**window functions**

Window functions in SQL are powerful tools that allow you to perform calculations across a set of table rows that are related to the current row. Unlike aggregate functions, which return a single result for a group of rows, window functions return a value for each row in the result set.

**Def: -**

* A window function performs a calculation across a set of table rows that are somehow related to the current row. This set of rows is called a window

**window\_function\_name**: The name of the window function, such as ROW\_NUMBER (), RANK (), SUM (), LEAD (), LAG () etc.

* **OVER ()**: This clause defines the window partition and order. It can include PARTITION BY: Divides the result set into partitions to which the window function is applied. ORDER BY: Specifies the order of rows within each partition. ROWS | RANGE: Defines the frame within the partition.

**Types of Window Functions**

Window functions can be categorized into three main types:

1. **Value Window Functions**: These functions return a single value for each row. Examples: FIRST\_VALUE (), LAST\_VALUE (), LAG (), LEAD ()
2. **Ranking Window Functions**: These functions assign a rank to each row within the partition. Examples: ROW\_NUMBER (), RANK (), DENSE\_RANK ()
3. **Aggregate Window Functions**: These functions perform aggregate calculations over a window. Examples: SUM (), AVG (), COUNT (), MAX(), MIN()

**Examples:**

**Aggregate Window Functions-**

**1. Using SUM () as a Window Function**

To calculate the total salary of all employees along with the salary of each individual employee:

A screenshot of a computer

Description automatically generated

A number of numbers and numbers

Description automatically generated with medium confidence

**2.AVG**- as an Analytical Function

To calculate the average price over all rows and include it in each row's result:

A screen shot of a computer

Description automatically generated



**3.COUNT-** as an Analytical Function

To count the number of employees in each department, but show the count alongside each employee's details:

A screenshot of a computer

Description automatically generated

A number of numbers on a white background

Description automatically generated

**4.MIN-** Function as an Analytical Function

To find the minimum salary within each department without collapsing the rows, you can use the MIN function as an analytical function with the OVER clause:

A close-up of a screen

Description automatically generated

A close-up of a number of numbers

Description automatically generated

**5.MAX**- as an Analytical Function

To find the maximum salary within each department, you can use the MAX function as an analytical function with the OVER clause:

A screen shot of a computer

Description automatically generated

A close-up of a number

Description automatically generated

**Ranking Window Functions-**

**1.ROW NUMBER ()-**

The ROW\_NUMBER () function is a type of window function that assigns a unique sequential integer to rows within a partition of a result set.

A screen shot of a computer

Description automatically generated

ROW\_NUMBER (): This function assigns a unique number to each row.

OVER: This clause defines the window for the function.

PARTITION BY department: This divides the result set into partitions by the department column. The ROW\_NUMBER () function will reset its count for each partition.

ORDER BY salary DESC: This orders the rows within each partition by the salary column in descending order.

A white background with black text

Description automatically generated

**2. Rank()-**

The RANK() function in SQL is a window function that assigns a rank to each row within a partition of a result set. The rank of a row is determined by one plus the number of ranks that come before it. This function is useful for ranking rows based on specific criteria, such assales volume, exam results, or employee performance.

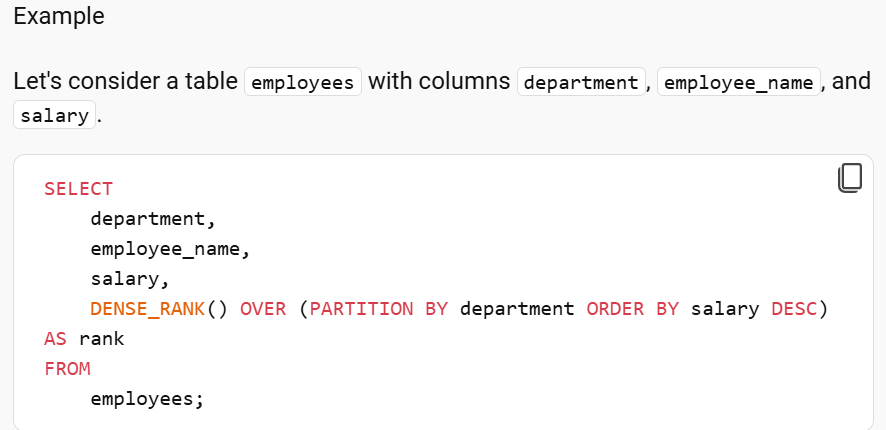
A screenshot of a chat

Description automatically generated

* **DENSE\_RANK ()**: Similar to **RANK ()**, but does not leave gaps in the ranking sequence when there are ties.

**3.DENSE\_RANK Function-**

The DENSE\_RANK function is a window function that assigns ranks to rows within a partition of a result set, with no gaps in ranking values. This means if two rows are tied, they receive the same rank, and the next rank is the immediate next integer.



A screenshot of a computer

Description automatically generated

**VALUE Window Functions-**

**1.LEAD Function-**

The LEAD function allows you to access data from a subsequent row in the same result set without the need for a self-join. This can be particularly useful for comparing values in consecutive rows.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**2.LAG Function**-

SQL LAG() is a window function that provides access to a row at a specified physical offset which comes before the current row. In other words, by using the LAG() function, from the current row, you can access data of the previous row, or from the second row before the current row, or from the third row before current row, and so on.

A white background with text

Description automatically generated

A screenshot of a graph

Description automatically generated

**3.FIRST\_VALUE**-

which returns the first value in an ordered set of values.

Now, let's use the FIRST\_VALUE function to find the first salary in each department when ordered by salary:

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

This way, you can easily identify the first value in an ordered set within each group of rows.

