21.05.18

오라클 DB설치하는 방법

<https://www.oracle.com/database/technologies/xe-downloads.html>에서 윈도우 버전 설치

🡪설치한 후에 cmd-sqlplus-id : system, pw : 1234입력했을 때 떠야 정상

(계정 비밀번호를 잃어 버렸을 땐 🡪 <https://m.blog.naver.com/baekmg1988/221298569322)-->>내용 상에서 sys를 system으로 바꿔서!

**alter session set container = XEPDB1;  
(만질 필요가 없음)  
-18c에만 있는 세션개념(예전 버전에는 없음)  
-세션 당 계정을 만들어야 한다.  
-세션 안에 계정들이 존재하는 거임**

**# create user java1(id) identified by 1234(pw);  
-java1이 아이디, 비밀번호 1234로 DB 사용자 계정 생성**

**grant create session to 계정이름;**

**grant connect to 계정이름;**

**grant resource to 계정이름;**

🡪SQL 계정에 필요한 권한들을 부여한다

\*show user : 현재 로그인한 계정을 보여줌

\*quit : sqlplus 나가기

#connect java1**(id)**/1234**(pw)**@localhost:1521**(포트번호)**/XEPDB1; : 만든 계정으로 접속할 수 있게 함  
connect java1/1234@localhost:1521/XEPDB1;

#alter user hr account unlock;  
-연습용 hr계정 잠금해제하기  
-오라클의 기본 연습용 계정

#exit(DB를 종료한다)

Sqldeveloper(GUI) 다운 받기

<https://www.oracle.com/tools/downloads/sqldev-downloads.html>에서 윈도우 버전으로 다운

**\*SQL로 Git 사용하기!!**

<https://git-scm.com/>에서 다운로드 받기(그리고 쭉쭉 next 해주면 됨 설치 뒤에 마지막에 Launch Git Bash를 체크하고 finish

그 후에 <https://desktop.github.com/>에서 다운 그리고 github(xoghkscc, 9513574!As)로 로그인

File-new local repo 후에 수정 후에는 Repository에서 푸쉬를 하면 됨(기존에 있는 레포를 추가하려면 local)

21.05.20

#GUI 환경에서 오라클 DB를 편하게 다루기 위해서 SQL Developer를 설치한다

**select\*from all\_users : 사용자 계정 확인**

#Oracle 접속

텍스트이(가) 표시된 사진

자동 생성된 설명

#관리자 계정 로그인

(비번 1234)

**#컨트롤 + 엔터 : 실행**

\*계정 비밀번호를 바꾸거나 잃어버렸을 때는 관리자 계정으로 들어가서 (**alter user 계정 identified by 비밀번호;**)을 치고 컨트롤+엔터를 누르면 됨

(sql 여러 줄 주석은 /\* \*/, 한 줄 주석 --)

\*단축키

새 행 삽입 : 쉬프트+엔터

선택된 줄 복사 : 컨트롤+쉬프트+D

자동완성 : 컨트롤 + 스페이스

자동정렬 : 컨트롤 + F7

**#데이터베이스란?**

* 데이터를 효율적으로 저장하기 위한 데이터 저장 전문 프로그램
* 파일 시스템의 많은 문제점과 한계점을 극복하기 위해 만든 프로그램  
  (보안성, 일관성, 저장효율 등…)

**# 파일 시스템의 문제점**

* 데이터 불일치가 발생할 수 있음  
  (ex : 헬스장 회원 정보.txt / 수영장 회원 정보.txt /PT회원 정보.txt 중 하나만 수정되는 경우)
* 다수 사용자를 위한 동시 제공이 불가능하다
* 중복 데이터를 필요 이상으로 많이 저장하게 될 수 있다.
* 파일 수정 도중 시스템에 문제가 발생했을 경우에 복구하는 기능이 없음

위 문제가 발생하지 않는 간단한 기능 구현에는 파일 시스템을 이용하는 것도 좋다 (ex : 환경설정)

**#DBMS(DataBase Management System)**

* 데이터베이스의 데이터를 관리하기 위해 필요한 프로그램
* 데이터베이스는 데이터를 보관하는 장소이고, 데이터 추가/수정/삭제/검색은 DBMS가 하는 작업이다.

**#RDBMS(Relational DataBase Management System)**

* 데이터간의 관계를 이용하여 데이터의 중복을 최소화 하는 **DBMS**
* 질의문(Query)으로 데이터를 관리한다., 쿼리문은 국제 표준을 따른다

**# 테이블(Table)**

* 관계형 데이터베이스는 데이터를 표 형태로 저장한다
* 필드 : 속성(attribute), 컬럼(column), 한 열(세로)에 저장될 데이터들의 특징을 저장하는 곳
* 레코드 : 튜플(tuple), 행(가로, row), 한 행에 저장되는 관련된 데이터들

**#NO-SQL**

* 관계형 데이터베이스의 문제점을 회피하기 위해 존재하는 DBMS
* 데이터의 중복을 해결하지 않는 대신 속도가 빠르다.

\*(좌측 접속에서 우클릭-워크시트 열기 누르면 새로운 코드 작성 판이 나타남)

**#SELECT 컬럼명 FROM 테이블명;**

* 데이터를 조회하는 쿼리문
* 컬럼명에 \*를 쓰는 것은 모든 컬럼을 뜻한다.
* **쿼리문도 대소문자를 구분하지 않는다.**
* **컬럼명/테이블명은 대소문자를 구분하지 않는다.**
* **데이터는 대소문자를 구분한다 (\*중요)**
* 컬럼명 as 바꿀 컬럼명이름🡪을 하면 출력될 때 바뀐 컬럼명 이름으로 출력 가능  
  **텍스트이(가) 표시된 사진

  자동 생성된 설명**

\* desc 테이블명 : 존재하는 컬럼을 보여줌

\*Select \* from **tab**(or **tabs**) : 어떤 테이블들이 있는지 확인할 수 있다.

