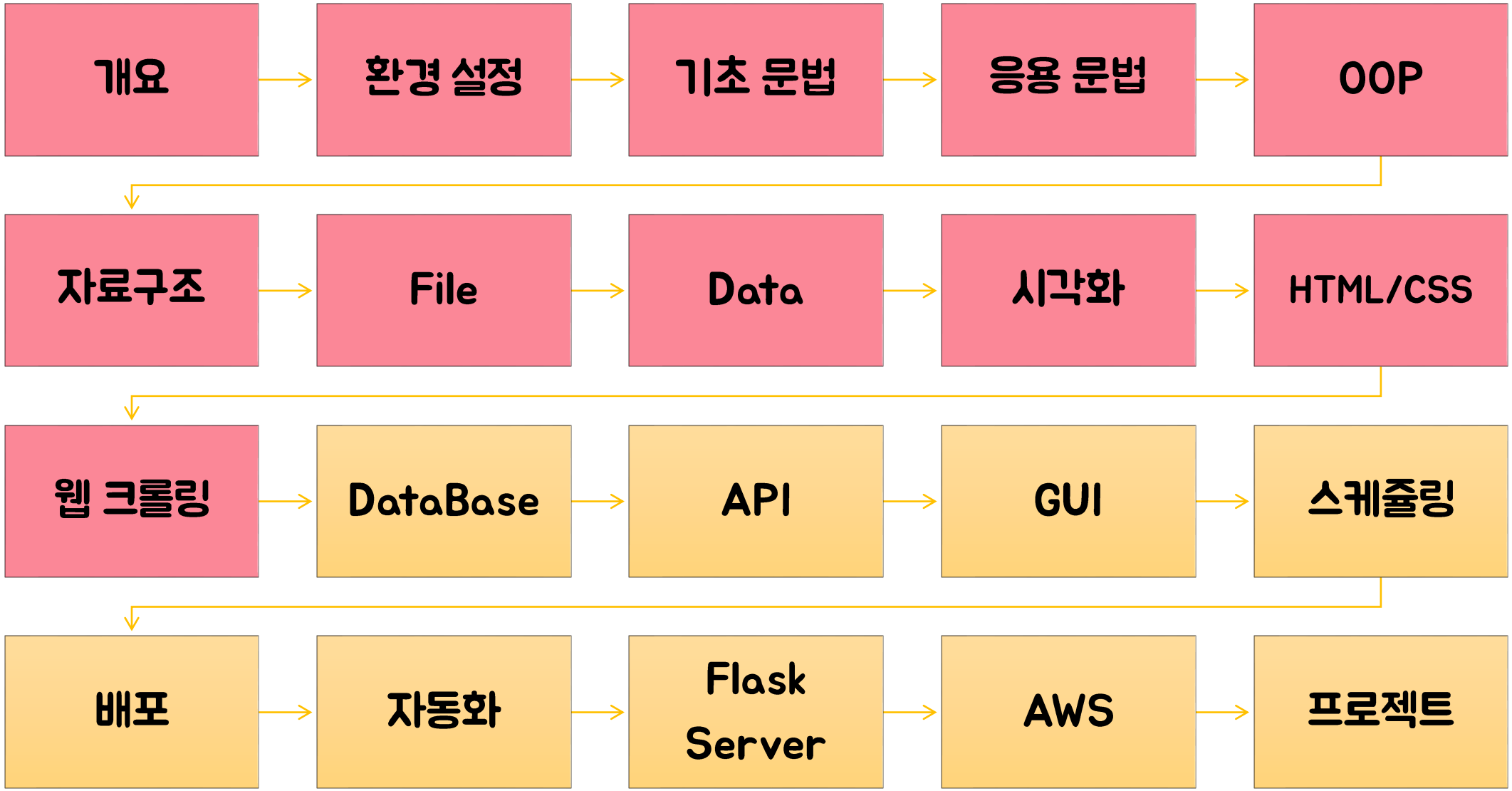
python

대한상공회의소
서울기술교육센터

나 예 호 교수

python 수업 진행

대한상공회의소
서울기술교육센터



데이터를 저장, 관리하는 시스템
대량의 데이터를 효율적으로 저장하고 검색하는 데 사용

SQLite란?
- 가볍고 설정이 필요 없는 파일 기반 데이터베이스.
- sqlite3 모듈을 사용하여 Python에서 쉽게 활용 가능.

