**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 | 손에 잡히는 10분 SQL 9-10, 12장 | | | | |
| 작 성 자 | 신예리 | | | **강 사** | 장민창 강사 |

|  |  |
| --- | --- |
| 학습안건 | 데이터 요약, 데이터 그룹핑, 테이블 조인  Erwin Data Modeler |

|  |  |
| --- | --- |
| 학습내용 | 내용 |
| 데아터 요약  >그룹 함수 : AVG() / COUNT() / MAX() / MIN() / SUM() <- (컬럼 or 계산식)  >Null값 무시. COUNT(\*)의 경우 Null값 포함  A {100 50 50 100}일 때, AVG(A)=75 , COUNT(A)=4, MAX(A)=100, MIN(A)=50  >중복 제거 : DINSTINCT  데이터 그룹핑  >집계 함수와 일반 컬럼은 같이 쓰일 수 없다. 🡺 GROUP BY & HAVING  -> 같이 쓰고 싶을 때, SELECT에 있는 일반 컬럼을 GROUP BY에 써준다  ※GROUP BY : 서버에 큰 부담을 줌. CPU 사망 가능 => 잘 사용하지 않음. -> Batch (주기적으로 돌아가는 App)  => 중복된 데이터가 있으면 안될 때 확인 시에만 사용됨.  >집계된 것을 필터링할 때 HAVING (GROUP BY 아래) <- 별칭 말고 집계 함수 그대로 적을 것  ★★ WHERE은 데이터가 그룹화 되기 전에 행을 필터링 / HAVING은 데이터가 그룹화된 후 그룹을 필터링  ERwin Data Modeler 7.3 설치  >라이선스 Key : W65DS-P42HT-FDNPL-KHXNV-NWUCA  >RDB (Relation DataBase) : 여러가지 Table로 구성  -TableA ---------------------------------------------------- TableB  ID(PK) ID(FK)  -admin admin\_\_\_\_  ■ admin\_\_\_\_  -user user\_\_\_\_\_  => ■ : 관계는 1 : 1 (정규화를 위해서만 사용, 만들어서는 X, 따로 사용할 일 없다.)  1 : N (1대다 / 하나가 여러 로우에 참조)  N : N (다대다 / 만들어서는 X, 표현할 수 없다)  -쇼핑몰  옷, 신발, RC카, 생수, 노트북 🡺 상품  손님 N | 1  파는 사람 1 | N  => 기준에 따라 1대다가 달라진다 (손님 기준에 파는 사람은 1:N / 파는 사람 기준 손님 1:N)  -부모자식  부모 1 | 1  자식 1 | N  => 부모 기준 자식 1:N / 자식 기준 부모 1:1  ※ 다대다 N : N  >만들어서는 안 된다.  >표현 할 수 없다.  -이마트 상품 vs 손님  PK 이름 가격 수량 PK 이름 포인트 상품PK  1 애호박 1000 1 1 류복기 1000 1  2 단호박 3000 2 2 정보경 2000 2  ︙ => 표현할 수 없다.  🡺 주문이라는 새로운 테이블이 필요하다!!! 1대다로  주문  PK 상품ID 손님ID  1 1 1  2 1 2  3 2 3  -수강신청 (다대다) : 과목 vs 학생 🡺 ‘수강’이라는 새로운 테이블 필요  PK 이름 정원 교수명 PK 이름 학번 ┗PK 과목ID 학생ID  -커뮤니티 : 게시글 ( N : 1) 회원  > 1:1 1:N <- 1이 있을 때 N이 존재  <- 1이 없어도 N이 존재  > ERwin Data Modeler 7.3  -Create Model > Logical/Physical  -Model > Natation > 둘다 IE로 바꿔주기  -Logical에서 만들면 Physical에 생김  -1대다의 경우 relationship (맨 오른족 점선) 클릭 후 1인 것부터 클릭, 다 인 것 클릭  -※그림에서 동그라미의 존재 중요!! 동그라미가 있는 테이블이 없어도 반대 테이블에 데이터가 존재 가능  🡺 선(그림) 더블 클릭 -> No Nulls 클릭  -PK가 2개인 경우 같은 값이 있을 수 있으나, PK 2개 모두가 동일할 때 중복  -PFK : PK이자 FK인 것 (실선으로 표현)  -재귀 참조 : 스스로를 참조하는 것  -Key Table : 참조를 가장 많이 해주는 Table (HR의 경우 Employees)  -Logical 한글 / Physical 영어…?  테이블 조인 ( = : 같다)  >inner join, equi join, natural join =교집합  SELECT \*  FROM EMPLOYEES  , DEPARTMENTS  WHERE EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.DEPARTMENT\_ID  ;  SELECT EMPLOYEE\_ID  , FIRST\_NAME  , LAST\_NAME  , DEPARTMENT\_ID  , DEPARTMENT\_NAME  FROM EMPLOYEES  , DEPARTMENTS  WHERE EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.DEPARTMENT\_ID  ; // 에러: [Error] Execution (74: 11): ORA-00918: column ambiguously defined (애매한 정의)  🡺 DEPARTMENT\_ID가 중복됨 (PK 인 테이블 이름을 적어줌)  SELECT E.EMPLOYEE\_ID  , E.FIRST\_NAME  , E.LAST\_NAME  , D.DEPARTMENT\_ID  , D.DEPARTMENT\_NAME  FROM EMPLOYEES E  , DEPARTMENTS D  WHERE E.EMPLOYEE\_ID = D.DEPARTMENT\_ID  ;  🡺 별칭 사용 가능 |

