**-- SQL Aggregate Functions, Like Operator, Wild card operator**

-----------------------------------------------------------------------

/\*

SQL Aggregate Functions

An aggregate function is a function that performs a calculation on a set of values, and returns a single value.

Aggregate functions are often used with the GROUP BY clause of the SELECT statement. The GROUP BY clause splits the result-set into groups of values and the aggregate function can be used to return a single value for each group.

The most commonly used SQL aggregate functions are:

MIN() - returns the smallest value within the selected column

MAX() - returns the largest value within the selected column

COUNT() - returns the number of rows in a set

SUM() - returns the total sum of a numerical column

AVG() - returns the average value of a numerical column

Aggregate functions ignore null values (except for COUNT(\*)).

\*/

-- SQL MIN() and MAX() Functions

-------------------------------------------------------------------------------------------

-- The SQL MIN() and MAX() Functions

-- The MIN() function returns the smallest value of the selected column.

-- The MAX() function returns the largest value of the selected column.

-- MIN

-- Find the lowest price in the Price column:

select \* from products;

SELECT MIN(Price)

FROM Products;

-- MAX

-- Find the highest price in the Price column:

SELECT MAX(Price)

FROM Products;

SELECT MIN(column\_name)

FROM table\_name

WHERE condition;

select \* from products where supplierid = 7;

select min(price) from products where supplierid = 7;

SELECT MAX(column\_name)

FROM table\_name

WHERE condition;

select \* from products where supplierid = 7;

select max(price) from products where supplierid = 7;

-- Set Column Name (Alias)

-- When you use MIN() or MAX(), the returned column will not have a descriptive name.

-- To give the column a descriptive name, use the AS keyword:

SELECT MIN(Price) AS SmallestPrice

FROM Products;

-- Use MIN() with GROUP BY

-- Here we use the MIN() function and the GROUP BY clause, to return the smallest price for each category in the Products table:

select \* from products;

SELECT MIN(Price) AS SmallestPrice, CategoryID

FROM Products

GROUP BY CategoryID;

SELECT MAX(Price) AS HeighestPrice, CategoryID

FROM Products

GROUP BY CategoryID;

-- SQL COUNT() Function

-------------------------------------------------------------------------------------------

-- The SQL COUNT() Function

-- The COUNT() function returns the number of rows that matches a specified criterion.

-- Find the total number of rows in the Products table:

SELECT COUNT(\*)

FROM Products; -- 78

-- SELECT COUNT(column\_name)

-- FROM table\_name

-- WHERE condition;

-- Specify Column

-- You can specify a column name instead of the asterix symbol (\*).

-- If you specify a column name instead of (\*), NULL values will not be counted.

-- Find the number of products where the ProductName is not null:

insert into products values(79,null,null,null,null,null);

select \* from products;

SELECT COUNT(ProductName)

FROM Products; -- 77(Excluded the nulls)

-- Add a WHERE Clause

-- You can add a WHERE clause to specify conditions:

-- Find the number of products where Price is higher than 20:

SELECT \*

FROM Products

WHERE Price > 20;

SELECT COUNT(ProductID)

FROM Products

WHERE Price > 20;

-- Ignore Duplicates

-- You can ignore duplicates by using the DISTINCT keyword in the COUNT() function.

-- If DISTINCT is specified, rows with the same value for the specified column will be counted as one.

-- How many different prices are there in the Products table:

SELECT COUNT(DISTINCT Price)

FROM Products;

SELECT count(DISTINCT categoryid)

FROM Products;

-- Use an Alias

-- Give the counted column a name by using the AS keyword.

-- Name the column "Number of records":

SELECT COUNT(\*) AS Number\_of\_records

FROM Products;

-- Use COUNT() with GROUP BY

-- Here we use the COUNT() function and the GROUP BY clause, to return the number of records for each category in the Products table:

SELECT COUNT(\*) AS Number\_of\_records, CategoryID

FROM Products

GROUP BY CategoryID;

---------------------------------------------------------------------------------------------------

-- SQL SUM() Function

-- The SQL SUM() Function

-- The SUM() function returns the total sum of a numeric column.

-- Return the sum of all Quantity fields in the OrderDetails table:

select \* from ORDER\_DETAILS;

SELECT SUM(Quantity)

FROM ORDER\_DETAILS; -- 12743

-- Syntax

-- SELECT SUM(column\_name)

-- FROM table\_name

-- WHERE condition;

-- Add a WHERE Clause

-- You can add a WHERE clause to specify conditions:

-- Return the sum of the Quantity field for the product with ProductID 11:

SELECT \*

FROM Order\_Details

WHERE ProductId = 11;

SELECT SUM(Quantity)

FROM Order\_Details

WHERE ProductId = 11;

-- Use an Alias

-- Give the summarized column a name by using the AS keyword.

