

# Istruzioni SQL con spiegazioni ed esempi

# DDL

Data Definition Language

## 1. CREATE DATABASE

Crea un nuovo database.

```
CREATE DATABASE Negozio;
```

## 2. DROP DATABASE

Elimina un database esistente.

```
DROP DATABASE Negozio;
```

### 3. BACKUP DATABASE

Crea un backup del database.

```
BACKUP DATABASE Negozio TO DISK = 'C:\Backup\Negozio.bak';
```

## 4. CREATE TABLE

Crea una nuova tabella in un database.

```
CREATE TABLE Utenti (  
    ID INT PRIMARY KEY,  
    Nome VARCHAR(100),  
    Età INT  
);
```

## 5. DROP TABLE

Elimina una tabella esistente.

```
DROP TABLE Utenti;
```

## 6. ALTER TABLE

Modifica una tabella esistente.

```
ALTER TABLE Utenti ADD Email VARCHAR(100);
```



## 7. CONSTRAINTS

Definisce regole sui dati in una tabella ( PRIMARY KEY , FOREIGN KEY , ecc.).

```
CREATE TABLE Ordini (  
    ID INT PRIMARY KEY,  
    UtenteID INT,  
    FOREIGN KEY (UtenteID) REFERENCES Utenti(ID)  
);
```

## 8. NOT NULL

Impedisce l'inserimento di valori nulli in una colonna.

```
CREATE TABLE Utenti (  
    Nome VARCHAR(100) NOT NULL,  
    Età INT  
);
```

## 9. UNIQUE

Garantisce che tutti i valori in una colonna siano univoci.

```
CREATE TABLE Utenti (  
    Email VARCHAR(100) UNIQUE  
);
```

## 10. PRIMARY KEY

Identifica un record in modo univoco in una tabella.

```
CREATE TABLE Utenti (  
    ID INT PRIMARY KEY,  
    Nome VARCHAR(100)  
);
```

## 11. FOREIGN KEY

Crea un legame tra due tabelle.

```
CREATE TABLE Ordini (  
    ID INT PRIMARY KEY,  
    UtenteID INT,  
    FOREIGN KEY (UtenteID) REFERENCES Utenti(ID)  
);
```

## 12. CHECK

Imposta una condizione sui valori di una colonna.

```
CREATE TABLE Utenti (  
    Età INT CHECK (Età >= 18)  
);
```

## 13. DEFAULT

Assegna un valore predefinito a una colonna.

```
CREATE TABLE Utenti (  
    Stato VARCHAR(50) DEFAULT 'Attivo'  
);
```

## 14. INDEX

Crea un indice per velocizzare le query.

```
CREATE INDEX idx_nome ON Utenti(Nome);
```



## 15. AUTO INCREMENT

Incrementa automaticamente i valori di una colonna.

```
CREATE TABLE Utenti (  
    ID INT AUTO_INCREMENT PRIMARY KEY,  
    Nome VARCHAR(100)  
);
```

## 16. DATES

Gestisce i dati temporali come `DATE` , `DATETIME` , ecc.

```
CREATE TABLE Eventi (  
    ID INT PRIMARY KEY,  
    Data DATE,  
    Ora DATETIME  
);
```

## 17. VIEWS

Crea una vista come risultato di una query.

```
CREATE VIEW VistaUtenti AS  
SELECT Nome, Età FROM Utenti WHERE Età >= 18;
```

## 18. SQL INJECTION

Attacco che sfrutta query mal formate. **Evita usando query parametrizzate.**

```
-- Vulnerabile
"SELECT * FROM Utenti WHERE Nome = '" + inputUtente + "';"

-- Sicuro (esempio in SQL parametrizzato)
SELECT * FROM Utenti WHERE Nome = @Nome;
```

## 19. HOSTING

Il database è ospitato su server locali o cloud.

```
-- Esempio generico per connettersi al database ospitato  
mysql -h host_remoto -u utente -p;
```

## 20. DATA TYPES

Definiscono il tipo di dati per ogni colonna.

```
CREATE TABLE Utenti (  
    ID INT,  
    Nome VARCHAR(100),  
    DataNascita DATE  
);
```

# **DML**

Data Manipulation Language

## 1. SELECT

Usata per recuperare dati da una tabella.

```
SELECT Nome, Età FROM Utenti;
```



## 2. SELECT DISTINCT

Rimuove i duplicati nei risultati.

```
SELECT DISTINCT Città FROM Clienti;
```

### 3. WHERE

Filtra i risultati in base a una condizione.

```
SELECT * FROM Utenti WHERE Età > 18;
```

## 4. ORDER BY

Ordina i risultati in ordine crescente o decrescente.

```
SELECT * FROM Prodotti ORDER BY Prezzo ASC;
```

## 5. AND

Combina più condizioni; tutte devono essere vere.

```
SELECT * FROM Utenti WHERE Età > 18 AND Città = 'Roma';
```

## 6. OR

Combina più condizioni; almeno una deve essere vera.

```
SELECT * FROM Utenti WHERE Età > 18 OR Città = 'Roma';
```

## 7. NOT

Esclude i record che soddisfano la condizione.

```
SELECT * FROM Utenti WHERE NOT Città = 'Roma';
```

## 8. INSERT INTO

Aggiunge nuovi record a una tabella.

```
INSERT INTO Utenti (Nome, Età) VALUES ('Luca', 25);
```

## 9. NULL VALUES

Indica l'assenza di un valore.

```
SELECT * FROM Utenti WHERE Telefono IS NULL;
```



## 10. UPDATE

Aggiorna i record esistenti.

```
UPDATE Utenti SET Età = 30 WHERE Nome = 'Luca';
```

