Aggregate function

What is Aggregate function in SQL?

★ Aggregate functions helps to summarize the large volumes of data.
★ This function can produced a single value for an entire group or table.
★ They operate on sets of rows and return results based on groups of rows.

List of Aggregate Functions

★ COUNT
★ SUM
★ AVERAGE
★ MAX
★ MIN

COUNT() function

The SQL COUNT function returns the number of rows in a table satisfying the criteria specified in the WHERE clause. It sets on the number of rows or non NULL column values.
SQL Syntax : COUNT(*) , COUNT([ALLlDISTINCT] expression)
MySQL, PostgreSQL, and SQL Server supports the SQL Syntax
DB2 and Oracle Syntax :
COUNT ({*l[DISTINCT] expression}) OVER (window_clause)

Example : COUNT()

Example : SELECT COUNT(*)
FROM product_mast;

Example : COUNT() with WHERE

Example : SELECT COUNT(*)
FROM product_mast
WHERE rate>=20;

Example : COUNT() with DISTINCT

Example : SELECT
COUNT(DISTINCT company)
FROM product_mast;

Example : COUNT() with GROUP BY

Example : SELECT company, COUNT(*)
FROM product_mast GROUP BY company;

Example : COUNT() with HAVING

Example : SELECT company, COUNT(*) FROM
product_mast GROUP BY company
HAVING COUNT(*)>2;

SUM() function

The SQL AGGREGATE SUM() function returns the sum of all selected column.
SQL Syntax : SUM ([ALL | DISTINCT] expression)
MySQL, PostgreSQL, and SQL Server supports the SQL Syntax
DB2 and Oracle Syntax :
SUM ([ALL | DISTINCT] expression) OVER (window_clause)

Example : SUM()

Example : SELECT SUM(cost)
FROM product_mast;

Example : SUM() with WHERE

Example : SELECT SUM(cost)
FROM product_mast
WHERE qty>3;

Example : SUM() with GROUP BY

Example : SELECT SUM(cost)
FROM product_mast
WHERE qty>3
GROUP BY
company;

Example : SUM() with HAVING

Example : SELECT company, SUM(cost)
FROM product_mast

GROUP BY company
HAVING SUM(cost)>=170;

AVG() function

The SQL AVG function calculates the average value of a column of numeric type.
It returns the average of all non NULL values.
SQL Syntax : AVG ([ALL | DISTINCT] expression)
MySQL, PostgreSQL, and SQL Server supports the SQL Syntax
DB2 and Oracle Syntax :
AVG ([ALL | DISTINCT] expression) OVER (window_clause)

Example : AVG()

Example : SELECT AVG(cost)
FROM product_mast;

Example : AVG() with HAVING

Example : SELECT company, AVG(cost)
FROM product_mast
GROUP BY company
HAVING AVG(cost)>=65;

MAX() function

The aggregate function SQL MAX() is used to find the maximum value or highest value of a certain column or expression. This function is useful to determine the largest of all selected values of a column.
SQL Syntax : MAX ([ALL | DISTINCT] expression)
MySQL, PostgreSQL, and SQL Server supports the SQL Syntax
DB2 and Oracle Syntax :
MAX ([ALL | DISTINCT] expression) OVER (window_clause)

Example : MAX()

Example : SELECT MAX(rate)
FROM product_mast;

Example : MAX() with HAVING

Example : SELECT company, MAX(rate)
FROM product_mast
GROUP BY company
HAVING MAX(rate)=30;

MIN() function

The aggregate function SQL MIN() is used to find the minimum value or lowest value of a column or expression. This function is useful to determine the smallest of all selected values of a column.
Syntax : MIN([ALL | DISTINCT] expression)
MySQL, PostgreSQL, and SQL Server supports the SQL Syntax
DB2 and Oracle Syntax :
MIN ([ALL | DISTINCT] expression) OVER (window_clause)

Example : MIN()

Example : SELECT MAX(rate)
FROM product_mast;

Example : MIN() with HAVING

Example : SELECT company, MIN(rate)
FROM product_mast
GROUP BY company
HAVING MIN(rate)<20;

Common Functions

- LENGTH(string): Returns the length of the provided string
- INSTR(string, substring, [start_position], [occurrence]): Returns the position of the substring within the specified string.
- TO_CHAR(input_value, [fmt_mask], [nls_param]): Converts a date or a number to a string
- TO_DATE(charvalue, [fmt_mask], [nls_date_lang]): Converts a string to a date value.
- TO_NUMBER(input_value, [fmt_mask], [nls_param]): Converts a string value to a number.
- ADD_MONTHS(input_date, num_months): Adds a number of months to a specified date.
- SYSDATE: Returns the current date, including time.
- CEIL(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, round_to): Rounds a number to a specified number of decimal places.
- TRUNC(input_value, dec_or_fmt): Truncates a number or date to a number of decimals or format.
- REPLACE(whole_string, string_to_replace, [replacement_string]): Replaces one string inside the whole string with another string.
- SUBSTR(string, start_position, [length]): Returns part of a value, based on a position and length.

