# SQL

**1. What is SQL?**
It stands for Structured Query Language. A programming language used for interaction with **Relational Database Management Systems** (***RDBMS***). This includes *fetching*, *updating*, *inserting*, and *removing* data from tables.

**2. What are SQL dialects? Give some examples.**
The various versions of SQL, both free and paid, are also called SQL dialects. All the flavors of SQL have a very similar syntax and vary insignificantly only in additional functionality. Some examples are **Microsoft SQL Server**, **PostgreSQL**, **MySQL**, **SQLite**, **T-SQL**, **Oracle**, and **MongoDB**.

**3. What are the main applications of SQL?**
Using SQL, we can:

- ***create***, ***delete***, and ***update*** tables in a database
- ***access***, ***manipulate***, and ***modify*** data in a table
- ***retrieve*** and ***summarize*** the necessary information from a table or several tables
- ***add*** or ***remove*** certain ***rows*** or ***columns*** from a table

All in all, SQL allows querying a database in multiple ways. In addition, SQL easily integrates with other programming languages, such as Python or R, so we can use their combined power.

1. **Database Management** : SQL manages relational databases, creating, modifying, and deleting databases and their objects.
2. **Data Querying** : SQL retrieves specific information from databases using queries, filtering, sorting, and joining data.
3. **Data Manipulation** : SQL alters data within databases, including inserting, updating, and deleting records.
4. **Data Definition** : SQL defines the structure of databases and their objects, creating, altering, and dropping tables, views, and indexes.
5. **Data Control** : SQL controls access to databases, granting and revoking privileges to users and roles, ensuring security and integrity.
6. **Data Administration** : SQL handles database administration tasks, such as backup, recovery, performance tuning, and monitoring.
7. **Reporting and Analysis** : SQL generates reports and performs analysis on data, aggregating, filtering, and summarizing information.
8. **Web Development** : SQL integrates with web development for backend data storage and retrieval, handling dynamic content and user input.

**4. What is an SQL statement? Give some examples.**

Also known as an SQL command. It's a string of characters interpreted by the SQL engine as a legal command and executed accordingly. Some examples of SQL statements are ***SELECT***, ***CREATE***, ***DELETE***, ***DROP***, ***REVOKE***, and so on.

**5. What types of SQL commands (or SQL subsets) do you know?**
**Data Definition Language (DDL)** – to define and modify the structure of a database.
**Data Manipulation Language (DML)** – to access, manipulate, and modify data in a database.
**Data Control Language (DCL)** – to control user access to the data in the database and give or revoke privileges to a specific user or a group of users.
**Transaction Control Language (TCL)** – to control transactions in a database.
**Data Query Language (DQL)** – to perform queries on the data in a database to retrieve the necessary information from it.

**6. Does SQL support programming language features?**
It is true that SQL is a language, but it does not support programming as it is not a programming language, it is a command language. We do not have conditional statements in SQL like for loops or if..else, we only have commands which we can use to *query*, *update*, *delete*, etc. data in the database. SQL allows us to manipulate data in a database.

**7. What is the difference between CHAR and VARCHAR2 datatypes in SQL?**
Both VARCHAR2 and CHAR are used for storing text, but they work differently:

- **VARCHAR2**: It's for text that can vary in length. For example, if you specify VARCHAR2(5), you can store any text up to 5 characters long.
- **CHAR**: It's for text with a fixed length. If you specify CHAR(5), you can only store text that is exactly 5 characters long. Any shorter text will be padded with spaces.

In simple terms, VARCHAR2 lets you store flexible-length text, while CHAR requires fixed-length text.

**8. What do you mean by data definition language?**
Data Definition Language or DDL allows the execution of queries like CREATE, DROP, and ALTER. That is those queries that define the data.

**9. What do you mean by data manipulation language?**
**Data Manipulation Language** or DML is used to access or manipulate data in the database.
It allows us to perform the below-listed functions:

- Insert data or rows in a database

- Delete data from the database
- Retrieve or fetch data
- Update data in a database

**10. What is the view in SQL?**

**Views** in SQL are a kind of virtual table. A view also has rows and columns as they are on a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain conditions.

The CREATE VIEW statement of SQL is used for creating views.

