\*\* 오라클 다운로드 및 설치

* [www.oracle.com](http://www.oracle.com)에서 다운로드
* 11g, 12c, 18c, 19c 버전이 있음
  + 11g와 12c는 설치를 할 때 sid를 이용해서 설치
  + 18c와 19c는 설치를 할 때 service name을 이용해서 설치
* sid는 오라클을 여러 개 설치할 때 구분하기 위한 이름
* 중복된 이름은 안됩니다.
* 기본 이름은 orcl(standard 또는 enterprise 버전) 이나 xe(express 버전)
* 기존에 설치가 되어 있는데 또 설치하면 sid가 중복되서 다른 이름을 입력해야 합니다.
* 공부를 할 때는 sid를 변경하지 않지만 실무에서는 변경을 합니다.
* 기본 포트는 1521번으로 설정됩니다.
  + 이전에 다른 오라클이 설치되어 있으면 1씩 증가합니다.
  + 8080 포트도 오라클이 사용합니다.
  + 오라클이 설치된 컴퓨터에서는 1521이나 8080을 함부로 사용하면 안됩니다.
* 실무에서는 포트번호도 변경을 합니다.
* 다른 데이터베이스와 달리 Oracle 이나 IBM DB2, Tibero는 개발자가 실무에서 설치할 가능성은 거의 없고 대부분은 판매한 기업에서 설치를 해줍니다.
  + 비밀번호만 수정해서 사용합니다.
  + 설치하는 운영체제도 Windows가 아니고 Linux 나 Unix 일 가능성이 높습니다.
* 설치를 하고나면 확인은 콘솔에서 sqlplus라고 입력해서 접속도구가 실행되는지 system계정으로 로그인 되는지 확인

\*\* Oracle, MySQL, MSSQL 과 같은 프로그램은 부팅이 되면 자동 실행됩니다.

* 서버라면 문제가 안되지만 공부를 위해서 데이터베이스를 설치한 경우 자동 실행되면 메모리를 많이 사용합니다.
* Windows의 경우는 서비스를 필요할 때만 실행하고 그렇지 않은 경우는 서비스를 중지해 두는 것이 좋습니다.
* 오라클의 경우는 오라클 서비스가 있고 TNS Listener가 있습니다.
  + 오라클 서비스는 오라클 자체를 중지하거나 실행시키는 것이고 TNS Listener는 외부에서 접속을 허용하는 서비스 입니다.

\*\* 샘플 데이터 이용

* 계정의 lock을 해제
* 관리자 계정은 sys와 system 으로 제공
* 샘플 계정은 scott과 hr 입니다.
  + 오라클 버전에 따라 scott이나 hr이 막혀있거나 없는 경우가 있습니다.
  + 11g에서는 막혀 있어서 scott 이나 hr을 사용할려면 계정의 lock을 풀어줘야 합니다.
* sys나 system으로 로그인을 한 후
  + alter user scott identified by tiger account unlock;
  + tiger는 비밀번호로 다른 것을 입력해도 됩니다.

\*\* 데이터베이스 접속 도구를 이용해서 접속

* 오라클의 sqlplus는 오라클이 설치된 컴퓨터에서만 사용할 수 있고 command line 기반이고 작업 내용이 바로 반영됩니다.
* 오라클이 설치되지 않은 컴퓨터나 oracle 명령어에 익숙하지 않은 사람이 사용하기에는 불편합니다.
* **sqlplus에서 작업할 때는 메모장과 같은 edito에 sql을 입력하고 확인한 후 복사해서 실행해야 합니다.**
* DBA(데이터베이스 관리자)를 제외하고는 sqlplus를 사용하지 않고 별도의 접속 프로그램을 이용해서 작업을 수행합니다.
* 오라클 접속도구는 많은데 금용에서 많이 사용하는 것은 toad, orange 이고 공부를 할 때는 sqldeveloper(java로 만들어진 오라클에서 제공하는 프로그램)를 많이 사용합니다.
  + 우리는 dbeaver를 사용하는데 하나만 설치하면 오라클, mysql 등 여러가지 데이터베이스를 접속할 수 있음

\*\* dbeaver를 이용한 오라클 접속

* 오라클 접속 드라이버 준비 : 오라클 웹 사이트에서 다운로드 받을 수 있고 오라클이 설치된 경우는 오라클 설치 디렉토리에서 가져올 수 있습니다.
  + 오라클 설치디렉토리에서 jdbc(java 와 데이터베이스 연동 기술) 디렉토리에 있습니다.
  + 드라이버에 붙는 숫자는 자바 버전입니다.
  + 지금은 6버전 이상을 사용합니다.(14는 1.4버전입니다.)

