★★★★★★★★★★★★★★★★★★★★★★★★ SQL 이론 정리 ★★★★★★★★★★★★★★★★★★★★★★★★

==========

========== 찾아가기 ==========

11g, 12c, sqlgate

1장.

2장.

3장.

4장.

9장.

1. DML
   - insert 문제 289
   - update 문제 294
   - delete 문제 298
   - merge 문제326
2. TCL
   - commit
   - rollback
   - savepoint
3. ed
4. lock 문제 338

10장.

1. DDL
   - create 문제 346
   - alter 문제 353
   - rename 문제360
   - drop 문제 360
   - truncate
   - ※ add, drop, modify, rename, unused 문제 371
2. 데이터 타입
3. 제약
   - primary key 문제 372
   - unique 문제 374
   - not null
   - check 문제 376
   - foreign key 문제 383
   - 걸린 제약 확인
4. 개체명 질의 문제 391

11장.

1. database object 5가지 (table,
   - index 문제 421
   - view 문제 399
   - synonym
   - sequence 문제 418
2. 복합 view, view옵션 문제 408
3. 함수기반 인덱스 문제 434
4. 힌트 문제 424

12장.

1. 권한 문제 447
   - grant 문제 450
   - revoke 문제 459
   - 권한 리스트
   - 권한 확인
2. 롤
3. 패스워드 변경 (유저확인, pw변경, 유저삭제)
4. 계층형 질의문 (level, sys_connect_by) 문제 460

13장.

1. 임시 테이블
   - commit delete
   - commit preserve
2. 외부테이블 문제 473
3. flashback 기술 (flashback drop, flashback query, flashback table) 문제 481

14장.

15장.

16장.

17 장.

튜닝스킬

==========

==========

★ 오라클이란 ?

" 데이터 베이스 소프트웨어 "

↓

데이터를 저장하는 저장소

★오라클 접속하는 방법

시작 -> 검색창 -> cmd -> 도스창 ->

sqlplus / as sysdba

관리자 권한으로 접속하겠다

오라클 데이터베이스에 접속하기 위한 프로그램 이름

show user -현재 오라클에 접속한 유저가 누구인지?
-user is "SYS" - 오라클의 최고 권한자 (최고권한자인 sys유저는 시스템 관리자나 dba 만 접속하게 하고 데이터 분석가나 일반 유저들은 필요한 데이터만 엑세스하면 되는 권한만 갖고 있으면 된다.)

★ 일반 유저를 생성하는 작업

```
create user scott       유저이름
  identified by tiger;  패스워드
```

★ scott 유저에게 기본적인 권한을 부여

```
grant connect, resource to scott;
```

★ scott 으로 접속을 한다.

```
SQL> connect scott/tiger
SQL> show user
```

★내일 접속할 때는 도스창에 sqlplus scott/tiger 입력하면 됨

★실습용 테이블을 생성하고 데이터 입력하는 방법
http://cafe.daum.net/oracleoracle 카페 검색창에 demobld 라고 치고 엔터를 친다.
맨위에꺼 눌러서 복사 붙여넣기(마우스로 해야함) - 도스창 설정에서 빠른 편집모드 하면 오른쪽 마우스 누르기만 해도 복사가 됨

★ emp 테이블에서 ename(이름)과 sal(월급)을 검색해본다
```
Select ename, sal       컬럼명
From emp;
```

★ SQL Gate 는 오라클 sqlplus가 다루기가 좀 어려우니 좀 더 쉽게 쿼리를 작성해서 데이터 베이스의 데이터를 검색할 수 있게 도와주는 툴
www.sqlgate.com
Host : localhost
유저이름 : scott
패스워드 : tiger
서비스이름 : xe

★ SQL

Sutucture Quary Language의 약자로 데이터 베이스의
구조적 질의 언어
데이터를 검색하고 조작할 수 있는 언어

★ SQL을 왜 배워야 하는가?

Small data (bussiness data) vs Big data (비정형화된 data)
↓
지금까지 한번도 분석해보지 않은 데이터

★ SQL의 종류 ?

1. Quary 문 --> database의 데이터를 검색하는 언어
   예 : select 문의 6가지 절
2. DML 문 -->database의 데이터를 조작 및 삭제하는 언어
   예 : insert, update, delete, merge
3. DDL 문 -->database의 테이블과 같은 object를 관리하는 언어
   예 : create, alter, drop, turngate, rename
4. DCL 문 --> database의 사용자들의 권한을 관리하는 언어
   예 : grant, revoke
5. TCL 문 --> database의 data를 저장할지 아니면 저장하지 않고 취소할 지 결정하는 언어
   예 : commit, rollback, savepoint

★ Quary 문

※ 설명 : SQL 작성 규칙 (P64)

1. SQL은 대소문자를 구분하지 않는다. (현업에서는 주로 소문자로 쓴다)
2. Sql은 한줄 또는 여러줄에 입력할 수 있다. (왠만하면 여러줄로 쓰는게 좋다)
3. 절은 별도의 줄에 입력해라 ~
4. 가독성을 높이기 위해 들여쓰기를 사용해라 ~

■ 12c
오라클
winx64_12102_database_1of2
winx64_12102_database_2of2
받아서 드래그해서 한번 여기에 풀기로 압축을 푼다 (database 폴더가 생성됨)
풀었을 때 2.7기가여야 한다
폴더 들어가서 setup을 관리자 권한으로 실행!
메일 적지마 -다음 - 다음 - 다음 - window 내장 계정으로 사용 -
비밀번호 oracle로 하고 그냥 설치하셈

* 오라클에서 scott계정 접속이 안된다면 ?

```
SQL> SELECT instance_name from v$instance;
            ※현재 오라클 버전 확인 xe인지 확인해

C:WUsersWAdministrator>set ORACLE_SID=xe

C:WUsersWAdministrator>sqlplus heaven/heaven
```

• 새로운 데이터베이스에 접속하는 방법

```
C:WUsersWAdministrator>set ORACLE_SID=orcl

C:WUsersWAdministrator>sqlplus / as sysdba

SQL> select instance_name
  2    from v$instance;

INSTANCE_NAME
----------------
orcl
```

11g	12c
데이터베이스 이름 : xe	데이터베이스 이름 : orcl
기능이 약한 소프트웨어	오라클의 모든 기능이 다 패키징 되어있다
scott 유저 생성	scott 유저가 이미 생성됨 emp, dept,salgrade
demobld 스크립트	다 이미 가지고 있다.

emp
emp2

★ 12c 오라클 접속하는 방법

1. 도스창을 연다.

2. C:₩Users₩stu>set ORACLE_SID=orcl

3. 리스너의 상태를 확인한다.
   ↓
   데이터 베이스 접속할때 통과해야하는 데몬

4. lsnrctl status

5. 오라클에 sys 유져로 접속을 한다.

```
c:> sqlplus  /  as  sysdba

SQL> show   user
```

6. 12c 의 데이터 베이스가 무엇이 있는지 확인

```
SQL> SELECT name, pdb
        FROM   v$services
        ORDER BY name;

NAME                 PDB
-------------------- -----------
SYS$BACKGROUND       CDB$ROOT
SYS$USERS            CDB$ROOT
orcl                 CDB$ROOT
orclXDB              CDB$ROOT
pdborcl              PDBORCL   <---  이것을 확인
```

7. 어느 데이터베이스로 접속이 되어있는지 확인

```
SQL> show con_name

CON_NAME
-----------
CDB$ROOT
```

8. pdborcl 데이터베이스를 사용하겠다라고 설정

```
SQL>  alter session set container=pdborcl;

SQL> show con_name

CON_NAME
-----------
PDBORCL
```

9. pdborcl 에 sys 유져로 접속한다. 최고 권한자로 들어가겠다

```
                                          ↓
SQL> connect sys/oracle@localhost:1522/pdborcl as sysdba
                          ↑       ↑       ↑
                     아이피주소  포트번호 디비이름
```

10. 현재 접속한 데이터 베이스가 open인지 확인

```
SQL> select instance_name , status
  2   from v$instance;

INSTANCE_NAME    STATUS
---------------- ------------
orcl             MOUNTED
```

11. pdborcl 데이터베이스를 올린다.

```
SQL> alter  database  pdborcl  open;

데이타베이스가 변경되었습니다.
```

12. 잘 올라갔는지 확인

```
SQL>  select instance_name , status
  2    from v$instance;
```

13. scott 유저의 lock 을 해제하고 패스워드를 tiger
    로 변경한다.

```
SQL> alter  user  scott
     account  unlock;

SQL> alter  user  scott
     identified  by  tiger;
```

10. scott 으로 접속한다.

```
SQL> connect scott/tiger@localhost:1522/pdborcl
연결되었습니다.
```

□□

★ 1장. SQL SELECT 문을 사용하여 데이터 검색

1. 기본 select 문 : SELECT [DISTINCT] { 속성명 [as] [컬럼명] , 속성명, …… }
FROM {테이블} [테이블별칭] ;

※ 1. 속성명 대신 * 을 사용하면 전체를 출력할 수 있다.
2. 속성부분에 산술연산자 활용 가능하다.