Basic Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2
FROM table_name
WHERE condition;
```

view_name: Name for the View

table_name: Name of the table

condition: Condition to select rows

**11. What do you mean by foreign key?**

A **Foreign Key** is like a connection between two tables in a database. It's a field in one table that points to the primary key in another table, allowing each row in one table to be linked to a unique row in another table.

This usually creates a kind of link between the two tables.

**12. What are Table and Field?**

**Table:** A table has a combination of rows and columns. Rows are called ***records*** and columns are called ***fields***. In MS SQL Server, the tables are being designated within the database and schema names.

**Field:** In DBMS, a database field can be defined as – a single piece of information from a record.

**13. What is the primary key?**

A **Primary Key** is one of the candidate keys. One of the candidate keys is selected as the most important and becomes the primary key. There cannot be more than one primary key in a table.

**14. What is a Default constraint?**

The **DEFAULT** constraint is used to fill a column with default and fixed values. The value will be added to all new records when no other value is provided.

**15. What is Normalization?**

**Normalization** in SQL organizes data to prevent redundancy and errors. By analyzing relationships and keys, it splits large tables into smaller ones for efficiency. This minimizes duplication and avoids issues when adding, deleting, or updating data. Tables not meeting standards are divided into smaller ones to ensure data integrity.

**16. What is Denormalization?**

**Denormalization** is a database optimization technique in which we add redundant data to one or more tables.Denormalization is a trick to make databases faster by adding duplicate info to tables. This can help us avoid costly joins in a relational database. It's not skipping normalization but optimizing after it

In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data. We may strive to have only one copy of each piece of data in the database.

**17. What is a query?**

An SQL query is used to retrieve the required data from the database. However, there may be multiple SQL queries that yield the same results but with different levels of efficiency. An inefficient query can drain the database resources, reduce the database speed or result in a loss of service for other users. So it is very important to optimize the query to obtain the best database performance.

**18. What is a subquery?**

In SQL a Subquery can be simply defined as a query within another query. In other words, we can say that a Subquery is a query that is embedded in the WHERE clause of another SQL query.

**19. What are the different operators available in SQL?**

There are three operators available in SQL namely:

1. **Arithmetic Operators** - Addition(+), Subtraction(-), Multiplication(*), Division(/)
2. **Logical Operators** - And, Or, Not
3. **Comparison Operators** - Equal to(=) , Not Equal to(!= or <>), greater than(>), Less than(<), greater than equal to(>=), less than equal to(<=)

**20. What is a Constraint?**

**Constraints** are the rules that we can apply to the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

In SQL, constraints are rules that control the data that can be stored in a column. They can be defined at the column or table level. ***Column Level Constraints*** apply to a single column, while ***Table Level Constraints*** apply to the entire table.

**21. What is Data Integrity?**

Data integrity refers to the *accuracy*, *consistency*, and *reliability* of data stored in a database. It ensures that data is complete, valid, and unchanged throughout its lifecycle, from creation to deletion. Data integrity is maintained through the enforcement of constraints, validation rules, and error checking mechanisms. It guarantees that data remains trustworthy and meaningful for its intended purpose.

**22. What is Auto Increment?**

Sometimes, while creating a table, we do not have a unique identifier within the table, hence we face difficulty in choosing Primary Key. So as to resolve such an issue, we've to manually provide unique keys to every record, but this is often also a tedious task. So we can use the  Auto-Increment feature that automatically generates a numerical Primary key value for every new record inserted. The Auto Increment feature is supported by all the Databases.

**23. What is MySQL collation?**

A *MySQL collation* is a well-defined set of rules which are used to compare characters of a particular character set by using their corresponding encoding. Each character set in MySQL might have more than one collation, and has, at least, one default collation. Two character sets cannot have the same collation.

MySQL collation refers to the set of rules used to compare and sort characters in a database. It determines how characters are sorted and compared based on their respective character sets and encoding. Collation settings influence various operations such as sorting data, comparing strings, and performing searches within the database. MySQL supports a wide range of collations to accommodate different languages, character sets, and sorting requirements.

**24. What are user-defined functions?**

We can use User-defined functions in PL/SQL or Java to provide functionality that is not available in SQL or SQL built-in functions. SQL functions and User-defined functions can appear anywhere, that is, wherever an expression occurs.

A user-defined function (UDF) in SQL is a custom function created by a user to perform specific operations or calculations within a database. Unlike built-in functions provided by the database management system (DBMS), UDFs allow users to extend the functionality of SQL by defining their own functions tailored to their specific requirements. UDFs can accept parameters, perform calculations, and return results, making them useful for complex data manipulations, calculations, and business logic implementation within SQL queries.

**25. What are all types of user-defined functions?**

User-Defined Functions allow people to define their own T-SQL functions that can accept 0 or more parameters and return a single scalar data value or a table data type. Different Kinds of User-Defined Functions created are:

**1. Scalar User-Defined Function** A Scalar user-defined function returns one of the scalar data types. Text, image, and timestamp data types are not supported. These are the type of user-defined functions that most developers are used to in other programming languages. You pass in 0 to many parameters and you get a return value.

**2. Inline Table-Value User-Defined Function** An Inline Table-Value user-defined function returns a table data type and is an exceptional alternative to a view as the user-defined function can pass parameters into a T-SQL select command and, in essence, provide us with a parameterized, non-updatable view of the underlying tables.

**3. Multi-statement Table-Value User-Defined Function** A Multi-Statement Table-Value user-defined function returns a table and is also an exceptional alternative to a view, as the function can support multiple T-SQL statements to build the final result where the view is limited to a single SELECT statement. Also, the ability to pass parameters into a TSQL select command or a group of them gives us the capability to, in essence, create a parameterized, non-updatable view of the data in the underlying tables. Within the create function command you must define the table structure that is being returned. After creating this type of user-defined function, it can be used in the FROM clause of a T-SQL command, unlike the behavior found when using a stored procedure which can also return record sets.

**26. What is a  stored procedure?**

A stored procedure is a reusable set of SQL statements stored in the database and given a name. It allows you to group and execute multiple SQL queries or commands as a single unit. Stored procedures help improve performance, enhance security, and simplify database management by reducing the need to send multiple queries from the application to the database server.

**27. What are aggregate and scalar functions?**

For doing operations on data SQL has many built-in functions, they are categorized into two categories:

**Aggregate Functions** perform calculations on a set of values and return a single result. Examples include functions like ***SUM***, ***AVG***, ***COUNT***, ***MAX***, and ***MIN***. These functions typically operate on multiple rows of data and are often used with the GROUP BY clause to group data before performing calculations.

**Scalar Functions**, on the other hand, operate on a single value and return a single result. They are applied to each row individually and can be used within SQL

statements wherever an expression is allowed. Examples of scalar functions include functions like ***UPPER***, ***LOWER***, ***CONCAT***, ***ROUND***, and ***DATE*** functions like ***CURRENT_DATE*** or ***NOW***().

**28. What is an ALIAS command?**

Aliases are the temporary names given to a table or column for the purpose of a particular SQL query. It is used when the name of a column or table is used other than its original name, but the modified name is only temporary.

- Aliases are created to make table or column names more readable.
- The renaming is just a temporary change and the table name does not change in the original database.
- Aliases are useful when table or column names are big or not very readable.

These are preferred when there is more than one table involved in a query.

**29. What are Union, Minus/Except and Interact commands?**

In SQL, UNION, MINUS (or EXCEPT), and INTERSECT are set operations used to combine or compare the results of multiple SELECT queries:

1. **UNION** : Combines the results of two or more SELECT queries into a single result set. It removes duplicate rows by default.
2. **MINUS** (or **EXCEPT**) : Returns the rows that are present in the result of the first SELECT query but not in the result of the second SELECT query.
3. **INTERSECT** : Returns the rows that are common to the result of both SELECT queries.

These commands allow for flexible querying and manipulation of data by combining or comparing results from different queries.

**30. What is T-SQL?**

T-SQL is an abbreviation for **Transact Structure Query Language**. T-SQL (Transact-SQL) is Microsoft's extension of SQL (Structured Query Language), used specifically with Microsoft SQL Server and Azure SQL Database. It includes additional features beyond standard SQL, such as procedural programming constructs like *variables*, *control-of-flow statements*, *error handling*, and *stored procedures*.

T-SQL is used for querying and managing relational databases in Microsoft SQL Server environments. It supports a wide range of operations including data manipulation (SELECT, INSERT, UPDATE, DELETE), data definition (CREATE, ALTER, DROP), transaction control, and administration tasks.

T-SQL is essential for developing applications, writing stored procedures, triggers, and user-defined functions, and performing various database-related tasks within the Microsoft SQL Server ecosystem.

### 31. What is ETL in SQL?

ETL is a process in Data Warehousing and stands for **Extract**, **Transform**, **Load**. In SQL, ETL refers to the process of extracting data from various sources, transforming it into a suitable format, and loading it into a target database or data warehouse for analysis or reporting purposes.

- **Extract** : Involves extracting data from different sources such as databases, files, or external systems.
- **Transform** : Includes transforming the extracted data to meet the requirements of the target system or database. This may involve *cleaning*, *filtering*, *aggregating*, and *converting* data.
- **Load** : Involves loading the transformed data into the target database or data warehouse using SQL commands or ETL tools.

ETL processes are essential for integrating data from multiple sources, ensuring data quality, and making it accessible for analysis and decision-making.

