**Keyword SQL SERVER**

ADD

Adds a column in an existing table

ADD CONSTRAINT

Adds a constraint after a table is already created

ALTER

Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table

ALTER COLUMN

Changes the data type of a column in a table

ALTER TABLE

Adds, deletes, or modifies columns in a table

ALL

Returns true if all of the subquery values meet the condition

AND

Only includes rows where both conditions is true

ANY

Returns true if any of the subquery values meet the condition

AS

Renames a column or table with an alias

ASC

Sorts the result set in ascending order

BACKUP DATABASE

Creates a back up of an existing database

BETWEEN

Selects values within a given range

CASE

Creates different outputs based on conditions

CHECK

A constraint that limits the value that can be placed in a column

COLUMN

Changes the data type of a column or deletes a column in a table

CONSTRAINT

Adds or deletes a constraint

CREATE

Creates a database, index, view, table, or procedure

CREATE DATABASE

Creates a new SQL database

CREATE INDEX

Creates an index on a table (allows duplicate values)

CREATE OR REPLACE VIEW

Updates a view

CREATE TABLE

Creates a new table in the database

CREATE PROCEDURE

Creates a stored procedure

CREATE UNIQUE INDEX

Creates a unique index on a table (no duplicate values)

CREATE VIEW

Creates a view based on the result set of a SELECT statement

DATABASE

Creates or deletes an SQL database

DEFAULT

A constraint that provides a default value for a column

DELETE

Deletes rows from a table

DESC

Sorts the result set in descending order

DISTINCT

Selects only distinct (different) values

DROP

Deletes a column, constraint, database, index, table, or view

DROP COLUMN

Deletes a column in a table

DROP CONSTRAINT

Deletes a UNIQUE, PRIMARY KEY, FOREIGN KEY, or CHECK constraint

DROP DATABASE

Deletes an existing SQL database

DROP DEFAULT

Deletes a DEFAULT constraint

DROP INDEX

Deletes an index in a table

DROP TABLE

Deletes an existing table in the database

DROP VIEW

Deletes a view

EXEC

Executes a stored procedure

EXISTS

Tests for the existence of any record in a subquery

FOREIGN KEY

A constraint that is a key used to link two tables together

FROM

Specifies which table to select or delete data from

FULL OUTER JOIN

Returns all rows when there is a match in either left table or right table

GROUP BY

Groups the result set (used with aggregate functions: COUNT, MAX, MIN, SUM, AVG)

HAVING

Used instead of WHERE with aggregate functions

IN

Allows you to specify multiple values in a WHERE clause

INDEX

Creates or deletes an index in a table

INNER JOIN

Returns rows that have matching values in both tables

INSERT INTO

Inserts new rows in a table

INSERT INTO SELECT

Copies data from one table into another table

IS NULL

Tests for empty values

IS NOT NULL

Tests for non-empty values

JOIN

Joins tables

LEFT JOIN

Returns all rows from the left table, and the matching rows from the right table

LIKE

Searches for a specified pattern in a column

LIMIT

Specifies the number of records to return in the result set

NOT

Only includes rows where a condition is not true

NOT NULL

A constraint that enforces a column to not accept NULL values

OR

Includes rows where either condition is true

ORDER BY

Sorts the result set in ascending or descending order

OUTER JOIN

Returns all rows when there is a match in either left table or right table

PRIMARY KEY

A constraint that uniquely identifies each record in a database table

PROCEDURE

A stored procedure

RIGHT JOIN

Returns all rows from the right table, and the matching rows from the left table

ROWNUM

Specifies the number of records to return in the result set

SELECT

Selects data from a database

SELECT DISTINCT

Selects only distinct (different) values

SELECT INTO

Copies data from one table into a new table

SELECT TOP

Specifies the number of records to return in the result set

SET

Specifies which columns and values that should be updated in a table

TABLE

Creates a table, or adds, deletes, or modifies columns in a table, or deletes a table or data inside a table

TOP

Specifies the number of records to return in the result set

TRUNCATE TABLE

Deletes the data inside a table, but not the table itself

UNION

Combines the result set of two or more SELECT statements (only distinct values)

UNION ALL

Combines the result set of two or more SELECT statements (allows duplicate values)

UNIQUE

A constraint that ensures that all values in a column are unique

UPDATE

Updates existing rows in a table

VALUES

Specifies the values of an INSERT INTO statement

VIEW

Creates, updates, or deletes a view

WHERE

Filters a result set to include only records that fulfill a specified condition

SQL

Select distinct

IN ( 1, 2, 3 …. Option al posto di or ) --- NOT IN

Select CONCAT ( name, cognomen, datana , … ) from … 🡪 unica stringa output

Select concat( ) AS nuova 🡪 out in una nuova Colonna

Operation

UPPER – LOWER SQRT AVG SUM SubQuery ASC DECR

SELECT FirstName, Salary FROM employees

WHERE  Salary > (SELECT AVG(Salary) FROM employees)

ORDER BY Salary DESC;

The **LIKE**keyword is useful when specifying a **search condition** within your WHERE clause.

SELECT column\_name(s)   
FROM table\_name   
WHERE column\_name **LIKE pattern**;

SQL **pattern**matching enables you to use "\_" to match any single character and "%" to match an arbitrary number of characters (including zero characters).