-- Name the column "total":

SELECT SUM(Quantity) AS total

FROM Order\_Details;

-- Use SUM() with GROUP BY

-- Here we use the SUM() function and the GROUP BY clause, to return the Quantity for each OrderID in the OrderDetails table:

SELECT OrderID

FROM Order\_Details

GROUP BY OrderID;

SELECT OrderID, SUM(Quantity) AS total

FROM Order\_Details

GROUP BY OrderID;

-- You will learn more about the GROUP BY clause later in this tutorial.

-- SUM() With an Expression

-- The parameter inside the SUM() function can also be an expression.

-- If we assume that each product in the OrderDetails column costs 10 dollars,

-- we can find the total earnings in dollars by multiply each quantity with 10:

-- Use an expression inside the SUM() function:

SELECT SUM(Quantity \* 10)

FROM Order\_Details;

-- We can also join the OrderDetails table to the Products table to find the actual amount, instead of assuming it is 10 dollars:

-- Join OrderDetails with Products, and use SUM() to find the total amount:

SELECT Order\_Details.\*,Products.CATEGORYID,Products.PRICE,Products.PRODUCTNAME

FROM Order\_Details

LEFT JOIN Products ON Order\_Details.ProductID = Products.ProductID;

select productid from Order\_Details

intersect

select productid from products; -- 77

SELECT SUM(Price \* Quantity)

FROM Order\_Details

LEFT JOIN Products ON Order\_Details.ProductID = Products.ProductID;

----------------------------------------------------------------------------------------------------------------

-- SQL AVG() Function

-- The SQL AVG() Function

-- The AVG() function returns the average value of a numeric column.

-- Find the average price of all products:

SELECT AVG(Price)

FROM Products;

-- Note: NULL values are ignored.

-- Syntax

-- SELECT AVG(column\_name)

-- FROM table\_name

-- WHERE condition;

-- Add a WHERE Clause

-- You can add a WHERE clause to specify conditions:

-- Return the average price of products in category 1:

SELECT \*

FROM Products

WHERE CategoryID = 1;

SELECT AVG(Price)

FROM Products

WHERE CategoryID = 1;

-- Use an Alias

-- Give the AVG column a name by using the AS keyword.

-- Name the column "average price":

SELECT AVG(Price) AS average\_price

FROM Products;

-- Higher Than Average

-- To list all records with a higher price than average, we can use the AVG() function in a sub query:

-- Return all products with a higher price than the average price:

SELECT \* FROM Products

WHERE price > (SELECT AVG(price) FROM Products);

-- Use AVG() with GROUP BY

-- Here we use the AVG() function and the GROUP BY clause, to return the average price for each category in the Products table:

SELECT AVG(Price) AS AveragePrice, CategoryID

FROM Products

GROUP BY CategoryID;

-----------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------

-- SQL LIKE Operator

-----------------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------------

/\*

SQL LIKE Operator

The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

The percent sign % represents zero, one, or multiple characters

The underscore sign \_ represents one, single character

You will learn more about wildcards in the next chapter.\*/

-- Select all customers that starts with the letter "a":

SELECT \* FROM Customers

WHERE CustomerName LIKE '\_\_\_a%';

SELECT \* FROM Customers

WHERE CustomerName LIKE '%a';

-- Syntax

-- SELECT column1, column2, ...

-- FROM table\_name

-- WHERE columnN LIKE pattern;

-- It can be any character or number, but each \_ represents one, and only one, character.

-- Return all customers from a city that starts with 'L' followed by one wildcard character, then 'nd' and then two wildcard characters:

SELECT \* FROM Customers

WHERE city LIKE 'L\_nd\_\_';

-----------------------------------------------------------------------------------------------------------------------------

-- The % Wildcard

-----------------------------------------------------------------------------------------------------------------------------

-- The % wildcard represents any number of characters, even zero characters.

-- Return all customers from a city that contains the letter 'L':

SELECT \* FROM Customers

WHERE city LIKE '%L%';

-- Starts With

-- To return records that starts with a specific letter or phrase, add the % at the end of the letter or phrase.

-- Return all customers that starts with 'La':

SELECT \* FROM Customers

WHERE CustomerName LIKE 'La%';

-- Tip: You can also combine any number of conditions using AND or OR operators.

-- Return all customers that starts with 'a' or starts with 'b':

SELECT \* FROM Customers

WHERE CustomerName LIKE 'A%' OR CustomerName LIKE 'B%';

-- To return records that ends with a specific letter or phrase, add the % at the beginning of the letter or phrase.

-- Return all customers that ends with 'a':

SELECT \* FROM Customers

WHERE CustomerName LIKE '%a';

-- Tip: You can also combine "starts with" and "ends with":

-- Return all customers that starts with "b" and ends with "s":

SELECT \* FROM Customers

WHERE CustomerName LIKE 'B%s';

-- Contains

-- To return records that contains a specific letter or phrase, add the % both before and after the letter or phrase.

-- Return all customers that contains the phrase 'or'

SELECT \* FROM Customers

WHERE CustomerName LIKE '%or%';

-- Combine Wildcards

-- Any wildcard, like % and \_ , can be used in combination with other wildcards.