SQLite 데이터베이스 생성 및 테이블 만들기

- SQLite 데이터베이스 파일은 .db 확장자로 저장됨.
- CREATE TABLE을 사용하여 테이블을 생성함.

```
import sqlite3
```

데이터베이스 연결 과정

1. DataBase 통로 생성
2. SQL 통로 생성
3. CRUD
4. 통로 닫기

데이터베이스 연결 과정

1. DataBase 통로 생성

```
# DataBase 통로 생성
# 데이터베이스 연결 (파일이 없으면 자동 생성)
conn = sqlite3.connect("users.db")
```

데이터베이스 연결 과정
2. SQL 통로 생성

```
# SQL 통로 생성
cursor = conn.cursor()
```

데이터베이스 연결 과정

3. CRUD

```
# CRUD 작업
# 현재는 테이블 생성을 위한
# CREATE 작업 진행
cursor.execute("""
CREATE TABLE users (
    name TEXT,
    age INTEGER,
    email TEXT
)
""")
```

* C, U, D 작업 시

```
# 변경사항 저장
conn.commit()
```

데이터베이스 연결 과정
4. 통로 닫기 (역순으로 닫는다)

```
# 통로 닫기
cursor.close()
conn.close()
```

Q. 테이블 생성

1. company.db 파일을 생성하고 연결하세요
 단, 데이터베이스 파일이 존재하는지 확인한 후 연결하시오
2. employees라는 테이블을 생성하시오
 단, 테이블이 존재하는 경우 새로 생성하지 않도록 설정하시오
3. 사용자로부터 테이블명을 입력 받아 동적으로 테이블을 생성하시오

컬럼명	데이터 타입	설명
id	INTEGER(PK, AUTOINCREMENT)	기본 키(자동 증가)
name	TEXT NOT NULL	사용자 이름
department	TEXT	소속 부서
email	TEXT UNIQUE NOT NULL	이메일(중복 불가)

```python
import sqlite3
import os

if not os.path.exists("company.db"):
    conn = sqlite3.connect("company.db")
```

```python
cursor = conn.cursor()
cursor.execute("""
CREATE TABLE IF NOT EXISTS employees (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    department INTEGER,
    email TEXT UNIQUE NOT NULL
)
""")
```

```python
# 변경사항 저장
conn.commit()

# 통로 닫기
cursor.close()
conn.close()
```

```
table_name = input("생성할 테이블 이름 입력: ")
cursor.execute(f"""
CREATE TABLE IF NOT EXISTS {table_name} (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    department INTEGER,
    email TEXT UNIQUE NOT NULL
)
""")
print(f"테이블 '{table_name}' 생성 완료.")
```

pandas.read_sql()을 활용하여 SQLite 데이터를 불러오기

```
df = pd.read_sql("SELECT * FROM users", conn)
```

SQL 데이터를 엑셀 파일로 변환하여 저장

```
df.to_excel("users.xlsx", index=False)
```

명령어 종류	명령어	설명
데이터 조작어 (DML : Data Manipulation Language)	SELECT	데이터베이스에 들어 있는 데이터를 조회하거나 검색하기 위한 명령어
	INSERT UPDATE DELETE	데이터베이스의 테이블에 들어 있는 데이터에 변형을 가하는 종류 (데이터 삽입, 수정, 삭제)의 명령어들을 말함.
데이터 정의어 (DDL : data Definition Language)	CREATE ALTER DROP RENAME TRUNCATE	테이블과 같은 데이터 구조를 정의하는데 사용되는 명령어(생성, 변경, 삭제, 이름변경) 데이터 구조와 관련된 명령어들을 말함.
데이터 제어어 (DCL : Data Control Language)	GRANT REVOKE	데이터베이스에 접근하고 객체들을 사용하도록 권한을 주고 회수하는 명령어들을 말함.
트랜잭션 제어어 (TCL : Transaction Control Language)	COMMIT ROLLBACK SAVEPOINT	논리적인 작업의 단위를 묶어서 DML에 의해 조작된 결과를 작업단위(트랜잭션) 별로 제어하는 명령어를 말함.

