HAPPY HACKING

WHERE

HAPPY HACKING

- Table users 생성

```
CREATE TABLE users (
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    age INTEGER NOT NULL,
    country TEXT NOT NULL,
    phone TEXT NOT NULL,
    balance INTEGER NOT NULL
);
```

WHERE

HAPPY HACKING

- csv파일 정보를 테이블에 적용하기

```
sqlite> .mode csv
sqlite> .import users.csv users
sqlite> .tables
classmates  examples    users
```

WHERE

HAPPY HACKING

- 특정 조건으로 데이터 조회하기

```
SELECT * FROM 테이블이름 WHERE 조건;
```

WHERE

HAPPY HACKING

- Q. users 테이블에서 age가 30 이상인 유저의 모든 컬럼 정보를 조회하려면?

WHERE

HAPPY HACKING

- Q. users 테이블에서 age가 30 이상인 유저의 모든 컬럼 정보를 조회하려면?

```sql
SELECT * FROM users WHERE age >= 30;
```

WHERE

HAPPY HACKING

- Q. users 테이블에서 age가 30 이상인 유저의 이름만 조회하려면?

WHERE

HAPPY HACKING

- Q. users 테이블에서 age가 30 이상인 유저의 이름만 조회하려면?

```sql
SELECT first_name FROM users WHERE age >= 30;
```

WHERE

HAPPY HACKING

- Q. users 테이블에서 age가 30 이상, 성이 '김'인 사람의 나이와 이름만 조회하려면?

WHERE

HAPPY HACKING

- Q. users 테이블에서 age가 30 이상, 성이 '김'인 사람의 나이와 이름만 조회하려면?

```
SELECT age, first_name FROM users
WHERE age >= 30 AND last_name='김';
```

HAPPY HACKING

- **WHERE절에서 사용할 수 있는 연산자**
  - 비교 연산자
    - =, 〉, 〉=, 〈, 〈= 는 숫자 혹은 문자 값의 대/소, 동일 여부를 확인하는 연산자
  - 논리 연산자
    - AND
      - 앞에 있는 조건과 뒤에 오는 조건이 모두 참인 경우
    - OR
      - 앞의 조건이나 뒤의 조건이 참인 경우
    - NOT
      - 뒤에 오는 조건의 결과를 반대로

WHERE

HAPPY HACKING

- 주의!

```
-- 1.
WHERE HEIGHT = 175 OR HEIGHT = 183 AND WEIGHT = 80
-- 2.
WHERE (HEIGHT = 175 OR HEIGHT = 183) AND WEIGHT = 80
```

WHERE

HAPPY HACKING

- 주의!

```
-- 1. 키가 175이거나, 키가 183이면서 몸무게가 80인 사람
WHERE HEIGHT = 175 OR HEIGHT = 183 AND WEIGHT = 80
-- 2. 키가 175 또는 183인 사람 중에서 몸무게가 80인 사람
WHERE (HEIGHT = 175 OR HEIGHT = 183) AND WEIGHT = 80
```