### 32. How to copy tables in SQL?

Sometimes, in SQL, we need to create an exact copy of an already defined (or created) table. MySQL enables you to perform this operation. Because we may need such duplicate tables for testing the data without having any impact on the original table and the data stored in it.

1. **CREATE TABLE Contact List(Clone_1) LIKE Original_table;**
2. **SELECT * INTO New_table_name FROM old_table_name;**

### 33. What is SQL injection?

SQL injection is a technique used to exploit user data through web page inputs by injecting SQL commands as statements. Basically, these statements can be used to manipulate the application's web server by malicious users.

- SQL injection is a code injection technique that might destroy your database.
- SQL injection is one of the most common web hacking techniques.
- SQL injection is the placement of malicious code in SQL statements, via web page input.

For example, an attacker might insert SQL code into a login form's input field to bypass authentication, retrieve sensitive information from the database, or even delete or modify data. SQL injection attacks can lead to data breaches, unauthorized access, data loss, and other security risks.

Preventing SQL injection involves using parameterized queries, input validation, and sanitization techniques to ensure that user input is properly handled and does not contain malicious SQL code. Additionally, keeping software up to date and following secure coding practices can help mitigate the risk of SQL injection vulnerabilities.

**34. Can we disable a trigger? If yes, how?**

Yes, we can disable a trigger in PL/SQL. If consider temporarily disabling a trigger and one of the following conditions is true:

- An object that the trigger references is not available.
- We must perform a large data load and want it to proceed quickly without firing triggers.
- We are loading data into the table to which the trigger applies.
- We disable a trigger using the ALTER TRIGGER statement with the DISABLE option.
- We can disable all triggers associated with a table at the same time using the ALTER TABLE statement with the DISABLE ALL TRIGGERS option.

To disable a trigger, you can use the ALTER TABLE statement with the DISABLE TRIGGER option.

**ALTER TABLE table_name DISABLE TRIGGER trigger_name**

To re-enable the trigger, you can use the ENABLE TRIGGER option:

**ALTER TABLE table_name ENABLE TRIGGER trigger_name**

**35. What are the differences between SQL and PL/SQL?**

1. **Purpose**:
    - SQL : Used for querying, manipulating, and managing relational databases.
    - PL/SQL: Extends SQL with procedural programming capabilities for building database applications.
2. **Focus** :
    - SQL: Primarily focuses on data manipulation tasks like SELECT, INSERT, UPDATE, DELETE.
    - PL/SQL: Allows writing procedural logic such as loops, conditional statements, and error handling.
3. **Usage** :
    - SQL : Used to retrieve data, add or modify data, and manage database objects like tables, views, indexes.
    - PL/SQL : Used to create stored procedures, functions, triggers, and packages for implementing complex business logic and application development within the database.
4. **Flexibility** :
    - SQL: Limited in its ability to handle procedural logic and control flow.
    - PL/SQL: Provides more flexibility and control over database operations with support for procedural programming constructs.
5. **Integration** :
    - SQL: We can embed SQL in PL/SQL
    - PL/SQL: We can not embed PL/SQL in SQL
6. **Nature** :

- SQL : SQL is very declarative in nature.
- PL/SQL : PL/SQL has a procedural nature.

**36. What is the difference between BETWEEN and IN operators in SQL?**

The BETWEEN and IN operators in SQL are used for different purposes:

1. **BETWEEN Operator:**
    - The BETWEEN operator is used to specify a range of values.
    - It checks if a value lies within a specified range, inclusive of both endpoints.
    - Syntax: `value BETWEEN low AND high`.
    - Example: SELECT * FROM products WHERE price BETWEEN 10 AND 50
2. **IN Operator :**
    - The IN operator is used to specify a list of values.
    - It checks if a value matches any value in a specified list.
    - Syntax: `value IN (value1, value2, ...)`.
    - Example : SELECT * FROM customers WHERE state IN ('NY', 'CA', 'TX');

In summary, BETWEEN is used to check if a value falls within a range, while IN is used to check if a value matches any value in a list.

**37. Write an SQL query to find the names of employees starting with 'A'.**

**SELECT * FROM Employees WHERE EmpName like 'A%' ;**

**38. What is the difference between primary key and unique constraints?**

The primary key cannot have NULL values, the unique constraints can have NULL values. There is only one primary key in a table, but there can be multiple unique constraints. The primary key creates the clustered index automatically but the unique key does not.

**39. What is a join in SQL? What are the types of joins?**

An SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are:

**INNER JOIN** : The INNER JOIN keyword selects all rows from both tables as long as the condition is satisfied. This keyword will create the result set by combining all rows from both the tables where the condition satisfies i.e. the value of the common field will be the same.

**LEFT JOIN** : This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result set will be null. LEFT JOIN is also known as *LEFT OUTER JOIN.*

**RIGHT JOIN** : RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result set will contain null. RIGHT JOIN is also known as *RIGHT OUTER JOIN*.
**FULL JOIN** : FULL JOIN creates the result set by combining the results of both LEFT JOIN and RIGHT JOIN. The result set will contain all the rows from both tables. For the rows for which there is no matching, the result set will contain NULL values.
**CROSS JOIN** : Returns the Cartesian product of rows from both tables (i.e., all possible combinations of rows).

**40. What is an index?**
In SQL, an index is like a roadmap for the database engine, helping it quickly locate specific rows in a table based on the values in certain columns. Just as an index in a book allows you to find information faster, a database index stores the values of selected columns along with pointers to the corresponding rows. This speeds up data retrieval operations like searching, sorting, and joining tables, making queries run more efficiently.
Creating an index on columns frequently used in search criteria or join conditions can significantly improve query performance. However, indexes also come with overhead, as they consume storage space and require maintenance. Therefore, it's important to carefully consider when and where to use indexes to balance the benefits of faster queries with the costs of additional storage and upkeep.

**41. What is the On Delete cascade constraint?**
An '**ON DELETE CASCADE**' constraint is used in MySQL to delete the rows from the child table automatically when the rows from the parent table are deleted.

**42. Explain  WITH clause in SQL?**
The **WITH** clause provides a way of defining a temporary relationship whose definition is available only to the query in which the with clause occurs. SQL applies predicates in the **WITH** clause after groups have been formed, so aggregate functions may be used.

**43. What are all the different attributes of indexes?**
The indexing has various attributes:
**Access Types** : This refers to the type of access such as value-based search, range access, etc.
**Access Time** : It refers to the time needed to find a particular data element or set of elements.
**Insertion Time** : It refers to the time taken to find the appropriate space and insert new data.

**Deletion Time** : Time is taken to find an item and delete it as well as update the index structure.
**Space Overhead** : It refers to the additional space required by the index.

### 44. What is a Cursor?

The cursor is a **Temporary Memory** or **Temporary Work Station**. It is Allocated by Database Server at the Time of Performing DML operations on the Table by the User. A cursor in SQL is like a pointer used to iterate over the rows of a result set returned by a query. It allows you to process individual rows one at a time, enabling more granular control over data manipulation operations. Cursors are commonly used in stored procedures, triggers, and other database programming scenarios where row-level processing is needed.

### 45. Write down various types of relationships in SQL?

There are various relationships, namely:

1. **One-to-One (1:1)** : Each record in one table is related to exactly one record in another table.
2. **One-to-Many (1:M)**: Each record in one table can be related to multiple records in another table.
3. **Many-to-One (M:1)** : Multiple records in one table can be related to exactly one record in another table.
4. **Many-to-Many (M:M)** : Multiple records in one table can be related to multiple records in another table.

### 46. What is a trigger?

The trigger is a statement that a system executes automatically a set of actions in response to certain database events, such as INSERT, UPDATE, DELETE operations on a table when there is any modification to the database. In a trigger, we first specify when the trigger is to be executed and then the action to be performed when the trigger executes. It enables you to enforce business rules, maintain data integrity, and automate tasks within the database. Triggers are used to specify certain integrity constraints and referential constraints that cannot be specified using the constraint mechanism of SQL.

### 47. What is the difference between SQL DELETE and SQL TRUNCATE commands?

1. **Functionality :**
    - DELETE: Removes specific rows based on conditions or all rows.
    - TRUNCATE: Removes all rows, resetting the table.
2. **Speed :**

- DELETE: Slower due to individual row deletions and transaction logging.
- TRUNCATE: Faster as it deallocates data pages directly.

3. **Rollback :**
    - DELETE: Allows rollback of individual deletions within a transaction.
    - TRUNCATE: Cannot be rolled back, making data loss permanent.
4. **Trigger :**
    - DELETE: Triggers can be fired for each row deleted.
    - TRUNCATE: Triggers are not fired, enhancing speed but reducing flexibility.
5. **Indexed views:**
    - DELETE : The delete can be used with indexed views.
    - TRUNCATE : Truncate cannot be used with indexed views.

**48. What is the difference between Cluster and Non-Cluster Index?**

Here's a simplified version of each point:

1. Clustered index: Faster.
   Non-clustered index: Slower.
2. Clustered index: Requires less memory.
   Non-clustered index: Requires more memory.
3. Clustered index: Index is the main data.
   Non-clustered index: Index is a copy of data.
4. Clustered index: Only one per table.
   Non-clustered index: Multiple per table.
5. Clustered index: Stores data on disk.
   Non-clustered index: Doesn't store data on disk inherently.
6. Clustered index: Stores pointers to blocks.
   Non-clustered index: Stores values and pointers to rows.
7. Clustered index: Leaf nodes contain actual data.
   Non-clustered index: Leaf nodes contain included columns.
8. Clustered index: Clustered key orders table data.
   Non-clustered index: Index key orders index data.
9. Clustered index: Reorders table records.
   Non-clustered index: Doesn't reorder table records.

**49. What is a Live Lock?**

A live lock in concurrent programming involves processes or threads continually changing states in response to each other's actions, but none make progress. Unlike deadlocks, where processes are blocked, in a live lock, they remain active yet ineffective. It occurs when processes try to avoid conflicts but end up in a cycle of unproductive actions, resulting in perpetual activity without accomplishing their tasks.

**50. What is Case WHEN in SQL?**

In SQL, the CASE WHEN statement is a conditional expression used to perform conditional logic within queries. It allows you to evaluate multiple conditions and return different values based on those conditions. The syntax typically follows this format:

