0. Cos’è SQL e cos’è un database relazionale

SQL (Structured Query Language) è il linguaggio usato per:

- creare tabelle
- inserire dati
- leggere dati
- modificare dati
- collegare tabelle tra loro

Un database relazionale organizza i dati in **tabelle** collegate tramite **chiavi**.

1. CREARE LE TABELLE (CREATE TABLE)

Hai già due tabelle perfette per imparare:

Tabella dipendenti

sql

```
CREATE TABLE dipendenti (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    cognome TEXT NOT NULL,
    email TEXT UNIQUE,
    reparto TEXT CHECK(reparto IN ('Vendite', 'IT', 'HR', 'Finanza',
'Logistica')),
    stipendio REAL,
    data_assunzione DATE,
    attivo BOOLEAN DEFAULT 1
);
```

Spiegazione dei campi

- **PRIMARY KEY AUTOINCREMENT** → ID unico che cresce da solo
- **NOT NULL** → campo obbligatorio
- **UNIQUE** → niente duplicati
- **CHECK** → limita i valori possibili
- **DEFAULT** → valore predefinito

Tabella progetti

sql

```
CREATE TABLE progetti (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome_progetto TEXT NOT NULL,
    budget REAL,
    data_inizio DATE,
    data_fine DATE,
    responsabile_id INTEGER,
    FOREIGN KEY (responsabile_id) REFERENCES dipendenti(id)
);
```

Spiegazione

- `responsabile_id` è una **chiave esterna**
- collega ogni progetto a un dipendente
- crea una relazione **1-a-molti** (un dipendente → molti progetti)

2. INSERIRE DATI (INSERT)

Dipendenti

sql

```
INSERT INTO dipendenti (nome, cognome, email, reparto, stipendio,
data_assunzione, attivo)
VALUES
('Marco', 'Rossi', 'marco.rossi@example.com', 'IT', 2800.50, '2022-03-15',
1),
('Laura', 'Bianchi', 'laura.bianchi@example.com', 'Vendite', 2500.00,
'2021-11-01', 1),
('Giulia', 'Verdi', 'giulia.verdi@example.com', 'HR', 2300.00, '2020-06-
10', 1),
('Paolo', 'Neri', 'paolo.neri@example.com', 'Finanza', 3200.00, '2019-09-
20', 1),
('Sara', 'Galli', 'sara.galli@example.com', 'Logistica', 2100.00, '2023-
01-05', 1);
```

Progetti

sql

```
INSERT INTO progetti (nome_progetto, budget, data_inizio, data_fine,
responsabile_id)
VALUES
('Sito Web Aziendale', 15000, '2023-02-01', '2023-06-30', 1),
('Espansione Mercato Europa', 50000, '2023-01-15', '2023-12-31', 2),
('Formazione Interna HR', 8000, '2023-03-01', '2023-04-15', 3),
```

```
('Ottimizzazione Contabilità', 12000, '2023-05-10', '2023-09-30', 4),
('Riorganizzazione Magazzino', 20000, '2023-04-01', '2023-10-01', 5);
```

3. LEGGERE I DATI (SELECT)

Tutti i dati

sql

```
SELECT * FROM dipendenti;
SELECT * FROM progetti;
```

Filtrare (WHERE)

sql

```
SELECT *
FROM progetti
WHERE nome_progetto = 'Sito Web Aziendale';
```

Ricerca parziale (LIKE)

sql

```
SELECT *
FROM progetti
WHERE nome_progetto LIKE '%Sito%';
```

4. ORDINARE I RISULTATI (ORDER BY)

sql

```
SELECT *
FROM dipendenti
ORDER BY stipendio DESC;
```

5. LIMITARE I RISULTATI (LIMIT)

sql

```
SELECT *
FROM dipendenti
ORDER BY stipendio DESC
LIMIT 3;
```

6. MODIFICARE DATI (UPDATE)

sql

```
UPDATE dipendenti
SET stipendio = 3000
WHERE id = 1;
```

7. CANCELLARE DATI (DELETE)

sql

```
DELETE FROM dipendenti
WHERE id = 5;
```

⚠️ Senza WHERE → cancella tutto

8. FUNZIONI DI AGGREGAZIONE

Quanti dipendenti ci sono

sql

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

Stipendio medio

sql

```
SELECT AVG(stipendio) FROM dipendenti;
```

Somma dei budget

sql

```
SELECT SUM(budget) FROM progetti;
```

9. RAGGRUPPARE (GROUP BY)

sql

```
SELECT reparto, AVG(stipendio)
FROM dipendenti
GROUP BY reparto;
```

10. COLLEGARE TABELLE (JOIN)

Fondamentale nei database relazionali.

Progetti + nome del responsabile

sql

```
SELECT p.nome_progetto, p.budget, d.nome, d.cognome
FROM progetti p
JOIN dipendenti d ON p.responsabile_id = d.id;
```

11. MODIFICARE UNA TABELLA (ALTER TABLE)

Aggiungere una colonna:

sql

```
ALTER TABLE dipendenti
ADD COLUMN telefono TEXT;
```

12. CREARE UNA VISTA (VIEW)

Una query salvata come fosse una tabella.

sql

```
CREATE VIEW vista_progetti AS
SELECT p.nome_progetto, p.budget, d.nome, d.cognome
FROM progetti p
JOIN dipendenti d ON p.responsabile_id = d.id;
```

13. STRUTTURA MENTALE DA IMPARARE

Ogni volta che lavori con SQL, pensa così:

1. **Cosa voglio?** → SELECT
2. **Da dove?** → FROM
3. **Con quali condizioni?** → WHERE
4. **In che ordine?** → ORDER BY
5. **Quante righe?** → LIMIT
6. **Devo collegare tabelle?** → JOIN
7. **Devo raggruppare?** → GROUP BY

Questa è la logica universale.