**로고학습일지**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 학습일시 | 2018. 08. 01 (수) | 장소 | kt ds University B관 201호 | **시 간** | 09:00~18:00 |
| 학습범위 | DB | | | | |
| 작 성 자 | 권석현 | | | **강 사** | 장민창 강사 |

|  |  |
| --- | --- |
| 학습안건 | DB 쿼리문, 연습문제 25번까지 |

|  |  |
| --- | --- |
| 학습내용 | 내용 |
| SQL 합계 함수  AVG() // 컬럼의 평균값을 반환한다.  SELECT ROUND ( AVG(SALARY), 0 )  FROM EMPLOYEES  ;  //평균연봉  COUNT() // 컬럼에 있는 행 개수를 반환한다.  MAX() // 컬럼의 최대값 반환  MIN() // 컬럼의 최소값 반환  SUM() // 컬럼의 합계 반환  SELECT ROUND ( AVG(SALARY), 0 ) AS AVG\_SALARY  , SUM( SALARY ) SUM\_SALARY  , MAX( SALARY ) MAX\_SALARY  , MIN( SALARY ) MIN\_SALARY  , COUNT( SALARY ) COUNT\_SALARY  , COUNT( COMMISSION\_PCT ) CNT\_COMMISSION  , COUNT( \* ) CNT\_ALL //전체  , COUNT( 1 ) // 전체  FROM EMPLOYEES  ;  실무에선 \*보다 1을 더 많이씀.  SELECT 1  FROM EMPLOYEES  ;  // EMPLOYEES의 로우 수만큼 1이 출력된다.  //괄호안에 인자 들어감.  AVG()함수는 NULL값을 가진 행은 무시한다.  MAX()무시  SUM()무시  COUNT() 무시하지만 아스터 써서 사용가능  데이터 그룹핑  -GROUP BY  잘안씀.  1. 정말안씀.  2. 데이터 양이 많을때 쓰면 DB서버가 다운될수도 있음.  3. 시스템 자원을 엄청 많이 씀.  그래서 DB -> BATCH프로그램(집계) -> DB  DB -> HADOOP -> SPARK -> DB  EX)  SELECT JOB\_ID  , MAX( SALARY ) AS MAX\_SALARY  FROM EMPLOYEES  GROUP BY JOB\_ID  집계 된것을 필터링 할때.  HAVING을 쓴다.  컬럼명을 쓰지 않는다. 집계함수를 써야함 해빙절엔  //중복체크 (실무에서 많이 씀)  SELECT HIRE\_DATE  ,COUNT(1)  FROM EMPLOYEES  GROUP BY HIRE\_DATE  HAVING COUNT(1) > 1  ;  WHERE는 행을 필터링하고 HAVING는 그룹을 필터링 한다.  WHERE은 데이터가 그룹화 되기 전에 필터링한다.  HAVING은 데이터가 그룹화된 후에 필터링한다.  er다이어그램  erwin 7.3  erwindatamodeller 클릭 하면 키 입력하라고 나옴  거기에 W65DS P42HT FDNPL KHXNV NWUCA  위에꺼 클릭 후 붙여넣기 후 ADD 후 CONTINUE  킨다음  관계의 종류  1:1, 1:N , N:N  1:1과 N:N은 좋지않다.  다대다 관계는 1:N 관계로 풀어줄 테이블이 필요하다. (물리적으로 구현할 수 없어서)  TOOLS -> OPTIONS  로지컬 만들고 피지컬 만들어야된다. 로지컬 -> 사람이 보는것(한글) , 피지컬 -> 데베 양식 (영어)  자기테이블을 스스로 참조하는것 -> 재귀참조  참조가 가장 많은 테이블 --> 키 테이블 (1 쪽이제일 많은것)  /\*  INNER JOIN, EQUIP JOIN, NATURAL JOIN 이라고 함(교집합)  \*/  SELECT \*  FROM EMPLOYEES  , DEPARTMENTS  WHERE EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.DEPARTMENT\_ID  연결되서 정보 모두 출력  SELECT EMPLOYEE\_ID  , FIRST\_NAME  , LAST\_NAME  , DEPARTMENT\_ID  , DEPARTMENT\_NAME  FROM EMPLOYEES  , DEPARTMENTS  WHERE EMPLOYEES.DEPARTMENT\_ID = DEPARTMENTS.DEPARTMENT\_ID  이렇게하면 오류남  DEPARTMENT\_NAME이 두 테이블에 다 있어서 어디것인지 명시 해주어야함.  