By definition, an aggregate function performs a calculation on a set of values and returns a single value.

Now, let’s get a real life example, where aggregate function might be needed:

For example, you cannot get the total amount of each order by simply querying from the orderdetails table because the orderdetails table stores only quantity and price of each item. You have to select the quantity and price of an item for each order and calculate the order’s total.

MySQL provides many aggregate functions that include AVG, COUNT, SUM, MIN, MAX, etc. An aggregate function ignores NULL values when it performs calculation except for the COUNT function

**AVG Function:**

SELECT AVG(buyPrice) average\_buy\_price FROM products;

**Count Function:**

SELECT COUNT(\*) AS Total FROM products;

**Sum Function:**

SELECT productCode,sum(priceEach \* quantityOrdered) total

FROM orderdetails GROUP by productCode;

**Max Function:**

SELECT MAX(buyPrice) highest\_price FROM Products;

**Min Function:**

SELECT MIN(buyPrice) lowest\_price FROM Products;

**These are some common aggregate functions.**

**Other Aggregate functions are:**

**[BIT\_AND(expr)](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_bit-and):**

It returns the bitwise AND of all bits in expr. The calculation is performed with 64-bit ([BIGINT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.2.1 Integer Types (Exact Value) - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)) precision.

If there are no matching rows, BIT\_AND() returns a neutral value (all bits set to 1).

**[BIT\_OR(expr)](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_bit-or):**

It Returns the bitwise OR of all bits in **expr**. The calculation is performed with 64-bit ([BIGINT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.2.1 Integer Types (Exact Value) - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)) precision.

If there are no matching rows, [BIT\_OR()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_bit-or) returns a neutral value (all bits set to 0).

**[BIT\_XOR(expr)](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_bit-xor):**

It returns the bitwise [XOR](https://dev.mysql.com/doc/refman/5.7/en/logical-operators.html" \l "operator_xor) of all bits in expr. The calculation is performed with 64-bit ([BIGINT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html" \o "11.2.1 Integer Types (Exact Value) - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT)) precision.

If there are no matching rows, [BIT\_XOR()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_bit-xor) returns a neutral value (all bits set to 0).

**COUNT(DISTINCT expr,[expr...]):**

Returns a count of the number of rows with different non-NULL expr values.

If there are no matching rows, COUNT(DISTINCT) returns 0.

SELECT COUNT(DISTINCT results) FROM student;

And some Other Aggregate Functions:

**[STD()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_std):** Return the population standard deviation

**[STDDEV()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_stddev):** Return the population standard deviation

**[STDDEV\_POP()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_stddev-pop):** Return the population standard deviation

**[STDDEV\_SAMP()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_stddev-samp):** Return the sample standard deviation

**[SUM()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_sum):** Return the sum

**[VAR\_POP()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_var-pop):** Return the population standard variance

**[VAR\_SAMP()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_var-samp):** Return the sample variance

**[VARIANCE()](https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html" \l "function_variance):** Return the population standard variance