Data Science World

all about data science...

30 Most Common SQL Interview Questions

DUKUL SINGLA / 20 FEBRUARY 2024 / SQL

1. **Order of execution of SQL**:
    - SQL statements are executed in a specific order to ensure proper processing of the query and to produce the desired results. The typical order of execution is as follows:
        1. **FROM**: Specifies the tables involved in the query.
        2. **WHERE**: Filters rows based on specified conditions.
        3. **GROUP BY**: Groups rows with the same values into summary rows.
        4. **HAVING**: Filters groups based on specified conditions.
        5. **SELECT**: Retrieves the columns specified in the query.
        6. **ORDER BY**: Sorts the result set based on specified columns.
2. **Difference between WHERE and HAVING**:

- **WHERE** clause is used to filter rows before grouping takes place, based on individual row values.
- **HAVING** clause is used to filter groups after the grouping has occurred, based on group-level summaries (aggregates).

3. **Use of GROUP BY**:
    - The **GROUP BY** clause is used to group rows that have the same values into summary rows. It is typically used in conjunction with aggregate functions like SUM, COUNT, AVG, etc. This allows for the analysis of data at a higher level, such as finding totals or averages for groups of data.
4. **Types of joins in SQL**:
    - **Inner Join**: Returns rows when there is at least one match in both tables.
    - **Left Join (or Left Outer Join)**: Returns all rows from the left table, and the matched rows from the right table.
    - **Right Join (or Right Outer Join)**: Returns all rows from the right table, and the matched rows from the left table.
    - **Full Join (or Full Outer Join)**: Returns all rows when there is a match in one of the tables.
    - **Cross Join**: Returns the Cartesian product of the sets of rows from the joined tables.
    - **Self Join**: Joins a table to itself.
5. **Triggers in SQL**:
    - **Triggers** are special types of stored procedures that are automatically executed or fired when certain events occur in the database. These events can include INSERT, UPDATE, and DELETE operations on a table. Triggers are useful for enforcing business rules, auditing changes, and maintaining data integrity.
6. **Stored procedure in SQL**:
    - A **stored procedure** is a precompiled collection of SQL statements and procedural logic that is stored in the database. It can be executed multiple times without needing to recompile the SQL code each time. Stored procedures can accept input parameters and return output parameters, making them reusable and efficient.
7. **Types of window functions**:
    - **Rank**: Assigns a unique rank to each row within a result set, with no gaps in the ranking values.
    - **Row_Number**: Assigns a unique sequential integer to each row within a result set.
    - **Dense_Rank**: Assigns a unique rank to each row within a result set, with no gaps in the ranking values, but may have duplicate ranking values.
    - **Lead**: Accesses data from subsequent rows in the result set.

- **Lag**: Accesses data from preceding rows in the result set.

8. **Difference between DELETE and TRUNCATE**:
    - **DELETE** is a DML (Data Manipulation Language) command used to remove one or more rows from a table based on specified conditions.
    - **TRUNCATE** is a DDL (Data Definition Language) command used to remove all rows from a table. It deallocates the data pages used by the table and resets any identity columns to their seed value.
9. **Difference between DML, DDL, and DCL**:
    - **DML (Data Manipulation Language)**: Used for managing data within the database, including operations like INSERT, UPDATE, DELETE.
    - **DDL (Data Definition Language)**: Used for defining the structure of the database, including operations like CREATE, ALTER, DROP.
    - **DCL (Data Control Language)**: Used for controlling access to data within the database, including operations like GRANT, REVOKE.
10. **Aggregate functions and their use**:
    - **Aggregate functions** perform calculations on a set of values and return a single value. They are often used with the GROUP BY clause to provide summarized data. Examples include:
        - **SUM**: Calculates the sum of values.
        - **AVG**: Calculates the average of values.
        - **COUNT**: Counts the number of rows.
        - **MIN**: Returns the minimum value.
        - **MAX**: Returns the maximum value.
11. **Performance comparison between CTE and subquery**:
    - The performance of CTEs (Common Table Expressions) and subqueries can vary depending on the specific query, database system, and indexing. In general, CTEs are often easier to read and maintain, while subqueries may sometimes offer better performance, particularly for simple queries.
12. **Constraints and types**:
    - **Constraints** are rules that enforce data integrity in a database. They can be applied to columns or tables. Types of constraints include:
        - **NOT NULL**: Ensures that a column cannot contain NULL values.
        - **UNIQUE**: Ensures that all values in a column are unique.
        - **PRIMARY KEY**: A combination of NOT NULL and UNIQUE constraints. It uniquely identifies each record in a table.
        - **FOREIGN KEY**: Enforces referential integrity by requiring that values in a column match values in another table's primary key.

    - **CHECK**: Ensures that all values in a column satisfy a specific condition.

