where there is a null value, and then we will replace it with the given default value. Else the original expression will be returned. Let's take an example to understand in detail.

```
SELECT CASE WHEN names IS NULL THEN 'N/A' ELSE names END FROM students_data
```

This code is for the same previous example. To replace 'N/A' in the names column when there are null entries.

Q3. What is the SQL Syntax for Auto Increment?

When a new entry is entered into a database, auto-increment enables the automatic generation of a unique number. We use the AUTO_INCREMENT keyword for auto increment in SQL. By default, the increment value is one.

Syntax:

```
CREATE TABLE table_name ( column_name datatype AUTO_INCREMENT, );
```

For example,

```
CREATE TABLE students_data ( id INT AUTO_INCREMENT, name varchar, phone_number INT );
```

Q4. What are the Different Rank Functions in SQL?

There are four rank functions in SQL

1. RANK()

2. DENSE_RANK()

3.ROW_NUMBER()

4.NTILE()

Let’s see them one by one in detail.

1. RANK(): This function will return a number that will be applied to each row in the output partition. Each row receives a rank equal to one plus the rank of the row before it. The RANK function assigns the same rank number to two values that it discovers to be identical within the same partition. The following ranking number will also include duplicate numbers in addition to the preceding rank. As a result, this method does not always assign the ranking of rows in numerical order.

Let’s see the syntax

```
SELECT column_name RANK() OVER ( PARTITION BY expression ORDER BY expression) AS result FROM table_name;
```

2. DENSE_RANK(): This is almost similar to that of the rank function. Here also, each row receives rank, adding one to the previous rank. If two rows are identical, then they receive the same rank, and the next row directly receives plus one to the current rank. For example, if the 1st and 2nd rows are identical, then both receive rank 1, and the next third row receives rank 2 instead of rank 3, as in the case of using the RANK() function. That’s the difference.

Let’s see the syntax

```
SELECT column_name DENSE_RANK() OVER ( PARTITION BY expression ORDER BY expression) AS result FROM table_name;
```

3.ROW_NUMBER(): The row number function differs from the rank and dense rank functions. Starting from 1, this gives ranks adding 1 to the previous row. No matter if any two rows are identical.

let’s see the syntax

```
SELECT column_name ROW_NUMBER() OVER ( PARTITION BY expression ORDER BY expression) AS result FROM table_name;
```

4. NTILE(): The NTILE() function is the one you want to use when you want to distribute groups of rows over a partition evenly. You must tell this ranking function how many groups you want the rows to be equally divided into. According to the specified requirement, each row group receives its rank.

let's see the syntax

```
SELECT column_name NTILE(N) OVER ( PARTITION BY expression ORDER BY expression) AS result FROM table_name;
```

Q5. Explain Normalization and Denormalization in SQL.

Normalization removes redundancy from the database, which means it is split across multiple tables instead of just one table. and non-redundant, consistent data is added. An improperly constructed database table is inconsistent and could cause problems when executing operations. Hence database normalization is an important step. An unnormalized table is transformed into a normalized table through this process.

Source: HiTechNectar

Denormalization is used to aggregate data from several tables into one to be easily queried. Redundancy is added using it. In contradiction to normalization, denormalization reduces the number of tables. Denormalization is used when joins are expensive, and table queries are run frequently. Wastage of memory is the main drawback of denormalization.

Conclusion

Social media app users frequently share photographs and posts, which involves databases that can update information and simultaneously display content to millions of users. There are many tables in the database, and each table contains rows of data. SQL helps to maintain these databases. We learned some important topics in SQL in this article.

- For both small and large organizations, SQL is ideal, and also SQL makes it simple to handle permissions.
- Currently, businesses are looking for someone with SQL knowledge. So it helps to pass interviews by learning SQL.
- Compared to other programming languages, SQL is simpler to learn.
- Many big companies, including Microsoft, Dell, Accenture, and stack Overflows, use SQL to maintain their databases.

Hope you found this article useful. Connect with me on LinkedIn.

Article Url - https://www.analyticsvidhya.com/blog/2023/03/top-5-sql-interview-questions-with-implementation/

Amrutha K