Ref
https://www.databasestar.com/oracle-sql-functions/
https://beginner-sql-tutorial.com/oracle-functions.htm

Oracle SQL offers a wide variety of built-in functions that can be used to perform operations on data. These functions are categorized into several types, such as string functions, numeric functions, date functions, aggregate functions, and more. Below are some examples:

1. **String Functions**
   - **`UPPER(string)`**: Converts all characters in the string to uppercase.
     ```sql
     SELECT UPPER('oracle') FROM dual;  -- Output: 'ORACLE'
     ```
   - **`LOWER(string)`**: Converts all characters in the string to lowercase.
     ```sql
     SELECT LOWER('ORACLE') FROM dual;  -- Output: 'oracle'
     ```
   - **`SUBSTR(string, start_position, length)`**: Extracts a substring from a string.
     ```sql
     SELECT SUBSTR('Oracle SQL', 8, 3) FROM dual;  -- Output: 'SQL'
     ```

2. **Numeric Functions**
   - **`ROUND(number, decimals)`**: Rounds a number to a specified number of decimal places.
     ```sql
     SELECT ROUND(123.4567, 2) FROM dual;  -- Output: 123.46
     ```
   - **`TRUNC(number, decimals)`**: Truncates a number to a specified number of decimal places.
     ```sql
     SELECT TRUNC(123.4567, 2) FROM dual;  -- Output: 123.45
     ```
   - **`MOD(number, divisor)`**: Returns the remainder of a division.
     ```sql
     SELECT MOD(10, 3) FROM dual;  -- Output: 1
     ```

3. **Date Functions**
   - **`SYSDATE`**: Returns the current date and time from the database server.
     ```sql
     SELECT SYSDATE FROM dual;
     ```
   - **`ADD_MONTHS(date, months)`**: Adds a specified number of months to a date.
     ```sql
     SELECT ADD_MONTHS(SYSDATE, 6) FROM dual;
     ```
   - **`MONTHS_BETWEEN(date1, date2)`**: Returns the number of months between two dates.
     ```sql
     SELECT MONTHS_BETWEEN('01-AUG-2023', '01-JAN-2023') FROM dual;  -- Output: 7
     ```

4. **Aggregate Functions**

  - **`SUM(expression)`**: Returns the sum of a set of values.
    ```sql
    SELECT SUM(salary) FROM employees;
    ```
  - **`COUNT(expression)`**: Returns the number of rows that match a specified condition.
    ```sql
    SELECT COUNT(*) FROM employees;
    ```
  - **`AVG(expression)`**: Returns the average value of a numeric column.
    ```sql
    SELECT AVG(salary) FROM employees;
    ```

5. **Conversion Functions**
  - **`TO_CHAR(expression, format)`**: Converts a date or number to a string in a specified format.
    ```sql
    SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD') FROM dual;  -- Output: '2024-08-11'
    ```
  - **`TO_DATE(string, format)`**: Converts a string to a date using a specified format.
    ```sql
    SELECT TO_DATE('2024-08-11', 'YYYY-MM-DD') FROM dual;
    ```
  - **`TO_NUMBER(string)`**: Converts a string to a number.
    ```sql
    SELECT TO_NUMBER('12345') FROM dual;
    ```

6. **Conditional Functions**
  - **`NVL(expr1, expr2)`**: Replaces `NULL` with a specified value.
    ```sql
    SELECT NVL(commission_pct, 0) FROM employees;
    ```
  - **`CASE`**: Provides conditional logic within SQL statements.
    ```sql
    SELECT
      CASE
        WHEN salary < 3000 THEN 'Low'
        WHEN salary BETWEEN 3000 AND 7000 THEN 'Medium'
        ELSE 'High'
      END AS salary_category
    FROM employees;
    ```

These are just a few examples of the many functions available in Oracle SQL. Depending on your needs, you can combine these functions to perform more complex operations in your SQL queries.