★ SELECT 문의 산술식 및 NULL값 (책 1-10)

※ 오라클의 기본 연산자

1. 산술 연산자 : * / + -
2. 비교 연산자 : >, <, >=, <=, <>, !=, ^=
3. 논리 연산자 : and, or, not

※ 논리연산자에서 헷갈리면 안되는 것

TRUE AND TRUE = TRUE
TRUE AND FALSE = FALSE
TRUE AND NULL = FALSE

TRUE OR TRUE =TRUE
TRUE OR FALSE = TRUE
TRUE OR NULL = TRUE

FALSE OR NULL = TRUE

FALSE AND NULL = FALSE

※ 레코드에 데이터가 비어 있는 부분을 null값이라고 한다.

• NULL 값이란 ? 1. 데이터가 없는 상태
2. 알 수 없는 값

NULL은 0과 공백과는 다르다.

★ NULL을 입력하는 방법 2가지 !

1. 암시적 입력방법

예 :

```
INSERT INTO EMP( empno, ename, sal)
   VALUES( 2919, 'allen', 4000) ;
```

2. 명시적 입력방법

- Null 그냥 널을 입력
- ''싱글 쿼테이션을 붙여서 입력"

예 :

```
INSERT INTO EMP (empno, ename, sal)
   VALUES(3945, NULL, 4500);

INSERT INTO EMP (empno, ename, sal)
   VALUES (3946, '', 4600);
```

★ nvl 함수

NULL값 대신에 지정된 다른 값을 출력하는 함수

형식 : nvl ({ 속성명 } , { 지정하고 싶은 값 })

★ 연결 연산자 ||

"연결연산자를 활용하여 두개의 속성을 하나의 레코드에 출력할 수 있다.

형식 : { 속성명 } || { 속성명 }

※테이블 구조 확인하는 방법 (describe)

desc { 테이블 명 }

★ 컬럼명 지정 (컬럼별칭 사용법)

Select { 속성명 } as [컬럼명], sal as 월급 from emp;

※ 설명 : as 는 생략 가능하다.
공백문자나 특수문자를 컬럼 별칭으로 사용하려면 양쪽에 더블 쿼테이션 마크를 사용해야 한다.

★ select 문의 6가지 절

1. Select 컬럼명
2. From 테이블 명
3. Where 검색조건
4. Group by 그룹핑할 컬럼명
5. Having 그룹함수를 사용한 검색조건
6. Order by 정렬해서 보고싶은 컬럼명

□□

■ 2장. Where 절과 order by 절 사용법

★ WHERE 절 : 원하는 검색결과를 얻기위해 검색 조건을 부여하는 절

```
                                                           실행순서
형식 :     SELECT  [ DISTINCT ]  { 속성명  [ as ]  [ 컬럼명 ] ,  속성명, …… }    3
      FROM {테이블}                                                1
      WHERE { 속성명 }  { 연산자 } { 검색조건 } ;                    2
```

※ WHERE 절 연산자 부분에 들어갈 수 있는 연산자

1. 산술연산자
2. 비교연산자
3. 기타 비교 연산자 (94 페이지)

※ between ~ and ~
양 끝 숫자를 포함하여 검색함(낮은 값부터 입력해야 함)

4. like : ~처럼, ~ 일것 같은
5. in
6. is null

※ 검색 조건에 들어갈 수 있는 기호들

1. % : wild card
   특징 : - 이자리에 뭐가와도 관계없다.
   - 철자의 개수가 몇 개가 되든 관계없다.
   - %가 특수문자 %가 아닌 wild card로 인식하려면 연산자를 반드시 =가 아닌 like를 사용하여야 한다.
2. _ : 이 자리에 뭐가 와도 관계없는데 철자의 개수는 딱 한 개여야 한다.

※ 숫자와는 다르게 문자와 날짜는 양쪽에 싱글 쿼테이션 마크를 사용해야 한다.
그리고 데이터가 대문자로 저장되어 있으면 대문자로 검색해야 한다.

※ SQL gate에서 위의 SQL이 에러가 나는 이유는 날짜 형식이 SQLPLUS와 달라서 이다.

• 현재 세션의 날짜 형식을 확인하는 방법

```
Select * from nls_session_parameters;
```

※ not like는 악성이 될 수 있다!

★★ 좌변을 함수로 가공하면 검색속도가 느려진다.

★ 치환변수

같은 SQL인데 값만 다르게 해서 반복 수행해야 하는 작업을 할 때 작업을 쉽게 할 수 있는 키워드

★ order by 절 (109 페이지)

"데이터 정렬하는 절"

옵션 : asc --> 낮은값부터 출력
       desc --> 높은값부터 출력

```
                                                           실행순서
형식 :     SELECT  [ DISTINCT ]  { 속성명  [ as ]  [ 컬럼명 ] ,  속성명, …… }    4
      FROM {테이블}                                                        1
      WHERE { 속성명 }  { 연산자 } { 검색조건 }                             2
      ORDER BY  { 속성명 } ,  { 속성명 } [ ASC / DESC ] ;                  3
```

□□

■ 3장. 함수 (단일행 함수)

- 함수가 필요한 이유 ?

  다양하게 데이터를 검색하고 분석하기 위해서 반드시 필요한 기능

  예 : 영화 겨울왕국에는 elsa가 많이 나오나 anna가 많이 나오나?

- 함수의 종류

  1. 단일행 함수 : 문자, 숫자, 날짜, 변환, 일반
  2. 복수행 함수 : 최대, 최소, 평균, 토탈, 개수, 분산, 표준편차

  데이터 -----------> 정보를 추출

  함수를 잘 알아야 한다

  예 : 내가 무슨 요일에 태어났는지?
  우리나라 시장에서 파는 가장 비싼 생필품은 무엇인지?
  우리나라에서 같은 이름(동명)이 가장 많은 이름이 무엇인지?

※ 함수는 중첩이 가능하다 !

★ 문자함수

1. upper 함수 [select절]

   "대문자로 출력하는 함수"

   형식 : upper (속성명)

2. lower 함수 [select절]

   "소문자로 출력하는 함수"

   형식 : lower (속성명)

3. initcap 함수 [select절]

   "문자를 출력할 때 첫번 째 철자는 대문자로 출력하고 나머지는 소문자로 출력하는 함수"

   형식 : inicap (속성명)

4. substr 함수 [select절]

   "문자에서 특정 철자만 출력하는 함수"

   형식 : substr (속성명, 1, 3)

   S M I T H (1번부터 3개라는 뜻)

   1 2 3 4 5

   ※ 오른쪽 부터 자리수를 세려면 (-) 를 붙이면 된다 !