```
CASE
   WHEN condition1 THEN result1
   WHEN condition2 THEN result2
   ...
   ELSE default_result
END
```

Each WHEN clause specifies a condition to be evaluated, and if that condition is true, the corresponding result is returned. If none of the conditions are true, the ELSE clause provides a default result. The CASE WHEN statement is commonly used in SELECT queries, UPDATE statements, and other SQL operations to manipulate data based on specified conditions.

**51. Name different types of case manipulation functions available in SQL.**

Different types of case manipulation functions available in SQL include:

1. **UPPER** Converts characters to uppercase.
2. **LOWER** Converts characters to lowercase.
3. **INITCAP** Capitalizes the first letter of each word.
4. **UCASE** Synonym for UPPER.
5. **LCASE** Synonym for LOWER.

These functions are useful for standardizing text data and performing case-insensitive comparisons in SQL queries.

**52. What are local and global variables and their differences?**

**Global Variable:** Global variables are variables that are defined outside of functions. These variables have global scope, so they can be used by any function without passing them to the function as parameters.

**Local Variable:** Local variables are variables that are defined within functions. They have local scope, which means that they can only be used within the functions that define them.

The main difference between local and global variables is their scope: local variables are limited to the block in which they are declared, while global variables are accessible throughout the entire program.

here's a simple example:

```
# Global variable
global_var = 10

def example_function():
   # Local variable
```

```
    local_var = 20
    print("Local variable:", local_var)
    print("Global variable:", global_var)

example_function()
```

In this example, *'**global_var**'* is a global variable accessible from within the '***example_function'***, while ***'local_var'*** is a local variable defined within the function and can only be accessed within it.

**53. Name the function which is used to remove spaces at the end of a string?**

In SQL the spaces at the end of the string are removed by a **trim function.**

**54. What is the difference between TRUNCATE and DROP statements?**

1. The DROP command removes both the table definition and its contents, while the TRUNCATE command deletes all rows from the table.
2. With DROP, table space is freed from memory, whereas TRUNCATE does not free table space.
3. Both DROP and TRUNCATE are DDL (Data Definition Language) commands.
4. In DROP a view of the table does not exist, while TRUNCATE does.
5. In DROP, integrity constraints are removed, whereas they are not removed with TRUNCATE.
6. DROP does not use undo space, while TRUNCATE does, albeit less than DELETE.
7. DROP is quick but can lead to complications, whereas TRUNCATE is faster than DROP.

**55. Which operator is used in queries for pattern matching?**

**LIKE** operator: It is used to fetch filtered data by searching for a particular pattern in the where clause.

```
SELECT column1,column2

FROM table_name

WHERE column_name LIKE pattern;

LIKE: operator name
```

**56. Define SQL Order by the statement?**

The ORDER BY statement in SQL is used to sort the fetched data in either ascending or descending according to one or more columns.

- By default ORDER BY sorts the data in ascending order.
- We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

**57. Explain SQL Having statement?**

**HAVING** is used to specify a condition for a group or an aggregate function used in the select statement. The **WHERE** clause selects before grouping. The **HAVING** clause selects rows after grouping. Unlike the **HAVING** clause, the **WHERE** clause cannot contain aggregate functions.

**58. Explain SQL AND OR statement with an example?**

In SQL, the **AND** & **OR** operators are used for filtering the data and getting precise results based on conditions.

The AND and OR operators are used with the WHERE clause.

These two operators are called ***conjunctive operators***.

**AND Operator**: This operator displays only those records where both conditions: condition 1 and condition 2 evaluate to True.

```
SELECT * FROM employees
WHERE department = 'HR' AND age > 30;
```

**OR Operator**: This operator displays the records where either one of the conditions condition 1 and condition 2 evaluates to True. That is, either condition1 is True or condition2 is True.

```
SELECT * FROM products
WHERE category = 'Electronics' OR category = 'Clothing';
```

**59. Define BETWEEN statements in SQL?**

The SQL **BETWEEN** condition allows you to easily test if an expression is within a range of values (inclusive). The values can be text, date, or numbers. It can be used in a *SELECT*, *INSERT*, *UPDATE*, or *DELETE* statement. The SQL BETWEEN Condition will return the records where the expression is within the range of value1 and value2.

**60. Why do we use  Commit and Rollback commands?**

- COMMIT permanently saves the changes made by the current transaction, while ROLLBACK undoes the changes made by the current transaction.
- After COMMIT execution, changes cannot be undone, whereas a ROLLBACK returns the transaction to its previous state.
- When the transaction is successful, COMMIT is applied, while When the transaction is aborted, ROLLBACK occurs.

**61. What are ACID properties?**

ACID properties are a set of four properties that guarantee the reliability and consistency of database transactions:

1. **Atomicity**: Transactions are treated as indivisible units, ensuring that either all operations within a transaction are completed successfully and applied to the database, or none of them are.
2. **Consistency** : Transactions bring the database from one valid state to another, maintaining data integrity and ensuring that the database remains in a consistent state before and after transaction execution.
3. **Isolation** : Concurrently executing transactions produce the same results as if executed serially, preventing interference between transactions and preserving data integrity in multi-user environments.
4. **Durability** : Once a transaction is committed, its changes are permanently saved to the database, ensuring that they persist even in the event of system failures, such as power outages or crashes.

Together, these properties ensure that database transactions are reliable, consistent, and durable, providing a robust foundation for data management in relational database systems.

**62. Are NULL values the same as zero or a blank space?**
In SQL, zero or blank space can be compared with another zero or blank space. whereas one null may not be equal to another null. null means data might not be provided or there is no data.

**63. What is the need for group functions in SQL?**
In database management, group functions, also known as **aggregate functions**, is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.
Group functions in SQL are needed to perform calculations on groups of rows rather than individual rows. They allow us to aggregate data and obtain summary information, such as finding the *total*, *average*, *maximum*, *minimum*, or *count* of values within a group. This is particularly useful for generating reports, analyzing trends, and summarizing data in databases.

**64. What is the need for a MERGE statement?**
The MERGE command in SQL is actually a combination of three SQL statements: ***INSERT***, ***UPDATE***, and ***DELETE***. In simple words, the **MERGE** statement in SQL provides a convenient way to perform all these three operations together which can be very helpful when it comes to handling large running databases. But unlike *INSERT*, *UPDATE*, and *DELETE* statements **MERGE** statement requires a source table to perform these operations on the required table which is called a *target table*. This is useful for implementing data synchronization processes, data warehousing, and managing slowly changing dimensions in database systems.

**65. How can you fetch common records from two tables?**

Specifically, you would use an INNER JOIN to combine rows from both tables based on a common key column. Here's an example:

```
SELECT table1.column1, table2.column2
FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

The INNER JOIN ensures that only records with matching values in the common column are returned, effectively fetching common records from both tables.

**66. What are the advantages of PL/SQL functions?**

The advantages of PL / SQL functions are as follows:

PL/SQL functions offer several advantages:

1. **Code Reusability** : Functions can be called from multiple locations, promoting modular code design.
2. **Encapsulation**: Allows hiding implementation details and exposing only necessary functionality.
3. **Parameterization** : Accepts input parameters to customize behavior, enhancing flexibility.
4. **Error Handling** : Supports exception handling to gracefully manage errors within functions.
5. **Performance** : Reduces network traffic by processing data on the server side.
6. **Security** : Enhances security by limiting direct access to database objects and data.
7. **Consistency** : Promotes consistent data processing logic across applications.

