What is SQL?

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views.

## RDBMS

* RDBMS stands for Relational Database Management System.
* The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.
* Every table is broken up into smaller entities called fields. A field is a column in a table that is designed to maintain specific information about every record in the table.
* A record, also called a row, is each individual entry that exists in a table.
* A record is a horizontal entity in a table. A column is a vertical entity in a table that contains all information associated with a specific field in a table.

## Data Types

**Common examples of data types,**

* [Boolean](https://www.computerhope.com/jargon/b/boolean.htm) (e.g., True or False)
* [Character](https://www.computerhope.com/jargon/c/charact.htm) (e.g., a)
* NVARCHAR :- it’s a letter and an alphabet together.
* Date:- Datetime the format is (e.g., 03/01/2016)
* [Integer](https://www.computerhope.com/jargon/i/integer.htm) (e.g., 1234)
* [Void](https://www.computerhope.com/jargon/v/void.htm) (e.g., no data)
* Money :- price of the item.

## SQL Statements

## Some of The Most Important SQL Commands

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database.

Most of the actions you need to perform on a database are done with SQL statements.

The following SQL statement selects all the records in the "Persons" table:

## The SQL SELECT Statement

The SELECT statement is used to select data from a database.The data returned is stored in a result table, called the result-set.

## SELECT Syntax

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

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table,

Example SELECT FirstName, LastName

FROM Persons;

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT \* FROM PERSONS;

## SELECT Column Example

The following SQL statement selects the "LastName" and "BirthDate" columns from the "Persons" table:

EXAMPLE:- SELECT LastName, BirthDate FROM Customers;

## The SQL SELECT DISTINCT Statement

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

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

Example SELECT DISTINCT LastName, IsActive, ...  
FROM Persons;

THE COUNT SYNTAX

This count syntax is used to list the number of data quired. In simple words it will give the no of data listed.

The following SQL statement lists the number of different (distinct) customer countries:

EXAMPLE SELECT COUNT(DISTINCT LastName)

 FROM Persons;

## The SQL WHERE Clause

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Example SELECT LastName, FirstName,...  
 FROM Persons  
 WHERE LastName=’Hailu’;

## 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 Syntax**

SELECT \*  
FROM Adresses  
WHERE City=‘ Fairfax’ AND State=’VA’ ;

**OR Syntax**

SELECT \*  
FROM Adresses  
WHERE City= ‘ Fairfax’  OR City=’Alexandria’

**NOT Syntax**

SELECT \*  
FROM Persons  
WHERE NOT lastName=’lucas’;

**Combining AND, OR and NOT**

* You can also combine the AND, OR and NOT operators.
* The following SQL statement selects all fields from "Adresses" where State is "VA" AND city must be "FairFax OR "Alexandria" (use parenthesis to form complex expressions):

Example SELECT \* FROM Adresses  
WHERE State='VA' AND (City='FairFax' OR City='Alexandria');

The following SQL statement selects all fields from "Adresses" where City is NOT "Fairfax" and NOT "Alexandria":

ExampleSELECT \* FROM Adresses  
WHERE NOT City='FairFax' AND NOT City='Alexandria';

**The SQL ORDER BY Keyword**

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.

Example The following SQL statements selects all from persons table sorted by LastName and BirthDate.

Example:- SELECT \* FROM Persons  
ORDER BY LastName, BirthDate DESC;

**The SQL INSERT INTO Statement**

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

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

INSERT INTO Persons (PersonID, FirstName, LastName, BirthDate, IsActive)  
VALUES ('1', 'Abeniet', 'Hailu', '18/4/1973', 'False');

**What is a NULL Value?**

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

**IS NULL Syntax**

SELECT LastName,FirstName,BirthDateFROM PERSONS  
WHERE BirthDate IS NULL;

**IS NOT NULL Syntax**

SELECT State, City, ZipCodeFROM Addresses  
WHERE ZipCode IS NOT NULL;

**The SQL UPDATE Statement**

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

EXAMPLE The following SQL statement updates the eighth Person (PersonID = 8) with a new FirstName *and* LastName.