Where name LIKE “a%” name che iniziano con a LIKE ‘%o’ name termina con o

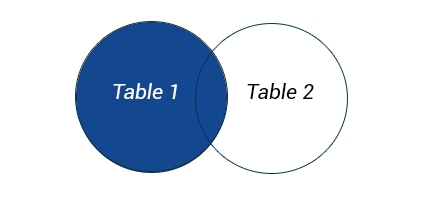
# Joining Tables

The following are the types of JOIN that can be used in MySQL:  
**- INNER** JOIN  
**- LEFT** JOIN  
**- RIGHT** JOIN  
  
INNER JOIN is equivalent to JOIN. It returns rows when there is a match between the tables.

SELECT column\_name(s)   
FROM table1 **INNER JOIN** table2   
**ON** table1.column\_name=table2.column\_name;

The **LEFT JOIN** returns all rows from the left table, even if there are no matches in the right table.  
  
This means that if there are no matches for the **ON**clause in the table on the right, the join will still return the rows from the first table in the result.

SELECT table1.column1, table2.column2...   
FROM table1 **LEFT OUTER JOIN** table2   
**ON** table1.column\_name = table2.column\_name;



The **RIGHT JOIN** returns all rows from the right table, even if there are no matches in the left table.

SELECT table1.column1, table2.column2...   
FROM table1 **RIGHT OUTER JOIN** table2   
ON table1.column\_name = table2.column\_name;

Occasionally, you might need to combine data from multiple tables into one comprehensive dataset. This may be for tables with similar data within the same database or maybe there is a need to combine similar data across databases or even across servers.  
  
To accomplish this, use the **UNION**and **UNION ALL** operators.  
  
**UNION**combines multiple datasets into a single dataset, and removes any existing duplicates.  
**UNION ALL** combines multiple datasets into one dataset, but does not remove duplicate rows.

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

All SELECT statements within the UNION must have the **same number of columns**. The columns must also have the same**data types**. Also, the columns in each SELECT statement must be in the same order.  
**The syntax of UNION is as follows:**

SELECT column\_name(s) FROM table1   
**UNION**  
SELECT column\_name(s) FROM table2;

# Inserting Data

SQL tables store data in rows, one row after another. The **INSERT INTO** statement is used to add **new rows** of data to a table in the database.  
The SQL **INSERT INTO** syntax is as follows:

**INSERT INTO** table\_name   
**VALUES** (value1, value2, value3,...);

Oppure

INSERT INTO table\_name **(column1, column2, column3, ...,columnN)**   
VALUES (value1, value2, value3,...valueN);

# Updating Data

The **UPDATE**statement allows us to alter data in the table.  
  
The basic syntax of an **UPDATE** query with a **WHERE** clause is as follows:

**UPDATE** table\_name   
**SET** column1=value1, column2=value2, ...   
WHERE condition;

It is also possible to UPDATE **multiple columns** at the same time by comma-separating them:

# Deleting Data

The **DELETE**statement is used to remove data from your table. DELETE queries work much like UPDATE queries.

DELETE FROM table\_name   
WHERE condition;

# Creating a Table

Assume that you want to create a table called "Users" that consists of four columns: UserID, LastName, FirstName, and City.  
Use the following **CREATE TABLE** statement:

CREATE TABLE Users   
(   
UserID **int**,   
FirstName **varchar**(100),   
LastName **varchar**(100),   
City **varchar**(100)   
);

Primary key

**The following are commonly used SQL constraints:**  
**NOT NULL** - Indicates that a column cannot contain any NULL value.  
**UNIQUE**- Does not allow to insert a duplicate value in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table.  
**PRIMARY KEY** - Enforces the table to accept unique data for a specific column and this constraint create a unique index for accessing the table faster.  
**CHECK** - Determines whether the value is valid or not from a logical expression.  
**DEFAULT** - While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT.  
  
For example, the following means that the **name**column disallows NULL values.

UserID int NOT NULL **AUTO\_INCREMENT**,   
PRIMARY KEY (UserID)

The **ALTER TABLE** command is used to add, delete, or modify columns in an existing table.  
You would also use the ALTER TABLE command to add and drop various constraints on an existing table.  
  
Consider the following table called **People**:

DROP TABLE ……. ;

ALTER TABLE …..

DROP COLUMN …..;

**RENAME TABLE** ​People **TO PEOPLES**

ALTER TABLE PEOPLE

RENAME COLUMN FirstNAMe to name ;

**CREATE VIEW** view\_name **AS**  
SELECT column\_name(s)   
FROM table\_name   
WHERE condition;

In SQL, a VIEW is a **virtual table** that is based on the result-set of an SQL statement.  
  
A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

Job

DECLARE @list varchar(23) = '1,2,3,4'

SELECT ...FROM tbl WHERE col IN (@list)

DECLARE @list int(10) = ‘101211, 101121 , 210101’

Select \*

From …..

Where codice in split\_string(@list)

* crea tabella temporanea

CREATE TABLE #values (n int NOT NULL PRIMARY KEY)

INSERT #values(n)

SELECT number FROM intlist\_to\_tbl('1,2,3,4', ',')

SELECT ...

FROM tbl

WHERE col IN (SELECT n FROM #values)