\*\* SCOTT에 제공되는 샘플 테이블

* emp, dept, salgrade, bonus 테이블이 제공됩니다.
* 이 테이블을 확인
  + 왼쪽 창에서 테이블을 더블 클릭하면 테이블 구조가 grid로 보여집니다.
  + 테이블의 구조를 확인하고 select 구문을 학습
    - NUMBER – 숫자
    - char, varchar2 – 문자열
    - date, time, timestamp – 날짜 및 시간
    - clob – 긴 문자열
    - blob – 파일의 내용 : 바이트 배열
    - 명령으로 확인할 때는 DESC 테이블이름;

\*\* select 구문

1. 기본구조

* select\* 또는 컬럼이름나열
* from 테이블이름
* where 조건
  + select와 from은 필수고 where은 선택
  + from 절의 테이블을 where는 행 단위로 분할(selection)하고 select는 열 단위로 분할(projection)
  + 테이블의 모든 컬럼을 확인할 때는 컬럼이름 대신에 \*
  + 중복된 데이터를 제거할 때는 select 절의 맨 앞에 distinct 기재

1. where 조건
   1. 크기 비교 연산자 사용
   * >, >=, <, <= : 숫자, 날짜, 문자열 모두 크기 비교가 가능
   1. 동일성 여부 판단 연산자
   * =(같다), != 또는 <> 다르다
   1. 문자열을 이용해서 날짜 만들기
   * to\_date(‘문자열’, ‘날짜서식’)
   * y : 년도
   * mm : 월
   * dd : 일
   * hh, hh24 : 시간
   * mi : 분
   * ss : 초
   1. 현재 시간
   * sysdate
   1. and 와 or를 이용해서 2개의 조건을 결합가능
   2. between A and B
   * A 와 B 사이로 A와 B를 포함
   * A가 B보다 작거나 같아야 함

--EMP 테이블에서 SAL 값이 1000-3000 사이인 데이터의 모든 컬럼을 조회 : BETWEEN을 이용

**SELECT**\*

**FROM** EMP

**WHERE** SAL **BETWEEN** 1000 **AND** 3000;

--EMP 테이블에서 HIREDATE 가 1981년에 속한 데이터의 ENAME과 HIREDATE 모두 조회 : BETWEEN을 이용

**SELECT** ENAME, HIREDATE

**FROM** EMP

**WHERE** HIREDATE **BETWEEN** **TO\_DATE**('1981-01-01', 'yyyy-mm-dd') **AND** **TO\_DATE**('1981-12-31', 'yyyy-mm-dd');

* 1. IN
  + IN(값의 리스트)를 이용하면 값의 리스트 안에 속한 모든 데이터가 리턴
    - OR로 대체 가능한 경우가 있지만 값의 리스트가 sub query인 경우에는 or로 대체가 안됨
    - 값의 리스트가 직접 입력한 값이면 OR로 대체 가능
  + SQL에서는 정답은 없고 원하는 결과만 나오면 답이 되는데 상황에 따라 속도가 다르므로 이 부분을 확인

--EMP 테이블에서 JOB이 CLECK 이거나 SALESMAN 인 데이터의 ENAME과 SAL의 값을 조회

**SELECT** ENAME, SAL

**FROM** EMP

**WHERE** JOB = 'CLERK' **OR** JOB = 'SALESMAN';

**SELECT** ENAME, SAL

**FROM** EMP

**WHERE** JOB **IN** ('CLERK', 'SALESMAN');

* 1. NULL 검색
  + IS NULL로 검색

--EMP 테이블에서 COMM 컬럼의 값이 NULL인 데이터의 ENAME과 SAL 그리고 COMM을 조회

**SELECT** ENAME, SAL, COMM

**FROM** EMP

**WHERE** COMM **IS** **NULL**;

* 1. LIKE
  + 부분 일치하는 데이터를 검색하는데 사용하는 연산자
  + WILDCARD 문자
    - \_ : 하나의 글자와 대치
    - % : 글자 수 상관없이 대치
    - \_\_\_ : 3글자
    - A% : A로 시작하는 모든 문자열
    - %A : A로 끝나는 모든 문자열