WHERE

HAPPY HACKING

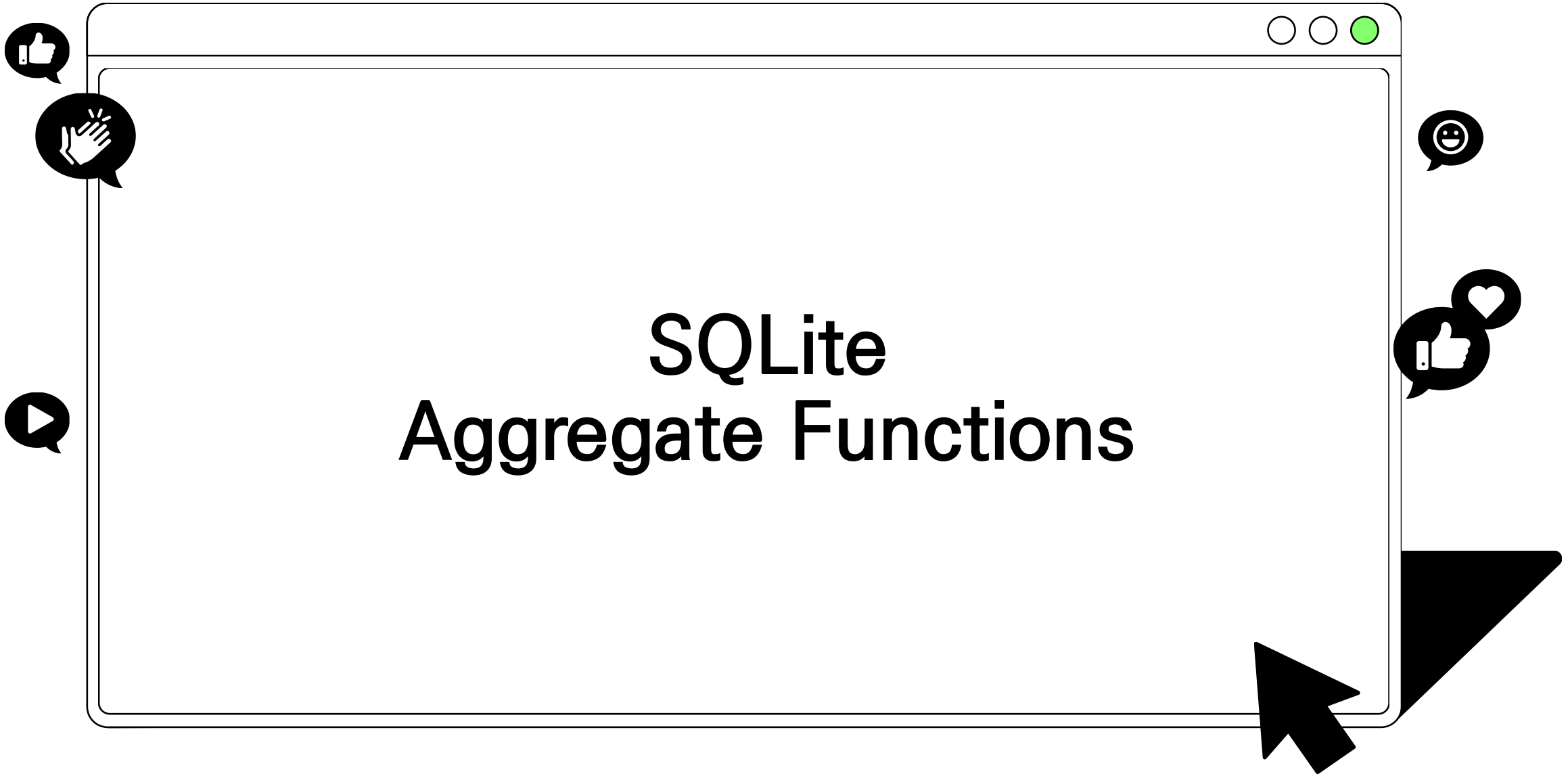
- **SQL 사용할 수 있는 연산자**
  - BETWEEN 값1 AND 값2
    - 값1과 값2 사이의 비교 (값1 <= 비교값 <= 값2)
  - IN (값1, 값2, …)
    - 목록 중에 값이 하나라도 일치하면 성공
  - LIKE
    - 비교 문자열과 형태 일치
    - 와일드카드 (% : 0개 이상 문자, _ : 1개 단일 문자)
  - IS NULL / IS NOT NULL
    - NULL 여부를 확인할 때는 항상 = 대신에 IS를 활용

- SQL 사용할 수 있는 연산자
  - 부정 연산자
    - 같지 않다. (!=, ^=, 〈〉)
    - ~와 같지 않다. (NOT 칼럼명 =)
    - ~보다 크지 않다. (NOT 칼럼명 〉)

```
WHERE 칼럼명1 != 비교값1
  AND 칼럼명2 ^= 비교값2
  AND 칼럼명3 <> 비교값3
  AND NOT 칼럼명4 = 비교값4
  AND NOT 칼럼명5 > 비교값5;
```

- **연산자 우선순위**
  - 1순위 : 괄호 ()
  - 2순위 : NOT
  - 3순위 : 비교 연산자, SQL
  - 4순위 : AND
  - 5순위 : OR

