**로고학습일지**

**kt ds University 자바 기반의 데이터 사이언티스트 양성과정**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 학습일시 | 2018. 08. 02 (목) | 장소 | kt ds University B관 201호 | **시 간** | 09:00~18:00 |
| 학습범위 | UI/UX | | | | |
| 작 성 자 |  | | | **강 사** | 장민창 강사 |

|  |  |
| --- | --- |
| 학습안건 | SQL |

|  |  |
| --- | --- |
| 학습내용 | 내용 |

★ **집계함수**

**100**

**AVG** ( ) 평균값 =75

**50**

**COUNT** ( ) 행 갯수 =4

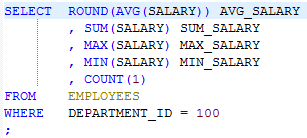
**50**

**MAX** ( ) 최대값 =100

**MIN** ( ) 최소값 =50

**100**

**SUM** ( ) 합계 =300

AVG( ), COUNT( ), SUM ( ) 는 NULL값을 무시

\*를 사용하면 NULL도 포함

SELECT ROUND(AVG(SALARY)) AVG\_SALARY

, SUM(SALARY) SUM\_SALARY

, MAX(SALARY) MAX\_SALARY

, MIN(SALARY) MIN\_SALARY

 , COUNT(1)

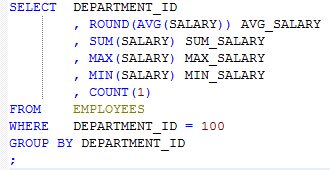
FROM EMPLOYEES

WHERE DEPARTMENT\_ID = 100

;

**이자리에 집계함수가 아닌**

**일반 컬럼을 넣으면 에러가 나서**

★ **데이터 그룹화**

**GROUP BY (CPU의 부하로 잘 사용하지 않음)**

SELECT DEPARTMENT\_ID

, ROUND(AVG(SALARY)) AVG\_SALARY

, SUM(SALARY) SUM\_SALARY

, MAX(SALARY) MAX\_SALARY

, MIN(SALARY) MIN\_SALARY

, COUNT(1)

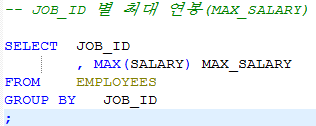
FROM EMPLOYEES

**이자리에 GROUP BY 를 한다**

WHERE DEPARTMENT\_ID = 100

**GROUP BY** DEPARTMENT\_ID

;

-- JOB\_ID 별 최대 연봉(MAX\_SALARY)

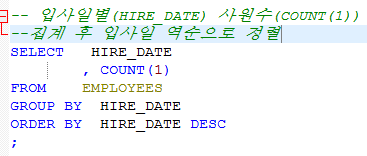
SELECT JOB\_ID

, MAX(SALARY) MAX\_SALARY

FROM EMPLOYEES

GROUP BY JOB\_ID

;

-- 입사일별(HIRE\_DATE) 사원수(COUNT(1))

집계 후 입사일 역순으로 정렬

SELECT HIRE\_DATE

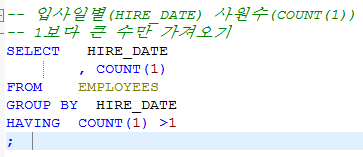
, COUNT(1)

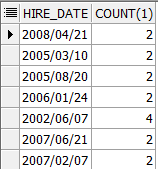
FROM EMPLOYEES

GROUP BY HIRE\_DATE

ORDER BY HIRE\_DATE DESC

;

-- 입사일별(HIRE\_DATE) 사원수(COUNT(1))

-- 1보다 큰 수만 가져오기

SELECT HIRE\_DATE

, COUNT(1)

FROM EMPLOYEES

GROUP BY HIRE\_DATE

**HAVING COUNT** (1) >1

;

**WHERE** 은 **행**을 필터링 하고, (데이터가 그룹화 되기 전에 필터링)

**HAVING** 은 **그룹**을 필터링 한다. (데이터가 그룹화 된 이후에 필터링)

\*하나의 문장에 둘 다 사용하는 경우 있음

EX) 지난 12개월간 2회이상 주문한 적이 있는 고객

> WHERE로 12개월간 주문했던 고객을 필터링.

> HAVING으로 2회이상 주문한 고객을 필터링.

★ **테이블 조인**

CA Erwin Data Modeler 7.3

Z49Y-8805-25R8-W439-66JS

W65DS-P42HT-FDNPL-KHXNV-NWUCA

RDB

------

* Table

**1 : N**

**1 : 1**

Table A ㅡ Table B

----------- -----------

ID(PK) ID(FK)

admin admin

user admin

\*정규화 : 한 테이블에 너무 많은 데이터를 넣으면 조회 능력이 떨어지므로 테이블을 분리, 1:1로 매칭

Logical : 사람이 알아보는 것

Phygical : DB와 같은 구조

**Inner Join, equip Join, Natural Join** (셋 다 같은 말)

-- EMPLOYEES 와 DEPARTMENTS에 공통되는 DEPARTMENT\_ID를 가져와라.