--EMP 테이블에서 ENAME이 J로 시작하는 데이터의 ENAME과 MGR을 조회

**SELECT** ENAME, MGR

**FROM** EMP

**WHERE** ENAME **LIKE** 'J%';

--EMP 테이블에서 LL이 포함된 데이터의 ENAME과 DEPTNO를 조회

**SELECT** ENAME, DEPTNO

**FROM** EMP

**WHERE** ENAME **LIKE** '%LL%';

--EMP 테이블에서 ENAME에 N로 끝나는 6자의 이름을 가진 데이터의 ENAME과 DEPTNO를 조회

**SELECT** ENAME, DEPTNO

**FROM** EMP

**WHERE** ENAME **LIKE** '\_\_\_\_\_N';

* 1. 검색 엔진은 검색할 때 공백을 기준으로 2개의 단어를 입력하면 2개의 단어가 포함된 데이터를 찾고 그리고 그 뒤에 한 개의 단어만 포함된 데이터를 조회해서 붙임
  + HELLO WORLD 라고 입력
    - SPLIT을 해서 HELLO, WORLD로 분할
    - SEARCH LIKE ‘%HELLO%’ AND ‘%WORLD%’
    - SEARCH LIKE ‘%HELLO%’
    - SEARCH LIKE ‘%WORLD%’
  1. NOT
  + NOT BETWEEN A AND B : A에서 B 사이가 아닌
  + NOT IN(목록) : 목록에 속하지 않은
  + NOT LIKE ‘패턴’ : 패턴과 일치하지 않은
  + IS NOT NULL : NULL이 아닌 데이터

--EMP 테이블에서 COMM의 값이 NULL 아닌 데이터의 ENAME과 JOB을 조회

**SELECT** ENAME, JOB

**FROM** EMP

**WHERE** COMM **IS** **NOT** **NULL**;

1. ORDER BY

* SELECT 구문의 마지막 절이고 마지막에 실행
* 관계형 데이터베이스에 데이터를 저장하면 컬럼의 순서나 행의 순서는 아무런 의미가 없습니다.
* 별도의 인덱싱(트라이나 B\*TREE)을 이용해서 데이터를 저장하기 때문입니다.
* 2개 이상의 컬럼을 조회할 때는 SELECT 절에서 컬럼의 순서를 정해주는 것이 좋고 2개 이상의 행을 조회할 때는 ORDER BY를 이용해서 정렬 조건을 설정해주는 것이 좋습니다.
* 정렬을 할 때는
  + ORDER BY 컬럼이름이나 연산식 [ASC | DESC], 다른 정렬 조건
  + ASC는 오름차순이고 DESC는 내림차순인데 생략하면 ASC
  + 정렬 조건을 2개 이상 설정하면 앞의 조건이 동일한 경우 적용
* ORDER BY는 마지막에 수행되기 때문에 SELECT 절에서 만든 별명을 사용할 수 있습니다.

--EMP 테이블의 데이터를 ENAME의 오름차순으로 정렬해서 전부 출력

**SELECT**\*

**FROM** EMP

**ORDER** **BY** ENAME **ASC**;

--EMP 테이블의 데이터를 SAL\*12 + COMM의 내림차순으로 정렬해서 ENAME과 계산된 값을 출력

--COMM의 값이 NULL이면 0으로 간주

**SELECT** ENAME, SAL\*12+**NVL**(COMM, 0) **AS** 연봉

**FROM** EMP

**ORDER** **BY** 연봉 **DESC**;

* 컬럼 이름 대신에 select 절에 입력한 컬럼의 순번을 정해도 됩니다.
* 프로그래밍 언어는 인덱스가 0부터 시작하지만 데이터베이스 인덱스가 1부터 시작합니다.

**SELECT** ENAME, SAL\*12+**NVL**(COMM, 0) **AS** 연봉

**FROM** EMP

**ORDER** **BY** 2 **DESC**;

\*\* 연습문제

1. EMP 테이블에서 EMPNO가 7788 인 데이터의 ENAME과 DEPTNO를 조회

**SELECT** ENAME, DEPTNO

**FROM** EMP

**WHERE** EMPNO = 7788;

1. EMP 테이블에서 DEPTNO 10 또는 20인데이터의 모든 컬럼을 조회
   1. 단 HIREDATE의 내림차순으로 정렬

**SELECT**\*

**FROM** EMP

--WHERE DEPTNO IN(10,20)