HAPPY HACKING



SQLite
Aggregate Functions

SQLite Aggregate Functions

HAPPY HACKING

- Aggregate function (집계 함수)
  - 값 집합에 대한 계산을 수행하고 단일 값을 반환
    - 여러 행으로부터 하나의 결괏값을 반환하는 함수
  - SELECT 구문에서만 사용됨
  - 예시
    - 테이블 전체 행 수를 구하는 **`COUNT(*)`**
    - age 컬럼 전체 평균 값을 구하는 **`AVG(age)`**

SQLite Aggregate Functions

HAPPY HACKING

- COUNT
  - 그룹의 항목 수를 가져옴
- AVG
  - 모든 값의 평균을 계산
- MAX
  - 그룹에 있는 모든 값의 최대값을 가져옴
- MIN
  - 그룹에 있는 모든 값의 최소값을 가져옴
- SUM
  - 모든 값의 합을 계산

https://www.sqlitetutorial.net/sqlite-count-function/
https://www.sqlitetutorial.net/sqlite-avg/
https://www.sqlitetutorial.net/sqlite-max/
https://www.sqlitetutorial.net/sqlite-min/
https://www.sqlitetutorial.net/sqlite-sum/

Sqlite Aggregate Functions

HAPPY HACKING

- COUNT(레코드의 개수 조회하기)

```
SELECT COUNT(컬럼) FROM 테이블이름;
```

Sqlite Aggregate Functions

HAPPY HACKING

- Q. users 테이블의 레코드 총 개수를 조회한다면?

Sqlite Aggregate Functions

HAPPY HACKING

- Q. users 테이블의 레코드 총 개수를 조회한다면?

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

Sqlite Aggregate Functions

HAPPY HACKING

- AVG, SUM, MIN, MAX
  - 위 함수들은 기본적으로 해당 컬럼이 숫자(INTEGER)일 때만 사용 가능

```
SELECT AVG(컬럼) FROM 테이블이름;
SELECT SUM(컬럼) FROM 테이블이름;
SELECT MIN(컬럼) FROM 테이블이름;
SELECT MAX(컬럼) FROM 테이블이름;
```

Sqlite Aggregate Functions

HAPPY HACKING

- Q. 30살 이상인 사람들의 평균 나이는?

Sqlite Aggregate Functions

HAPPY HACKING

- Q. 30살 이상인 사람들의 평균 나이는?

```
SELECT AVG(age) FROM users WHERE age>=30;
```

Sqlite Aggregate Functions

HAPPY HACKING

- Q. 계좌 잔액(balance)이 가장 높은 사람과 그 액수를 조회하려면?

Sqlite Aggregate Functions

HAPPY HACKING

- Q. 계좌 잔액(balance)이 가장 높은 사람과 그 액수를 조회하려면?

```
SELECT first_name, MAX(balance) FROM users;
```

Sqlite Aggregate Functions

HAPPY HACKING

- Q. 나이가 30 이상인 사람의 계좌 평균 잔액을 조회하려면?

Sqlite Aggregate Functions

HAPPY HACKING

- Q. 나이가 30 이상인 사람의 계좌 평균 잔액을 조회하려면?

```
SELECT AVG(balance) FROM users WHERE age>=30;
```

HAPPY HACKING

LIKE

HAPPY HACKING

- “query data based on pattern matching”
- 패턴 일치를 기반으로 데이터를 조회하는 방법
- SQLite는 패턴 구성을 위한 2개의 wildcards를 제공
  - % (percent sign)
    - 0개 이상의 문자
  - _ (underscore)
    - 임의의 단일 문자

HAPPY HACKING

와일드 카드 2가지 패턴

- wildcards

%

(percent sign)

이 자리에 문자열이

있을 수도, 없을 수도 있다.

_

(underscore)

반드시 이 자리에

한 개의 문자가 존재해야 한다.