\* \*를 적으면 모든 컬럼을 조회하고, 컬럼명을 적으면 해당 컬럼만 조회한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**# NUMBER(n), NUMBER(n,m)**

* 숫자 데이터만 저장할 수 있는 컬럼
* 숫자 하나만 적혀있으면 정수를 저장, 숫자가 두개 적혀있으면 실수를 저장
* NUMBER(10) : 10자리 정수를 저장하는 컬럼이 된다
* **NUMBER(10, 4) : 6자리 정수와 소수점 4자리를 저장하는 컬럼이 된다**

**# VARCHAR2(n)**

* 가변 길이 문자 데이터를 저장하는 컬럼 타입
* 저장되는 데이터의 크기에 따라 알맞은 공간만 사용한다
* VARCHAR2(10)으로 설정된 컬럼의 데이터로 2글자가 들어오면 2바이트만 사용한다.
* 가변적인 데이터를 저장할 때는 데이터가 절약되지만 크기 계산이 필요해 속도가 약간 느리다.

**# CHAR(n)**

* 고정 길이 문자 데이터를 저장하는 컬럼 타입
* 설정된 기링보다 적은 양의 데이터가 들어오더라도 설정된 길이를 모두 차지한다
* CHAR(10)으로 설정된 컬럼의 데이터로 2글자가 들어오면 10바이트를 모두 사용한다.
* 데이터가 낭비될 수는 있지만 불필요한 크기 계산이 없기 때문에 속도가 더 빠르다(VARCHAR2보다 더 빠르다)

**# DATE**

* 날짜 및 시간을 저장하는 컬럼 타입

\*산술 연산자를 이용해 계산된 값 출력하기  
select last\_name, salary+500 from employees;

**\*산술 연산할 때 null값이 있는 경우 nvl**🡪해당 컬럼에서 null값이 나왔을 때 대체값을 사용한다.  
select last\_name, salary \* (1 + nvl(commission\_pct, 0)) from employees;  
🡪commision\_pct에는 일부 NULL값이 존재하는데 이때 그 NULL값을 0으로 대체한다는 의미

**\*select distinct : 중복되는 내용이 여러 개 있어도 한번만 출력된다.  
🡪중복되는 직업은 한번만 출력**

**#SELECT 컬럼명 FROM 테이블명 WHERE 조건절;**

* SELECT문에 WHERE절을 추가하여 해당 조건을 만족하는 데이터만 조회할 수 있다  
  🡪월급이 15000달러인 사람들만 조회

**#오라클의 비교 연산자들**

* = : 같다
* <, >, <=, >= : 비교
* !=, <>, ^= : 다르다

\*오라클의 문자열 리터럴은 ‘’만 사용한다  
🡪성이 ‘Steven’인 사람들만 조사

\*날짜 타입도 가볍게 비교가 가능하다  
🡪입사일이 적힌 날짜보다 적힌 데이터들을 조회

\*and, or, not  
  

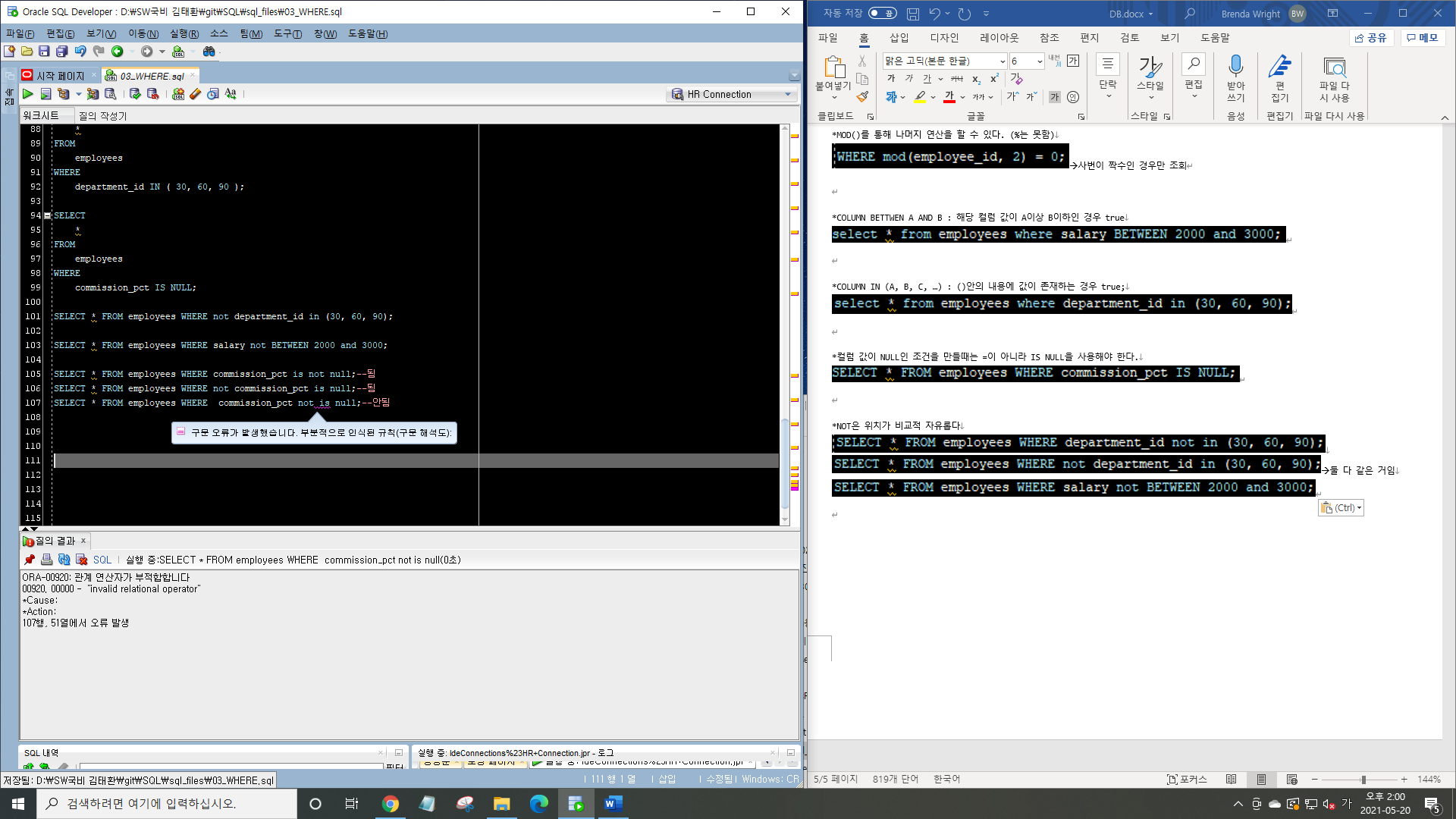

\*MOD()를 통해 나머지 연산을 할 수 있다. (%는 못함)  
🡪사번이 짝수인 경우만 조회

\*COLUMN BETTWEN A AND B : 해당 컬럼 값이 A이상 B이하인 경우 true  


\*COLUMN IN (A, B, C, …) : ()안의 내용에 값이 존재하는 경우 true;  


\*컬럼 값이 NULL인 조건을 만들때는 =이 아니라 IS NULL을 사용해야 한다.  