|  |  |
| --- | --- |
| 특이사항 | 7기 수강생 블로그 http://cocomo.tistory.com |

■ORACLE

--집계함수

SELECT ROUND( AVG(SALARY), 0 ) AVG\_SALARY

, SUM(SALARY) SUM\_SALARY

, MAX(SALARY) MAX\_SALARY

, MIN(SALARY) MIN\_SALARY

, COUNT(SALARY) CNT\_SALARY

, COUNT(COMMISSION\_PCT) CNT\_COMMISSION --NULL은 제외

, COUNT(\*) CNT\_ALL --NULL 포함

, COUNT(1) --이렇게 많이 쓰임

FROM EMPLOYEES

;

SELECT 1

FROM EMPLOYEES

;

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

;

SELECT DEPARTMENT\_ID

, ROUND(AVG(SALARY)) AVG\_SALARY

, SUM(SALARY) SUM\_SALARY

, MAX(SALARY) MAX\_SALARY

, MIN(SALARY) MIN\_SALARY

, COUNT(1)

FROM EMPLOYEES

GROUP BY DEPARTMENT\_ID

ORDER BY DEPARTMENT\_ID

;

--에러남. EMPLOYEE\_ID로 그룹핑된 것이 아니기 때문에

SELECT DEPARTMENT\_ID

, 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

GROUP BY EMPLOYEE\_ID

;

--JOB\_ID별 최대연봉 구하기

SELECT JOB\_ID

, MAX(SALARY) MAX\_SALARY

FROM EMPLOYEES

GROUP BY JOB\_ID

;

SELECT HIRE\_DATE

, COUNT(1)

FROM EMPLOYEES

GROUP BY HIRE\_DATE

HAVING COUNT(1) > 1

ORDER BY HIRE\_DATE

;

--INNER JOIN, EQUI JOIN, NATURAL JOIN

SELECT \*

FROM EMPLOYEES

, DEPARTMENTS

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

;

--에러남. DEPARTMENT\_ID가 중복됨 -> PK 테이블 이름 지정해줄 것

SELECT EMPLOYEE\_ID

, FIRST\_NAME

, LAST\_NAME

, DEPARTMENT\_ID

, DEPARTMENT\_NAME

FROM EMPLOYEES

, DEPARTMENTS

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

;

SELECT EMPLOYEE\_ID

, FIRST\_NAME

, LAST\_NAME

, DEPARTMENTS.DEPARTMENT\_ID

, DEPARTMENT\_NAME

FROM EMPLOYEES

, DEPARTMENTS

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

;

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

;

SELECT E.EMPLOYEE\_ID

, E.FIRST\_NAME

, E.LAST\_NAME

, J.JOB\_ID

, J.JOB\_TITLE

FROM EMPLOYEES E

, JOBS J

WHERE E.JOB\_ID = J.JOB\_ID

ORDER BY E.EMPLOYEE\_ID

;

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 E.JOB\_ID = J.JOB\_ID

AND E.DEPARTMENT\_ID = D.DEPARTMENT\_ID

ORDER BY J.JOB\_TITLE

;

--사원이 속한 대륙명 조회

SELECT E.EMPLOYEE\_ID

, R.REGION\_NAME

FROM EMPLOYEES E

, DEPARTMENTS D

, LOCATIONS L

, COUNTRIES C

, REGIONS R

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

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

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

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

;

■SQL 문제

-- 1번 부서번호가 10번인 부서의 사람 중 사원번호, 이름, 월급을 출력하라.

SELECT EMPLOYEE\_ID

, FIRST\_NAME

, LAST\_NAME

, SALARY

FROM EMPLOYEES

WHERE DEPARTMENT\_ID = 10

;

-- 2번 사원번호가 130인 사람 중 이름, 입사일, 부서번호를 출력하라.

SELECT FIRST\_NAME

, LAST\_NAME

, HIRE\_DATE

, DEPARTMENT\_ID

FROM EMPLOYEES

WHERE EMPLOYEE\_ID = 130

;

-- 3번 이름이 ALLEN인 사람의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE FIRST\_NAME = 'Allen'

OR LAST\_NAME = 'Allen'

;

-- 4번 입사일이 2007/01/12인 사원의 이름, 부서번호, 월급을 출력하라.

SELECT FIRST\_NAME

, LAST\_NAME

, DEPARTMENT\_ID

, SALARY

FROM EMPLOYEES

WHERE HIRE\_DATE = TO\_DATE('2007/01/14', 'YYYY-MM-DD')

;

-- 5번 직업이 MANAGER가 아닌 사람의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES E

, JOBS J

WHERE J.JOB\_ID = E.JOB\_ID

AND J.JOB\_TITLE NOT LIKE '%Manager'

;

-- 6번 입사일이 2007/01/14 이후에 입사한 사원의 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE HIRE\_DATE > TO\_DATE('2007/01/14', 'YYYY-MM-DD')

;

-- 7번 급여가 $8000 이상인 사람의 이름, 급여, 부서번호를 출력하라.

