SQL 빠른 참조

SELECT, JOIN, 서브쿼리, 인덱스, 트랜잭션

SELECT

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

WHERE

비교 연산자

`= <> (!=)`	같음 / 다름
`< > <= >=`	비교 연산자
`AND OR NOT`	논리 연산자
`IS NULL / IS NOT NULL`	Null 확인

패턴 일치

```
SELECT * FROM users WHERE name LIKE 'A%';
-- % = 임의 문자, _ = 단일 문자
SELECT * FROM users WHERE age IN (20, 25, 30);
SELECT * FROM users WHERE age BETWEEN 18 AND 30;
```

JOIN

조인 유형

`INNER JOIN`	두 테이블 모두에서 일치하는 행
`LEFT JOIN`	왼쪽 모든 행 + 일치하는 오른쪽
`RIGHT JOIN`	오른쪽 모든 행 + 일치하는 왼쪽
`FULL OUTER JOIN`	두 테이블의 모든 행
`CROSS JOIN`	두 테이블의 카르테시안 곱

조인 문법

```
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

삽입

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

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

수정

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

삭제

```
DELETE FROM users WHERE id = 1;
DELETE FROM users; -- 모든 행 삭제
```

CREATE TABLE

문법

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

일반 데이터 타입

`INTEGER`	정수
`REAL`	부동소수점 숫자
`TEXT`	문자열 / 텍스트 데이터
`BLOB`	이진 데이터
`BOOLEAN`	TRUE / FALSE (0/1로 저장)
`DATE / DATETIME`	날짜 및 타임스탬프 값

제약 조건

`PRIMARY KEY`	고유 행 식별자
`NOT NULL`	값 필수
`UNIQUE`	중복 값 없음
`DEFAULT val`	생략 시 기본값
`CHECK (expr)`	커스텀 유효성 검사 규칙
`FOREIGN KEY`	다른 테이블 참조

집계 함수

`COUNT(*)`	행 수
`COUNT(col)`	열의 NULL이 아닌 값 수
`SUM(col)`	숫자 열의 합계
`AVG(col)`	숫자 열의 평균
`MIN(col)`	최솟값
`MAX(col)`	최댓값

예시

```
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;
```

WHERE는 그룹화 전 행을 필터링하고, HAVING은 집계 후 그룹을 필터링

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;  -- 20개 건너뛰고 10개 가져오기
```

서브쿼리

WHERE 절에서

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

파생 테이블로

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

인덱스

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

인덱스 사용 시점

`WHERE의 열`	필터링 속도 향상
`JOIN ON의 열`	조인 조회 속도 향상
`ORDER BY의 열`	정렬 속도 향상
`고카디널리티 열`	고유값이 많을수록 효과적