\*NOT은 위치가 비교적 자유롭다  
  
🡪둘 다 같은 거임  


(not is는 안됨)

**#LIKE**

* **데이터의 일부분으로 원하는 내용을 검색할 수 있다.**
* 문자 타입과 날짜 타입에 사용할 수 있다
* % : 길이 제한 없이 아무 문자열이나 와도 되는 와일드카드  
  (심지어 없는 것도 가능)
* \_ : 하나의 아무 문자가 반드시 와야하는 와일드카드  
     
  텍스트, 벽, 실내이(가) 표시된 사진

  자동 생성된 설명🡪마지막꺼는 유용하게 쓸 듯  
  텍스트, 화면, 검은색이(가) 표시된 사진

  자동 생성된 설명

**\* NOT LIKE도 가능함!**

\*UNION :합집합(중복은 알아서 제거됨)🡪단 같은 테이블끼리 해야함(단 같은 열을 지니고 있으면 다른 테이블도 가능)

텍스트, 오렌지, 실내, 화면이(가) 표시된 사진

자동 생성된 설명

\*UNION ALL : 합집합(중복 제거 안함)  
텍스트, 오렌지, 설정, 스크린샷이(가) 표시된 사진

자동 생성된 설명

\*INTERSECT : 교집합  
텍스트, 실내, 오렌지, 설정이(가) 표시된 사진

자동 생성된 설명

\*MINUM : 차집합  
텍스트, 오렌지, 화면, 설정이(가) 표시된 사진

자동 생성된 설명  
🡪즉 10~40 – 30~50이므로 10~29이라는 의미(30제외)

텍스트이(가) 표시된 사진

자동 생성된 설명

**🡪이렇게 만들면 타 컴퓨터와 연동된 서버 가능(단 기준이 되는 컴퓨터는 항상 켜놔야 접속이 가능함)**

AWS 원격 접속하는 방법 참고(https://docs.oracle.com/cd/E17781\_01/server.112/e18804/network.htm#ADMQS157)  
(aws홈페이지 보안그룹해서 인바운드 규칙에 대해서 포트 범위를 인정해줘야함

EXEC DBMS\_XDB.SETLISTENERLOCALACCESS(FALSE);🡪입력

21.05.21

**#dual 테이블**

* 연습용 테이블
* 하나의 행만 가지고 있음

\*ABC(n) : n의 절대값

\*FLOOR(d) : 내림

\*ROUND(n) : 반올림

\*CEIL(n) : 올림

\*MOD(n, m) : 나머지 연산

\*TRUNC(d, m) : 지정한 자리 값 이하를 잘라낸다  
텍스트이(가) 표시된 사진

자동 생성된 설명

**\*sysdate : 오늘 날짜(단 TRUNC로 표시하고 싶은 곳을 정할 수 있음)**  
텍스트이(가) 표시된 사진

자동 생성된 설명

\*TO\_CHAR로 DATE타입 데이터를 원하는 포맷으로 변경하기  
텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

\*LOWER(‘STRING’) : 소문자로 바꾸기, UPPER(‘string’) : 대문자로 바꾸기  


\*SUBSTR : 문자열 자르기(자바와는 달리 첫 인덱스는 1임)  


\*LENGTH(‘STRING’) : 문자열의 문자 개수 반환(공백까지도 포함)  

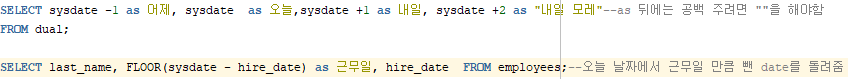

\*LPAD(원본, 길이, 채울문자)-왼쪽에 채우고 원본, RPAD(원본, 길이, 채울문자)-원본 채우고 그리고 채울문자  
🡪채울 문자가 공백인 경우 생략 가능  


\*LTRIM(원본, 제거할 문자), RTRIM(원본, 제거할 문자)  
🡪제거할 문자가 공백인 경우 생략 가능  
텍스트이(가) 표시된 사진

자동 생성된 설명

\*TRIM(제거할 문자, 원본) : 앞 뒤의 특정 문자를 제거한다  


**\* as 뒤에는 공백 주려면 ""을 해야함**

**\*날짜 연산하기(날짜끼리 연산하면 두 날짜 사이의 시간을 구할 수 있다)**  
🡪 테이블이(가) 표시된 사진

자동 생성된 설명

\*MONTHS\_BETWEEN(피감소날짜, 감소날짜) : 두 날짜 사이의 개월 수를 구한다  
텍스트이(가) 표시된 사진

자동 생성된 설명

\*ADD\_MONTHS(date, n) : DATE타입 데이터에 원하는 개월 수를 더한다  


\*NEXT\_DAY(date, 요일) : DATE타입 데이터를 기준으로 다음 요일을 찾는다  


**🡪즉 sysdate -n은 일자를 연산하기 때문에 월이나 요일 같은 경우는 따로 함수를 써야한다**

\*LAST\_DAY(date) : DATE타입 데이터를 기준으로 해당 달의 마지막 날짜를 반환한다  


**#오라클의 날짜 포맷**

* YYYY : 년도
* YY : 년도(2자리)
* MM : 월
* MON : 월을 문자로
* DD : 날짜
* DAY : 요일
* DY : 요일(줄임)
* HH, HH12 : 12시
* HH24 : 24시
* MI : 분
* SS : 초
* AM, PM : 오전, 오후 표시(현재 시각이 오전이라면 PM이라고 써도 오전이라고 나옴



**#오라클의 숫자 포맷**

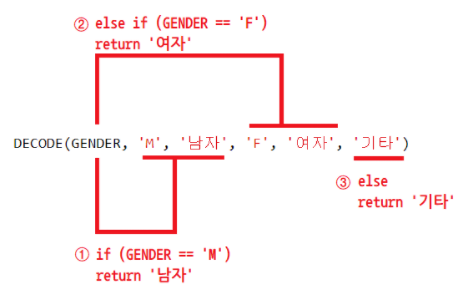
* 0 : 자릿수를 나타낸다. 숫자가 비면 0으로 채운다
* 9 : 자릿수를 나타낸다. 숫자가 비면 빈칸으로 채운다.
* L : 각 지역별 통화 기호를 붙인다
* . : 소수점을 출력한다
* , : 천 단위를 구분하는 쉼표를 출력한다.