UPDATE Persons  
SET FirstName = 'Etete', LastName= 'Birru'  
WHERE PersonID = 8;

**The SQL DELETE Statement**

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

Example:- The following SQL statement deletes the crew "Nebiyou" from the "Crews" table.

DELETE FROM Crews WHERE FirstName='Nebiyou';

**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.

Example of the **MIN SYNTAX**

Use the MIN function to select the record with the smallest value of the Price column.

SELECT MIN (Price)

FROM Products ;

Example of MAX Syntax

Use the MAX function to select the record with the smallest value of the Price column.

SELECT MAX (Price)

FROM Products ;

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

### **COUNT() Syntax**

SELECT COUNT(ProductID)  
FROM Products;

**AVG() Syntax**

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

SELECT AVG(Price)  
FROM Products;

**SUM() Syntax**

SELECT SUM(Quantity)  
FROM OrderDetails;

**The SQL LIKE Operator**

|  |  |
| --- | --- |
| **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 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.

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

Examples of the **LIKE SYNTAX**

The following SQL statement selects all Products with a Products Name starting with "a"

SELECT \*

FROM Products  
WHERE Products Name LIKE 'a%';

**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](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| % | Represents zero or more characters | bl% finds bl, black, blue, and blob |
| \_ | Represents a single character | h\_t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |
| ^ | Represents any character not in the brackets | h[^oa]t finds hit, but not hot and hat |
| - | Represents any single character within the specified range | c[a-b]t finds cat and cbt |

All the wildcards can also be used in combinations!

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |  |
| --- | --- |
| LIKE Operator | Description |
| WHERE CustomerName LIKE 'a%' | Finds any values that starts with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that ends 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 starts with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that starts with "a" and ends with "o" |

## 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.

Example of **IN Syntax**

## 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.

Example:-The following SQL statement selects all Addresses that are located in "Fairfax", or "Alexandria"

SELECT \* FROM Addresses  
WHERE City IN ('Fairfax', 'Alexandria');

Example:- The following SQL statement selects all Adresses that are NOT located in "Fairfax" or "Alexandria":

SELECT \* FROM Addresses  
WHERE City NOT IN ('Fairfax','Alexandria');

**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.

**BETWEEN Syntax**

The following SQL statement selects all products with a price between 40 and 120

Example

SELECT \* FROM Products  
WHERE Price BETWEEN 40 AND 120;

**NOT BETWEEN Example**

To display the products outside the range of the previous example, use NOT BETWEEN:

Example

SELECT \* FROM Products  
WHERE Price NOT BETWEEN 40 AND 120;

**BETWEEN with IN Example**

The following SQL statement selects all products with a price between 10 and 20. In addition; do not show products with a ProductID of 1,2, or 3:

Example

SELECT \* FROM Products  
WHERE Price BETWEEN 40 AND 120  
AND ProductID NOT IN (1,2,3)

**BETWEEN Text Values Example**

The following SQL statement selects all products with a ProductName between Bread and Soft Cake:

Example

SELECT \* FROM Products  
WHERE ProductName BETWEEN 'Bread' AND 'Soft Cake'  
ORDER BY ProductName;

The following SQL statement selects all products with a ProductName between Cookies and Cream Cake:

Example

SELECT \* FROM Products  
WHERE ProductName BETWEEN "Cookies" AND "Cream Cake"  
ORDER BY ProductName;

**NOT BETWEEN Text Values Example**

The following SQL statement selects all products with a ProductName not between Bread and Soft Cake:

Example

SELECT \* FROM Products  
WHERE ProductName NOT BETWEEN 'Bread' AND 'Soft Cake'  
ORDER BY ProductName;

**SQL Aliases**

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

**Alias Column Syntax**

The following SQL statement creates two aliases, one for the PersonID column and one for the FirstName column:

Example

SELECT PersonID AS ID, FirstName AS Fn  
FROM Persons;

**Alias Table Syntax**

The following SQL statement creates two aliases, one for the AddressesID column and one for the ZipCode column:

SELECT AddressesID AS ID, ZipCode AS Zc  
FROM Addresses;