Different Types of SQL Functions

SQL, or Structured Query Language, is a programming language used for managing and manipulating relational databases. One of the most powerful features of SQL is the ability to use functions to perform various operations on the data in a database. In this article, we'll discuss the different categories of SQL functions and provide code examples to help illustrate their use.

Aggregate Functions

Aggregate functions are used to perform calculations on a set of values and return a single result. Some of the most commonly used aggregate functions in SQL include –

- **COUNT()** - Returns the number of rows in a table or the number of non-NULL values in a column
- **SUM()** - Returns the sum of all non-NULL values in a column
- **AVG()** - Returns the average of all non-NULL values in a column
- **MIN()** - Returns the minimum value in a column
- **MAX()** - Returns the maximum value in a column

Here's an example of using the COUNT() function to find the number of rows in a table called "orders" –

```
SELECT COUNT(*) FROM orders;
```

And here's an example of using the SUM() function to find the total cost of all orders in the table –

```
SELECT SUM(total_cost) FROM orders;
```

Scalar Functions

Scalar functions are used to perform calculations on a single value and return a single result. Some examples of scalar functions in SQL include –

- **LENGTH()** - Returns the number of characters in a string
- **UPPER()** - Converts a string to uppercase
- **LOWER()** - Converts a string to lowercase

- **CONCAT()** - Concatenates two or more strings together
- **ROUND()** - Rounds a number to a specified number of decimal places

Here's an example of using the UPPER() function to display the names of all customers in uppercase –

```
SELECT UPPER(customer_name) FROM customers;
```

And here's an example of using the ROUND() function to round the total cost of an order to two decimal places –

```
SELECT ROUND(total_cost, 2) FROM orders;
```

Date and Time Functions

SQL also provides a number of functions for working with date and time values. Some examples of date and time functions in SQL include –

- **NOW()** - Returns the current date and time
- **CURRENT_DATE()** - Returns the current date
- **CURRENT_TIME()** - Returns the current time
- **YEAR()** - Returns the year of a date
- **MONTH()** - Returns the month of a date
- **DAY()** - Returns the day of a date

Here's an example of using the NOW() function to find the current date and time –

```
SELECT NOW();
```

And here's an example of using the MONTH() function to find the month of an order's date –

```
SELECT MONTH(order_date) FROM orders;
```

String Functions

SQL also provides a number of string manipulation function. Some examples of string functions in SQL include –

- **LTRIM()** - Removes the leading whitespace of the string
- **RTRIM()** - Removes the trailing whitespace of the string
- **TRIM()** - Removes both leading and trailing whitespace of the string
- **SUBSTRING()** - Extracts a specific portion of a string
- **REPLACE()** - Replaces all occurrences of a specified string with another string

Conditional Functions

SQL also provides a number of functions that perform different actions based on certain conditions. Some examples of conditional functions in SQL include –

- **CASE** - evaluates a list of conditions and returns a result for the first condition that is met
- **IF** - return a specified value if the condition is met, otherwise return another specified value
- **COALESCE** - return the first non-null expression among multiple expressions.

Here's an example of using the CASE function to assign a label to each order based on the total cost –

```
SELECT order_id, total_cost,
   CASE
      WHEN total_cost > 100 THEN 'expensive'
      WHEN total_cost > 50 THEN 'moderately priced'
      ELSE 'inexpensive'
   END as "price range"
FROM orders;
```

Here's an example of using the IF function to check the availability of stock of a product

```
SELECT product_name,
   IF(stock_quantity > 0, 'In Stock', 'Out of Stock') as
   "Availability"
FROM products;
```

Here's an example of using the COALESCE function to display the primary phone number and the secondary phone number of a customer –

```
SELECT customer_name,
   COALESCE(primary_phone, secondary_phone) as "Phone Number"
FROM customers;
```

Logical Functions

SQL provides a set of logical functions that return a Boolean value, which can be either true or false. Some examples of logical functions in SQL include –

- **AND** - Returns true if both the conditions are true
- **OR** - Returns true if at least one of the conditions is true
- **NOT** - Negates a boolean value

Here's an example of using the AND function to find all customers who live in a specific city and have an account balance greater than a certain amount –

```
SELECT customer_name, city, account_balance
FROM customers
WHERE city = 'New York' AND account_balance > 1000;
```

Conversion Functions

SQL provides a number of functions that can be used to convert data from one type to another. Some examples of conversion functions in SQL include –

- **CAST()** - Converts a value from one data type to another
- **CONVERT()** - Converts a value from one data type to another (This function is specific for some database vendors like SQL Server)
- **TO_DATE()** - Converts a string to a date value
- **TO_TIME()** - Converts a string to a time value
- **TO_TIMESTAMP()** - Converts a string to a timestamp value

Here's an example of using the CAST() function to convert a float value to an int –

```
SELECT CAST(price AS INT) as "Integer Price"
FROM products;
```

Here's an example of using the TO_DATE() function to convert a string to a date value –

```
SELECT TO_DATE(order_date, 'yyyy-mm-dd') as "Formatted Order Date"
FROM orders;
```

Window Functions

SQL provides a set of functions that can be used to perform calculations across a set of rows that are related to the current row. These functions are known as window functions. Some examples of window functions in SQL include –

- **RANK()** - Assigns a unique rank to each row within a result set, based on the values in one or more columns
- **DENSE_RANK()** - Assigns a unique rank to each row within a result set, based on the values in one or more columns, but does not leave gaps in the ranking sequence when there are ties
- **ROW_NUMBER()** - Assigns a unique number to each row within a result set, based on the order specified in the ORDER BY clause of the function

Here's an example of using the RANK() function to find the rank of each customer based on their account balance –

```
SELECT customer_name, account_balance, RANK() OVER (ORDER BY account_balance DESC) as
"Rank"
FROM customers;
```

Here's an example of using the ROW_NUMBER() function to find the row number of each customer in the table –

```
SELECT customer_name, ROW_NUMBER() OVER (ORDER BY customer_id) as "Row Number"
FROM customers;
```

These are just a few examples of the many functions that SQL provides for working with and manipulating data in a relational database. Each category of functions serves its own unique purpose, and understanding when and how to use them can help to make working with SQL and relational databases more efficient and effective.

Conclusion

SQL functions are an incredibly powerful tool for working with and manipulating data in a relational database. In this article, we've discussed the different categories of SQL functions, including aggregate functions, scalar functions, date and time functions, string functions, and conditional functions, and provided examples of how they can be used. Understanding and being proficient in the use of these functions is an essential part of working with SQL and relational databases.