SELECT FIRST\_NAME

, LAST\_NAME

, SALARY

, DEPARTMENT\_ID

FROM EMPLOYEES

WHERE SALARY > 8000

;

-- 8번 부서번호가 20번 이상인 사원의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE DEPARTMENT\_ID >= 20

;

-- 9번 이름이 K로 시작하는 사람보다 높은 이름을 가진 사람의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE FIRST\_NAME > 'K'

ORDER BY FIRST\_NAME

;

-- 10번 입사일이 2007/01/14 보다 먼저 입사한 사람들의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE HIRE\_DATE < TO\_DATE('2007/01/14', 'YYYY-MM-DD')

;

-- 11번 사원번호가 보다 작거나 같은 사람들의 입사번호와 이름을 출력하라.

SELECT EMPLOYEE\_ID

, FIRST\_NAME

, LAST\_NAME

FROM EMPLOYEES

WHERE EMPLOYEE\_ID <= 200

;

-- 12번 입사일이 2007/01/14보다 늦고 2009/01/14보다 빠른 사원의 이름, 월급, 부서번호를 출력하라.

SELECT FIRST\_NAME

, LAST\_NAME

, SALARY

, DEPARTMENT\_ID

, HIRE\_DATE

FROM EMPLOYEES

WHERE HIRE\_DATE > TO\_DATE('2007/01/14', 'YYYY-MM-DD')

AND HIRE\_DATE < TO\_DATE('2009/01/14', 'YYYY-MM-DD')

;

-- 13번 급여가 $16000보다 크고 $23,000보다 작은 사람의 이름, 직업, 급여를 출력하라.

SELECT E.FIRST\_NAME

, E.LAST\_NAME

, J.JOB\_TITLE

, E.SALARY

FROM EMPLOYEES E

, JOBS J

WHERE E.SALARY > 16000

AND E.SALARY < 23000

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

;

-- 14번 사원번호가 100과 200 사이 이외의 사원의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE EMPLOYEE\_ID < 100

OR EMPLOYEE\_ID > 200

;

-- 15번 이름이 B와 J사이의 모든 사원의 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE (FIRST\_NAME > 'B' AND FIRST\_NAME < 'J')

OR (LAST\_NAME > 'B' AND LAST\_NAME < 'J')

;

-- 16번 입사일이 2003년 이외의 입사한 사람의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE HIRE\_DATE < TO\_DATE('2003/01/01', 'YYYY-MM-DD')

OR HIRE\_DATE > TO\_DATE('2003/12/31', 'YYYY-MM-DD')

;

-- 17번 직업이 MANAGER와 SALESMAN인 사람의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES E

, JOBS J

WHERE E.JOB\_ID = J.JOB\_ID

AND J.JOB\_TITLE LIKE '%Manager'

OR J.JOB\_TITLE LIKE '%Salesman'

;

-- 18번 부서번호롸 20, 30번을 제외한 모든 사람의 이름, 사원번호, 부서번호를 출력하라.

SELECT FIRST\_NAME

, LAST\_NAME

, EMPLOYEE\_ID

, DEPARTMENT\_ID

FROM EMPLOYEES

WHERE DEPARTMENT\_ID != 20

AND DEPARTMENT\_ID != 30

;

-- 19번 S로 시작하는 사원의 사원번호, 이름, 입사일, 부서번호를 출력하라.

SELECT EMPLOYEE\_ID

, FIRST\_NAME

, LAST\_NAME

, HIRE\_DATE

, DEPARTMENT\_ID

FROM EMPLOYEES

WHERE FIRST\_NAME LIKE 'S%'

;

-- 20번 입사일이 2008년도인 사람의 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE HIRE\_DATE > TO\_DATE('2008/01/01', 'YYYY-MM-DD')

AND HIRE\_DATE < TO\_DATE('2008/12/31', 'YYYY-MM-DD')

;

-- 21번 이름 중 S자가 들어가 있는 사람만 모든 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE FIRST\_NAME LIKE '%S%'

OR LAST\_NAME LIKE '%S%'

;

-- 22번 이름이 S로 시작하고 마지막 글자가 T인 사람의 모든 정보를 출력하라(단, 이름은 전체 5자리이다.)

SELECT \*

FROM EMPLOYEES

WHERE FIRST\_NAME LIKE 'S\_\_\_T'

OR LAST\_NAME LIKE 'S\_\_\_T'

;

-- 23번 첫 번쨰 문자는 관계없고, 두 번쨰 문자가 a인 사람의 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE FIRST\_NAME LIKE '\_a%'

;

-- 24번 커미션이 NULL인 사람의 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE COMMISSION\_PCT IS NULL

;

-- 25번 커미션이 NULL이 아닌 사람의 정보를 출력하라.

SELECT \*

FROM EMPLOYEES

WHERE COMMISSION\_PCT IS NOT NULL

;