**\*TO\_CHAR : 숫자를 문자로 바꾸거나, 날짜를 문자로 바꾼다, 바꿀 때 형식을 이용할 수 있어서 편리하다.**



**\*TO\_NUMBER : 문자를 숫자로 변환. 포맷을 이용해 다시 숫자로 변환할 수 있다**

**#DECODE(컬럼명, 조건1, 결과1, 조건2, 결과2, ……)**

* 특정 컬럼의 값을 통해 표시할 데이터를 선택한다
* Switch case 같은 역할을 한다**(단 LIKE 구문 쓰지 못함)**

텍스트이(가) 표시된 사진

자동 생성된 설명

🡪department\_id가 90이면 Executive, 60이면 soso를 department\_name이라는 컬럼에 나타남(해당하지 않는 id는 (null)이 나타남

텍스트이(가) 표시된 사진

자동 생성된 설명

**#DECODE는 해당 컬럼의 값과 정확하게 일치하는 값만 이용할 수 있지만 CASE문은 조건을 사용할 수 있다  
(DECODE와 달리 LIKE 구문 사용 가능)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**#그룹 함수(GROUP BY)**

* 여러 행의 값을 특정 컬럼을 기준으로 그룹화 하는 함수들
* 특정 집단의 총합, 개수, 평균 등을 구할 수 있다
* GROUP BY에 여러 컬럼을 설정할 수도 있다
* 그룹의 기준이 되는 컬럼은 GROUP BY절을 통해 선택(안해도 됨)
* 그룹 함수의 결과는 일반 컬럼과 함께 출력될 수 없다  
  🡪 텍스트이(가) 표시된 사진

  자동 생성된 설명🡪job\_ib는 group by의 기준이 되기 때문에 가능하나 salary는 안됨
* SUM(colum) : 그룹 별로 해당 컬럼의 총 합을 구한다  
  
* AVG(column) : 평균  
  
* MAX(column) : 최대값  
  
* MIN(column) : 최소값
* COUNT(column) : 개수(단 count는 \*이 가능)  
  

테이블이(가) 표시된 사진

자동 생성된 설명  
텍스트이(가) 표시된 사진

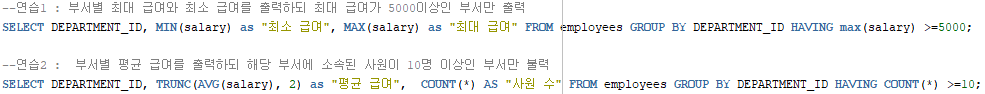
자동 생성된 설명





**🡪WHERE은 그룹이 생성되기 전에 모든 행에 적용되는 조건이다**

\*GROUP BY로 나눠진 그룹에 조건을 주고 싶을 때는 HAVING절을 이용한다  
**(즉 일반 쿼리에 대한 조건문은 WHERE이 되지만 그룹 함수에 대한 조건문은 HAVING으로 해야함!)**  

텍스트이(가) 표시된 사진

자동 생성된 설명  
🡪방법 1은 WHERE을 통해 애초에 출력을 해당하는 직급만 출력하도록 하는 방법  
방법 2는 HAVING을 통해 인상한 월급과 기존 월급과의 평균의 차가 0 이상인 직급들의 평균만 출력(결과는 방법 1과 같음)

**\*GROUP BY는 2개 이상도 가능함!**

21.05.24

\*기본적으로는 PK 순서로 조회된다.

**# ORDER BY절(단 ASC나 DESC를 생략하면 ASC로 자동 적용됨)**

* 원하는 컬럼 기준으로 정렬하여 조회할 수 있다
* ORDER BY 컬럼명 [ASC or DESC]
* ASC : ASCENDING, 오름차순(생략 가능)
* DESC : DESCENDING, 내림차순  
  🡪salary 기준으로 오름차순 정렬해서 보여짐
* 해당 컬럼에 null이 포함되어 있는 경우  
  🡪오름차수(ASC)로 조회하면 가장 나중에 등장하고 내림차순(DESC)로 조회하면 가장 먼저 등장한다.
* 여러 개의 정렬 기준을 적용할 수 있다.job\_id를 1차로 정렬하고 중복되는 것들에 대해서는 first\_name으로 정렬하고 싶으면  
  🡪정렬 기준을 추가함

텍스트이(가) 표시된 사진

자동 생성된 설명

**# 기본키 (Primary Key, PK)**

* 테이블에서 하나의 행을 유일하게 구분할 수 있는 컬럼(전화번호, 주민번호와 같은 것들임)
* 한 테이블 당 하나 밖에 설정할 수 없다
* 기본키가 될 수 있는 컬럼이지만 기본키로 설정되지 않은 컬럼은 **‘후보키(Candidate Key)’**라고 한다.
* **기본키 컬럼에는 중복된 값이 없어야 하며 Null도 없어야 한다.**

**# 외래키 (Foreign Key, FK)**

* 다른 테이블에서는 기본키 또는 후보키이지만 해당 테이블에서는 중복되는 값인 컬럼  
  ex: employees의 department\_id는 departments의 기본키이고, employees에 가져와서 활용되고 있다
* 외래키가 설정된 테이블 간에는 관계가 형성된다
* 외래키로 설정된 컬럼에는 반드시 참조하는 테이블의 기본키로 이미 존재하는 값만 추가할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명  
🡪즉 departments 테이블에서는 department\_id는 기본키이지만 employees 테이블에서는 외래키가 된다  
이를 통해 departments과 employees 테이블 간에 관계를 만들 수 있다  
다른 예 : 팀과 소속 선수, 수강생과 담당 상담원, 소대장과 분대원, 제품과 제조회사

