|  |  |
| --- | --- |
| For grouping along with counting | SELECT owner, COUNT(\*) FROM table GROUP BY owner; |
| For selecting maximum values | SELECT owner, max(\*) FROM table GROUP BY owner; |

Operators in The WHERE Clause

The following operators can be used in the WHERE clause:

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| <> | Not equal. **Note:** In some versions of SQL this operator may be written as != |
| BETWEEN | Between a certain range |
| LIKE | Search for a pattern |
| IN | To specify multiple possible values for a column |

## ORDER BY Several Columns Example

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName:

### **Example**

SELECT \* FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

## The SQL AND Operator

The WHERE clause can contain one or many AND operators.

The AND operator is used to filter records based on more than one condition, like if you want to return all customers from Spain that starts with the letter 'G':

### **Example**

Select all customers from Spain that starts with the letter 'G':

SELECT \*  
FROM Customers  
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';

SELECT \* FROM Customers  
WHERE Country = 'Germany'  
AND City = 'Berlin'  
AND PostalCode > 1200;

SELECT \* FROM Customers  
WHERE Country = 'Spain'

AND (CustomerName LIKE 'G%' OR CustomerName LIKE 'R%');

## The NOT Operator

The NOT operator is used in combination with other operators to give the opposite result, also called the negative result.

In the select statement below we want to return all customers that are NOT from Spain:

### **Example**

Select only the customers that are NOT from Spain:

SELECT \* FROM Customers  
WHERE NOT Country = 'Spain';

## NOT LIKE

### **Example**

Select customers that does not start with the letter 'A':

SELECT \* FROM Customers  
WHERE CustomerName NOT LIKE 'A%';

se

## NOT BETWEEN

### **Example**

Select customers with a customerID not between 10 and 60:

SELECT \* FROM Customers  
WHERE CustomerID NOT BETWEEN 10 AND 60;

## NOT IN

### **Example**

Select customers that are not from Paris or London:

SELECT \* FROM Customers  
WHERE City NOT IN ('Paris', 'London');

## Step 1: Create a Database

## CREATE DATABASE my\_database;

## Step 2: Use the Database

## Before creating tables, you need to select the database.

## USE my\_database;

## Step 3: Create a Table

## Now, you can create a table within the database. Below is an example of creating a users table with some common columns.

## CREATE TABLE users (

## id INT AUTO\_INCREMENT PRIMARY KEY,

## username VARCHAR(50) NOT NULL,

## City VARCHAR(50) NOT NULL,

## password VARCHAR(255) NOT NULL,

## created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

## );

## Explanation:

## id: An integer that automatically increments with each new record.

## username: A string with a maximum of 50 characters, cannot be null.

## email: A string with a maximum of 100 characters, cannot be null.

## password: A string with a maximum of 255 characters, cannot be null.

## created\_at: A timestamp that defaults to the current date and time when the record is created.

## INSERT INTO Example

The INSERT INTO statement is used to insert new records in a table.

The following SQL statement inserts a new record in the "Customers" table:

### **Example**

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

insert into customers (customer\_id ,customer\_fname , customer\_lname )values (1000521 , 'mahmoud' ,'khorshed' );

SELECT \* FROM customers WHERE Customer\_ID=1000521;

## UPDATE Table

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person and a new city.

### **Example**

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

## The SQL DELETE Statement

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

### **DELETE Syntax**

DELETE FROM table\_name WHERE condition;

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM Customers;

## Delete a Table

To delete the table completely, use the DROP TABLE statement:

### **Example**

Remove the Customers table:

DROP TABLE Customers;

## 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 Example**

Find the lowest price:

SELECT MIN(Price)  
FROM Products;

### **MAX Example**

Find the highest price:

SELECT MAX(Price)  
FROM Products;

## Set Column Name (Alias)

When you use MIN() or MAX(), the returned column will be named MIN(field) or MAX(field) by default. To give the column a new name, use the AS keyword:

### **Example**

SELECT MIN(Price) AS SmallestPrice  
FROM Products;

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

### **Example**

Name the column "number of records":

SELECT COUNT(\*) AS [number of records]  
FROM Products;

### **Example**

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

SELECT COUNT(ProductID)  
FROM Products  
WHERE Price > 20;

### **Example**

Insert a NULL value to better understand the example above:

UPDATE Products  
SET ProductName = NULL  
WHERE ProductName = 'Chang';

**Sum Example**

Return the number of orders made for the product with ProductID 11:

SELECT SUM(Quantity) as total  
FROM OrderDetails  
WHERE ProdictId = 11;

## 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:

### **Example**

Use an expression inside the SUM() parenthesis:

SELECT SUM(Quantity \* 10)  
FROM OrderDetails;

select order\_item\_product\_id , sum(order\_item\_product\_price ) from order\_items group by order\_item\_product\_id order by sum(order\_item\_product\_price) desc;

## The SQL AVG() Function

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

### **Example**

Find the average price of all products:

SELECT AVG(Price) as average\_price  
FROM Products  
WHERE CategoryID = 1;

## Higher Than Average

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

### **Example**

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

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

select order\_item\_product\_id , order\_item\_product\_price from order\_items where order\_item\_product\_price > (select AVG(order\_item\_product\_price) from order\_items)

The SQL LIKE Operator

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

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

|  |  |
| --- | --- |
| **LIKE Operator** | **Description** |
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

The following SQL statement selects all customers with a CustomerName starting with "a":

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a%';

## Using the [charlist] Wildcard

The following SQL statement selects all customers with a City starting with "b", "s", or "p":

### **Example**

SELECT \* FROM Customers  
WHERE City LIKE '[bsp]%';

## 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 \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');

The following SQL statement selects all customers that are from the same countries as the suppliers:

### **Example**

SELECT \* FROM Customers  
WHERE Country IN (SELECT Country FROM Suppliers);