**SELECT**: Used to get all the records present in a table.



\* 🡪 Specifies we want to see all the columns data present in table.

We can also go with specific column data. Like **SELECT *name* from *cats*;**

Also **SELECT *name*, *age* from *cats*;**

**WHERE**

The WHERE clause is used to extract only those records that fulfill a specified criterion.

SELECT <column\_name> from <table\_name> WHERE <column\_Name> operator value;

SELECT \* FROM cats WHERE age=4;

SELECT \* FROM cats WHERE name='Egg';

**ALIAS**

You can give a table or a column another name by using an alias. This can be a good thing to

do if you have very long or complex table names or column names.

An alias name could be anything, but usually it is short.

SELECT column\_name AS alias\_name FROM table\_name;

**UPDATE**

The UPDATE statement is used to modify the existing records in a table.

UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition;

**DELETE**

The DELETE statement is used to delete existing records in a table.

DELETE FROM table\_name WHERE condition;

**DISTINCT**

The **SELECT DISTINCT** statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT column1, column2, ... FROM table\_name;

**ORDER BY**

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ***ascending order by default***. To sort the records in descending order, use the DESC keyword.

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

**LIMIT**

The LIMIT clause is used to specify the number of records to return. The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

SELECT column\_name(s) FROM table\_name WHERE condition LIMIT number;

NOTE: LIMIT is supported by only MYSQL.

**LIKE**

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

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

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() Syntax

SELECT MIN(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

MAX() Syntax

SELECT MAX(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

The SQL COUNT(), AVG() and SUM() Functions

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

SELECT COUNT(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

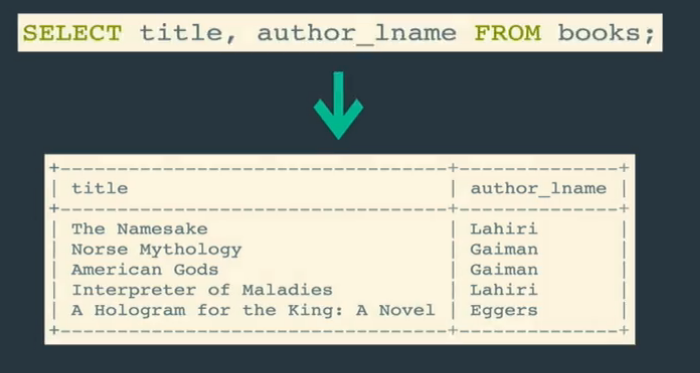
The AVG() function returns the average value of a numeric column.

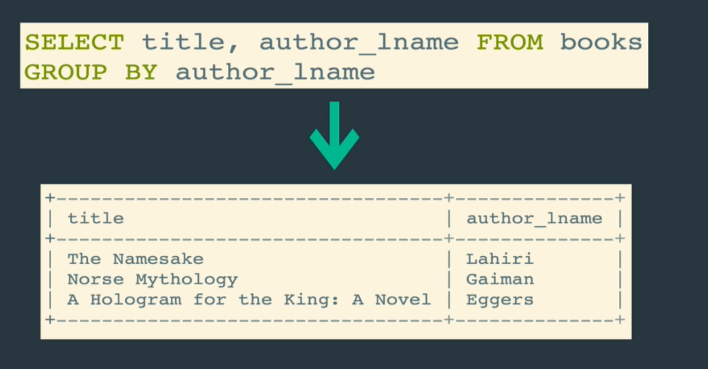
SELECT AVG(column\_name)  
FROM table\_name  
WHERE condition;

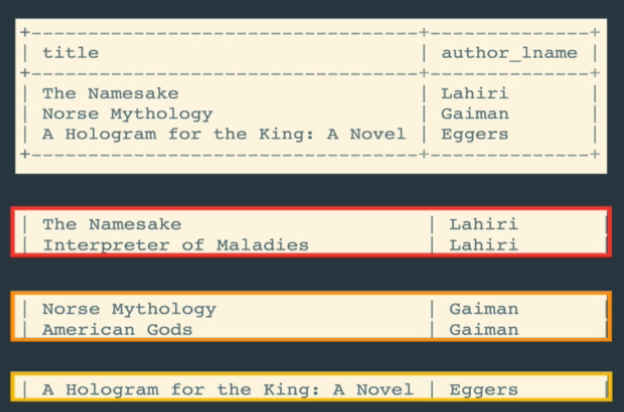
The SUM() function returns the total sum of a numeric column.

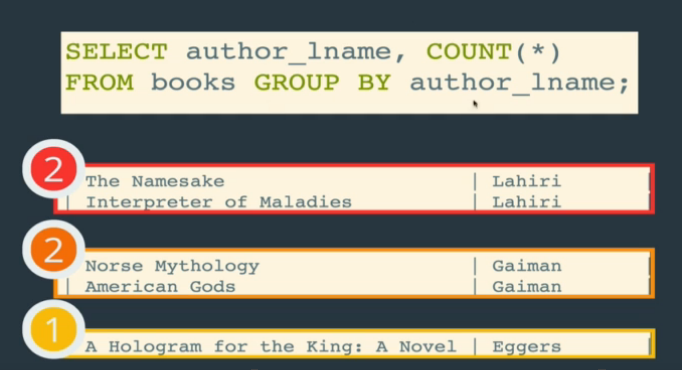
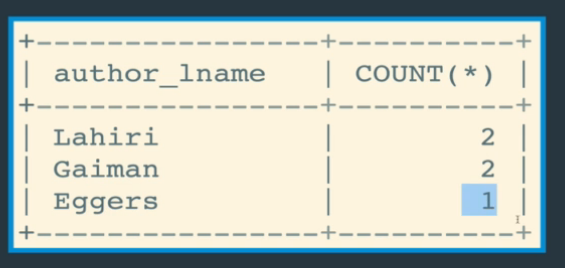
SELECT SUM(column\_name)  
FROM table\_name  
WHERE condition;

**GROUP BY**





****



The SQL AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

AND

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

OR

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

NOT

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

GREATER THAN

SELECT title, stock\_quantity FROM books WHERE stock\_quantity >= 100;

LESS THAN

SELECT title, released\_year FROM books WHERE released\_year <= 2000;

The SQL BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

The SQL IN Operator

The IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name* IN (*SELECT STATEMENT*);

SELECT \* FROM Customers WHERE Country NOT IN ('Germany', 'France', 'UK');

The SQL CASE Expression

The CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

If there is no ELSE part and no conditions are true, it returns NULL.

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

SELECT OrderID, Quantity,  
CASE  
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'  
    WHEN Quantity = 30 THEN 'The quantity is 30'  
    ELSE 'The quantity is under 30'  
END AS QuantityText  
FROM OrderDetails;