

Riferimento Rapido SQL

SELECT, JOIN, subquery, indici, transazioni

SELECT

```
SELECT * FROM users;
SELECT name, email FROM users;
SELECT DISTINCT city FROM users;
SELECT name AS full_name FROM users;
```

WHERE

Operatori di Confronto

= <> (!=)	Uguale / diverso
< > <= >=	Operatori di confronto
AND OR NOT	Operatori logici
IS NULL / IS NOT NULL	Controlli null

Pattern Matching

```
SELECT * FROM users WHERE name LIKE 'A%';
-- % = any chars, _ = single char
SELECT * FROM users WHERE age IN (20, 25, 30);
SELECT * FROM users WHERE age BETWEEN 18 AND 30;
```

JOIN

Tipi di Join

INNER JOIN	Righe corrispondenti in entrambe le tabelle
LEFT JOIN	Tutte le righe sinistre + corrispondenti destre
RIGHT JOIN	Tutte le righe destre + corrispondenti sinistre
FULL OUTER JOIN	Tutte le righe di entrambe le tabelle
CROSS JOIN	Prodotto cartesiano di entrambe le tabelle

Sintassi Join

```
SELECT u.name, o.total
FROM users u
INNER JOIN orders o ON u.id = o.user_id;

SELECT u.name, o.total
FROM users u
LEFT JOIN orders o ON u.id = o.user_id;
```

INSERT / UPDATE / DELETE

Insert

```
INSERT INTO users (name, email)
VALUES ('Alice', 'alice@example.com');

INSERT INTO users (name, email) VALUES
('Bob', 'bob@ex.com'),
('Carol', 'carol@ex.com');
```

Update

```
UPDATE users SET email = 'new@ex.com'
WHERE id = 1;
```

Delete

```
DELETE FROM users WHERE id = 1;
DELETE FROM users; -- delete all rows
```

CREATE TABLE

Sintassi

```
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  email TEXT UNIQUE,
  age INTEGER DEFAULT 0,
  score REAL
);
```

Tipi di Dato Comuni

INTEGER	Numeri interi
REAL	Numeri in virgola mobile
TEXT	Dati stringa/testo
BLOB	Dati binari
BOOLEAN	TRUE / FALSE (memorizzato come 0/1)
DATE / DATETIME	Valori di data e timestamp

Vincoli

PRIMARY KEY	Identificatore univoco di riga
NOT NULL	Valore obbligatorio
UNIQUE	Nessun valore duplicato
DEFAULT val	Valore predefinito se omissso
CHECK (expr)	Regola di validazione personalizzata
FOREIGN KEY	Riferimento a un'altra tabella

Funzioni di Aggregazione

COUNT(*)	Numero di righe
COUNT(col)	Valori non-null nella colonna
SUM(col)	Somma della colonna numerica
AVG(col)	Media della colonna numerica
MIN(col)	Valore minimo
MAX(col)	Valore massimo

Esempio

```
SELECT COUNT(*) AS total,
       AVG(age) AS avg_age,
       MAX(score) AS top_score
FROM users;
```

GROUP BY / HAVING

```
SELECT city, COUNT(*) AS num_users
FROM users
GROUP BY city;

SELECT city, AVG(age) AS avg_age
FROM users
GROUP BY city
HAVING AVG(age) > 25;
```

WHERE filtra le righe prima del raggruppamento; HAVING filtra i gruppi dopo l'aggregazione

ORDER BY / LIMIT

```
SELECT * FROM users ORDER BY name ASC;
SELECT * FROM users ORDER BY age DESC;
SELECT * FROM users
ORDER BY city, name
LIMIT 10 OFFSET 20; -- skip 20, take 10
```

Subquery

Nella Clausola WHERE

```
SELECT name FROM users
WHERE id IN (
  SELECT user_id FROM orders
  WHERE total > 100
);
```

Come Tabella Derivata

```
SELECT city, avg_age FROM (
  SELECT city, AVG(age) AS avg_age
  FROM users GROUP BY city
) WHERE avg_age > 30;
```

Indici

```
CREATE INDEX idx_name ON users(name);
CREATE UNIQUE INDEX idx_email
  ON users(email);
DROP INDEX idx_name;
```

Quando Usare gli Indici

Colonne in WHERE	Velocizza il filtraggio
Colonne in JOIN ON	Velocizza le ricerche nei join
Colonne in ORDER BY	Velocizza l'ordinamento
Colonne ad alta cardinalità	Molti valori unici traggono maggior beneficio