CREATE, READ, UPDATE, DELETE

명령어	설명
INSERT INTO	데이터 삽입
SELECT	데이터 조회
UPDATE	데이터 수정
DELETE	데이터 삭제

기본 SELECT 문

```
SELECT *
FROM  table;
```

- SELECT는 표시할 대상열을 지정합니다.
- FROM은 대상 열을 포함하는 해당 테이블을 지정합니다.

SQL 문 작성

- SQL 문은 대소문자를 구분하지 않습니다.
- SQL 문은 하나 이상의 줄에 입력할 수 있습니다.
- 키워드는 약어로 쓰거나 여러 줄에 나눠 쓸 수 없습니다.
- 절은 일반적으로 서로 다른 줄에 씁니다.

산술식

산술 연산자를 사용하여 숫자 및 날짜 데이터에 대한 표현식을 작성합니다.

연산자	설명
+	더하기
-	빼기
*	곱하기
/	나누기

열 별칭 정의

- 열 머리글의 이름을 변경합니다.
- 계산식에 대한 열머리를 지정할 때 유용합니다.
- 열 이름 바로 뒤에 사용합니다.
- 열 이름과 별칭 사이에 선택적으로 AS 키워드를 사용할 수 있습니다.
- 공백 또는 특수 문자가 있거나 대소문자를 구분할 경우 큰 따옴표를 사용합니다.

연결 연산자

- 열 또는 문자열을 다른 열에 연결합니다.
- 두 개의 세로선(||)으로 표시합니다.
- 문자식에 해당하는 결과 열을 생성합니다.

중복 행 제거

```
SELECT DISTINCT department_id
FROM  employees;
```

- SELECT절에서 DISTINCT 키워드를 사용하여 중복 행을 제거합니다.

선택되는 행 제한

```
SELECT *
FROM  table;
WHERE  condition(s);
```

- WHERE 절을 사용하여 반환되는 행을 제한합니다.
- WHERE 절은 FROM 절 다음에 옵니다.

1. 연봉이 120000 이상되는 사원들의 성씨 및 연봉을 출력하시오

```
SELECT LAST_NAME, SALARY*12
FROM EMPLOYEES
WHERE SALARY*12 >= 120000;
```

2. 사원번호가 176 인 사원의 성씨와 부서 번호를 출력하시오

```
SELECT LAST_NAME, DEPARTMENT_ID
FROM EMPLOYEES
WHERE EMPLOYEE_ID = 176;
```

문자열 및 날짜

- 문자열 및 날짜 값은 작은 따옴표로 묶습니다.
- 문자 값은 대소문자를 구분하며 날짜 값은 날짜 형식을 구분 합니다.
- 기본 날짜 형식은 DD-MON-RR입니다.

비교 조건

연산자	의미
=	같음
>	보다 큼
>=	크거나 같음
<	보다 작음
<=	작거나 같음
<>	같지 않음

연산자	의미
BETWEEN 하한 AND 상한	두 값 사이(지정 값 포함)
IN(set)	값 목록 중의 값과 일치
LIKE	문자 패턴 일치
IS NULL	널 값

논리 조건

연산자	의미
AND	구성 요소 조건이 모두 TRUE이면 TRUE반환
OR	구성 요소 조건 중 하나라도 TRUE이면 TRUE반환
NOT	뒤따르는 조건이 FALSE이면 TRUE반환 TRUE라면 FALSE 반환

우선 순위 규칙

계산 순서	연산자
1	산술 연산자
2	연결 연산자 ||
3	비교 조건
4	IS NULL, LIKE, IN
5	BETWEEN
6	NOT 논리조건
7	AND 논리조건
8	OR 논리조건

ORDER BY 절

```
SELECT last_name, job_id, department_id, hire_date
FROM  employees;
ORDER BY  hire_date;
```

- ORDER BY 절을 사용하여 행을 정렬합니다.
  - ASC : 오름차순, 기본 값 / DESC : 내림차순
- ORDER BY 절은 SELECT 문의 가장 끝에 둡니다.