예제 1. 셜록홈즈 대본을 오라클 데이터 베이스에 입력하고 셜록홈즈에서 가장 많이 나오는 단어가 무엇인지 1위부터 10위까지 출력하시오 !

```
select COUNT(regexp_substr(lower(sl_text), '[^ ]+',1,187)) aaa1 from sherlock;
```

※단어로 문장을 쪼갰을 때 한문장당 단어의 숫자

5. concat 함수 [select절]
"두개의 컬럼의 데이터를 연결해서 출력하는 함수"

형식 : concat (속성명, 속성명)

6. Length 함수 [select절]
"철자의 개수를 세는 함수"

형식 : length (속성명)

7. instr 함수 [select절]
"특정 철자의 위치번호(인덱스)를 출력하는 함수"

형식 : instr (속성명, ' 철자 ')

8. trim 함수 [select절]
"특정 철자나 공백문자를 잘라내고 출력하는 함수"
Rtrim ---> 오른쪽에 있는 특정 철자나 공백 문자를 잘라내는 함수
Ltrim ---> 왼쪽에 있는 특정 철자나 공백 문자를 잘라내는 함수
Trim ---> 양쪽에 있는 특정 철자나 공백 문자를 잘라내는 함수

형식 : ○trim (속성명, ' 철자 ')
※ trim(속성명)은 양쪽에 공백을 자름

9. replace 함수 [select절]
"특정 철자를 다른 철자로 변경하는 함수"

형식 : replace (속성명, ' 기존 철자 ', ' 변경할 철자 ')

※ 문서 암호화에 쓰임
regexp_replace (속성명, ' [0 - 3] ' , ' * ')
출력할 때에 숫자 0부터 3까지는 *로 출력
설명 : regular expression (정규 표현식) 함수를 이용해서 출력하면 된다.

10. lpad, rpad 함수[pad (속성, 자리수, 넣고싶은거)] [select절]
"lpad 함수는 문자나 숫자를 출력할 때 오른쪽에 원하는 철자를 {숫자}만큼 출력하는 함수이고
rpad 함수는 문자나 숫자를 출력할 때 왼쪽에 원하는 철자를 {숫자}만큼 출력하는 함수이다."

형식 : pad (속성 , {숫자} , ' 원하는 철자 ')

★ 숫자 함수 (p 142)

1. round 함수 [select절]
"반올림하는 함수"

형식 : round (속성명 , 반올림 자리수)
※ 0 자리수는 생략 가능!

2. trunc 함수 [select절]
"값을 버리는 함수"

형식 : trunc (속성명 , 절사 자리수)

3. mod 함수 [select절]
"나눈 나머지값을 출력하는 함수"
형식 : mod (속성명, 나눌 수)

★ 날짜 함수
날짜의 산술 연산
1. 날짜 - 날짜 = 숫자
2. 날짜 - 숫자 = 날짜
3. 날짜 + 숫자 = 날짜
※ sysdate : 오늘날짜를 보는 키워드

1. months_between 함수
" 두 날짜 사이의 개월수를 출력하는 함수"

형식 : months_between (최근날짜, 나중날짜)

2. add_months 함수
" 날짜에 개월수를 더한 날짜를 출력하는 함수"

형식 : add_months(날짜, 개월수)

3. next_day 함수
" 지정날짜를 기준으로 돌아올 요일을 출력하는 함수"

형식 : next_day (날짜, '요일')

4. last_day 함수
" 지정 날짜를 기준으로 그 달의 마지막 날짜를 출력하는 함수"

형식 : last_day (날짜)

□□□

■ 4장. 변환 함수 및 조건부 표현식 사용

★ 형변환 함수 (명시적 형변환)
"데이터의 유형을 변환하는 함수"

1. To_char [select절]
"날짜or숫자 형 데이터를 문자형 데이터로 변환"

형식 : to_char (숫자or날짜, format)
설명 : to_char의 날짜 ---> 문자로 변경할 시 format의 종류
년도 : RRRR, RR, YYYY, YY
달 : MM, MON
일 : DD
시간 : HH, HH24
분 : MI
초 : SS
요일 : DAY, DY, D
주 : IW, WW

■ RR 과 YY 의 차이

RR	YY
81 = 1981	81 = 2081
1981 2081	1981 2081
2018	2018
현재 연도에서 가장 가까운 연도	현재 세기

※ 데이터 베이스에 년도를 2자리로 입력해서 저장했다면 반드시 RR을 사용해야 한다.

■ WW와 IW의 차이점
ww : 요일에 관계 없이 7일을 기준으로 주차를 구분
iw : 요일(월화수목금토일 순)을 기준으로 주차를 구분
※ 설명 : iw는 iso 표준에 따른 연의 주를 의미(한주의 시작을 iso 표준은 월요일로 본다)
Ww는 7일씩 끊어서 주를 표시한다.

2. to_number [select절]
"문자형 데이터를 숫자형 데이터로 변환"
형식 : to_number (속성명)

3. to_date [select절]
"문자형 데이터를 날짜형 데이터로 변환"
형식 : to_date (속성명, 'format')

★형변환의 종류 2가지

1. 명시적 형변환 : 아래의 3가지 형변환 함수를 사용해서 형변환을 하는것을 말한다.
   To_char
   To_number
   To_date

2. 암시적 형변환 : 오라클이 자동으로 형변환을 수행하는 것을 말한다.
   ※ 우선순위 : 숫자 > 문자

   ※ 오리클은 암시적 형변환을 수행을 해서 에러가 나지않고 결과를 출력하게 해주지만
   암시적 형변환이 성능을 느리게 하는 원인이 될 수 있다.
   그래서 가급적 암시적 형변환이 발생하지 않도록 SQL을 작성하는 것이 중요하다.

★ 일반 함수

1. nvl [select절]
   "null값을 다른 값으로 대치시키는 함수"
   형식 : nvl(속성명, 변경 값)

2. nvl2 [select절]
   "null 이 아니면 두번째가 출력되고 널이면 세번째가 출력되게 하는 함수"
   형식 : nvl2 (속성명, 변경값, 변경값)

3. decode [select절]
   " IF 문으로 나열해서 프로그래밍 해야 볼 수 있는 결과를 SQL함수인 decode 하나로
   프로그래밍 없이 간단하게 조회 할 수 있는 함수"
   형식 : decode (속성명, 조건, true출력값, 조건, true출력값, ... , false출력값)

※ false 출력값을 입력하지 않으면 자동으로 null이 출력된다.

4. case [select절]
   " decode함수가 등호비교만 가능하다면 case는 부등호 비교가 가능한 함수"
   형식 : case when [속성명] [연산자] [조건] then [true출력] else [false출력] end

□□□

■ 5장. 그룹 함수
여러 개의 행의 데이터가 입력되어서 하나의 값으로 출력되는 함수

• 그룹함수의 종류 (p 211)
1. Max(최댓값)
2. Min(최솟값)
3. Avg(평균)
4. Count(개수값)
5. Sum(토탈값)
6. Stddev(표준편차)
7. Variance(분산)

※ group 함수로 검색 조건을 주는것을 where 절에 줄 수 없다.
group 함수로 검색 조건을 줄 때는 having 절을 사용해야 한다.

※ 그룹함수는 null값을 무시한다 !

1. max 함수 [select절]
" 최대값을 출력하는 함수"
형식 : 1. max (속성명)
group by [속성명]
2. max (속성명) over (order by 속성명)

※ group by 절의 역할 ? Grouping을 하는 역할을 수행한다.

2. min 함수 [select절]
"최소값을 출력하는 함수'
형식 : 1. min (속성명)
group by [속성명]
2. min (속성명) over (order by 속성명)

3. avg 함수 [select절]
"평균값을 출력하는 함수"
형식 : 1. avg (속성명)
group by [속성명]
2. avg (속성명) over (order by 속성명)

4. count 함수 [select절]
"데이터의 건수를 세는 함수"
형식 : 1. count(속성명)
or count (*)
group by [속성명]
2. count (속성명) over (order by 속성명)

5. sum 함수 [select절]
" 합계를 출력하는 함수"
형식 : 1. sum (속성명)
2. sum (속성명) over (order by 속성명)
group by [속성명]

예제 : select deptno, ename, sal, sum(sal) over (order by empno rows
between unbounded preceding
and current row) 누적치
from emp;
※설명 :1. Unbounded preceding : 제일 첫번째 행
2. Current row : 현재 행
3. Unbounded flowing : 맨 마지막 행

★ 레포팅 함수

1. Rollup [group by 절]

"전체토탈, 전체 맥스, 전체 미니멈, 전체 카운트를 맨 아래 레코드에 표시해주는 함수"

형식 : group by rollup (속성1, 속성2)

※ 결과

1. Deptno, job
2. Deptno
3. 전체

2. Cube [group by절]

"전체토탈, 전체 맥스, 전체 미니멈, 전체 카운트를 맨 위 레코드에 표시해주는 함수"

형식 : group by cube (속성명)

3. grouping [group by]

"grouping sets은 rollup, cube와 다르게 grouping된 결과를 좀더 세밀하게 지정할 수 있다라는 것이다."