**WHERE** DEPTNO = 10 **OR** DEPTNO = 20

**ORDER** **BY** HIREDATE **DESC**;

1. EMP 테이블에서 COMM이 SAL 보다 많은 데이터의 ENAME, SAL, COMM을 조회

**SELECT** ENAME, SAL, COMM

**FROM** EMP

**WHERE** COMM > SAL;

1. EMP 테이블에서 ENAME에 L이 2개 이상 포함되어 있고 DEPTNO가 30인 데이터 또는 MGR이 7782인 데이터의 모든 컬럼을 조회

**SELECT**\*

**FROM** EMP

**WHERE** ENAME **LIKE** '%L%L%' **AND** DEPTNO = 30 **OR** MGR = 7782;

1. BOARD 테이블이 있고 이 테이블에는 TITEL(제목)과 CONTENT(내용) 컬럼이 존재

* BOARD 테이블에서 제목이나 내용에 이벤트라는 단어가 포함된 데이터의 모든 컬럼을 조회하는 SQL

**SELECT**\*

**FROM** BOARD

**WHERE** TITLE **LIKE** '%이벤트%' **OR** CONTENT **LIKE** '%이벤트%';

\*\* 데이터 분류

1. 기본형과 참조형

* 기본형은 데이터의 값을 저장하고 참조형은 데이터의 참조를 저장

1. 데이터의 개수에 따른 분류
   1. scala data : 1개의 데이터
   2. vector data : 0개 이상의 데이터
   * 1개의 데이터가 할 수 있는 일이 있고 0개 이상의 데이터를 가지고 할 수 있는 작업이 다르기 때문입니다.
2. 정형과 비정형
   1. 정형 : 모양이 일정한 데이터 – 테이블 구조로 만들어지는 데이터
   2. 비정형 : 모양이 일정하지 않은 데이터 – 음성이나 사운드처럼 테이블 구조로 만들기는 어렵고 행렬구조로 만드는 데이터
   3. 반정형 : 비정형처럼 생겼는데 정형으로 변환이 가능한 데이터 – 센서

\*\* 오라클에서 제공하는 함수(Maker Function)

* 데이터베이스에서는 함수가 반드시 리턴을 해야 합니다.

1. 함수의 종류
   1. 단일 행 함수 : 함수의 수행 결과가 1개인 함수
   2. 다중 행 함수 : 함수의 수행 결과가 1개 이상인 함수
2. DUAL

* oracal에서 제공하는 가상의 테이블
* 테이블에 존재하지 않는 데이터를 출력해 볼 목적으로 만든 테이블
* sysdate는 현재 시간을 가지고 있는데 확인이 필요할 때는 dual 테이블을 이용해야 합니다.
  + sysdate는 실제 테이블에 존재하지 않기 때문입니다.
* sequence 값 확인할 때도 이용

--현재 시간 확인

**SELECT** **SYSDATE**

**FROM** DUAL;

1. 숫자 관련 함수

* 행 단위로 작업해서 행의 개수 만큼 리턴해주는 단일 행 함수
* ROUND 함수 : 반올림 해주는 함수
  + ROUND(데이터, 자릿수) : 자릿수를 생략하면 소수 전체를 반올림
  + 자릿수에 양수를 대입하면 양수에 해당하는 자리까지 반올림을 해주고 음수를 대입하면 정수 부분을 반올림

--EMP 테이블에서 ENAME과 SAL을 조회

**SELECT** ENAME, SAL

**FROM** EMP;

--SAL의 데이터를 소수 없이 표시

**SELECT** ENAME, **ROUND**(SAL,0)

**FROM** EMP;

--SAL의 값을 10자리 반올림해서 표시

**SELECT** ENAME, **ROUND**(SAL,-2)

**FROM** EMP;

* 비슷한 역할을 수행하는 FLOOR, TRUNC 라는 함수도 있음

1. 문자 관련 함수

* 영문자 비교할 때는 대소문자를 구분할 것인지 구분하지 않을 것인지 고민
* CHAR를 사용했을 때 뒤의 공백 문제
  1. LOWER 와 UPPER
  + 모두 소문자나 대문자로 변경해주는 함수
  1. TRIM
  + 좌우 공백을 제거해주는 함수
  + 공백 뿐 아니라 옵션으로 다른 문자를 대입해서 문자를 제거할 수도 있습니다.
  + 왼쪽이나 오른쪽에서만 제거하는 LTRIM과 RTRIM 함수도 존재

