Referência Rápida de SQL

SELECT, JOIN, subconsultas, índices, transações

SELECT

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

WHERE

Operadores de Comparação

`= <> (!=)`	Igual / diferente
`< > <= >=`	Operadores de comparação
`AND OR NOT`	Operadores lógicos
`IS NULL / IS NOT NULL`	Verificações de nulo

Correspondência de Padrão

```
SELECT * FROM users WHERE name LIKE 'A%';
-- % = quaisquer caracteres, _ = caractere único
SELECT * FROM users WHERE age IN (20, 25, 30);
SELECT * FROM users WHERE age BETWEEN 18 AND 30;
```

JOIN

Tipos de JOIN

`INNER JOIN`	Linhas correspondentes em ambas as tabelas
`LEFT JOIN`	Todas as linhas da esquerda + correspondentes da direita
`RIGHT JOIN`	Todas as linhas da direita + correspondentes da esquerda
`FULL OUTER JOIN`	Todas as linhas de ambas as tabelas
`CROSS JOIN`	Produto cartesiano de ambas as tabelas

Sintaxe do 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

Inserir

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

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

Atualizar

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

Apagar

```
DELETE FROM users WHERE id = 1;
DELETE FROM users; -- apaga todas as linhas
```

CREATE TABLE

Sintaxe

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

Tipos de Dados Comuns

`INTEGER`	Números inteiros
`REAL`	Números de ponto flutuante
`TEXT`	Dados de string/texto
`BLOB`	Dados binários
`BOOLEAN`	TRUE / FALSE (armazenado como 0/1)
`DATE / DATETIME`	Valores de data e timestamp

Restrições

`PRIMARY KEY`	Identificador único de linha
`NOT NULL`	Valor obrigatório
`UNIQUE`	Sem valores duplicados
`DEFAULT val`	Valor padrão se omitido
`CHECK (expr)`	Regra de validação personalizada
`FOREIGN KEY`	Referência a outra tabela

Funções de Agregação

`COUNT(*)`	Número de linhas
`COUNT(col)`	Valores não nulos na coluna
`SUM(col)`	Soma da coluna numérica
`AVG(col)`	Média da coluna numérica
`MIN(col)`	Valor mínimo
`MAX(col)`	Valor máximo

Exemplo

```
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 linhas antes do agrupamento; HAVING filtra grupos após a agregação

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;  -- pula 20, pega 10
```

Subconsultas

Na Cláusula WHERE

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

Como Tabela Derivada

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

Índices

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

Quando Indexar

`Colunas em WHERE`	Acelera a filtragem
`Colunas em JOIN ON`	Acelera pesquisas de JOIN
`Colunas em ORDER BY`	Acelera a ordenação
`Colunas de alta cardinalidade`	Muitos valores únicos se beneficiam mais