형식 : group by grouping sets ((속성1, 속성2), 속성2, ())

★ pivot 문 : 세로 -----> 가로 ※ 피벗은 회전한다는 뜻

예제 :

아래의 결과를 pivot문으로 구현하시오!

	23	24	25	26	27	28	30	31	32	33	40
sk	0	0	1	3	3	2	1	2	0	1	0
lg	0	0	0	1	1	1	0	0	1	0	1
kt	1	1	1	3	3	0	0	0	0	0	0

```
select *
   from (select lower(telecom), age from emp2)
   pivot (count(*) for age in (23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 40));
```

★ unpivot 문 <--------- 컬럼을 데이터 속으로 입력하는 방법

데이터 ------> 컬럼 : pivot 문
컬럼 -------->대이터 : unpivot 문

• Unpivot : 컬럼 --------> 데이터

10	20	30
8750	21973	9400

deptno	sum(sal)
30	9400
20	21973
10	8750

```
========================================
CREATE TABLE EMP406 ( "10" number(10),
                      "20" NUMBER(10),
                      "30" NUMBER(10) );

INSERT INTO emp406 VALUES(8750,21973,9400);

COMMIT;

SELECT * FROM emp406;

SELECT * FROM emp406
unpivot( bbb FOR aaa IN ("10","20","30"));
========================================
```

★ 데이터 분석 함수

1. Rank [select]
   "괄호안의 내용을 해석해서 순위를 출력하는 함수"
   형식 : 1. rank() over ([partition by 속성명] [order by 속성명] [asc/desc])
   2. rank(값) within group ([order by 속성명] [asc/desc])

2. Dense_rank [select]
   "순위(숫자를 건너뛰지 않고) 를 출력하는 함수"
   형식 : 1. dense_rank() over ([partition by 속성명] [order by 속성명]
   [asc/desc])
   2. dense_rank(값) within group ([order by 속성명] [asc/desc])

   ※ partition은 group by 랑은 다른것이다 분석함수 쓸 때 쓰는 옵션 키워드

3. listagg [select]
   "데이터를 가로로 출력하는 함수"
   형식 : listagg (속성, '구분자') within group (order by 속성명)
   group by [속성명]

   ※ order by와 group by 는 필수로 들어가야 한다.

4. ntile [select]
   " 등급(분위수) 를 출력하는 함수 "
   형식 : ntile(등급수) over (order by 속성명 [asc/desc])

5. lead [select]
   " 선택된 행의 {숫자}만큼 다음행을 출력하는 함수 "
   형식 : lead (속성명, {숫자}) over ([partition by 속성] [order by 속성])

6. lag [select]
   " 선택된 행의 {숫자}만큼 이전 행을 출력하는 함수 "
   형식 : lag (속성명, {숫자}) over ([partition by 속성] [order by 속성])

□□□

■ 6장. 조인

" 여러 개의 테이블에 있는 컬럼들을 다 모아서 하나의 결과로 출력하기 위한 SQL 문법 "

★ 오라클 조인 문법

1. equi join

"연결고리의 형식이 = 인 조인을 말한다 (강한 연결고리) "

```
형식 : select
        from 테이블1, 테이블2
        where 테이블1연결고리 = 테이블2연결고리
```

2. outer join

" equi join으로는 볼 수 없는 결과를 볼때 사용하는 조인 문법 "

```
형식 : select
        from 테이블1, 테이블2
        where 테이블1연결고리 [ (+) ] = 테이블2연결고리 [ (+) ]
```

※ 한쪽에만 [(+)]을 쓸 수 있다 !

3. Non equi join

" 조인의 연결고리가 =(이퀄)이 아닌 경우의 조인문법 "

```
형식 : select
        from 테이블1, 테이블2
        where 테이블 연결고리 (ex : e.sal between s.losal and s.hisal)
```

4. Self join

" 자기 자신의 테이블과 조인하는 조인문법 "

```
형식 : select
        from 테이블1 별칭1, 테이블1 별칭2
        where 테이블 연결고리
```

※ 테이블이 3개면 연결고리는 두개 써야한다!

★ 1999 ansi 조인 문법 (Ansi : american national standards institute)

※ Oracle, mssql, mysql, postgre sql, maria DB 등 어느 데이터베이스에서든 다 사용할 수 있다

1. on join

```
형식 : select
        from 테이블1 join 테이블2
        on( 테이블 연결고리 )
```

2. left, right, Full outer join

" outer join에서 [(+)] 를 입력한 효과가 있는 조인 문법 "

```
형식 : select
        from 테이블1 [left/ right/ full] outer join 테이블2
        on( 테이블 연결고리 )
```

※ on 절의 역할과 WHERE 절의 역할이 다르다

WHERE 대신 AND를 써도 되는데 왠만하면 WHERE 써라

3. using 절을 사용한 조인문법 (거의 안씀)

" using절에서 equi join의 연결고리를 수행"

형식 : select
from 테이블1 join 테이블2
using(연결고리가 되는 속성)

※주의사항 ! Using절에 테이블 별칭을 사용해서는 안된다.

4. natural join (거의 안씀)

"두 테이블에서 데이터 유형과 이름이 일치하는 열을 기반으로 자동으로 테이블을 조인"

형식 : select
from 테이블1 natural join 테이블2

5. cross join (더 거의 안씀)

" 오라클 조인문법에서 연결고리가 없는것과 같은 효과의 조인 방법"

형식 : select
from 테이블1 cross join 테이블2

□□

★ 7장. Subquery

1. where 절의 subquery

```
select [속성], [속성], ...
from [테이블]          <<----------- main query
   where [속성] [연산자] (select [속성]
                        from [테이블]   <<-----------subquery
                        where [ 조건 ] );
```

※ 문장이 실행되는 것은 subquery부터 실행한다.

★ select 문에서 subquery를 쓸 수 있는 절

Select	Subquery 가능(scalar Subquery)
From	Subquery 가능(in line view)
Where	Subquery 가능
Group by	불가능
Having	Subquery 가능
Order by	Subquery 가능(scalar Subquery)

※ 스칼라 서브쿼리의 특징 !
"스칼라 서브쿼리는 하나의 값만 리턴한다."

★ where절의 subquery의 종류 3가지

1. Single row subquery : 서브쿼리에서 메인 쿼리로 하나의 값이 리턴되는 경우
   ☆ 사용 가능 연산자 : =, >, <, >=, <=, !=, ^=, <>

   ※ SUBQUERY의 테이블과 메인쿼리의 테이블이 달라도 상관 없다

2. Multiple row subquery : 서브쿼리에서 메인쿼리로 여러 개의 값이 리턴되는 경우
   ☆ 사용 가능 연산자 : in, not in, >all, <all, >any, <any

   ※ >all 활용
   >all : 이 모든 값보다 큰 월급을 찾아라 (해석 : 멀티를 싱글로 ??)
   ※ 결과의 순서가 다르다. 이유 : >all 은 메인쿼리로 시작해서 값을 하나하나 대입(느림)
   > 는 서브쿼리부터 수행함! (빠름)
   수직선으로 나타내면 가장 큰값보다 크다 라는 뜻
   ※ >any : 수직선으로 나타내면 가장 작은것 보다 큰 것들 모두

3. Multiple columm subquery : 서브쿼리에서 메인쿼리로 여러 개의 컬럼값들이 리턴되는 경우
   ☆ 사용 가능 연산자 : in, not in

   1. Non pair wise 방식

```
select [속성명], [속성명], ....
  from [테이블명]
  where [속성1] [연산자] ( [SUBQUERY] )
     and [속성2] [연산자] ( [SUBQUERY] );
```

훨씬 더 많은 데이터를 가지고 옴
(내 생각에 WHERE절에 AND가 OR과 비슷하게 연산이 된다는 거 같아
서브쿼리 각각의 집합에 둘다 속하기만 하면 출력되는거 같아)
UPDATE EMP SET sal = 1250, COMM = 300 WHERE ename = 'KING';
COMMIT;   해보면 알수있음

2. Pair wise 방식

```
select [속성명], [속성명], ....
  from [테이블명]
  where ( [속성1], [속성2] ) [연산자] ( SUBQUERY )
```

※ 성능의 문제가 아니라 결과가 다르게 나옴

★★★ in : ○ or ○ or ○ or ○ or ○ or ○ or ○ or null 이라면 ?