--EMP 테이블에서 ENAME이 king인 사원의 ENAME과 JOB을 조회

--대소문자 구분없이 조회

**SELECT** ENAME, JOB

**FROM** EMP

**WHERE** **LOWER**(ENAME) = 'king';

-- ' ORACLE '에서 좌우 공백을 제거하고 조회

**SELECT** **TRIM**(' ORACLE ')

**FROM** DUAL;

* 1. LENTGH, LENGTHB
  + 문자열의 길이를 리턴하는 메소드인데 LENGTH는 글자의 개수를 LENGTHB는 바이트 수를 리턴
  + 한글 1글자는 글자 수로는 1이지만 바이트 수로는 2나 3이 됩니다.
    - 오라클에서는 보통의 경우는 2

-- 한글의 글자 수와 바이트 수를 확인

**SELECT** **LENGTH**('안녕하세요'), **LENGTHB**('안녕하세요')

**FROM** DUAL;

결과는 LENGTH는 5 LENGTHB는 15입니다.

UTF-8에서는 한글은 3BYTE 입니다.

테이블을 생성할 때 한글을 삽입하는 컬럼은 글자수\*3해서 설정해야 합니다.

* 1. SUBSTR, SUBSTRB
  + 문자열에서 위치를 가지고 문자열을 분할해주는 함수
  + 매개변수로 숫자를 1개 또는 2개를 설정
  + 1개일 때는 시작위치 2개일 때는 시작위치와 가져올 개수
    - 다른 언어나 데이터베이스에서 이런 함수를 사용할 때는 2번째 정수를 확인해야 합니다.
    - 프로그램에 따라 2번째 정수가 개수가 아니라 종료 다음위치인 경우가 있습니다.
  + 많은 양의 문자열이 있을 때 앞에서 몇 개만 추출하고자 할 때 많이 사용
  + YES24처럼 도서에 대한 검색을 하게 되면 목차나 서평을 출력해주는 이 항목들은 많은 문자열인데 모든 유저가 전부 이 부분을 확인하지 않습니다. 이런 경우에는 일부분만 가지고 가서 보여주고 클릭하면 나중에 전체를 넘겨주면 됩니다.
  + 스마트 폰의 경우는 화면이 작기 때문에 불필요한 데이터를 처음부터 가지고 가서 출력할 필요가 없습니다.

--EMP 테이블에서 ENAME의 첫 3글자만 조회

**SELECT** **SUBSTR**(ENAME, 1, 3)

**FROM** EMP;

* 1. 형 변환 함수
  + 숫자 <-> 문자 <-> 날짜
    1. TO\_CHAR
       - 숫자를 입력하면 문자로 변환
       - 날짜를 대입할 때는 뒤에 변환서식을 지정
       - yyyy : 년도를 4자리로
       - yy : 년도를 2자리로
       - mm : 월을 2자리로
       - mon : 월을 영문자 약자로
       - day : 요일을 표현
       - dy : 요일을 약자로 표현
       - dd : 일을 2자리로 표현
       - hh : 시간을 2자리로 표현 - 12시간제
       - hh24 : 시간을 2자리로 표현 - 24시간제
       - mi : 분
       - ss : 초
  + 개발자들은 날짜나 숫자를 문자로 변환하는 작업을 잘 하지 않습니다.
  + 개발자는 프로그래밍 언어를 통해서 데이터베이스를 사용하기 때문에 이러한 변환작업을 프로그램에서 하려고 합니다.
    1. TO\_DATE
       - 문자열을 날짜로 변환
       - TO\_DATE(‘문자열’, ‘날짜서식’)
    2. TO\_NUMBER
       - 문자열을 숫자로 변환
  1. NVL
  + 자바에서는 NULL 과 연산을 하면 예외가 발생합니다.
  + 데이터베이스에서는 NULL과 연산을 하면 결과는 무조건 NULL
  + 데이터에 NULL이 있을 때 NULL을 치환하는 함수가 NVL
  + NVL(데이터, 대체값) : 데이터에 NULL이 있는 경우 대체값을 변환해서 수행

\*\* 그룹함수

* 여러 개를 가지고 연산해서 결과를 하나의 행으로 리턴하는 함수
* 집계함수라고도 합니다.

