Referensi Cepat SQL

SELECT, JOIN, subquery, index, transaksi

SELECT

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

WHERE

Operator Perbandingan

`= <> (!=)`	Sama / tidak sama
`< > <= >=`	Operator perbandingan
`AND OR NOT`	Operator logika
`IS NULL / IS NOT NULL`	Pengecekan null

Pencocokan Pola

```
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

Jenis Join

`INNER JOIN`	Baris yang cocok di kedua tabel
`LEFT JOIN`	Semua baris kiri + yang cocok dari kanan
`RIGHT JOIN`	Semua baris kanan + yang cocok dari kiri
`FULL OUTER JOIN`	Semua baris dari kedua tabel
`CROSS JOIN`	Hasil kartesian dari kedua tabel

Sintaks 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

Sintaks

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

Tipe Data Umum

`INTEGER`	Bilangan bulat
`REAL`	Bilangan titik mengambang
`TEXT`	Data string / teks
`BLOB`	Data biner
`BOOLEAN`	TRUE / FALSE (disimpan sebagai 0/1)
`DATE / DATETIME`	Nilai tanggal dan timestamp

Constraint

`PRIMARY KEY`	Identifier baris yang unik
`NOT NULL`	Nilai wajib diisi
`UNIQUE`	Tidak ada nilai duplikat
`DEFAULT val`	Nilai default jika tidak diisi
`CHECK (expr)`	Aturan validasi kustom
`FOREIGN KEY`	Referensi ke tabel lain

Fungsi Agregat

`COUNT(*)`	Jumlah baris
`COUNT(col)`	Nilai non-null di kolom
`SUM(col)`	Jumlah kolom numerik
`AVG(col)`	Rata-rata kolom numerik
`MIN(col)`	Nilai minimum
`MAX(col)`	Nilai maksimum

Contoh

```
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 memfilter baris sebelum pengelompokan; HAVING memfilter grup setelah agregasi

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

Dalam Klausa WHERE

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

Sebagai Derived Table

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

Index

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

Kapan Membuat Index

`Kolom di WHERE`	Mempercepat filtering
`Kolom di JOIN ON`	Mempercepat pencarian join
`Kolom di ORDER BY`	Mempercepat pengurutan
`Kolom high-cardinality`	Banyak nilai unik paling banyak untung