True or null = 이게 중요한데

False

True ----> TRUE가 나오기 때문에 245는 되는거고

Not IN : ○ AND ○ AND ○ AND ○ AND ○ AND NULL 이기 때문에

TRUE AND NULL = 이것이

FALSE

TRUE --------> NULL이 나오기 때문에 246은 안되서 nvl 함수를 써야 하는것이다

★ subquery문에서 not in을 사용할 때에는 subquery에 null 처리를 반드시 해야한다.

★ 상호관련 서브쿼리

"메인쿼리의 컬럼이 서브쿼리 안으로 들어가게 되는 SQL 을 상호관련 서브쿼리라고 한다."

예 : 직업별 인원수가 4명 이상인 직업인 사원들의 이름과 직업을 출력하시오 !

```
select ename, job
  from emp m
  where 4 <= (select count(*)
               from emp s
               where job = m.job);
```

```
=====================================
ENAME                JOB
-------------------- ------------------
MARTIN               SALESMAN
ALLEN                SALESMAN
TURNER               SALESMAN
JAMES                CLERK
WARD                 SALESMAN
SMITH                CLERK
ADAMS                CLERK
MILLER               CLERK
=====================================
```

※ 상호관련 서브쿼리는 main query 부터 실행이 된다.

메인쿼리의 컬럼을 하나씩 서브쿼리에서 읽으면서 서브쿼리가 완성되고

서브쿼리가 완성된 후 메인쿼리가 수행되어 완성되어서 출력이 된다.

★ exists 문

예제 1. 부서 테이블의 부서번호를 출력하는데 사원 테이블에 존재하는 부서번호만 출력하시오 !

```
답1
select deptno
  from dept
  where deptno in (select deptno
                     from emp);

답2
select deptno
  from dept d
  where exists (select 'A'
                  from emp e
                  where e.deptno = d.deptno);
```

```
==========
    DEPTNO
----------
        10
        30
        20
==========
```

※ exists 문은 메인 쿼리부터 수행한다.
exists 문은 메인쿼리의 데이터를 서브쿼리에서 찾을때
존재하면 더 이상 찾지 않고 멈춘다.
그래서 검색속도가 빠르다.

예제 2. telecom_price에는 존재하는데 우리반 테이블에는 존재하지 않는 통신사를 telecom_price에서 지우시오 !

```
delete from telecom_price t
  where not exists (select 'A'
                      from emp2 e
                      where e.telecom = t.telecom_name);

select * from telecom_price;
```

```
================================
TELECOM_ID TELECOM_NAME
---------- --------------------
         1 sk
         2 lg
         3 kt
================================
```

★ with 절 (p 255)

"복잡한 쿼리내에 동일한 쿼리 블럭이 두번 이상 발생하는 경우에 사용하면 코드도 단순해지고 성능도 좋아지는 SQL"

※전치사 with : ~와 함께

* with절을 사용했을 때의 이점 ?

1. 하나의 SQL안에서 반복되는 쿼리문을 반복작성하지 않고 단순하게 작성할 수 있는 장점

※ view와 with절의 차이는 ?

뷰를 만들어서 코드는 단순해졌는데 뷰는 데이터를 저장하지않고 쿼리를 저장하기 때문에
필요할때마다 뷰쿼리를 수행하므로 성능이 화살표 이전의 상태와 비슷하다.

view는 view 셀렉트 할때 마다 무거운 쿼리 수행해야 한다. 그러나 with절은 temp table만들 때
한번만 무거운 SQL을 수행하면 된다

2. 급하게 테이블을 만들어야 할 때 만들려고 DBA에게 요청하지 않아도 되고 with절로 temp table을
   만들어서 쿼리할 수 있다.
   view도 만드려면 dba허락을 받아야 한다.

예 : 수학자 가우스가 초등학교때 알아낸 공식을 이용해서 1부터 10까지 다 더한 숫자의 합을
구하시오!

```
CREATE TABLE number10
AS SELECT ROWNUM AS rnum FROM dual CONNECT
by ROWNUM <11;
        테이블 만들어야 함!
with number10 as (SELECT ROWNUM AS rnum FROM dual CONNECT
                        by ROWNUM <11)
select sum(rnum) from number10;
        테이블 안만들어도 됨
```

예제 1. 직업, 직업별 토탈월급을 출력하는데 직업별 토탈월급들의 평균값보다 더 큰것만 출력하시오 !

```
select job, sum(sal)
  from emp
  group by job
  having sum(sal) > (select avg( sum(sal) )
                     from emp
                     group by job);
```

```
============================
JOB                 SUM(SAL)
------------------ ----------
MANAGER                 8275
ANALYST                 6000
============================
```

※ 동일한 쿼리블럭(직업별 토탈월급) 두번이상 발생 (비슷한 SQL)

예제 2. 위의 SQL을 with절로 변경하시오 !

```
with job_sumsal as (select /*+ inline */ job, sum(sal) 토탈월급
                    from emp
                    group by job)
select job, 토탈월급
  from job_sumsal
  where 토탈월급 > ( select avg(토탈월급)
                     from job_sumsal);
```

```
-------------------------------------------------------------------
| Id  | Operation                  | Name                       |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                            |
|   1 |  TEMP TABLE TRANSFORMATION |                            |
```

```
|   2 |   LOAD AS SELECT           | SYS_TEMP_0FD9D6601_8A8F2 |
|   3 |    HASH GROUP BY           |                          |
|   4 |     TABLE ACCESS FULL      | EMP                      |
|*  5 |   VIEW                     |                          |
|   6 |    TABLE ACCESS FULL       | SYS_TEMP_0FD9D6601_8A8F2 |
|   7 |    SORT AGGREGATE          |                          |
|   8 |     VIEW                   |                          |
|   9 |      TABLE ACCESS FULL     | SYS_TEMP_0FD9D6601_8A8F2 |
-------------------------------------------------------------
```

※ with절 문장은 마치 job_sumsal 이라는 테이블을 하나 만들어 놓은 효과이다.
효력은 이 문장에서만 발생 나중에 job_sumsal 해도 검색 안됨!(TEMP TABLE TRANSFORMATION)
왜 이게 더 빠른 SQL이냐 동일한 쿼리블럭을 select 하는 것은 같은것을 두번함
이 쿼리는 with절에서 한번 select 하기 때문에 빠름 (거의 절반으로 시간 단축)

※ with 절의 유명한 힌트 2가지 ?

1. /*+ inline */ ---> temp 테이블 안만들겠다.
(with절이 아니라 그냥 서브쿼리로 수행)
temp 저장공간이 부족한데 with절 남발하면 오히려 오래걸림

2. /*+ materialize */ ---> temp 테이블을 만들겠다.
temp 저장공간이 다시 충분해 지면 materialize써서 temp저장공간 사용

□□□

■ 8장. 집합 연산자

"조인이 서로 다른 테이블에 있는 컬럼들을 양 옆으로 붙여서 출력하는 SQL문법이 있다면
집합연산자는 다른 테이블에 있는 컬럼들을 위 아래로 붙여서 출력하는 SQL 문법이다. "

★집합 연산자

1. UNION ALL 합집합 연산자
   " 여러 쿼리의 결과를 하나로 합쳐서 출력해주는 연산자이다"

2. UNION 합집합 연산자
   "union all 과는 다른점은 데이터의 중복을 제거하면서 데이터가 정렬이 되어서 출력된다"

3. INTERSECT 교집합 연산자

4. MINUS 차집합 연산자

※ 작성시 유의사항 !

1. 위아래의 컬럼의 개수와 데이터 타입이 맞아야 한다
2. Order by 절은 맨 아래의 쿼리에만 사용할 수 있다.

※ union 보다 union all이 성능은 좋음 (union은 정렬작업에 메모리를 더 쓰기 때문이다.)

□□

★ 9장. 데이터 조작언어 (DML)

* DML문이란?

Data maniplation language(데이터를 조작하는 언어)

* DML문의 종류 ?