**4.LAST\_VALUE-**

which returns the last value in an ordered set of values.

A screenshot of a computer

Description automatically generated

**Explanation**

* **PARTITION BY product\_id**: This clause divides the result set into partitions by product\_id.
* **ORDER BY sale\_date**: This clause orders the rows within each partition by sale\_date.
* **ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING**: This frame clause ensures that the window includes all rows in the partition.
* **UNBOUNDED PRECEDING**: All rows before current row are considered
* **UNBOUNDED FOLLOWING**: All rows after the current row are considered

A screenshot of a computer

Description automatically generated

**Conclusion**

Window functions in SQL provide a flexible and powerful way to perform complex calculations over a set of rows related to the current row. They are essential tools for data analysis and reporting, allowing you to perform tasks such as ranking, running totals, and moving averages efficiently.

**SQL Performance Tuning**-  
[SQL performance tuning](https://www.geeksforgeeks.org/sql-performance-tuning/) is the process of optimizing SQL queries to improve the **speed**and **efficiency**of [database](https://www.geeksforgeeks.org/what-is-database/)**operations**. It involves various techniques to optimize the **execution of queries**, manage **system resources** more effectively, and ensure that the database responds quickly to user requests.

**Factors Affecting SQL Speed -**

Some of the major factors that influence the computation and execution time in SQL are:

* **Table Size:** Larger tables with millions of rows can slow down query performance if the query hits a large number of rows.
* **Joins:** The use of complex joins, especially when joining multiple tables, can significantly affect query execution time.
* **Aggregations:** Queries that aggregate large datasets require more processing time and resources.
* **Concurrency:** Simultaneous queries from multiple users can overwhelm the database, leading to slow performance.
* **Indexes:** Proper indexing speeds up data retrieval but, when misused, can lead to inefficiencies.

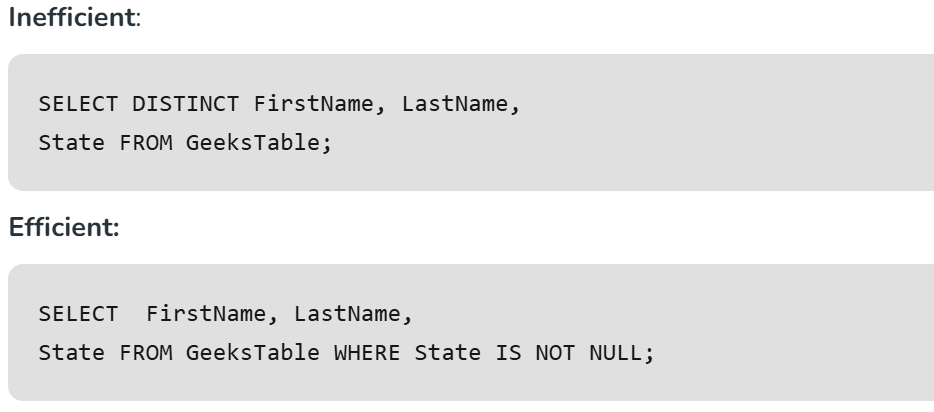
**SQL Query Optimization Techniques**

**1. SELECT fields instead of using SELECT \***

**A screenshot of a computer code

Description automatically generated**

**2. Avoid SELECT DISTINCT**



**3. Use INNER JOIN Instead of WHERE for Joins**

A screenshot of a computer

Description automatically generated

**4. Use WHERE Instead of HAVING**

A screenshot of a computer code

Description automatically generated

**Indexing in SQL**- is a technique used to improve the speed of data retrieval operations on a database table. An index is a schema object that provides a quick way to look up rows based on the values of one or more columns. It works similarly to the index in a book, which helps you find information quickly without having to read through every page.

**Creating an Index-**

**A screenshot of a computer

Description automatically generated**

**Types of Indexes**

**There are several types of indexes, including:**

* **Primary Index**: Created on the primary key of a table. It ensures that the primary key values are unique and helps in quick retrieval of rows.
* **Clustered Index**: Arranges the data rows in the table based on the index key. Each table can have only one clustered index.
* **Secondary Index**: Created on non-primary key columns to speed up queries that use those columns.
* **Dense Index**: Contains an index entry for every search key value in the data file.
* **Sparse Index**: Contains index entries for only some of the search key values.

**SQL Performance Tuning Tools-**

Several SQL performance tuning tools can help identify and optimize database performance. Some of the popular tools include:

* **SQL Sentry (SolarWinds)**
* **SQL Profiler (Microsoft)**
* **SQL Index Manager (Red Gate)**
* **SQL Diagnostic Manager (IDERA)**

**open-source performance tuning tools for SQL:**

**1.pgBadger:**

**Description**: A fast PostgreSQL log analyzer that generates detailed reports on database performance.

**Features**: Provides insights into query performance, lock analysis, and connection statistics.

**Usage**: Ideal for PostgreSQL users looking to optimize their database performance through comprehensive log analysis.

**2.Percona Toolkit:**

**Description**: A collection of advanced command-line tools for MySQL, MariaDB, and MongoDB.

**Features**: Includes tools for query analysis, replication management, and server diagnostics.

**Usage**: Perfect for database administrators who need to perform complex database tasks efficiently.

**3.pg\_stat\_statements:**

**Description**: An extension for PostgreSQL that tracks execution statistics of all SQL statements.

**Features**: Provides detailed information on query execution times, frequency, and resource usage.

**Usage**: Useful for identifying slow queries and understanding their impact on overall database performance.

**Conclusion-**

SQL performance tuning is important for ensuring efficient [database](https://www.geeksforgeeks.org/what-is-database/)operations, especially as the size and complexity of databases grow