3. 연봉이 150,000 에서 200,000의 범위 이외인 사원들의 성씨 및 연봉을

출력하시오. 단 연봉은 AnnSal로 출력하시오

```
SELECT LAST_NAME, 12*SALARY AS “AnnSal”
FROM EMPLOYEES
WHERE 12*SALARY NOT BETWEEN 150000 and 200000;
```

4. 2020/01/01 일부터 2025/02/17일 사이에 고용된 사원들의 성씨, 사번,

   고용일자를 출력하시오. 고용일자 순으로 정렬하시오

```
SELECT LAST_NAME, EMPLOYEE_ID, HIRE_DATE
FROM EMPLOYEES
WHERE HIRE_DATE BETWEEN '03/01/01' AND '05/05/30'
ORDER BY HIRE_DATE DESC;
```

5. 20번 혹은 50 번 부서에서 근무하는 모든 사원들의 성씨 및 부서 번호를

알파벳순으로 출력하시오

```
SELECT LAST_NAME, DEPARTMENT_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN ( 20, 50 )
ORDER BY LAST_NAME ASC;
```

6. 20 번 혹은 50 번 부서에 근무하며, 연봉이 200,000 ~ 250,000 사이인 사원

들의 성씨 및 연봉을 출력하시오

```
SELECT LAST_NAME, 12*SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN ( 20, 50 )
AND 12*SALARY BETWEEN 20000 AND 250000;
```

7. 2020년도에 고용된 모든 사람들의 성씨 및 고용일을 조회한다

```
SELECT LAST_NAME, HIRE_DATE
FROM EMPLOYEES
WHERE HIRE_DATE LIKE '2020%';
```

8. 매니저가 없는 사람들의 성씨 및 업무를 출력하시오

```
SELECT LAST_NAME , JOB_ID
FROM EMPLOYEES
WHERE MANAGER_ID IS NULL;
```

9. 매니저가 있는 사람들의 성씨 및 업무, 매니저번호를 조회한다.

```
SELECT LAST_NAME , JOB_ID, MANAGER_ID
FROM EMPLOYEES
WHERE MANAGER_ID IS NOT NULL;
```

10. 커미션을 받는 모든 사원들의 성씨, 연봉 및 커미션을 출력하시오.

- 연봉을 역순으로 정렬하고, 연봉은 ANNSAL로 출력하시오

```
SELECT LAST_NAME, 12*SALARY AS ANNSAL, COMMISSION_PCT
FROM EMPLOYEES
WHERE COMMISSION_PCT IS NOT NULL
ORDER BY ANNSAL  DESC;
```

11. 성씨의 네번째 글자가 a인 사원의 성씨를 조회하시오

```
SELECT LAST_NAME
FROM EMPLOYEES
WHERE LAST_NAME LIKE '___a%';
```

12. 성씨에 a 및 e 글자가 있는 사원의 성씨를 조회하시오

```
SELECT LAST_NAME
FROM EMPLOYEES
WHERE LAST_NAME LIKE '%a%' AND LAST_NAME LIKE '%e%';
```

13. 급여가 2500,3500,7000이 아니며 직업이 SA_REP나 ST_CLERK인 사원의 성씨와,

급여, 직업을 출력하시오.

```
SELECT LAST_NAME, SALARY, JOB_ID
FROM EMPLOYEES
WHERE SALARY NOT IN ( 2500, 3500, 7000 )
AND JOB_ID IN ( 'SA_REP', 'ST_CLERK' );
```

14. 30번 부서 혹은 90번 부서의 모든 직업들을 유일한 값으로 출력하시오.
단, 직업을 오름차순으로 출력하시오

```
SELECT DISTINCT JOB_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN ( 30, 90 )
ORDER BY JOB_ID;
```

그룹 함수

- 단일 행 함수와 달리 그룹함수는 행 집합에 작용하여
  그룹 당 하나의 결과를 생성

그룹 함수 종류

함수	설명
AVG	평균 값(NULL 값은 무시)
COUNT	NULL이 아닌 행의 수 *를 사용하면 중복 행 및 NULL이 있는 행을 포함하여 모든 행을 센다
MAX	최댓값(NULL 값은 무시)
MIN	최솟값(NULL 값은 무시)
STDDEV	표준 편차(NULL 값은 무시)
SUM	합계(NULL 값은 무시)
VARIANCE	분산(NULL 값은 무시)

함수	설명
NVL	NVL(NULL여부 판단 column, NULL일 경우 대체 값)
NVL2	NVL2(NULL여부 판단 column, NOT NULL일 경우 반환 값, NULL일 경우 대체 값)
COALESCE	COALESCE(data1, data2, data3, ...) data1이 NOT NULL일 경우 반환, NULL일 경우 data2를 조회 data2가 NOT NULL일 경우 반환, NULL일 경우 data3을 조회 ... 모든 값이 NULL일 경우 NULL을 반환