**67. What is the SQL query to display the current date?**

```
SELECT CURRENT_DATE;
```

**68. What are Nested Triggers?**

A trigger can also contain ***INSERT***, ***UPDATE***, and ***DELETE*** logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger.

Nested triggers are triggers within a ***database management system (DBMS)*** that, when executed due to a data modification event, can themselves cause additional triggers to be fired. In other words, nested triggers are triggers that contain data modification logic within their own trigger body. This can create a cascading effect, where the firing of one trigger leads to the firing of another trigger, and so on. Proper management of nested triggers is important to avoid unintended consequences and ensure efficient performance in database systems.

**69. How to find the available constraint information in the table?**

In SQL Server the data dictionary is a set of database tables used to store information about a database's definition. One can use these data dictionaries to check the constraints on an already existing table and to change them(if possible).

To find available constraint information in a table, you can query the system catalogs or data dictionary views provided by your database management system (DBMS). These views contain metadata about the constraints defined on tables in the database.

**70. How do we avoid getting duplicate entries in a query without using the distinct keyword?**

DISTINCT is useful in certain circumstances, but it has drawbacks that it can increase the load on the query engine to perform the sort (since it needs to compare the result set to itself to remove duplicates). We can remove duplicate entries using the following options:

- Remove duplicates using row numbers.
- Remove duplicates using self-Join.
- Remove duplicates using group by.

**71. The difference between NVL and NVL2 functions?**

The NVL and NVL2 functions are used in SQL to handle NULL values, but they differ in functionality:

1. **NVL :**
    - Syntax: 'NVL(expr1, expr2)'
    - Returns 'expr2' if 'expr1' is NULL; otherwise, returns 'expr1'.
    - NVL replaces NULL values with a specified default value.
2. **NVL2 :**
    - Syntax: 'NVL2(expr1, expr2, expr3)'
    - Returns 'expr2' if 'expr1' is not NULL; otherwise, returns 'expr3'.
    - NVL2 allows you to specify different expressions based on whether the first expression is NULL or not.

In summary, NVL provides a single default value to replace NULL, while NVL2 allows for different values based on the presence or absence of NULL.

**72. What is the difference between COALESCE() & ISNULL()?**

The COALESCE() and ISNULL() functions are used to handle NULL values in SQL, but they differ in their implementation:

1. **COALESCE() :**
    - Syntax: **SELECT column(s), COALESCE(expression_1,….,expression_n)FROM table_name;**
    - Returns the first non-NULL expression in the list.

- COALESCE allows you to specify multiple expressions and returns the first non-NULL value found.

2. **ISNULL() :**
   - Syntax: **SELECT column(s), ISNULL(column_name, value_to_replace)FROM table_name;**
   - Returns `expr2` if `expr1` is NULL; otherwise, returns `expr1`.
   - ISNULL is specific to SQL Server and replaces NULL values with a specified default value.

In summary, COALESCE() returns the first non-NULL expression from a list of expressions, while ISNULL() replaces NULL values with a specified default value, but it is specific to SQL Server.

**73. Name the operator which is used in the query for appending two strings?**

In many SQL dialects, including Oracle, PostgreSQL, and SQLite, the concatenation operator is represented by the "**||**" symbol. Here's an example:

```
SELECT 'Hello' || 'World' AS ConcatenatedString
```

This query would result in the string "HelloWorld" being concatenated.

# Phase II

**1. What is a self-join, and how would you use it?**

A self-join is a type of join where a table is joined with itself. It is useful when creating relationships within the same table, such as finding hierarchical relationships or comparing rows with related data.

To perform a self-join, you typically use table aliases to differentiate between the different instances of the same table. Here's a basic example:

Suppose you have a table named "employees" with columns for employee ID, name, and manager ID. You want to retrieve the names of employees along with the names of their managers. You can achieve this with a self-join as follows:

```
SELECT e.name AS employee_name, m.name AS manager_name
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id;
```

In this example, "employees" is joined with itself using aliases "e" and "m". The join condition compares the manager ID of each employee (e) with the employee ID (m) to find the corresponding manager's name. This allows you to retrieve the names of employees and their managers in the same query.

**2. How do you optimize SQL queries?**

SQL query optimization aims to enhance query performance by reducing resource usage and execution time. It involves strategies such as utilizing appropriate indexes,

optimizing query structure, and avoiding resource-intensive operations like full table scans.
Key optimization techniques include:

1. Employing indexes for faster data retrieval.
2. Minimizing wildcard usage in WHERE clauses.
3. Avoiding SELECT * and fetching only necessary columns.
4. Restricting the number of returned rows with LIMIT or TOP.
5. Optimizing joins and avoiding unnecessary ones.
6. Analyzing and rewriting complex queries for improved efficiency.
7. Utilizing EXPLAIN or equivalent to scrutinize query execution plans.
8. Regularly updating statistics for the database optimizer to make well-informed decisions.

**3. What is the difference between UNION and UNION ALL?**
The UNION and UNION ALL operators are used in SQL to combine the results of two or more SELECT queries into a single result set, but they differ in how they handle duplicate rows:

1. **UNION :**
    - Removes duplicate rows from the combined result set.
    - Only distinct rows are retained, eliminating duplicates.
2. **UNION ALL :**
    - Retains all rows from the combined result set, including duplicates.
    - Does not remove duplicate rows, resulting in all rows being included in the output.

In summary, UNION removes duplicate rows, while UNION ALL includes all rows, regardless of duplicates.

**4. What are correlated subqueries?**
Correlated subqueries are subqueries that reference columns from the outer query. They are evaluated for each row processed by the outer query, making them dependent on the outer query's results. Correlated subqueries can be used to filter or retrieve data based on conditions related to each row of the outer query's result set.

**5. What is a transaction in SQL?**
A transaction in SQL is a sequence of one or more SQL operations treated as a single unit of work. It represents a sequence of operations that should be performed together as a single logical unit. Transactions ensure that database operations are either completed successfully or rolled back entirely in case of failure.