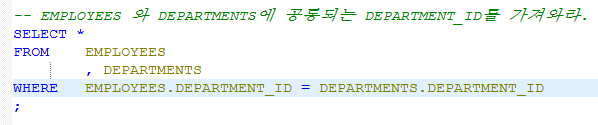
SELECT \*

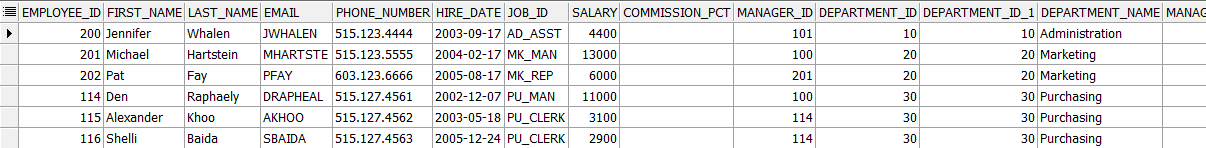
FROM EMPLOYEES

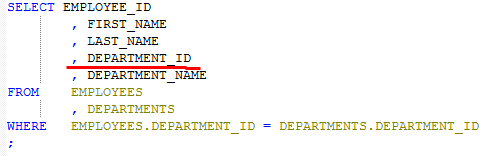
, DEPARTMENTS

WHERE EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.DEPARTMENT\_ID

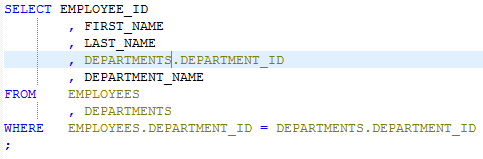
;



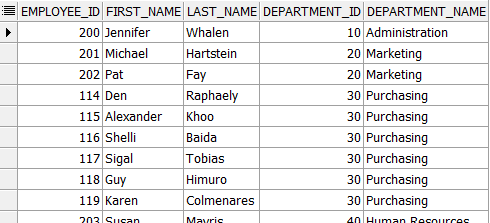




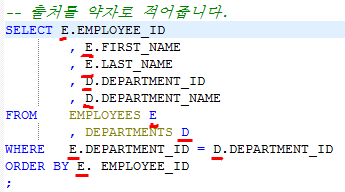
**이 부분이 애매해서 에러발생**



**이렇게 수정하면(출처를 적어준다)**



**결과**



-- 출처를 약자로 적어줍니다.

SELECT E.EMPLOYEE\_ID

, E.FIRST\_NAME

, E.LAST\_NAME

, D.DEPARTMENT\_ID

, D.DEPARTMENT\_NAME

FROM EMPLOYEES E

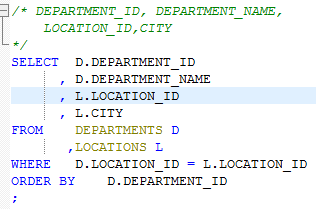
, DEPARTMENTS D

WHERE E.DEPARTMENT\_ID = D.DEPARTMENT\_ID

ORDER BY E. EMPLOYEE\_ID

;





/\* DEPARTMENT\_ID, DEPARTMENT\_NAME,

LOCATION\_ID,CITY

\*/

SELECT D.DEPARTMENT\_ID

, D.DEPARTMENT\_NAME

, L.LOCATION\_ID

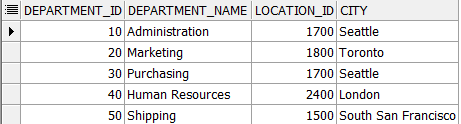
, L.CITY

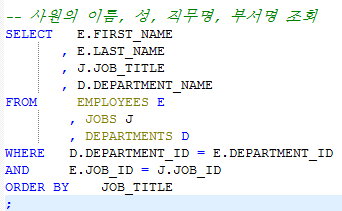
FROM DEPARTMENTS D

,LOCATIONS L

WHERE D.LOCATION\_ID = L.LOCATION\_ID

ORDER BY D.DEPARTMENT\_ID

;



-- 사원의 이름, 성, 직무명, 부서명 조회

SELECT E.FIRST\_NAME

, E.LAST\_NAME

, J.JOB\_TITLE

, D.DEPARTMENT\_NAME

FROM EMPLOYEES E

, JOBS J

, DEPARTMENTS D

WHERE D.DEPARTMENT\_ID = E.DEPARTMENT\_ID

AND E.JOB\_ID = J.JOB\_ID

**WHERE 에**

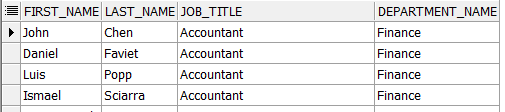
**불러오는 열이 아니어도**

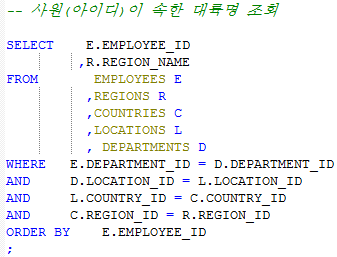
**해당하는 Table에 공통되는(겹치는) column을**

**적어준다**

ORDER BY JOB\_TITLE

;





-- 사원(아이디)이 속한 대륙명 조회

SELECT E.EMPLOYEE\_ID

,R.REGION\_NAME

FROM EMPLOYEES E

,REGIONS R

,COUNTRIES C

,LOCATIONS L

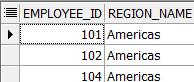
, DEPARTMENTS D

WHERE E.DEPARTMENT\_ID = D.DEPARTMENT\_ID

AND D.LOCATION\_ID = L.LOCATION\_ID

AND L.COUNTRY\_ID = C.COUNTRY\_ID

AND C.REGION\_ID = R.REGION\_ID

ORDER BY E.EMPLOYEE\_ID

**WHERE 에**

**겹치는 칼럼중 KF-KF인**

**MANAGER\_ID는 넣지 않는다.**

;