1. 함수의 종류
   1. SUM
   2. AVG
   3. COUNT
   4. MAX
   5. MIN
   6. STDDEV : 표준 편차
   7. VARIANCE : 분산
      * 표준 편차나 분산을 구하는 이유는 데이터의 분포를 알아보기 위해서
      * 그룹 함수는 GROUP BY 절에서 사용한 조건과 함께하는 경우가 아니라면 단독으로만 출력
      * 그룹화하지 않은 컬럼과는 같이 출력할 수 없습니다.
2. SUM은 합계
   * SUM에 대입된 컬럼 또는 연산식의 합계를 리턴
3. AVG는 평균
   * AVG는 NULL인 데이터는 제외하고 평균을 구함

--EMP 테이블에서 COMM의 컬럼의 합계와 평균을 조회

--NVL을 사용하면 전체 데이터의 평균을 만들지만 NVL을 사용하지 않으면 NULL아닌 데이터의 평균이 됩니다.

**SELECT** **SUM**(COMM), **AVG**(COMM), **AVG**(**NVL**(COMM, 0))

**FROM** EMP;

결과 : SUM 2,200 AVG(COMM) : 550, AVG(NVL(COMM,0)) : 157.1428571429

1. MAX, MIN
   * 최대값과 최소값을 구해주는 함수
2. COUNT
   * 데이터의 개수를 구해주는 함수
   * NULL 인 데이터를 제외하고 데이터의 개수를 구해줍니다.
   * COUNT에는 컬럼 이름 대신에 \*을 넣는 경우가 있습니다.
     + 전체 데이터 개수가 됩니다.

--EMP 테이블의 전체 데이터 개수와 COMM 컬럼의 데이터 개수 조회

**SELECT** **COUNT**(\*), **COUNT**(COMM)

**FROM** EMP;

결과 : COUNT(\*)의 경우는 테이블의 전체 데이터 개수가 14가 나오지만 COUNT(COMM)의 경우는 COMM 컬럼에서 NULL이 아닌 4개 리턴

1. 집계함수와 일반 컬럼은 같이 조회할 수 없습니다.

--EMP 테이블에서 SAL이 가장 큰 데이터의 ENAME과 SAL을 조회 : 에러

--ENAME 컬럼이 그룹화되어 있지 않으므로 집계함수와 같이 출력을 못합니다.

**SELECT** ENAME, **MAX**(SAL)

**FROM** EMP;

1. GROUP BY
   * SELECT 구문에서 데이터를 그룹화하기 위해 사용하는 절
     + 4 - SELECT \* 또는 조회할 컬럼이름 또는 연산식 나열
     + 1- FROM 조회할 테이블이름 나열
     + 2 - WHERE 조회할 조건 나열
     + 3 - GROUP BY 그룹화 할 컬럼이름 또는 연산식 나열
     + 5 - ORDER BY 정렬할 컬럼이름 또는 연산식 [ASC | DESC]

--EMP 테이블에서 JOB의 중복을 제거하고 JOB을 출력

**SELECT** **DISTINCT** JOB

**FROM** EMP;

--JOB 별로 그룹화를 하게되면 JOB별로 묶이게 되서 중복은 제거 됩니다.

**SELECT** JOB

**FROM** EMP

**GROUP** **BY** JOB;

--EMP 테이블에서 JOB 별로 SAL의 평균을 조회

**SELECT** JOB, **ROUND**(**AVG**(SAL))

**FROM** EMP

**GROUP** **BY** JOB

**ORDER** **BY** **ROUND**(**AVG**(SAL)) **DESC**;

--GROUP BY 절의 컬럼은 집계함수와만 같이 출력, 다른 컬럼과는 같이 출력을 못함 - 에러

**SELECT** JOB, ENAME, **ROUND**(**AVG**(SAL))

**FROM** EMP

**GROUP** **BY** JOB

**ORDER** **BY** **ROUND**(**AVG**(SAL)) **DESC**;

1. HAVING
   * GROUP BY 이후의 조건 설정을 하기 위한 절
   * 실행 순서 때문에 GROUP BY 이후의 조건을 WHERE에 기재할 수 없습니다.
   * WHERE 절에는 집계함수를 사용할 수 없습니다.
     + 집계함수는 GROUP BY 이후에 수행

--EMP 테이블에서 JOB의 개수가 2개 이상인 데이터의 SAL의 평균

--WHERE 절에는 집계함수를 사용할 수 없음 - 에러

**SELECT** JOB, **ROUND**(**AVG**(SAL))

**FROM** EMP

--WHERE COUNT(JOB) >= 2 - 에러

**GROUP** **BY** JOB

**HAVING** **COUNT**(JOB) >= 2

**ORDER** **BY** **ROUND**(**AVG**(SAL)) **DESC**;

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING

(6) ORDER BY

이 순서를 기억해서 데이터를 필터링 할려고 할 때는 되도록 앞에서 필터링을 해야 합니다.