**6. How do you implement error handling in SQL?**
Error handling in SQL can be implemented using the following techniques:

1. **TRY...CATCH** : Encapsulate code in a **TRY** block, with error handling logic in a **CATCH** block.
2. **RAISEERROR** : Generate custom error messages and set severity levels using the **RAISEERROR** function.
3. **@@ERROR** : Check the value of **@@ERROR** after executing SQL statements to determine if an error occurred.
4. **XACT_ABORT** : Set **XACT_ABORT** to ON to automatically roll back transactions on run-time errors, ensuring database consistency.

**7. Describe the data types in SQL**

In SQL, data types define the type of data that can be stored in a column or variable. Common data types include:

1. **Numeric** : Integers (INT), floating-point numbers (FLOAT), decimal numbers (DECIMAL), etc.
2. **Character** : Fixed-length strings (CHAR), variable-length strings (VARCHAR), text (TEXT), etc.
3. **Date and Time** : Date (DATE), time (TIME), timestamp (TIMESTAMP), etc.
4. **Boolean** : Stores true or false values (BOOLEAN).
5. **Binary** : Stores binary data, such as images or files (BLOB, BYTEA).
6. **Others** : Various specialized types like UUID, XML, JSON, etc.

Different database systems may have additional data types or variations of these types, and each type has specific storage requirements and allowed values. Choosing the appropriate data type is important for efficient storage and retrieval of data.

**8. What are the different types of triggers?**

Triggers in SQL are pieces of procedural code executed in response to specific events in a database. There are three main types:

1. **DML Triggers** : Execute additional code when data is modified using insert, update, or delete statements.
2. **DDL Triggers** : Execute code in response to changes in the database structure or server events, such as table creation or user login.
3. **Logon Triggers** : Special server-scoped triggers that fire when a user session is established.

**9. Explain the concept of a database schema.**

In SQL, a database schema acts as a conceptual container for storing tables, views, indexes, and procedures. It organizes and separates these elements while defining their structure and relationships.

**10. What is a deadlock in SQL? How can it be prevented?**

A deadlock in SQL occurs when two or more transactions cannot proceed because they are waiting for resources held by each other. Deadlocks can be prevented or resolved by using techniques such as locking hierarchies, timeouts, or deadlock detection and resolution mechanisms.

**11. Explain different isolation levels in SQL.**

Different isolation levels in SQL determine how transactions interact with each other and with the database:

1. **Read Uncommitted** : Allows transactions to read data that has been modified but not committed by other transactions. This can lead to dirty reads.
2. **Read Committed** : Ensures that transactions only read data that has been committed by other transactions. However, it may still result in non-repeatable reads and phantom reads.
3. **Repeatable Read** : Guarantees that transactions always see a consistent snapshot of the database, preventing non-repeatable reads but allowing phantom reads.
4. **Serializable** : Provides the highest level of isolation, ensuring transactions are executed as if they were executed one after another. This prevents dirty reads, non-repeatable reads, and phantom reads, but can impact concurrency.

**12. How does the clustered index work?**

A clustered index in SQL physically sorts table rows based on the indexed column(s). It determines the physical order of data on disk, allowing for efficient retrieval of rows using range scans. As new rows are added, they are inserted into the appropriate position in the sorted order, maintaining the physical order of the data.

**13. How does the non-clustered index work?**

A non-clustered index in SQL creates a separate data structure containing indexed column(s) and pointers to the corresponding rows in the table. It does not affect the physical order of data on disk but provides a quick lookup mechanism for retrieving specific rows based on the indexed column(s). This allows for efficient retrieval of data without rearranging the physical storage of the table.

**14. Discuss SQL server reporting services.**

SQL Server Reporting Services is a reporting tool provided by Microsoft for creating, managing, and delivering interactive, tabular, graphical, and free-form reports. SSRS allows users to design and generate reports from various data sources, making it a valuable asset for businesses needing comprehensive reporting capabilities.

**15. What are CTEs (Common table expressions)?**

**Common Table Expressions** (CTEs) are temporary result sets that you can reference within SQL statements, like **SELECT**, **INSERT**, **UPDATE**, or **DELETE** operations. They streamline complex queries by breaking them into more manageable parts, defined using the **WITH** keyword. CTEs are useful for recursive queries, where a query refers to its own output. They resemble creating a view but offer more flexibility and are defined inline within the query itself. With CTEs, you can improve query readability and maintainability by organizing logic into reusable components within a single SQL statement.

**16. How do you use a window function in SQL?**

Window functions are employed to carry out computations on a group of table rows that are associated with the current row. They enable the generation of result sets containing aggregated data while retaining the distinct details of each row. Typical window functions encompass **ROW_NUMBER(), RANK(), DENSE_RANK(), SUM()** and **OVER**().

**17. Explain Row_Number, Rank, Dense_Rank**

1. **ROW_NUMBER()** : Assigns a unique sequential integer to each row in the result set, regardless of duplicates.
2. **RANK()** : Assigns a unique integer to each distinct value in the result set, with gaps in ranking for ties.
3. **DENSE_RANK()** : Assigns a unique integer to each distinct value in the result set, without gaps in ranking for ties.

**18. What is a pivot table and how do you create one in SQL?**

A pivot table is a technique used to ***rotate*** or ***transpose*** rows into columns to better analyze and summarize data. You can create pivot tables in SQL using the **'PIVOT'** operator to convert row-based data into a column-based format.

**19. Describe the process of database mirroring.**

Database mirroring is a high-availability solution in SQL Server that involves creating and maintaining redundant copies of a database on separate servers. It ensures data availability and disaster recovery by automatically failing over to the mirror server in case of a primary server failure.

**20. Explain the concept of table partitioning.**

Partitioning a table involves the strategy of breaking down a sizable table into smaller, more easily handled segments known as **partitions**. This method can enhance query efficiency by permitting SQL Server to focus solely on particular partitions while executing queries. Typically, partitioning is carried out using a column characterized by a high cardinality, such as date or region.

**21. How do you handle transactions in distributed databases?**
Handling transactions in distributed databases involves ensuring the ***ACID*** (Atomicity, Consistency, Isolation, Durability) properties across multiple databases or nodes. This can be achieved through distributed transaction management protocols like Two-Phase Commit (2PC) or by using distributed database systems designed for this purpose.