**# 테이블 JOIN**

* 기본키와 외래키로 관계가 형성되어 있는 테이블들의 정보를 종합하여 조회하는 것

**# CROSS JOIN**

* 조인에 사용되는 테이블들의 데이터를 조합하여 나올 수 있는 모든 경우를 출력하는 JOIN
* 모든 경우를 출력하는 쓸모없는 정보  
  🡪걍 두 테이블에 대해 관계 신경 안쓰고 출력할 수 있는 모든 경우를 출력  
  테이블이(가) 표시된 사진

  자동 생성된 설명  
  🡪즉 원순열처럼 할 수 있는 모든 조합을 출력하는 거임

**# EQUI JOIN**

* 두 테이블 간의 서로 동일한 값을 지닌 컬럼(기본키와 외래키)를 이용하여 **CROSS JOIN**의 결과에서 결과에서 의미있는 데이터만 걸러내는 JOIN  
    
  이때 양쪽 테이블 모두에 같은 이름을 지닌 컬럼이 있는 경우 어느쪽의 것인지 명확하게 명시해야함  
  (employees.department\_id or departments.department\_id)

텍스트이(가) 표시된 사진

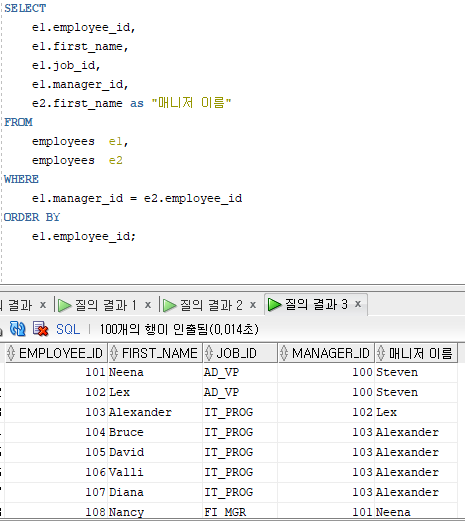
자동 생성된 설명

**이때 테이블 명을 자주 사용해야 하는 경우 별칭을 사용할 수 있다.**텍스트이(가) 표시된 사진

자동 생성된 설명🡪FROM에서 별칭 지정

**# SELF JOIN**

* 하나의 테이블 내에서 자기 자신과 JOIN하여 원하는 데이터를 얻어내는 조인 방식
* 이때 SELF JOIN을 할 때에느 테이블 별칭을 사용할 수 밖에 없다

🡪 employees 테이블에서 SELF JOIN을 통해 매니저 이름까지 출력

텍스트이(가) 표시된 사진

자동 생성된 설명

21.05.25

**# OUTER JOIN**

* JOIN 조건을 만족하지 못해서 등장하지 못했던 행들을 확인할 때 사용하는 JOIN
* (+)를 붙인쪽의 컬럼에 null을 추가해서라도 등장하지 못했던 행들을 확인할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명  
**\*이거 시험문제에 낸다고 했음**

텍스트이(가) 표시된 사진

자동 생성된 설명

**# ANSI JOIN**

* ANSI (American National Standards Institute) : 여러가지 표준을 재정하는 민간 단체
* 미국 국가 표준 협회에서 정한 표준을 따르는 JOIN문법
* 다른 RDBMS에도 이 문법이 존재한다**(즉 공용어라는 의미)  
  기존에 배웠던 거랑 결과물은 다르지 않아 이걸 안써도 되나 다른 RDBMS(MYSQL 등등)에서도 쓸 수 있음**

**# ANSI CROSS JOIN  
**

**# ANSI INNER JOIN**

* 조건에 맞는 행만 등장하는 JOIN ****
* **JOIN에 이용되는 컬럼명이 같은 경우 USING을 이용해 쿼리문을 간소화 할 수 있다.**  
    
  🡪위에 쿼리문랑 같은 거임

**# ANSI OUTER JOIN**

* 조건에 맞는 않는 행도 등장하는 JOIN
* LEFT/RIGHT/FULL을 선택해야 한다
* LEFT OUTER JOIN : 왼쪽에 나오는 아우터 조인, 즉 오른쪽에 (+)을 붙인 것과 같다
* RIGHT OUTER JOIN : 오른쪽에 나오는 아우터 조인, 즉 왼쪽에 (+)을 붙인 것과 같다
* FULL OUTER JOIN : 양쪽에 나오는 아우터 조인, 즉 양쪽에 (+)을 붙인 것과 같다

텍스트이(가) 표시된 사진

자동 생성된 설명🡪얘는 USING을 쓸 수가 없음(조건 쿼리명이 다르기 때문)

텍스트이(가) 표시된 사진

자동 생성된 설명

**🡪즉 첫번째 INNER를 통해 departments와 연결하고 이를 통해 departments와 연결된 locations을 통해 location\_id와 연결 가능**

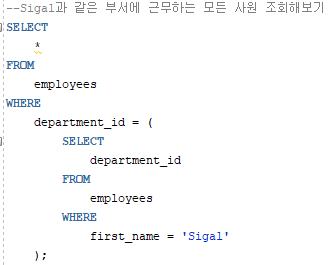
**\* ANSI JOIN을 사용하면 조인 조건과 WHERE 조건절을 구분하여 사용할 수 있고   
USING에 사용된 컬럼을 어느 테이블의 컬럼인지 명시할 필요가 없어진다**

**# 서브쿼리**

* 하나의 SELECT문 내부에 포함된 또 하나의 SELECT문
* 서브쿼리를 포함하고 있는 쿼리를 메인 쿼리라고 한다.
* 서브쿼리는 메인 쿼리가 실행되기 전에 한번만 실행된다



**# 단일 행 서브 쿼리**

* 서브 쿼리의 수행 결과가 단 하나의 행만 반환하는 서브쿼리
* =, >, >=, <= 등의 비교 연산자들은 단일 값끼리 비교하는 연산자이므로 단일행 서브쿼리와 함께 사용해야 한다 🡪()안에 결과가 이므로 department\_id = 30인 데이터에 대해 출력  
  **🡪이와 같이 서브쿼리에 결과가 단일 행이면 단일 행 서브 쿼리라고 함**

**# 다중 행 서브 쿼리**

* 서브 쿼리의 실행 결과로 반환되는 행이 2줄 이상인 서브 쿼리
* **다중 행 연산자**와 함께 사용해야 한다\*(다중 행 연산자는 아래와 같다)
* IN : 여러 개의 값들 중 포함되어 있으면 참
* ANY, SOME : 여러 개의 값들 중 하나 이상 만족시키면 참
* ALL : 여러 개의 값들을 모두 만족시켜야 참
* EXIST : 서브쿼리의 실행 결과가 존재하면 참, 서브쿼리 실행 결과가 1행이라도 존재하면 참

