Contents

SQL keywords are NOT case sensitive

SELECT

```
SELECT DISTINCT column1, column2, ...
FROM table_name;

SELECT COUNT(DISTINCT Country) FROM Customers;

SELECT COUNT(*) AS DistinctCountries
FROM (SELECT DISTINCT Country FROM Customers);

SELECT * FROM Customers
WHERE City IN ('Paris','London');

SELECT * FROM Customers
WHERE City LIKE 's%';
```

```
SELECT * FROM Products
WHERE Price BETWEEN 50 AND 60;
```

Not equal. Note: In some versions of SQL this operator may be written as !=

```
SELECT * FROM Products
WHERE Price <> 18;
```

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT Country='USA';
```

ORDER BY

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

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

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

INSERT INTO

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

NULL

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

```
SELECT ProductName, UnitPrice * (UnitsInStock + IFNULL(UnitsOnOrder, 0))
FROM Products;

SELECT ProductName, UnitPrice * (UnitsInStock + COALESCE(UnitsOnOrder, 0))
FROM Products;
```

UPDATE

If you omit the WHERE clause, ALL records will be updated

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

DELETE

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

```
DELETE FROM table_name WHERE condition;
```

Function

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;

SELECT MAX(column_name)
FROM table_name
WHERE condition;

SELECT COUNT(column_name)
FROM table_name
WHERE condition;

SELECT AVG(column_name)
FROM table_name
WHERE condition;

SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

LIKE

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

The percent sign (%) represents zero, one, or multiple characters
The underscore sign (_) represents one, single character

starts with "a" and are at least 3 characters in length

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a__%';

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

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

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

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

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

IN

```
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');

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

BETWEEN

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

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

between '01-July-1996' and '31-July-1996'

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN #07/01/1996# AND #07/31/1996#;
```

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
```

Aliases

```
SELECT column_name AS alias_name
FROM table_name;
```

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

requires double quotation marks or square brackets if the alias name contains spaces

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

```
SELECT CustomerName, Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS
Address
FROM Customers;
```

MySQL:

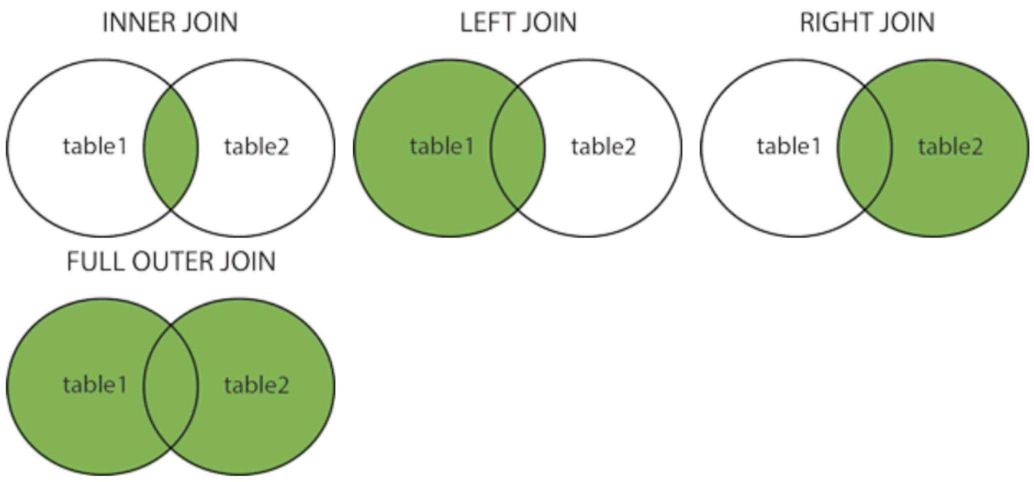
```
SELECT CustomerName, CONCAT(Address,', ',PostalCode,', ',City,', ',Country) AS
Address
FROM Customers;
```

```
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;
```

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;



SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;

JOIN Three Tables
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);

Self Join

SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;

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

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ⍰ distinct UNION ALL ⍰ duplicate

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

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

GROUP BY

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

```
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 column_name(s)
```

```
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);

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
WHERE LastName = 'Davolio' OR LastName = 'Fuller'
GROUP BY LastName
HAVING COUNT(Orders.OrderID) > 25;
```

EXISTS

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);

SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID =
Suppliers.supplierID AND Price < 20);
```

ANY

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
  (SELECT column_name
  FROM table_name
  WHERE condition);

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

ALL

```
SELECT ALL column_name(s)
FROM table_name
WHERE condition;

SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

lists ALL the product names

```
SELECT ALL ProductName
FROM Products
WHERE TRUE;

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

SELECT INTO

```
SELECT *
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

SELECT INTO can also be used to create a new, empty table

```
SELECT * INTO newtable
FROM oldtable
WHERE 1 = 0;
```

INSERT INTO SELECT

The existing records in the target table are unaffected

```
INSERT INTO table2
SELECT * FROM table1
WHERE condition;

INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;

INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers
WHERE Country='Germany';
```

CASE

returns a value when the first condition is met
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;

SELECT CustomerName, City, Country
FROM Customers
ORDER BY
(CASE
   WHEN City IS NULL THEN Country
   ELSE City
END);
```

Stored Procedures

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;

EXEC procedure_name;

CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode
nvarchar(10)
AS
SELECT * FROM Customers WHERE City = @City AND PostalCode = @PostalCode
GO;

EXEC SelectAllCustomers @City = 'London', @PostalCode = 'WA1 1DP';
```

Comments

```
--Select all:
SELECT * FROM Customers;

/*Select all the columns
of all the records
in the Customers table:*/
SELECT * FROM Customers;
```