LIKE

HAPPY HACKING

- LIKE statement : 패턴을 확인하여 해당하는 값을 조회하기

```
SELECT * FROM 테이블이름 WHERE 컬럼 LIKE '패턴';
```

LIKE

HAPPY HACKING

- wildcards 사용 예시

```
SELECT * FROM 테이블이름 WHERE 컬럼 LIKE '패턴';
```

와일드카드패턴	의미
2%	
%2	
%2%	
_2%	
1___	
2_%_% / 2__%	

LIKE

HAPPY HACKING

- wildcards 사용 예시

```
SELECT * FROM 테이블이름 WHERE 컬럼 LIKE '패턴';
```

와일드카드패턴	의미
2%	2로 시작하는 값
%2	2로 끝나는 값
%2%	2가 들어가는 값
_2%	아무 값이 하나 있고 두 번째가 2로 시작하는 값
1___	1로 시작하고 총 4자리인 값
2_%_% / 2__%	2로 시작하고 적어도 3자리인 값

LIKE

HAPPY HACKING

- Q. users 테이블에서 나이가 20대인 사람만 조회한다면?

LIKE

HAPPY HACKING

- Q. users 테이블에서 나이가 20대인 사람만 조회한다면?

LIKE

HAPPY HACKING

Q. users 테이블에서 나이가 20대인 사람만 조회한다면?

```
SELECT * FROM users WHERE age LIKE '2_';
```

LIKE

HAPPY HACKING

- Q. users 테이블에서 지역 번호가 02인 사람만 조회한다면?

LIKE

HAPPY HACKING

- Q. users 테이블에서 지역 번호가 02인 사람만 조회한다면?

LIKE

HAPPY HACKING

- Q. users 테이블에서 지역 번호가 02인 사람만 조회한다면?

```
SELECT * FROM users WHERE phone LIKE '02-%';
```

LIKE

HAPPY HACKING

- Q. users 테이블에서 이름이 '준'으로 끝나는 사람만 조회한다면?

LIKE

HAPPY HACKING

- Q. users 테이블에서 이름이 '준'으로 끝나는 사람만 조회한다면?

LIKE

HAPPY HACKING

- Q. users 테이블에서 이름이 '준'으로 끝나는 사람만 조회한다면?

```
SELECT * FROM users WHERE first_name LIKE '%준';
```

LIKE

HAPPY HACKING

- Q. users 테이블에서 중간 번호가 5114인 사람만 조회한다면?

LIKE

HAPPY HACKING

- Q. users 테이블에서 중간 번호가 5114인 사람만 조회한다면?

LIKE

HAPPY HACKING

- Q. users 테이블에서 중간 번호가 5114인 사람만 조회한다면?

```
SELECT * FROM users WHERE phone LIKE '%-5114-%;
```

HAPPY HACKING

ORDER BY

ORDER BY

HAPPY HACKING

- **ORDER BY**
  - “sort a result set of a query”
  - 조회 결과 집합을 정렬
  - SELECT 문에 추가하여 사용
  - 정렬 순서를 위한 2개의 keyword 제공
    - ASC – 오름차순 (default)
    - DESC – 내림차순

ORDER BY

HAPPY HACKING

- 특정 컬럼을 기준으로 데이터를 정렬해서 조회하기
  - ASC : 오름차순 (default)
  - DESC : 내림차순

```
SELECT * FROM 테이블이름 ORDER BY 컬럼 ASC;
SELECT * FROM 테이블이름 ORDER BY 컬럼 DESC;
```

ORDER BY

HAPPY HACKING

- Q. users 에서 나이 순으로 오름차순 정렬하여 상위 10개만 조회한다면?

ORDER BY

HAPPY HACKING

- Q. users 에서 나이 순으로 오름차순 정렬하여 상위 10개만 조회한다면?

```
SELECT * FROM users ORDER BY age ASC LIMIT 10;
```

ORDER BY

HAPPY HACKING

- Q. 나이 순, 성 순으로 오름차순 정렬하여 상위 10개만 조회한다면?

ORDER BY

HAPPY HACKING

- Q. 나이 순, 성 순으로 오름차순 정렬하여 상위 10개만 조회한다면?

```
SELECT * FROM users ORDER BY age, last_name
ASC LIMIT 10;
```

ORDER BY

HAPPY HACKING

- Q. 계좌 잔액 순으로 내림차순 정렬하여 해당 유저의 성과 이름을 10개만 조회한다면?

ORDER BY

HAPPY HACKING

- Q. 계좌 잔액 순으로 내림차순 정렬하여 해당 유저의 성과 이름을 10개만 조회한다면?

```
SELECT last_name, first_name FROM users
ORDER BY balance DESC LIMIT 10;
```