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명(SOME으로 바꿔도 똑같음)

텍스트이(가) 표시된 사진

자동 생성된 설명🡪월급이 가장 많은 사원의 정보를 출력

21.05.26

**\*SQL 계정 만들기**

**# create user 계정이름 identified by 비밀번호;  
-계정이름이 아이디, 비밀번호 1234로 DB 사용자 계정 생성**

**grant create session to 계정이름;**

**grant connect to 계정이름;**

**grant resource to 계정이름;**

**🡪SQL 계정에 필요한 권한들을 부여한다**

**ALTER USER awsconnection DEFAULT tablespace users quota unlimited on users;  
🡪해당 계정에 대해 용량을 무제한으로!**

**🡪위와 같은 과정을 거쳐야함**

**# DML (Data Manipulation Language, 데이터 조작어) \*DB의 CRUD**

* SELECT : 테이블의 데이터 조회
* INSERT : 테이블의 데이터 생성
* DELETE : 테이블의 데이터 삭제
* UPDATE : 테이블의 데이터 수정

\*CRUD : Creat, Read, Update, Delete. 모든 프로그램에는 CRUD가 다양한 형태로 존재한다.

**# DDL (Data Definitaion Language, 데이터 정의어)**

* 테이블, 시퀀드, 뷰.. 등 DB에서 사용하는 **오브젝트 구조를** 생성할 때 사용하는 명령어
* CREATE : DB오브젝트 생성
* DROP : DB 오브젝트 삭제
* ALTER : DB 오브젝트 수정
* TRUNCATE : DB 오브젝트 완전 삭제

**# DCL (Data Control Language, 데이터 조작어)-DBMS마다 명령이가 조금씩 다를 수도 있음**

* DB의 사용자들이 관리하는 명령어
* GRANT : 사용자에게 권한을 부여
* REVOKE : 사용자의 권한을 회수

\* CREAT TABLE 테이블명 (컬럼이름1 컬럼타입1, 컬럼이름2 컬럼타입2, …)🡪테이블 생성  
텍스트이(가) 표시된 사진

자동 생성된 설명

\* DROP TABLE 테이블명🡪테이블 삭제  


\* 삭제한 테이블은 휴지통(recyclebin)에 담겨있다.  


\* 휴지통 복구하기-flashback table 테이블명 to before drop;  


\*휴지통 비우기 – purge recyclebin;

**#데이터 딕셔너리**

* 데이터에 대한 데이터 (메타 데이터)
* DB는 우리가 추가한 데이터를 알아서 관리한다
* 현재 상태는 DB의 데이터 딕셔너리를 통해서만 알 수 있다
* 사용자가 직접 데이터 딕셔너리를 수정할 수는 없다

**#데이터 딕셔너리 뷰**

* 데이터 딕셔너리를 사용자가 확인하기 편리한 형태로 제공하는 뷰
* **user\_**가 붙으면 해당 계정의 데이터 딕셔너리임을 나타낸다  
  텍스트, 테이블이(가) 표시된 사진

  자동 생성된 설명🡪커피 테이블을 다시 만들었음
* all\_가 붙으면 모든 계정의 데이터 딕셔너리임을 나타낸다  
  🡪단 권한이 있어야함(AwsConnection에는 HR을 볼 수 있는 권한이 없기에 Sys에서 확인하면 다 보임)

**# 테이블 데이터 추가하기**

\*INSERT INTO 테이블명(컬럼명1, 컬럼명2, …) VALUES(값1, 값2, …)  
텍스트이(가) 표시된 사진

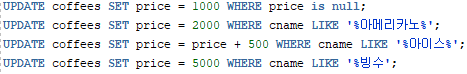
자동 생성된 설명

\*일부 컬럼에만 데이터 추가하기  
🡪넣지 않은 컬럼에 대해서는 NULL로 들어감

\*컬럼명을 생략하고 데이터 추가하기  
 🡪반드시 모든 데이터를 컬럼 순서대로 넣어줘야 한다.

\*컬럼명을 모두 적어주면 순서는 상관없다

\*INSERT INTO 테이블명 (서브쿼리) : 서브쿼리를 테이블에 INSERT 할 수 있다.  


* **#테이블 데이터 수정하기(UPDATE 테이블명 SET 값을 바꾸고 싶은 컬럼 WHERE 조건;**)
* \*UPDATE 테이블명 SET 컬럼=값, 컬럼=값, … WHERE 조건;
* **UPDATE 조건을 설정하지 않으면 모든 행을 수정한다**
* 조건이 있다면 조건을 만족하는 모든 행을 수정한다
* 하나의 행을 구분할 수 있는 기본키와 함께 사용되는 경우가 많다  
  
* 여러 컬럼을 수정할 때는 쉼표를 사용한다  
  
* Raplace(컬럼 이름, 바꾸고 싶은 데이터, 바꿀려는 데이터)  
    
  🡪coffees 테이블에 cname 컬럼에 대해서 cname 중에 ‘아메리’가 있으면 ‘이탈리’로 바꿔줌

텍스트이(가) 표시된 사진

자동 생성된 설명

**# 테이블 데이터 삭제하기**

* DELETE FROM 테이블명 WHRER 조건
* 조건에 맞는 행 전체를 삭제한다 (특정 컬럼 값만 삭제하는 것은 UPDATE에 해당한다)
* 삭제 조건을 설정하지 않으면 모든 행을 삭제한다
* 조건이 있다면 조건을 만족하는 모든 행을 삭제한다
* 하나의 행을 구분할 수 있는 기본키와 함께 사용되는 경우가 많다

\*ROLLBACK : 방금 실행시킨 쿼리를 되돌림

\*COMMIT : 쿼리문 확정

\*CRATE TABLE 테이블명 AS(서브쿼리)로 테이블을 생성할 수 있다  


\*테이블에 새 컬럼 추가하기  
ALTER TABLE 테이블명 ADD (컬럼명 컬렴타입, …)  
 