**22. What is the use of the explain plan?**
The **EXPLAIN** plan is a valuable feature found in numerous relational database management systems. This tool offers a comprehensive view of the database engine's strategy for executing a query, encompassing details such as the selected execution plan, join techniques, index utilization, and projected costs. Database administrators (DBAs) and developers rely on **EXPLAIN** plans to enhance the performance of their queries.

**23. What are indexed views?**
Indexed views, are precomputed result sets stored as physical tables in the database. They improve query performance by allowing the database engine to access pre-aggregated or pre-joined data directly from the indexed view, reducing the need for complex query processing.

**24. Explain the concept of database sharding.**
Database sharding is a horizontal partitioning technique that distributes data across multiple database instances or servers. It's commonly used in large-scale systems to improve scalability and performance. Each shard contains a subset of the data, and a sharding strategy determines how data is distributed.

**25. How do you manage large-scale databases for performance?**
Managing large-scale databases for performance involves various strategies, including proper **indexing**, **partitioning**, **query optimization**, **hardware optimization**, and **caching**. **Monitoring** and **fine-tuning** the database is crucial to ensure optimal performance as data volumes grow.

**26. What is a materialized view?**
Materialized views are a type of database component designed to maintain the outcomes of a query in the form of a tangible table. These views undergo periodic updates to ensure that the stored data remains current. They are employed to enhance the efficiency of database queries, particularly for intricate or frequently executed ones.
**27. Difference between Indexed View and Materialized View:**

- **Indexed View:**

1. Stores precomputed results of a query as a regular view with a unique clustered index.
2. Data is not physically stored separately but is indexed for quick retrieval.
3. Automatically updated by the SQL Server engine when underlying data changes, but may suffer from performance overhead during updates.

- **Materialized View:**
    1. Stores precomputed results of a query as a physical table-like structure.
    2. Data is physically stored separately, reducing the need for re-computation during query execution.
    3. Requires manual refreshing or updating to synchronize with changes in the underlying data, providing better control over update frequency and performance impact.

**28. Discuss the strategies for database backup and recovery.**
Ensuring data availability and disaster recovery relies on the implementation of vital backup and recovery strategies. These strategies encompass various methods, such as **full backups**, **differential backups**, **transaction log backups**, and **regular testing of restoration procedures**.

**29. What are the best practices for securing a SQL database?**
Securing a SQL database involves implementing access **controls**, **encryption**, **auditing**, and **regular security assessments**. Best practices include using **strong authentication**, **limiting permissions**, and keeping **database systems** and **software up to date**.

**30. Explain the concept of database replication.**
It is the process of copying and synchronizing data from one database to another. It ensures data availability, load balancing, and disaster recovery. Common replication types include snapshot replication, transactional replication, and merge replication.

**31. What is a database warehouse?**
A database warehouse is a centralized repository that stores data from various sources for analytical and reporting purposes. It is optimized for querying and analysis and often contains historical data.

**32. Explain the use of full-text search in SQL.**
Full-text search in SQL allows users to search for text-based data within large text fields or documents. It uses advanced indexing and search algorithms to provide efficient and accurate text-searching capabilities.

**33. What are the challenges in handling big data in SQL?**

Handling big data in SQL involves dealing with large volumes of data that exceed the capabilities of traditional database systems. Challenges include **data storage**, **processing**, **scalability**, and **efficient querying**. Solutions may include **distributed databases** and **big data technologies** like ***Hadoop*** and ***Spark***.

**34. How do you implement high availability in SQL databases?**
High availability in SQL databases ensures that the database remains accessible and operational despite failures. Techniques like **clustering**, **replication**, and **failover mechanisms** help achieve high availability.

**35. Explain the use of XML data type in SQL server.**
The XML data type allows to **store**, **retrieve**, and **manipulate** XML data. It provides support for querying XML documents using ***XQuery***, and it's commonly used in applications that deal with XML data structures.

**36. Discuss the concept of NoSQL databases and their interaction with SQL.**
NoSQL databases are non-relational databases designed for handling large volumes of unstructured or semi-structured data. They interact with SQL databases through various integration methods, such as **data pipelines**, **ETL processes**, and **API-based data transfers**.

**35. What is a spatial database?**
A spatial database stores and queries geometric and geographic data, such as maps, GPS coordinates, and spatial objects. It provides specialized functions and indexing methods to support spatial queries and analysis.

**36. How do you migrate a database from one server to another?**
Database migration involves moving a database from one server or platform to another. This undertaking demands careful **planning**, **data transfer**, **schema conversion**, and **thorough testing** to ensure a smooth transition while **minimizing** the **potential** for **data loss** or **system downtime**.

**37. Discuss advanced optimization techniques for SQL queries.**
Advanced optimization techniques for SQL queries include using query hints, indexing strategies, query rewriting, and understanding the query execution plan. Profiling tools and performance monitoring are essential for identifying and resolving performance bottlenecks.

**38. Explain different types of key in SQL**

1. **Unique Key** : Ensures that each value in a column or combination of columns is unique within a table, similar to a primary key but allows null values.
2. **Primary Key** : Uniquely identifies each row in a table and ensures data integrity.
3. **Foreign Key** : Establishes a relationship between two tables by referencing the primary key of another table, enforcing referential integrity.
4. **Composite Key** : Consists of multiple columns that, together, uniquely identify each row in a table.
5. **Candidate Key** : A key that could potentially become the primary key of a table, fulfilling the uniqueness requirement.
6. **Super Key** : A set of attributes that uniquely identifies each tuple in a relation, which may include keys and additional attributes.

```
SELECT COALESCE(a.user_id ,b.user_id) as user_id ,
case
when b.user_id is null then 'CHURN'
when a.status = 'NEW' then 'EXISTING'
when a.status = 'CHURN' then 'RESURRECT'
when a.status = 'EXISTING' then 'EXISTING'
when a.status = 'RESURRECT' then 'EXISTING'
else 'NEW' end as "new_status"
FROM advertiser as a
FULL OUTER JOIN daily_pay as b
on a.USER_ID = b.user_id
order by 1
```

**I need to practice on windows functions**

- `RANK() -`
- `DENSE_RANK()`
- `ROW_NUMBER()`
- `NTILE()`
- `LAG()`
- `LEAD()`
- `NTH_VALUE()`