**SQL**

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

1.Tables

-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.

-Column names are field and row names are record.

-fields +records=enrtities.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **CustomerName** | **ContactName** | **Address** | **City** | **PostalCode** | **Country** |
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |
| 4 | Around the Horn | Thomas Hardy | 120 Hanover Sq. | London | WA1 1DP | UK |

2. DBMS DATABASE OBJECTS

-A **database object** is any defined object in a database that is used to store or reference data.

-Anything which we make from **create command**is known as Database Object

-It can be used to hold and manipulate the data.

Some of the examples of database objects are :

* **Table –** Basic unit of storage; composed rows and columns
* **View –** Logically represents subsets of data from one or more tables
* **Sequence –** Generates primary key values
* **Index –** Improves the performance of some queries
* **Synonym –** Alternative name for an object

## **3.** Some of The Most Important SQL Commands are:

* **SELECT** - extracts data from a database

-The data returned is stored in a result table, called the result-set.

-Eg. SELECT CustomerName, City FROM Customers;

SELECT \* FROM Customers;

SELECT DISTINCT Country FROM Customers;

SELECT COUNT(DISTINCT Country) FROM Customers;

select count(\*) as distcountries from (select distinct country from customers)

* WHERE -The WHERE clause is used in SELECT, UPDATE, DELETE statement, etc.!

-SELECT \* FROM Customers

WHERE Country='Mexico'; or “Maxico”

-SELECT \* FROM Customers

WHERE CustomerID=1;

## **Operators in The WHERE Clause**

The following operators can be used in the WHERE clause:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_equal_to) |
| > | Greater than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than) |
| < | Less than | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than) |
| >= | Greater than or equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_greater_than2) |
| <= | Less than or equal | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_less_than2) |
| <> | Not equal. **Note:** In some versions of SQL this operator may be written as != | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_not_equal_to) |
|  |  |  |
|  |  |  |
| BETWEEN | Between a certain range | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_between) |
| LIKE | Search for a pattern | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_like) |
| IN | To specify multiple possible values for a column | [Try it](https://www.w3schools.com/sql/trysql.asp?filename=trysql_op_in) |

Eg.- SELECT \* FROM Products

WHERE Price = 18;

SELECT \* FROM PRODUCTS

WHERE PRICE >30

SELECT \* FROM Products

WHERE Price < 30;

SELECT \* FROM Products

WHERE Price >= 30;

SELECT \* FROM Products

WHERE Price <= 30;

SELECT \* FROM Products

WHERE Price != 18; OR <> 18

SELECT \* FROM Products

WHERE Price BETWEEN 50 AND 60;

SELECT \* FROM Customers

WHERE City LIKE 'San%'; // CITY NAME STARTS WITH

San

SELECT \* FROM Customers

WHERE City IN ('Paris','London');

// CITY IS EITHER PARIS OR LONDON

SELECT \* FROM CUSTOMERS

WHERE NOT CITY = ‘BERLIN’; //every city other than berlin

SELECT \* FROM Customers

WHERE Country='Germany' AND City='Berlin';

SELECT \* FROM Customers

WHERE City='Berlin' OR City='München';

SELECT \* FROM Customers

WHERE NOT Country='Germany';

SELECT \* FROM Customers

WHERE Country='Germany' AND (City='Berlin' OR City='München');

SELECT \* FROM Customers

WHERE NOT Country='Germany' AND NOT Country='USA';

//neither Germany nor usa

## **-ORDER BY Keyword**

The ORDER BY keyword sorts the records in ascending order by default.

To sort the records in descending order, use the DESC keyword.

- SELECT \* FROM Customers

ORDER BY Country; //ascending order of country acc to dictionary

- SELECT \* FROM Customers

ORDER BY Country DESC;

- SELECT \* FROM Customers

ORDER BY Country, CustomerName; //1ST acc to country then by

customer name both asc

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

* **UPDATE** - updates data in a database

- Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

* + - UPDATE Customers

SET ContactName='sUMIT KUMAR', City='rANCHI'

WHERE CustomerID=1;

* + - UPDATE Customers  
      SET ContactName='Juan'  
      WHERE Country='Mexico';
    - DELETE FROM Customers;

* **DELETE** - deletes data from a database

-If you omit the WHERE clause, all records in the table will be deleted!

* + - DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste'; // The following SQL statement deletes all

rows in the "Customers" table, without deleting the table

* **INSERT INTO** - inserts new data into a database

- INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES ('Cardinal','Tom B. Erichsen','Skagen 21','Stavanger','4006','Norway');

- INSERT INTO Customers (CustomerName, City, Country)  
VALUES ('Cardinal', 'Stavanger', 'Norway');

* **CREATE** **DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

**IS NULL:**

**- 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!

- SELECT CustomerName, ContactName, Address

FROM Customers

WHERE Address IS NULL;

**IS NOT NULL:**

SELECT CustomerName, ContactName, Address

FROM Customers

WHERE Address IS NOT NULL;

**SQL SELECT TOP:**

- The SELECT TOP clause is used to specify the number of

records to return.

- The SELECT TOP clause is useful on large tables with

thousands of records. Returning a large number of records

can impact on performance.

- Not all database systems support the SELECT TOP clause.

MySQL supports the LIMIT clause to select a limited number of records,

while Oracle uses ROWNUM.

- The following SQL statement selects the first three records from the "Customers" table:

1. SELECT TOP 3 \* FROM Customers;

2. SELECT \* FROM Customers  
 LIMIT 3;

3. SELECT \* FROM Customers  
 WHERE ROWNUM <= 3;

## **SQL TOP PERCENT :**

- The following SQL statement selects the first 50% of the records from the "Customers" table:

1. SELECT TOP 50 PERCENT \* FROM Customers;

ADD WHERE CLAUSE:

1. SELECT TOP 3 \* FROM Customers  
 WHERE Country='Germany';

2. SELECT \* FROM Customers  
 WHERE Country='Germany'  
 LIMIT 3;

3. SELECT \* FROM Customers  
 WHERE Country='Germany' AND ROWNUM <= 3;

# 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.

- SELECT MIN(Price) AS SmallestPrice

FROM Products;

Number of Records: 1

|  |
| --- |
| **SmallestPrice** |
| 2.5 |

# - SELECT MAX(Price) AS LargestPrice FROM Products;

## **SQL COUNT(), AVG() and SUM() Functions:**

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

- NULL values are not counted.

* SELECT COUNT(ProductID)

FROM Products; //give no. of products

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

-  NULL values are ignored.

* SELECT AVG(Price)

FROM Products;

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

- NULL values are ignored.

* SELECT SUM(Quantity)  
  FROM OrderDetails; //  finds the sum of the

"Quantity" fields in the

"OrderDetails" table

## **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 represents a single character
* Sql MS ACCESS

% \*

\_ ?

- Here are some examples showing different LIKE operators with

'%' and '\_' wildcards:

|  |  |
| --- | --- |
| **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 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

## SQL LIKE Examples:

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

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

2. The following SQL statement selects all customers with a CustomerName ending with "a":

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

3. The following SQL statement selects all customers with a CustomerName that have "or" in any position:

- SELECT \* FROM Customers  
WHERE CustomerName LIKE '%or%';

4. The following SQL statement selects all customers with a CustomerName that have "r" in the second position:

- SELECT \* FROM Customers  
WHERE CustomerName LIKE '\_r%';

5. The following SQL statement selects all customers with a CustomerName that starts with "a" and are at least 3 characters in length:

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

6. The following SQL statement selects all customers with a ContactName that starts with "a" and ends with "o":

- SELECT \* FROM Customers  
WHERE ContactName LIKE 'a%o';

7. The following SQL statement selects all customers with a CustomerName that does NOT start with "a":

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

**SQL Wildcard Characters**

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

Wildcard characters are used with the [SQL 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.

### **Wildcard Characters in MS Access**

|  |  |  |
| --- | --- | --- |
| **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 a range of characters | c[a-b]t finds cat and cbt |
| # | Represents any single numeric character | 2#5 finds 205, 215, 225, 235, 245, 255, 265, 275, 285, and 295 |

### **Wildcard Characters in SQL Server**

|  |  |  |
| --- | --- | --- |
| **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 a range of characters | c[a-b]t finds cat and cbt |

All the wildcards can also be used in combinations!