\*\* 연습문제

1. EMP 테이블에서 JOB 별 인원 수를 조회

**SELECT** JOB, **COUNT**(JOB)

**FROM** EMP

**GROUP** **BY** JOB;

1. EMP 테이블에서 HIREDATE에서 년도가 동일한 인원 수를 조회

**SELECT** **TO\_CHAR**(HIREDATE, 'yyyy'), **COUNT**(\*)

**FROM** EMP

**GROUP** **BY** **TO\_CHAR**(HIREDATE, 'yyyy');

* TO\_CHAR 함수를 이용해서 년도만 가져온 후 그룹화해서 해결
* SUBSTR을 이용해도 되는데 이 때는 날짜가 저장된 서식을 알아야 합니다.
* 년도와 월을 그룹화 하는 것을 많이 합니다.

***\*\* JOIN(존나게 외워라)***

* 2개의 테이블을 합쳐서 하나의 테이블로 만드는 것입니다.
* FROM 절에 테이블 이름을 2개 이상 기재하면 JOIN이 발생합니다.

1. CROSS JOIN – CARTESIAN PRODUCT
   * FROM 절에 테이블 이름을 2개 이상 기재하고 JOIN 조건을 만들지 않은 경우
   * 2개 테이블의 모든 조합을 가지고 새로운 테이블을 생성
   * 열(컬럼)의 개수는 양쪽 테이블의 열의 개수의 합
   * 행의 개수는 양쪽 테이블 행의 개수의 곱
2. EMP 테이블의 확인

SELECT\*

FROM EMP;

8개의 열과 14개의 행으로 구성

1. DEPT 테이블 확인

SELECT\*

FROM DEPT;

3개의 열과 4개의 행으로 구성

1. EMP 테이블과 DEPT 테이블의 모든 조합

SELECT \*

FROM EMP, DEPT;

11개의 열과 56개의 행으로 구성

1. EQUI JOIN
   * 양쪽 테이블의 존재하는 동일한 의미의 열(컬럼)의 값이 일치하는 경우에만 결합에 참여시키는 것
   * WHERE 절에 동일한 의미의 컬럼이 값이 같다 라는 조건을 추가합니다.
   * 양쪽 테이블의 열(컬럼)이름이 같다면 열(컬럼)이름 앞에 테이블 일므을 추가해야 합니다.
     1. EMP 테이블에는 DEPTNO 컬럼이 부서 번호를 나타내는 컬럼이고

DEPT 테이블에서 DEPTNO 컬럼이 부서 번호를 나타내는 컬럼입니다.

* + 1. EMP 테이블과 DEPT 테이블에서 DEPTNO가 동일한 데이터만 조회
    2. SELECT \*

FROM EMP, DEPT

WHERE EMP.DEPTNO = DEPT.DEPTNO;

* + 1. 회원 테이블에 아이디와 이름이 있고 게시판 테이블에서 작성자 아이디가 있고 게시글 정보가 있는 경우 게시글 정보와 이름을 같이 출력할 때 회원 테이블과 게시판 테이블을 아이디를 가지고 조인해서 데이터를 생성해야 합니다.

1. 테이블이름에 별명 사용 가능
   * FROM 절에서 테이블이름을 기재할 때 한칸 공백을 주고 별명을 입력하면 되고 이후에는 테이블 이름 대신에 별명을 사용합니다.
2. SELECT\*

FROM EMP E, DEPT D

WHER E.DEPTNO = D.DEPTNO;

별명을 부여하게 되면 이후에는 계속 별명만 사용해야 합니다.

별명을 사용하는 이유는 코드를 줄여서 사용하기 위한 경우가 있고 코드를 알아보기 쉽게 하기 위해서 사용하는 경우도 있습니다.

프로그래밍에서 FINAL FIELD 나 ENUM을 사용하는 경우가 별명을 사용하는 것과 동일한 목적

스레드의 우선 순위는 1부터 10까지 있습니다.

static final int MAX\_PRIORITY = 10

우선순위 설정할 때 10이나 MAX\_PRIORITY로 설정가능