SQL 문법 순서

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

SQL 실행 순서

FROM	각 테이블 확인
ON	조인 조건 확인
JOIN	테이블 조인(병합)
WHERE	데이터 추출 조건 확인
GROUP BY	특정 컬럼으로 데이터 그룹화
HAVING	그룹화 이후 데이터 추출 조건 확인
SELECT	데이터 추출
DISTINCT	중복 제거
ORDER BY	데이터 정렬

조인 : 여러 테이블에서 데이터 얻기

- 여러 테이블의 데이터가 필요한 경우 사용
- 관련된 둘 이상의 테이블에 있는 데이터를 표시하기 위해서는
  WHERE 절에서 간단한 조인 조건을 작성
- 여러 테이블에 동일한 열 이름이 있는 경우 열 이름에 테이블 이름을 접두어로 표기

조인의 종류

- 등가 조인
- 비등가 조인
- 포괄 조인
- 자체 조인

15. 모든 사원들의 성씨, 부서 성씨 및 부서 번호를 출력하시오.

```
SELECT E.EMPLOYEE_ID, E. LAST_NAME, D.DEPARTMENT_NAME,
        D.DEPARTMENT_ID
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID;
```

16. 커미션을 받지 않는 모든 사람들의 성씨, 부서 명, 지역 ID 및 도시 명을 출력하시오

```
SELECT E.LAST_NAME, D.DEPARTMENT_NAME, L.LOCATION_ID, L.CITY
FROM  EMPLOYEES E, DEPARTMENTS D, LOCATIONS L
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
AND   D.LOCATION_ID = L.LOCATION_ID
AND   E.COMMISSION_PCT IS NULL;
```

17. 자신의 매니저보다 먼저 고용된 사원들의 성씨 및 고용일을 출력하시오.

```
SELECT  EMP.LAST_NAME, EMP.EMPLOYEE_ID, EMP.HIRE_DATE
FROM   EMPLOYEES EMP, EMPLOYEES MGR
WHERE  EMP.MANAGER_ID = MGR.EMPLOYEE_ID
AND    EMP.HIRE_DATE < MGR.HIRE_DATE
ORDER BY EMP.LAST_NAME;
```

18. 회사 전체의 최대 급여, 최소 급여, 급여 총 합 및 평균 급여를 출력하시오

```
SELECT MAX(SALARY), MIN(SALARY), SUM(SALARY), AVG(SALARY)
FROM EMPLOYEES;
```

19. 각 직업별, 최대 급여, 최소 급여, 급여 총 합 및 평균 급여를 출력하시오.
단 최대 급여는 MAX, 최소 급여는 MIN, 급여 총 합은 SUM 및
평균 급여는 AVG로 출력하고, 직업을 오름차순으로 정렬하시오

```
SELECT  JOB_ID, MAX(SALARY) MAX, MIN(SALARY) MIN ,
        SUM(SALARY) SUM, AVG(SALARY) AVG
FROM   EMPLOYEES
GROUP BY JOB_ID
ORDER BY JOB_ID;
```

20. 동일한 직업을 가진 사원들의 총 수를 출력하시오

```
SELECT  JOB_ID, COUNT(EMPLOYEE_ID)
FROM   EMPLOYEES
GROUP BY JOB_ID
ORDER BY JOB_ID;
```

21. 매니저로 근무하는 사원들의 총 수를 출력하시오

```
SELECT COUNT(DISTINCT MANAGER_ID)
FROM EMPLOYEES;
```

22. 사내의 최대 급여 및 최소 급여의 차이를 출력하시오.

```
SELECT MAX(SALARY) - MIN(SALARY)
FROM EMPLOYEES;
```

23. 매니저의 사번 및 그 매니저 밑 사원들 중 최소 급여를 받는 사원의 급여를 출력하시오
    - 매니저가 없는 사람들은 제외한다.
    - 최소 급여가 5000 미만인 경우는 제외한다.
    - 급여 기준 역순으로 조회한다.

```
SELECT MANAGER_ID, MIN(SALARY)

FROM EMPLOYEES

WHERE MANAGER_ID IS NOT NULL

GROUP BY MANAGER_ID

HAVING NOT MIN(SALARY) < 5000

ORDER BY MIN(SALARY) DESC;
```