-- Return all customers that starts with "a" and are at least 3 characters in length:

SELECT \* FROM Customers

WHERE CustomerName LIKE 'A\_\_%';

-- Return all customers that have "r" in the second position:

SELECT \* FROM Customers

WHERE CustomerName LIKE '\_r%';

-- Without Wildcard

-- If no wildcard is specified, the phrase has to have an exact match to return a result.

-- Return all customers from Spain:

SELECT \* FROM Customers

WHERE Country LIKE 'Spain';

-----------------------------------------------------------------------------------------------------------------------------

-- SQL Wildcards

-----------------------------------------------------------------------------------------------------------------------------

-- SQL Wildcard Characters

-- A wildcard character is used to substitute one or more characters in a string.

-- Wildcard characters are used with the LIKE operator.

-- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

-- Return all customers that starts with the letter 'a':

SELECT \* FROM Customers

WHERE CustomerName REGEXP 'a%';

/\*

Wildcard Characters

Symbol Description

% Represents zero or more characters

\_ Represents a single character

[] Represents any single character within the brackets \*

^ Represents any character not in the brackets \*

- Represents any single character within the specified range \*

{} Represents any escaped character \*\*

\* Not supported in PostgreSQL and MySQL databases.

\*\* Supported only in Oracle databases.

\*/

-- Using the % Wildcard

-- The % wildcard represents any number of characters, even zero characters.

-- Return all customers that ends with the pattern 'es':

SELECT \* FROM Customers

WHERE CustomerName LIKE '%es';

-- Return all customers that contains the pattern 'mer':

SELECT \* FROM Customers

WHERE CustomerName LIKE '%mer%';

/\*Using the \_ Wildcard

The \_ wildcard represents a single character.

It can be any character or number, but each \_ represents one, and only one, character.

Return all customers with a City starting with any character, followed by "ondon":\*/

SELECT \* FROM Customers

WHERE City LIKE '\_ondon';

-- Return all customers with a City starting with "L", followed by any 3 characters, ending with "on":

SELECT \* FROM Customers

WHERE City LIKE 'L\_\_\_on';

-- Using the [] Wildcard

-- The [] wildcard returns a result if any of the characters inside gets a match.

-- Return all customers starting with either "b", "s", or "p":

SELECT \* FROM Customers

WHERE CustomerName LIKE 'bsp%';

-- Using the - Wildcard

-- The - wildcard allows you to specify a range of characters inside the [] wildcard.

-- Return all customers starting with "a", "b", "c", "d", "e" or "f":

SELECT \* FROM Customers

WHERE CustomerName LIKE '[A-F]%';

-- Combine Wildcards

-- Any wildcard, like % and \_ , can be used in combination with other wildcards.

-- Return all customers that starts with "a" and are at least 3 characters in length:

SELECT \* FROM Customers

WHERE CustomerName LIKE 'a\_\_%';

-- Return all customers that have "r" in the second position:

SELECT \* FROM Customers

WHERE CustomerName LIKE '\_r%';

-- Without Wildcard

-- If no wildcard is specified, the phrase has to have an exact match to return a result.

-- Return all customers from Spain:

SELECT \* FROM Customers

WHERE Country LIKE 'Spain';

-- The % wildcard represents any number of characters, even zero characters.

-- Return all customers from a city that contains the letter 'L':

SELECT \* FROM Customers

WHERE city LIKE '%L%';

-- Starts With

-- To return records that starts with a specific letter or phrase, add the % at the end of the letter or phrase.

-- Return all customers that starts with 'La':

SELECT \* FROM Customers

WHERE CustomerName LIKE 'La%';

-- Tip: You can also combine any number of conditions using AND or OR operators.

-- Return all customers that starts with 'a' or starts with 'b':

SELECT \* FROM Customers

WHERE CustomerName LIKE 'A%' OR CustomerName LIKE 'B%';

-- To return records that ends with a specific letter or phrase, add the % at the beginning of the letter or phrase.

-- Return all customers that ends with 'a':

SELECT \* FROM Customers

WHERE CustomerName LIKE '%a';

-- Tip: You can also combine "starts with" and "ends with":

-- Return all customers that starts with "b" and ends with "s":

SELECT \* FROM Customers

WHERE CustomerName LIKE 'B%s';

-- Contains

-- To return records that contains a specific letter or phrase, add the % both before and after the letter or phrase.

-- Return all customers that contains the phrase 'or'

SELECT \* FROM Customers

WHERE CustomerName LIKE '%or%';

-- Combine Wildcards

-- Any wildcard, like % and \_ , can be used in combination with other wildcards.

-- Return all customers that starts with "a" and are at least 3 characters in length:

SELECT \* FROM Customers

WHERE CustomerName LIKE 'A\_\_%';

-- Return all customers that have "r" in the second position:

SELECT \* FROM Customers

WHERE CustomerName LIKE '\_r%';

-- Without Wildcard

-- If no wildcard is specified, the phrase has to have an exact match to return a result.

-- Return all customers from Spain:

SELECT \* FROM Customers

WHERE Country LIKE 'Spain';