SELECT EMPLOYEE\_ID  , FIRST\_NAME  , LAST\_NAME  , D.DEPARTMENT\_ID  , DEPARTMENT\_NAME  FROM EMPLOYEES E  , DEPARTMENTS D  WHERE E.DEPARTMENT\_ID = D.DEPARTMENT\_ID  다쓰기 힘드니 저렇게 이름을 다르게 정의해서 편리하게 사용 가능.  SELECT D.DEPARTMENT\_ID  , D.DEPARTMENT\_NAME  , D.LOCATION\_ID  , L.CITY  FROM DEPARTMENTS D  , LOCATIONS L  WHERE L.LOCATION\_ID = D.LOCATION\_ID  ORDER BY D.DEPARTMENT\_ID  ;  ERD만 잘 그리면 조인은 줄 따라가고 1의 PK가 N쪽의 FK와 같다만 정의하면 된다.  --1번 부서번호가 1-번인 부서의 사람의 사원번호, 이름, 월급을 출력하라  SELECT EMPLOYEE\_ID  , FIRST\_NAME  , SALARY  FROM EMPLOYEES  WHERE DEPARTMENT\_ID = 10  ORDER BY EMPLOYEE\_ID  ;  --2번 사번이130인 사람의 이름, 입사일, 부서번호를 출력  SELECT FIRST\_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 2007/01/14인 사원의 이름, 부서번호, 월급을 출력  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')  ORDER BY HIRE\_DATE  ;  -- 7번 급여가 $800 이상인 사람의 이름, 급여, 부서번호를 출력하라.  SELECT FIRST\_NAME  , LAST\_NAME  , SALARY  , DEPARTMENT\_ID  FROM EMPLOYEES  WHERE SALARY > 800  ORDER BY SALARY  ;  -- 8번 부서번호가 20번 이상인 사원의 모든 정보를 출력하라.  SELECT \*  FROM EMPLOYEES  WHERE DEPARTMENT\_ID > 20  ORDER BY EMPLOYEE\_ID  ;  -- 9번 이름이 K로 시작하는 사람보다 높은 이름을 가진 사람의 모든 정보를 출력하라.  SELECT \*  FROM EMPLOYEES  WHERE FIRST\_NAME > 'L'  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' )  ORDER BY HIRE\_DATE  ;  -- 11번 사원번호가 200보다 작거나 같은 사람들의 사원번호와 이름을 출력하라.  SELECT EMPLOYEE\_ID  , FIRST\_NAME  , LAST\_NAME  FROM EMPLOYEES  WHERE EMPLOYEE\_ID <= 200  ORDER BY EMPLOYEE\_ID  ;  -- 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' )  ORDER BY HIRE\_DATE  ;  -- 13번 급여가 $16000보다 크고 $23000보다 작은 사람의 이름, 직무명, 급여를 출력하라.  SELECT E.FIRST\_NAME  , E.LAST\_NAME  , E.SALARY  , J.JOB\_TITLE  FROM EMPLOYEES E  , JOBS J  WHERE SALARY > 16000  AND SALARY < 23000  AND J.JOB\_ID = E.JOB\_ID  ;  -- 14번 사원번호가 100와 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'  ORDER BY FIRST\_NAME  ;  -- 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' )  ORDER BY HIRE\_DATE  ;  -- 17번 직업이 MANAGER와 SALESMAN인 사람의 모든 정보를 출력하라.  SELECT \*  FROM EMPLOYEES E, JOBS J  WHERE J.JOB\_ID = E.JOB\_ID  AND JOB\_TITLE LIKE '%Manager'  OR 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 NOT IN( 20, 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 FIRST\_NAME LIKE '%s'  OR FIRST\_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\_\_\_\_n'  ;  -- 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; |

|  |  |
| --- | --- |
| 특이사항 |  |