\*테이블의 컬럼 이름 변경하기  
ALTER TABLE 테이블명 RENAME COLUMN 현재이름 TO 바꿀이름  


\*테이블 이름 변경하기

RENAME 현재테이블명 TO 바꿀테이블명  


\*컬럼 타입 변경하기  
ALTER TABLE 테이블명 MODIFY (컬럼명 컬럼타입, …);  
텍스트이(가) 표시된 사진

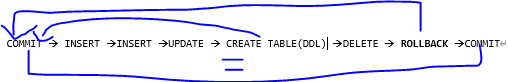
자동 생성된 설명

\*컬럼 삭제하기

ALTER TABLE 테이블명 DROP COLUMN 컬럼명  


**\*coffee\_id에 대해서** **마지막 번호를 찾지 않아도 그냥 넣을 수 있는 쿼리**  
  
🡪이러면 bingsoo 쿼리문을 보지 않아도 데이터를 추가할 때 그 데이터의 coffee\_id는 마지막 번호+1이 되어 순서대로 들어감

**#트랜잭션 (Transaction)**

* 데이터 처리의 한 단위
* 하나의 논리적인 작업이 완전하게 마무리 될 때 까지의 절차  
  ex: 송금 트랜잭션  
  1. 보내는 사람의 통장에서 돈이 줄어든다  
  2. 받는 사람의 통장의 돈이 늘어난다
* 하나의 트랜잭션은 여러 명령어의 집합이다
* 트랜잭션의 모든 과정이 정상적으로 완료되는 경우에만 변경 사항을 확정짓는다 (all or nothing)
* 트랜잭션을 관리하기 위해 사용되는 TCL로는 **COMMIT, ROLLBACK, SAVEPOINT**가 있다.
* 트랜잭션은 마지막으로 실행된 커밋(또는 롤백)이후부터 새로운 커밋을 시랭하는 시점까지 수행하는 모든 **DML**을 의미한다  
     
  🡪첫번째 COMMIT부터 쭈욱 진행하다가 **ROLLBACK**을 시전하면 첫번째 COMMIT으로 돌아가게 되는데 이때 CREATE TABLE은 **DDL**이므로  
  다시 돌아간 COMMIT에는 CREATE TABLE까지 반영된다.(나머지는 반영 X)
* **INSERT. UPDATE, DELETE는 DML이기 때문에 ROLLBACK으로 다시 되둘릴 수 있다.**
* **CREATE TABLE, DROP TABLE, TRUNCATE TABLE 등의 DDL은 트랜잭션으로 관리할 수 없다**
* SAVAPOINT로 잡아놓고 ROLLBACK TO로 세이브 포인트로 해놨던 곳으로 이동 가능  
  (단 ROLLBACK을 하면 마지막 COMMIT으로 이동!)  
  텍스트이(가) 표시된 사진

  자동 생성된 설명

🡪이때 ROLLBACK TO save02을 하면 감자탕이 추가되어 있고 product\_number이 3~10이 사라진 쿼리로 이동함 (단 아메리와 이탈리는 바뀌지 않은 상태) 이때 sava01은 가능하나 save03은 불가능

그리고 이때 ROLLBACK을 하면 마지막 COMMIT으로 이동

||과 뒤에 ‘’을 붙이면 원하는 형식으로 출력 가능

21.05.27

**# 데이터 무결성**

* 데이터에는 결점이 없어야 한다는 뜻
* 개체 무결성, 참조 무결성, 도메인 무결성 ……

**# 개체 무결성**

* 테이블의 데이터는 반드시 한 행을 구분할 수 있어야 한다.
* 개체 무결성을 지키기 위한 제약 조건으로 PK(기본키)를 사용한다

**# 참조 무결성**

* 참조 관계에 있는 데이터는 항상 일관된 값을 가져야 한다  
  (employees의 department\_id는 departments의 department\_id에 존재하는 값만 사용해야 한다)
* 참조 무결성을 지키기 위한 제약 조건으로 FK(외래키)를 사용한다

**# 도메인 무결성**

* 해당 컬럼을 구성하는 도메인들은 항상 타입이 일정해야 한다
* 테이블의 컬럼 타입 설정 및 CHECK를 통해서 도메인 무결성을 지킬 수 있다

**# 데이터 무결성약 조건**

* NOT NULL : 해당 컬럼에 null을 허용하지 않는 규칙
* UNIQUE : 해당 컬럼에 중복되는 값을 허용하지 않는 규칙. 항상 유일한 값을 갖도록 한다
* **PRIMARY KEY : 해당 컬럼을 기본키로 설정한다. NOU NULL + UNIQUE. 테이블 당 하나**
* FOREIGN KEY : 해당 컬럼을 외래키로 설정한다.
* CHECK : 원하는 데이터의 조건을 직접 지정하여 도메인 무결성을 유지할 수 있다

\*제약조건 데이터 딕셔너리를 통해 현재 계정의 모든 제약조건을 확인할 수 있다.  
텍스트이(가) 표시된 사진

자동 생성된 설명

**# 제약조건 테이블 CONSTRAINT\_TYPE**

* P : Primary Key(기본키)
* R : Foreign Key(외래키)
* C : CHECK / NOT NULL
* U : UNIQUE

**\*제약조건을 건 테이블 만들기**테이블이(가) 표시된 사진

자동 생성된 설명

**\*CHECK에 대해서는 CHECK()를 사용**

**# 테이블에 존재하는 제약조건 제거하기**

* ALTER TABLE 테이블명 DROP CONSTRAINT 제약조건 명;  
  테이블이(가) 표시된 사진

  자동 생성된 설명🡪이게 CONSTRAINT임  
  🡪fcOlor가 RED, GREEN, BLUE만 와야 하는 조건을 삭제  
  (이는 컬럼이 사라지는 것이 아니라 컬럼에 있는 제약조건이 사라지는 거임!)