## 11. DELETE

Elimina i record da una tabella.

```
DELETE FROM Utenti WHERE Età < 18;
```

## 12. SELECT TOP

Limita il numero di record restituiti.

```
SELECT TOP 5 * FROM Prodotti;
```

## 13. Aggregate Functions

Calcola valori aggregati su un gruppo di dati (es.: SUM , COUNT ).

```
SELECT COUNT(*) FROM Utenti;
```

## 14. MIN and MAX

Trova il valore minimo o massimo in una colonna.

```
SELECT MIN(Prezzo), MAX(Prezzo) FROM Prodotti;
```

## 15. COUNT

Conta il numero di righe nei risultati.

```
SELECT COUNT(*) FROM Utenti WHERE Città = 'Roma';
```

## 16. SUM

Calcola la somma dei valori in una colonna.

```
SELECT SUM(Prezzo) FROM Prodotti;
```

## 17. AVG

Calcola la media dei valori in una colonna.

```
SELECT AVG(Prezzo) FROM Prodotti;
```



## 18. LIKE

Cerca valori che corrispondono a un modello.

```
SELECT * FROM Utenti WHERE Nome LIKE 'Lu%';
```

## 19. WILDCARDS

Utilizzato con `LIKE` per trovare valori parziali.

```
SELECT * FROM Utenti WHERE Nome LIKE '_a%';
```

## 20. IN

Confronta un valore con un elenco di valori.

```
SELECT * FROM Utenti WHERE Città IN ('Roma', 'Milano');
```

## 21. BETWEEN

Filtra valori compresi in un intervallo.

```
SELECT * FROM Prodotti WHERE Prezzo BETWEEN 10 AND 50;
```

## 22. ALIASES

Crea un alias temporaneo per colonne o tabelle.

```
SELECT Nome AS Cliente, Città AS Località FROM Utenti;
```

## 23. JOINS

Unisce tabelle basate su una relazione.

```
SELECT Utenti.Nome, Ordini.Totale FROM Utenti INNER JOIN Ordini ON Utenti.ID = Ordini.UtenteID;
```

## 24. INNER JOIN

Restituisce solo le righe con corrispondenze in entrambe le tabelle.

```
SELECT * FROM Utenti INNER JOIN Ordini ON Utenti.ID = Ordini.UtenteID;
```

## 25. LEFT JOIN

Restituisce tutte le righe dalla tabella sinistra e le corrispondenze dalla destra.

```
SELECT * FROM Utenti LEFT JOIN Ordini ON Utenti.ID = Ordini.UtenteID;
```



## 26. RIGHT JOIN

Restituisce tutte le righe dalla tabella destra e le corrispondenze dalla sinistra.

```
SELECT * FROM Utenti RIGHT JOIN Ordini ON Utenti.ID = Ordini.UtenteID;
```

## 27. FULL JOIN

Restituisce tutte le righe quando c'è una corrispondenza in una delle tabelle.

```
SELECT * FROM Utenti FULL JOIN Ordini ON Utenti.ID = Ordini.UtenteID;
```

## 28. SELF JOIN

Unisce una tabella a se stessa.

```
SELECT A.Nome, B.Nome FROM Utenti A, Utenti B WHERE A.ID = B.RefID;
```

## 29. UNION

Combina i risultati di due query, eliminando i duplicati.

```
SELECT Nome FROM Utenti UNION SELECT Nome FROM Clienti;
```

## 30. GROUP BY

Raggruppa i risultati in base a una colonna.

```
SELECT Città, COUNT(*) FROM Utenti GROUP BY Città;
```

## 31. HAVING

Filtra gruppi dopo l'uso di una funzione aggregata.

```
SELECT Città, COUNT(*) FROM Utenti GROUP BY Città HAVING COUNT(*) > 5;
```

## 32. EXISTS

Controlla se una sottoquery restituisce risultati.

```
SELECT * FROM Utenti WHERE EXISTS (SELECT * FROM Ordini WHERE Utenti.ID = Ordini.UtenteID);
```

## 33. ANY, ALL

Confronta un valore con uno o tutti i valori restituiti da una sottoquery.

```
SELECT * FROM Prodotti WHERE Prezzo > ALL (SELECT Prezzo FROM Prodotti WHERE Categoria = 'Elettronica');
```



## 34. SELECT INTO

Copia i dati in una nuova tabella.

```
SELECT * INTO UtentiBackup FROM Utenti;
```

## 35. INSERT INTO SELECT

Inserisce dati in una tabella da un'altra tabella.

```
INSERT INTO NuoviUtenti (Nome, Età) SELECT Nome, Età FROM Utenti WHERE Città = 'Roma';
```

## 36. CASE

Crea istruzioni condizionali nei risultati.

```
SELECT Nome, CASE WHEN Età >= 18 THEN 'Adulto' ELSE 'Minorenne' END AS Categoria FROM Utenti;
```

## 37. NULL FUNCTIONS

Gestisce valori `NULL` .

```
SELECT COALESCE(Telefono, 'Non disponibile') AS Contatto FROM Utenti;
```

## 38. STORED PROCEDURES

Blocchi di codice SQL salvati nel database.

```
CREATE PROCEDURE SpUtenti  
AS  
BEGIN  
    SELECT * FROM Utenti;  
END;
```

## 39. COMMENTS

Aggiunge commenti al codice SQL.

```
-- Questo è un commento su una sola riga
/*
Questo è un commento
su più righe
*/
```

## 40. OPERATORS

Esegue calcoli e confronti.

```
SELECT Prezzo * Quantità AS Totale FROM Prodotti;
```