24. 부서 명, 부서위치ID, 각 부서 별 사원 총 수, 각 부서 별 평균 급여를
출력하되, 부서위치를 오름차순으로 출력하시오

```
SELECT D.DEPARTMENT_NAME , D.LOCATION_ID, COUNT(E.EMPLOYEE_ID)
        , AVG(E.SALARY) AVG_SALARY
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
GROUP BY D.DEPARTMENT_NAME, D.LOCATION_ID
ORDER BY D.LOCATION_ID;
```

25. Ryan 와 동일한 부서에 근무하는 모든 사원들의 사번 및 고용날짜를 출력하시오.

```
SELECT EMPLOYEE_ID, HIRE_DATE
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN ( SELECT DEPARTMENT_ID
                         FROM EMPLOYEES
                         WHERE LAST_NAME = 'Ryan')
AND LAST_NAME != 'Ryan';
```

26. 회사 전체 평균 급여보다 더 급여를 많이 받는 사원들의 사번 및 성씨를 출력하시오

```
SELECT EMPLOYEE_ID, LAST_NAME
FROM EMPLOYEES
WHERE SALARY > ( SELECT AVG(SALARY)
                 FROM EMPLOYEES);
```

27. 성씨에 u 가 포함되는 사원들과 동일 부서에 근무하는 사원들의
사번 및 성씨를 출력하시오.

```
SELECT EMPLOYEE_ID, LAST_NAME
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN ( SELECT DEPARTMENT_ID
                         FROM EMPLOYEES
                         WHERE LAST_NAME LIKE '%u%');
```

28. 시애틀에 근무하는 사람 중 커미션을 받지않는 모든 사람들의 성씨, 부서 명, 지역 ID를 출력하시오

```
SELECT E.LAST_NAME, D.DEPARTMENT_NAME, D.LOCATION_ID
FROM EMPLOYEES E, DEPARTMENTS D
WHERE E.DEPARTMENT_ID = D.DEPARTMENT_ID
AND D.LOCATION_ID = ( SELECT LOCATION_ID
                      FROM LOCATIONS
                      WHERE CITY = 'Seattle')
AND E.COMMISSION_PCT IS NULL;
```

29. 성씨가 Ryan 인 사람보다 먼저 고용된 사원들의 성씨 및 고용일자를 출력하시오. 고용일자를 역순으로 출력하시오

```
SELECT LAST_NAME, HIRE_DATE
FROM EMPLOYEES
WHERE HIRE_DATE < ( SELECT HIRE_DATE
                    FROM EMPLOYEES
                    WHERE LAST_NAME = 'Ryan')
ORDER BY HIRE_DATE;
```

30. Davis 을 매니저로 두고 있는 모든 사원들의 성씨 및 급여를 출력하시오

```
SELECT LAST_NAME, SALARY, MANAGER_ID
FROM EMPLOYEES
WHERE MANAGER_ID IN ( SELECT EMPLOYEE_ID
                      FROM EMPLOYEES
                      WHERE LAST_NAME = 'Davis');
```

31. 회사 전체 평균급여보다 더 많이 받는 사원들 중 성씨에 u 가 있는 사원들
이 근무하는 부서에서 근무하는 사원들의 사번, 성씨 및 급여를 출력하시오

```
SELECT EMPLOYEE_ID, LAST_NAME, SALARY
FROM EMPLOYEES
WHERE DEPARTMENT_ID IN ( SELECT DEPARTMENT_ID
                         FROM EMPLOYEES
                         WHERE LAST_NAME LIKE '%u%'
                         AND SALARY >= ( SELECT AVG(SALARY)
                                         FROM EMPLOYEES));
```

INSERT 문

```
INSERT INTO table
VALUES (column1 value, column2 value, ..., )
```

- table 뒤에 column명 생략 시 value들은 순서대로 삽입됩니다.

UPDATE 문

```
UPDATE table
SET  column = value
WHERE 조건;
```

- WHERE절은 생략 가능합니다..

DELETE 문

```
DELETE FROM table
WHERE 조건;
```

- WHERE절 생략 시 모든 데이터가 삭제됩니다.
- TABLE 자체를 삭제하고 싶을 경우, DROP TABLE 명령어를 사용합니다.