#테이블에 이미 존재하는 컬럼에 제약조건 추가하기

* ALTER TABLE 테이블명 ADD CONSTRAINT CONSTRAINT 제약조건(적용될 컬럼);
* ALTER TABLE 테이블명 MODIFY 컬럼명 (새로 설정);

 🡪아까 삭제한 fcOlor에 제약조건을 추가

**\*이때 CONSTRAINT\_NAME은 자동 생성되나 사용자가 직접 정할 수 있다.(웬만하면 정하는 걸로!)**

제약조건 이름은 테이블명\_컬럼명\_제약조건으로 하는 것이 좋다

**# 제약조건을 추가하는 방법**

1. 테이블 생성과 동시에 제약조건 추가하기  
   -컬럼명 컬럼타임 CONSTRAINT 제약조건명 제약조건타입  
   -반드시 제약조건 이름은 팀 규칙에 맞게 짓도록 하자  
   텍스트이(가) 표시된 사진

   자동 생성된 설명
2. 테이블 생성 후 제약조건 추가하기  
   텍스트이(가) 표시된 사진

   자동 생성된 설명

🡪제약조건 확인하기

**#default 제약 조건**

* 해당 컬럼에 기본값을 설정해놓는 제약 조건
* Default가 설정된 컬럼은 행 추가시 값을 넣지 않으면 null대신 default로 설정한 값이 추가된다.

# Foreign Key 제약조건

* 제약조건을 설정할 때 어떤 테이블의 어떤 컬럼을 참조할지 반드시 지정해야 한다
* 다른 테이블의 PK또는 UK제약조건이 걸린 컬럼만 지정할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**#컬럼을 참조하는 방법**

* 컬럼을 만듦과 동시에 참조하게 하는 쿼리문  
  ALTER TABLE (테이블명) **ADD** ((참조할새컬럼) (유형) CONSTRAINT (CONSTRAINT이름) REFERENCES (참조연결할 테이블 이름(컬럼));  
  
* 존재하는 컬럼을 참조하는 쿼리문  
  ALTER TABLE (테이블명) **MODIFY** ((참조할컬럼) (유형) CONSTRAINT (CONSTRAINT이름) REFERENCES (참조연결할 테이블 이름(컬럼));  
  

**# 제약조건 추가하는 방법3 (테이블 레벨로 추가하기)**

* 컬럼을 모두 정의한 CREATE TABKE의 맨 뒤에 따로 제약조건을 지정하는 방식
* DEFAULT와 NOT NULL은 테이블 레벨에서 정의할 수 없다  
  텍스트이(가) 표시된 사진

  자동 생성된 설명

21.05.28

\*다른 테이블에서 FK로 참조되는 테이블의 행을 삭제할시 아무 생각없이 삭제하면 무결성을 해칠 위험이 있음  
테이블이(가) 표시된 사진

자동 생성된 설명

**#ON DELETE 제약조건**

* 자식 레코드가 참조하고 있는 부모 레코드를 삭제할 때의 정책을 결정할 수 있다
* CASCADE, SET NULL, RESTRICT

**#ON DELETE CASCADE**

* 부모 레코드를 삭제하면 자식 레코드를 모두 함께 삭제한다

**#ON DELETE SET NULL**

* 부모 레코드를 삭제하면 참조하던 자식 레코드의 값을 null로 변경한다

**#RESTRICT**

* 자식 레코드가 있으면 부모 레코드를 삭제할 수 없게 한다

**# CASCADE TEST  
**🡪이후에 을 하면   
lid = 10을 참조한 fruit3의 모든 데이터들은 삭제

**# SET NULL TEST**

  
을 하고 fruits3의 lid = 10에 대한걸 지우면 참조한 fruit3 테이블의 해당 lid = null이 됨

**#엔티티간의 다대다 관계가 발생하는 경우에 대하여**

* 두 개의 1:다 관계를 만들어 해결한다

테이블이(가) 표시된 사진

자동 생성된 설명

* 그냥 포기하고 한 테이블에 다 적는다 (성능이 떨어지지 않는다. 물리적 용량을 더 많이 차지한다)  
  🡪그래서 나온게 NO-SQL(관계를 나누지 않는 것)

#**시퀀스**

* 기본키로 사용하기 편하도록 설계된 자동 번호 생성기
* User\_sequences 데이터 딕셔너리로 확인할 수 있다.

CREATE SEQUENCE 시퀀스명

[START WITH n] – 시퀸스의 시작 번호 설정

[INCREMENT BY n] – 시퀀스의 증가값 설정

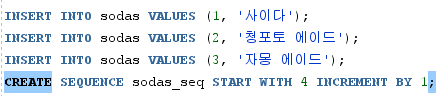
[MAXVALUE n | NOMAXVALUE] – 최대값 설정

[MINVALUE n | NOMINVALUE] – 최소값 설정

[CYCLE | NOCYCLE] -최대값에 도달했을 때 순환 여부 설정

[CACHE n | NOCACHE] – 메모리상에 미리 만들어 놓을 시퀀스의 개수 설정

\*시퀀스데이터딕셔너리  


예제  
  
🡪앞으로 4부터 시작으로 1씩 증가하는 시퀀스 생성

\*시퀀스 쓰는 법  
  
텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

\*현재 몇 번째 번호까지 사용증인지 알고 싶을 때  


\*시퀀스 삭제  
-DROP SEQUENCE 시퀀스명

\*시퀀스 수정  
-ALTER SEQUENCE 시퀀스명 (이후 CREATE SEQUENCE와 같음)

**#뷰**

* 물리적인 기본 테이블들을 이용해 생성한 논리적인 가상의 테이블
* 기본 테이블에서 파생된 DB 객체
* 사용자는 주어진 뷰를 통해 기본 테이블을 제한적으로 사용하게끔 할 수 있다.

\*뷰 만들기  
-CREATE [OR REPLACE] VIEW 뷰이름 AS 서브쿼리 [WITH CHECK OPTION] [WITH READ ONLY];

-CREATE OR RAPLACE : 이미 존재하는 뷰라면 수정을 한다. 없다면 생성한다.

-WITH CHECK OPTION : 해당 뷰의 조건을 만족하는 DML이 가능해진다.

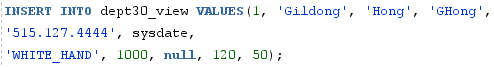
-WITH READ ONLY : 해방 뷰는 SELECE만 가능해진다.(DML 금지)  
텍스트이(가) 표시된 사진

자동 생성된 설명  
텍스트이(가) 표시된 사진

자동 생성된 설명🡪읽기 전용 뷰를 통해서는 DML작업이 불가능

텍스트이(가) 표시된 사진

자동 생성된 설명🡪이렇게 했을 때 department\_id= 50인 경우만 기입 가능



🡪이때 뷰를 통해 테이블을 수정하는 거임

\*이렇게 하는 이유는 뷰를 통해서 수정을 할 때 올바르게 넣을 수 있도록 거르는 역할을 함

--끝?