13. **Types of Keys**:
    - **Primary Key**: A unique identifier for each record in a table.
    - **Foreign Key**: A field in one table that refers to the primary key in another table.
    - **Alternate Key**: A candidate key that is not selected as the primary key.
    - **Composite Key**: A key that consists of two or more columns.
    - **Super Key**: A set of attributes that uniquely identifies each tuple in a relation.
14. **Different types of Operators**:
    - **Arithmetic Operators**: Perform mathematical operations (+, -, *, /).
    - **Comparison Operators**: Compare values (=, <>, >, <, >=, <=).
    - **Logical Operators**: Combine conditions (AND, OR, NOT).
15. **Difference between Group By and Where**:
    - **GROUP BY** is used to group rows that have the same values into summary rows, typically used with aggregate functions.
    - **WHERE** is used to filter rows based on specified conditions before grouping takes place.
16. **Views**:
    - **Views** are virtual tables derived from one or more tables. They can be used to simplify complex queries, provide a level of security by restricting access to certain columns or rows, and abstract the underlying data structure.
17. **Difference between VARCHAR and NVARCHAR**:
    - **VARCHAR** stores variable-length character data with a maximum size specified in bytes.
    - **NVARCHAR** stores variable-length Unicode character data with a maximum size specified in bytes.
18. **Difference between CHAR and NCHAR**:
    - **CHAR** stores fixed-length character data with a specified length in bytes.
    - **NCHAR** stores fixed-length Unicode character data with a specified length in bytes.
19. **Index and their types**:
    - **An index** is a database object used to speed up the retrieval of rows from a table by using a pointer.
    - Types of indexes include:
        - **Clustered Index**: Defines the physical order of data in a table.
        - **Non-Clustered Index**: A separate structure that contains a sorted list of references to the table's rows.

        - **Unique Index**: Ensures that no two rows have duplicate values in the indexed columns.
        - **Composite Index**: Index that includes multiple columns.
20. **Types of relationships in SQL**:
    - **One-to-One**: Each record in the first table is related to only one record in the second table, and vice versa.
    - **One-to-Many**: Each record in the first table can be related to one or more records in the second table, but each record in the second table can only be related to one record in the first table.
    - **Many-to-Many**: Many records in the first table can be related to many records in the second table.
21. **Difference between UNION and UNION ALL**:
    - **UNION**: Combines the result sets of two or more SELECT statements, removing duplicate rows.
    - **UNION ALL**: Combines the result sets of two or more SELECT statements, including all rows, even if they are duplicates.
22. **Types of clauses in SQL**:
    - There are several types of clauses in SQL, including:
        - **SELECT**: Retrieves data from one or more tables.
        - **FROM**: Specifies the tables from which to retrieve data.
        - **WHERE**: Filters rows based on specified conditions.
        - **GROUP BY**: Groups rows based on specified columns.
        - **HAVING**: Filters groups based on specified conditions.
        - **ORDER BY**: Sorts the result set based on specified columns.
23. **Difference between Primary Key and Secondary Key**:
    - **Primary Key**: A column or a set of columns that uniquely identifies each row in a table.
    - **Secondary Key**: Any column or set of columns that is not the primary key but still provides an efficient way to search for data.
24. **Find the second highest salary of an employee**:
    - SELECT MAX(salary) AS SecondHighestSalary FROM employees WHERE salary < (SELECT MAX(salary) FROM employees);
25. **Retention query in SQL**:
    - A retention query typically involves deleting old or obsolete records from a table. For example

    - SQL Code DELETE FROM tablename WHERE datecolumn < ‘2022-01-01’;

26. **Year-on-year growth in SQL**:
    - Year-on-year growth is calculated by comparing data from the current year to the previous year. For example:

    - SQL code `SELECT (SUM(sales_current_year) - SUM(sales_previous_year)) / SUM(sales_previous_year) AS YoYGrowth FROM sales_data WHERE year IN (current_year, previous_year);`
27. **Query for cumulative sum in SQL**:
    - A cumulative sum query calculates the running total of a numerical column. For example:

    - SQL code `SELECT column1, column2, SUM(column3) OVER (ORDER BY column1) AS CumulativeSum FROM tablename;`
28. **Difference between Function and Stored Procedure**:
    - **Function**: Returns a single value and cannot perform data manipulation operations. It is called in the SELECT statement.
    - **Stored Procedure**: Can perform data manipulation operations and does not necessarily return a value. It can be called independently or within other SQL statements.
29. **Use of variables in views**:
    - Views in SQL cannot directly use variables. Variables are typically used within stored procedures, functions, or dynamic SQL queries.
30. **Limitations of views**:
    - **Performance**: Views can sometimes impact performance, especially if they involve complex joins or calculations.
    - **Limited Updateability**: Some views are not directly updatable if they involve multiple tables or complex expressions.
    - **Complexity**: Views may become difficult to manage and maintain if they are overly complex or nested.
    - **Dependency**: Changes to underlying tables may affect views, potentially leading to unintended consequences.
    - **Security**: Views may not always provide sufficient security, especially if they expose sensitive data without proper access controls.

Dukul Singla

ARTICLES: 3

PREVIOUS POST

Introduction to SQL (Structured Query Language)

Search

Latest Posts

30 Most Common SQL Interview Questions

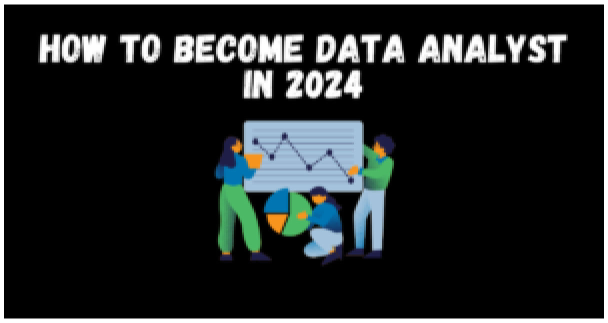
by Dukul Singla

20 February 2024

Introduction to SQL (Structured Query Language)

by Dukul Singla

25 April 2023



How to Become Data Analyst in 2024

by Dukul Singla

16 April 2023

Related Posts

Introduction to SQL (Structured Query Language)

25 April 2023 / 1 Comment

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Add Comment *

Save my name, email, and website in this browser for the next time I comment.

Post Comment

HOME ABOUT CONTACT