1. INSERT 문

"테이블에 데이터를 입력하는 문법"

형식 : insert into 테이블 (컬럼1, 컬럼2, ...)
values (컬럼1데이터, 컬럼2데이터, ...);

※ 설명 : insert 절에는 컬럼명 리스트를 나열하고 values 절에는 컬럼명리스트에 맞게 데이터를 나열한다.
※ 데이터를 작성할 때는 숫자와는 다르게 문자와 날짜는 양쪽에 싱글 쿼테이션 마크를 써야 한다.
※ 컬럼을 입력하지 않으면 Values절에 테이블의 컬럼의 순서에 맞게 다 적어줘야 한다.
※ 날짜 입력할 때 TO_DATE
※ sysdate를 이용해서 날짜 데이터를 입력했다면 시, 분, 초 까지 데이터가 다 들어간다.

1-1. Subquery를 사용한 insert 문

형식 : insert into 테이블
select 절

※ 질의문과 테이블의 컬럼 순서가 다르면 * 를 쓸수 없다. 이거 좀 중요하다
※ 백업할 때 유용

2. UPDATE 문

"테이블에 데이터를 수정하는 문법"

형식 : update 테이블
set 수정컬럼1 = 수정값1, 수정컬럼2 = 수정값2
where 조건;

2-2. update 문의 subquery를 쓸 수 있는 절

Update 서브쿼리 가능(튜닝할 때 사용) [이거 아는사람 진짜 별로 없음]
Set 서브쿼리 가능
Where 서브쿼리 가능

• 갱신 방법 3가지
  § Merge (튜닝된 SQL)
  § Update절의 서브쿼리문 (튜닝된 SQL)
  § 상호관련 서브쿼리문을 이용한 update문 (악성 SQL)

※ update절의 subquery를 쓸때 제약조건에 걸리는 경우가 많을 수 있다.

3. DELETE : 데이터 삭제

"테이블의 데이터를 삭제하는 문법"

형식 : delete from 테이블
where 조건; <------- subquery 가능

※ 조건을 적지 않으면 테이블을 모든 데이터를 삭제한다.

★★★ 꿀팁 ! 만약 실수로 데이터를 날렸다면 10분안에 플래시백을 해야하므로 바로 보고해!

4. MERGE

"INSERT, UPDATE, DELETE 를 하나의 문장으로 한번에 수행하는 SQL"

형식 : merge into 테이블1 별칭1
using 테이블2 별칭2
on (테이블 연결 조건)
when matched then
[update set 수정사항 | delete | insert]

※ 테이블2에 서브쿼리를 사용하려면 중복제거가 필요함
※ 다른 DML문으로 하려다가 제약을 걸어야 하는 경우에 사용하는 듯!
※ merge가 진짜 중요해 현업에서 !!!

★ TCL문 3가지

" Transaction Control Language "

1. Commit

"데이터 베이스에 변경사항을 영구히 저장하겠다."

형식 : commit;

☆ commit 의 종류 2가지

1. 암시적 commit (어설프게 알고있다가 낭패를 볼 수도 있음)
- 정상종료 (exit)
- DDL문 실행 (CREATE, ALTER, DROP, TRUNCATE, RENAME)
- DCL문 실행 (GRANT, REVOKE)

1. 명시적 commit

2. Rollback

"최종적으로 commit 한 이후에 작업했던 DML 작업들을 취소하겠다."

형식 : 1. rollback;
2. rollback to [A-Z];

☆ rollback 종류 2가지

1. 암시적 rollback
- 비정상 종료
□ 도스창(사용자가)을 꺼버렸을 때 !
□ 시스템이 다운 되었을 때

2. 명시적 rollback

3. Savepiont

"롤백할 지점을 지정하는 SQL"

형식 : savepoint [A-Z];

※ ed

Demobld 저장하는 법 sqlplus에 들어가서 ed demobld.sql 만들어서 저장 그리고 @demobld로 불러오기

★ lock

"데이터를 수정/ 삭제/ 입력하지 못하게끔 행이나 테이블을 잠궈버리는 기능"

왜 잠궈버리는가? 데이터의 일관성을 유지시키기 위해서 이다.

```
A세션(도스 창 1)          |      B세션(도스창 2)
-------------------------------------------------
1. Update emp             |
Set sal = 9000            |
Where ename = 'SCOTT';    |
-------------------------------------------------
                          |      2. Update emp
                          |      Set sal = 0
                          |      Where ename = 'SCOTT';
-------------------------------------------------
3. Select ename, sal      |
From emp                  |
Where ename = 'SCOTT';    |
```

※ 2번하다가 락이 걸림 풀어주려면 1도스창에서 커밋을 해야함
내가 변경한 데이터를 다른 사람(세션)이 볼려면 내가 COMMIT을
해줘야 다른사람이 볼 수 있다. <------ " 읽기 일관성 "

LOCK???
내가 변경하고 있는 그 데이터를 내가 변경하고 있는 동안에는 그 누구도 변경하지 못하도록
그 데이터의 행(ROW)에 락을 걸어 잠궈버린다.

그리고 그 LOCK은 내가 COMMIT(ROLLBACK)을 하면 풀린다.

★ 오라클의 데이터를 삭제하는 방법 3가지

	Delete (DML)	Truncate (DML)	Drop (DDL)
데이터	삭제	삭제	삭제
저장공간	유지	삭제	삭제
저장구조	유지	유지	삭제
롤백 유무	가능	불가능	불가능
Flashback 유무	가능	불가능	가능
	Delete from emp;	Truncate table emp;	Drop table emp;

※ TRUNCATE는 무서운 명령어, 바로 지워지고 롤백도 안되고 골든타임도 없다. 백업파일로만 복구할 수 있다.
※ flashback table emp to before drop;

□□

■ 10장. DDL문

• DDL문이 필요한 이유?

데이터 분석가 입장에서 본다면 ?
분석해야 할 데이를 테이블로 생성해야 SQL을 작성할 수 있기 때문이다.
(예 : 엑셀파일, CSV 파일)

딥러닝 개발 및 연구원 입장에서 본다면 ?
5기 학생들 5 딥노이드 (정상 폐사진 VS 폐결절 사진 컴퓨터가 인식할 수 있도록 프로그래밍을 하는 회사)

★ DDL문이란?

Data Definition Language : create, alter, drop, rename
(정의)

1. create

"데이터베이스 object를 생성하는 문법"

```
형식 : 1. create [object] [object 이름]
          (컬럼명 데이터타입,
           컬럼명 데이터타입,
           .....              );

       2. create [object] [object 이름]
          as                            <------ ctas라고 한다.
          select ~~~~
```

※ 2번형식은 테이블 구조와 데이터만 가져오고 제약은 가져오지 않는다.
※ 2번형식에서 where 1=2 를 추가하면 데이터 구조만 가져올 수 있다.

※ 테이블 명 지을 때 유의사항 !

- 문자로 시작해야 한다.
- 25자 이내여야 한다
- 오라클 예약어를 사용하면 안된다. (select 같은 것들)
  - □ 예 : create table select (empno number(10));

※ 데이터 타입의 종류 3가지

- 문자형 : char, varchar2, long, clob, blob
- 숫자 : number
- 날짜 : date
- rowid

• Rowid : 해당 row의 유니크한 물리적 주소
AAAE5OAABAAALCxAAC
File번호 + block번호 + row 번호

★ long, clob, blob 데이터 타입

1. Long : 긴 텍스트 파일을 저장할 때 사용하는 데이터 타입 (최대 2GB)
2. Clob, Blob : 이미지 데이터를 저장할 때 사용하는 데이터 타입 (최대 4GB)

예 :

```
CREATE TABLE RESUME (ename      VARCHAR2(20),
                     age        NUMBER(10),
                     mobile     VARCHAR2(20),
                     self_intro LONG );

INSERT INTO RESUME (ename, age, mobile, self_intro)
   VALUES('chio jae hyeok', 27, '01053693647',
   'when i was young, my family was poor.
    my mother said, she doesn''t like chineese noodle.
    Yahee yahee Ya');
```

2. alter
"테이블의 컬럼을 추가, 삭제, 변경할 때 사용하는 명령어 "
형식 : alter [object] [object 이름]
[add | drop | modify | rename | unused] [컬럼명 데이터타입]
※ add : 추가
drop : 제거
modify : 데이터 타입 변경
rename : 이름 변경
unused : 감춘다
※ unused설명 : 다시 used 한다고 나오지 않는다.
당장 drop하면 DB가 느려지기 때문에 unused를 이용해서 컬럼을 숨기고
속도를 유지하면서 나중에 밤 10시쯤 drop 하려고 쓴다.

