# Distinct

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

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

SELECT COUNT(DISTINCT Country) FROM Customers;

# Where

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

SELECT \* FROM Customers  
WHERE Country='Germany' AND (City='Berlin' OR City='München');

# Order By

SELECT \* FROM Customers  
ORDER BY Country;

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

# Insert

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

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

# IS NULL

SELECT LastName, FirstName, Address FROM Persons  
WHERE Address IS NOT NULL;

# UPDATE

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

# DELETE

DELETE FROM Customers  
WHERE CustomerName='Alfreds Futterkiste';

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 table\_name;

# TOP

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact on performance.

SELECT TOP 5 \* FROM Products order by Price desc SELECT TOP 50 PERCENT \* FROM Customers;

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

# Min Max

SELECT MAX(Price) AS LargestPrice  
FROM Products;

Count AVG SUM

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

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

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

SELECT COUNT(ProductID)  
FROM Products;

SELECT AVG(Price)  
FROM Products;

SELECT SUM(Quantity)  
FROM OrderDetails;

# Like

|  |  |
| --- | --- |
| **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" |

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

# Wildcards

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

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

The following SQL statement selects all customers with a City starting with "a", "b", or "c"

SELECT \* FROM Customers  
WHERE City LIKE '[a-c]%';

SELECT \* FROM Customers  
WHERE City NOT LIKE '[bsp]%';

# IN

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

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

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

SELECT Distinct Country as UniqCountry FROM Customers

WHERE Country in (SELECT Country FROM Suppliers);

# BETWEEN

SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20;

SELECT \* FROM Products  
WHERE (Price BETWEEN 10 AND 20)  
AND NOT CategoryID IN (1,2,3);

The following SQL statement selects all products with a ProductName BETWEEN 'Carnarvon Tigers' and 'Mozzarella di Giovanni'

SELECT \* FROM Products  
WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'  
ORDER BY ProductName;

The following SQL statement selects all orders with an OrderDate BETWEEN '04-July-1996' and '09-July-1996'

SELECT \* FROM Orders  
WHERE OrderDate BETWEEN #07/04/1996# AND #07/09/1996#;

# Aliases

The following SQL statement creates two aliases, one for the CustomerName column and one for the ContactName column. **Note:** It requires double quotation marks or square brackets if the alias name contains spaces

SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;

The following SQL statement creates two aliases, one for the CustomerName column and one for the ContactName column. **Note:** It requires double quotation marks or square brackets if the alias name contains spaces

SELECT CustomerName AS Customer, ContactName AS [Contact Person]  
FROM Customers;

The following SQL statement selects all the orders from the customer with CustomerID=4 (Around the Horn). We use the "Customers" and "Orders" tables, and give them the table aliases of "c" and "o" respectively (Here we use aliases to make the SQL shorter)

SELECT o.OrderID, o.OrderDate, c.CustomerName  
FROM Customers AS c, Orders AS o  
WHERE c.CustomerName="Around the Horn" AND c.CustomerID=o.CustomerID;

The following SQL statement is the same as above, but without aliases

SELECT Orders.OrderID, Orders.OrderDate, Customers.CustomerName  
FROM Customers, Orders  
WHERE Customers.CustomerName="Around the Horn" AND Customers.CustomerID=Orders.CustomerID;

# JOIN

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

Here are the different types of the JOINs in SQL:

* **(INNER) JOIN**: Returns records that have matching values in both tables
* **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
* **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
* **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table

SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName  
FROM ((Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)  
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);

# LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;

# FULL OUTTER JOIN

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID  
ORDER BY Customers.CustomerName;

# SELF JOIN

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City   
ORDER BY A.City;

# UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.

* Each SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in each SELECT statement must also be in the same order

SELECT City FROM Customers  
UNION  
SELECT City FROM Suppliers  
ORDER BY City;

SELECT City FROM Customers  
UNION ALL  
SELECT City FROM Suppliers  
ORDER BY City;

SELECT City, Country FROM Customers  
WHERE Country='Germany'  
UNION  
SELECT City, Country FROM Suppliers  
WHERE Country='Germany'  
ORDER BY City;

SELECT 'Customer' As Type, ContactName, City, Country  
FROM Customers  
UNION  
SELECT 'Supplier', ContactName, City, Country  
FROM Suppliers;

# GROUP BY

The following SQL statement lists the number of customers in each country

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;

The following SQL statement lists the number of customers in each country, sorted high to low

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
ORDER BY COUNT(CustomerID) DESC;

SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders  
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID  
GROUP BY ShipperName;

# HAVING

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5;

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country  
HAVING COUNT(CustomerID) > 5  
ORDER BY COUNT(CustomerID) DESC;

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
FROM (Orders  
INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID)  
GROUP BY LastName  
HAVING COUNT(Orders.OrderID) > 10;

# EXIST

SELECT SupplierName  
FROM Suppliers  
WHERE EXISTS (SELECT ProductName FROM Products WHERE SupplierId = Suppliers.supplierId AND Price < 20);

# ANY

he ANY operator returns TRUE if any of the subquery values meet the condition.

The following SQL statement returns TRUE and lists the productnames if it finds ANY records in the OrderDetails table that quantity = 10

SELECT ProductName  
FROM Products  
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);