3, rename
"테이블의 이름 변경 방법"
형식 : rename [object 이름] to [변경할 이름];

4. drop
"object를 제거하는 방법"
5. truncate

★ 제약 (constraint)
• 테이블에 제약이 필요한 이유 ?
"데이터의 품질을 높이기 위해서 필요하다"

예 : 잘못된 DATA가 입력되지 못하도록 설정을 한다.

• 제약의 종류 (P 82)
1. Primary key
"중복된 data와 null을 입력하지 못하게"
형식 : 1. create [object] [object 이름]
(컬럼명 데이터타입 [constraint 제약이름] [primary key],
컬럼명 데이터타입,
.....);
형식 : 2. alter table [테이블명]
add constraint [제약이름] [primary key]([속성])

2. unique
"중복된 data를 입력하지 못하게"
형식 : 1. create [object] [object 이름]
(컬럼명 데이터타입 [constraint 제약이름] [unique],
컬럼명 데이터타입,
.....);
형식 : 2. alter table [테이블명]
add constraint [제약이름] [unique]([속성])

※ 형식 2는 현재 테이블에 중복된 데이터가 없어야 제약이 걸린다.

3. Not null

"Null을 입력하지 못하게"

형식 : 1. create [object] [object 이름]

```
(컬럼명 데이터타입 [constraint 제약이름] [ Not null],
 컬럼명 데이터타입,
 .....              );
```

형식 : 2. alter table [테이블명]

```
add constraint [제약이름] [Not null]( [속성] )
```

4. check

"지정된 data만 입력되게"

형식 : 1. create [object] [object 이름]

```
(컬럼명 데이터타입 [constraint 제약이름] [ check ( [속성][조건] ) ],
 컬럼명 데이터타입,
 .....              );
```

형식 : 2. alter table [테이블명]

```
add constraint [제약이름] [ check ( [속성] [조건] ) ],
```

5. Foreign key

"부모테이블의 부모키 데이터만 입력될 수 있게 참조할 때 사용하는 제약"

형식 : 1. create [object] [object 이름]

```
(컬럼명 데이터타입 [constraint 제약이름] [Foreign key]
                    references [참조테이블 ( [참조테이블 속성] ) ],
 컬럼명 데이터타입,
 .....              );
```

형식 : 2. alter table [테이블명]

```
add constraint [제약이름] [Foreign key]( [속성] )
    references [참조테이블 ( [참조테이블 속성] ) ]
```

※ 제약이름(마음대로 줘도 되나 의미있게 줘야 나중에 삭제할 때 쉽다)

• 테이블의 제약 확인

```
SELECT *
  FROM USER_constraints
  WHERE TABLE_name = '[테이블 명]' ;   <---- EMP20 대문자로 작성

SELECT *
  FROM user_cons_columns
  WHERE TABLE_name = '[테이블 명]';
```

※컬럼명 확인

• 테이블의 제약 삭제

```
ALTER TABLE [테이블명]
  DROP CONSTRAINT [제약 이름] ;
```

아이디어! :

```
SELECT ' alter table ' || TABLE_name ||
       ' drop  constraint ' || CONSTRAINT_name || ';'
  FROM USER_cons_columns;
```

※ 연결연산자로 쿼리내용을 만들어서 삭제

★ foreign key 제약

Emp900 ----------dept900
(자식) (부모)
Deptno deptno
10 10
20 20
20 30
10 40
30
10
'
'
'

※ 설명 : 만약 제약이 없어서 emp 테이블에 deptno에 80번과 같이 dept 테이블의 deptno에 없는 데이터가 들어오게 되면 나중에 조인할 때 outer join을 사용해야 한다.
그런데 outer join을 사용하게 되면 equi join일때 보다는 더 성능이 나빠질 수 있기 때문에 특별한 SQL튜닝방법이 필요하다.

※개체명 질의

문제 391. 숫자 1부터 100까지를 출력하는 쿼리를 작성하시오 !

```
SELECT ROWNUM FROM dual CONNECT
 BY ROWNUM < 101;
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

■ 11장. 기타 스키마 생성

- 데이터 베이스 오브젝트(객체) 5가지
  1. Table : 데이터를 저장하는 기본 저장 단위
  2. Index : 데이터 검색 속도를 높이는 오브젝트
  3. View : 하나 이상의 테이블에 있는 데이터의 부분집합
  4. Synonym : 동의어

★ view 가 무엇인가 ?

"쿼리의 결과 집합"

- View가 필요한 이유?

  현업에서는 테이블을 만들면서 작업하는 것이 결제 절차가 복잡해서 엄청 오래걸리고 번거롭다.
  원본테이블이 변경되면 만들었던 테이블을 최신화 해야하는 번거로움이 있다
  (뷰는 데이터가 아닌 쿼리를 저장하기 때문에 변경된 원본테이블의 최신 데이터를 반영한다)

★ view 의 종류 2가지 (p 115)

	단순 view	복합 view
테이블의 개수	1개	2개이상
함수	포함안함	포함
그룹함수	포함안함	포함
DML 여부	가능	불가능할 수도 있다.

★ 복합 뷰

예제 1. 사원이름, 부서위치를 출력하는 view를 생성하시오 (뷰 이름 : emp303)

```
CREATE VIEW emp303
AS
SELECT e.ename, d.loc
  FROM EMP e, DEPT d
  WHERE e.deptno = d.deptno;
```

※ 생성된 뷰는 e. D. 이 안붙는다

※ 생성된 뷰는 복합뷰 이다.

예제 2. 위의 emp303을 수정하는데 KING의 부서위치를 SEOUL로 변경하시오 !

```
UPDATE EMP303 SET LOC = 'SEOUL' WHERE ENAME = 'KING';
```

ORA-01779: cannot modify a column which maps to a non key-preserved table

※복합 뷰 이기 때문에 수정이 불가능 할 수도 있어서 안된다.

원래는 바꿀 수 있는데 바꾸려니 찝찝한거임 그래서 확실하게 정해줘라

```
ALTER TABLE DEPT
  ADD CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno);

UPDATE EMP303 SET ENAME = 'AAA' WHERE ENAME = 'KING';
```

※ 이건 된다.

예제 3. 직업, 직업별 토탈월급을 출력하는 view를 생성하시오 (view 이름 : dept_sumsal)

```
CREATE VIEW dept_sumsal
AS
SELECT job, SUM(sal) sumsal     <--------view 생성시 컬럼별칭을 줘야 한다.(그룹함수)
  FROM EMP
  GROUP BY job;
```

예제 4. dept_sumsal 뷰를 수정하는데 job이 MANAGER의 토탈월급을 2000으로 수정하시오 !

```
UPDATE dept_sumsal SET sumsal = 2000 WHERE job = 'MANAGER';
        ORA-01732: data manipulation operation not legal on this view
        ※안됨
```

★ view의 옵션 2가지

1. With check option : 뷰 생성시 where 절에서 기술한 조건에 위배되게끔 뷰를 수정 못하게하는 옵션
2. With read only : 뷰 전체를 수정 못하게 하는 옵션

예 :

```
CREATE VIEW emp45
as
SELECT empno, ename, job, sal
  FROM EMP
  WHERE job = 'SALESNAM'
  WITH CHECK option;

CREATE VIEW EMP525
AS
SELECT empno, ename, sal, job
  FROM EMP
  WITH READ ONLY;
```

★ sequence

"번호를 생성하는 db object"

```
CREATE SEQUENCE seq1;
SELECT SEQ1.NEXTVAL FROM dual;

INSERT INTO dept(deptno, loc,dname)
  VALUES (SEQ1.NEXTVAL, 'seoul', 'sales');
```

예제 1. 아래의 테이블을 생성하고 아래의 테이블에 번호를 1번부터 1000번 까지 입력하시오 !

```
CREATE SEQUENCE seq1;

CREATE TABLE emp418
(empno NUMBER(10) );

BEGIN FOR i IN 1 .. 100 loop
INSERT INTO emp418 VALUES(seq1.nextval);
END LOOP;
END;
/

SELECT * FROM emp418;
```

★ index

" 데이터 검색속도를 높이는 object (예 : 책의 목차) "

형식 : create index [인덱스 이름]
on [테이블명 (속성명)]

※ 인덱스의 구조 : 1. 컬럼값 + rowid로 구성
2. 컬럼값이 ascending 하게 정렬이 되어있다.
Where 절에 검색조건에서 인덱스를 한다.
그래서 인덱스를 사용하려면 where절이나 힌트가 반드시 필요하다.

※ 사람들이 현업에서 SQL을 사용하는 유저들이 교육을 받으러 올때 가장 배우고 싶은 내용?
"SQL의 검색속도를 높이는 방법을 알기 원한다"

예제 1. 아래의 SQL을 튜닝하시오 ! (order by 절 없이 이미 데이터가 정렬되어있는 인덱스에서 읽어오게끔 튜닝하시오 !)

튜닝전 :

```
SELECT ename, sal
  FROM EMP
  ORDER BY sal DESC;
```

튜닝전 :

```
SELECT /*+ index_desc(emp emp_sal) */ ename, sal
  FROM EMP
  WHERE sal >-1 ;
```

※ 힌트를 써야 한다.

• 힌트(hint) ? 오라클에게 어떠한 데이터를 보여달라고 하는것은 select 문이고 오라클에게 지금 수행하는 select 문을 어떻게 실행해달라고 하는것이 힌트이다.

예 : 커피를 주문할 때 카페라떼 주세요 ~ (SQL)
카페라떼를 주시는데요, 원드는 하우스 블랜드로 갈아주시고 우유는 120도로 데워주시고 시럽을 2번 반 넣어주세요 ! (힌트)

```
Select /*+ 힌트 */ 컬럼명, …..

Select /*+ index_desc(테이블명 컬럼명) */ 컬럼명, ……
```

• index힌트의 종류 (간략)

1. Index_desc ----> 인덱스를 descending(역순) 으로 읽어라 !
2. Index_asc ------> 인덱스를 ascending (순방향) 으로 읽어라 !

Emp_sal의 인덱스의 모습 (컬럼값 + rowid)

```
SELECT sal, ROWID
  FROM EMP
  WHERE sal > 0;
```

예제 2. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal, job
  FROM EMP
  WHERE ename || sal = 'SCOTT3000';
---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
|*  1 |  TABLE ACCESS FULL| EMP  |
---------------------------------
```

튜닝후 :

```
SELECT ename, sal, job
  FROM EMP
  WHERE ename = 'SCOTT' AND sal = 3000;
-------------------------------------------------
| Id  | Operation                    | Name      |
-------------------------------------------------
|   0 | SELECT STATEMENT             |           |
|*  1 |  TABLE ACCESS BY INDEX ROWID| EMP       |
|*  2 |   INDEX RANGE SCAN           | EMP_ENAME |
-------------------------------------------------
```

※ ename, sal 모두 인덱스가 있다.
오라클에는 인공지능프로그램이 있기 때문에 두개중 좋은 인덱스(emp_ename)를
선택해서 사용함!
왜냐하면 emp_sal 은 두건이 나오지만 emp_ename은 한건이라 간단하기 때문이다 !
월급을 인덱스로 사용하고 싶다면 ?

```
SELECT /*+ index(emp emp_sal) */ ename, sal, job
  FROM EMP
  WHERE ename = 'SCOTT' AND sal = 3000;
```

예제 3. 아래의 sql을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal
  FROM EMP
  WHERE sal LIKE '30%';
---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
|*  1 |  TABLE ACCESS FULL| EMP  |
---------------------------------
```

※ 좌변이 전혀 가공되지 않았는데 인덱스를 타지않았지만 실행계획을 보면
1 - filter(TO_CHAR("SAL") LIKE '30%') 로 변경되어 검색이 되었다
Like는 문자형 데이터를 위한 검색 명령어이다.
숫자가 우선순위가 더 높기 때문에 문자를 숫자로 변경해줘야 하는데 %를 숫자로
못바꾸니까 숫자를 문자로 변경해준다.

튜닝후 : 쿼리문에서 like를 주로 사용할 것 같은 컬럼은 처음부터 문자로 만들었어야 한다. (모델링)
위의 경우의 해결방법은 함수기반 인덱스를 생성하는 것이다 !

```
CREATE INDEX emp_sal_func
  ON EMP( TO_CHAR(sal) );
-------------------------------------------------
| Id  | Operation                    | Name         |
-------------------------------------------------
|   0 | SELECT STATEMENT             |              |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP          |
|*  2 |   INDEX RANGE SCAN           | EMP_SAL_FUNC |
-------------------------------------------------
```

★ synonym

"하나의 객체에 대해서 기존 이름외에 다른 이름을 부여하는 것"

예 : 핸드폰 기계는 하나인데 번호가 2개일 수 있다 .

```
CREATE SYNONYM employee
  FOR EMP;

SELECT * FROM employee;
```

※ emp를 앞으로 employee라고도 부르겠다!

```
DROP TABLE employee;
```

(안됨! 이걸로 일부러 드롭 못하게 synonym으로 테이블을 주는 경우도 있다.)

```
create public synonym emp for heaven.emp;
select * from emp;
```

※ 퍼블릭을 주게되면 모든 유저기 이 시너님을 사용할 수 있다 !

□□

■ 12장. 유저 엑세스 제어

"데이터 베이스의 권한 관리에 대한 내용"

• SQL의 종류

1. Query
2. DML
3. DDL
4. DCL ------> Database Control Language : grant, revoke
5. TCL

1. grant

"유저에게 권한을 부여하는 방법"

형식 : grant [권한 내용] to [유저이름];

2. revoke

" 유저에게 권한을 회수하는 방법"

형식 : revoke [권한내용] on [테이블] from [유저이름];

* 대표적인 권한 리스트

connect	role
create table	
unlimited tablespace	
select	
resource	role
dba	role
all	connect + resource

예제 1. resource 라는 role 이 가지고 있는 권한이 무엇인지 확인하시오 !

```
SELECT * FROM ROLE_sys_privs
  WHERE ROLE = 'RESOURCE';
        RESOURCE        CREATE SEQUENCE
        RESOURCE        CREATE TRIGGER
        RESOURCE        CREATE CLUSTER
        RESOURCE        CREATE PROCEDURE
        RESOURCE        CREATE TYPE
        RESOURCE        CREATE OPERATOR
        RESOURCE        CREATE TABLE
        RESOURCE        CREATE INDEXTYPE
```

1. System권한 확인 (객체를 만들 수 있는 권한)

```
SQL> connect jones/tiger

SQL> SELECT * FROM session_privs;

PRIVILEGE
----------------------------------
CREATE SESSION
UNLIMITED TABLESPACE    땅덩어리를 사용할 수 있는 권한
CREATE TABLE
CREATE CLUSTER
CREATE SEQUENCE
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE
```

2. 객체 권한 확인 (특정 데이터를 엑세스 할 수 있는 권한)
★★★Select * from user_tab_privs;★★★

★ 권한의 집합인 롤을 부여

1. 어제 유저생성하고 권한 부여했던 방법

```
CREATE USER jack
  IDENTIFIED BY tiger;

GRANT CONNECT TO jack;
          롤 (role) --------> 역할에 맞는 권한들의 집합
GRANT CREATE TABLE TO jack;
               인덱스를 생성할 수 있는 권한까지 포함된다.
GRANT CREATE VIEW TO jack;
```

2. 오늘 유저생성하고 권한 부여하는 방법

```
CREATE USER jack2
  IDENTIFIED BY tiger;

GRANT CONNECT, RESOURCE TO jack2;
               롤(role) ----->여러 권한들의 집합
```

★ 패스워드 변경

• 오라클 데이터 베이스에 유저들이 누가누가 잇는지 확인하시오 !

```
SELECT * FROM dba_users;
```

• Smih의 패스워드를 tiger로 변경하시오 !

```
ALTER USER smith
  IDENTIFIED BY tiger;
```

• 변경한 패스워드를 확인하는 방법 ?

```
SELECT * FROM dba_users;
        없음!
```

• 유저 삭제

```
DROP USER smith CASCADE;
```

※설명 : 유저를 삭제하면 해당유저가 가지고 있는 모든 객체들이 다 삭제가 된다 !

★ 계층형 질의문 (책에는 없음)

"데이터의 서열을 결과로 시각화하는 SQL"

예 :

```
SELECT LEVEL, empno, ename, mgr
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr;
```

※ 레벨은 emp 테이블에 없는 컬럼이다 밑의 두개 절을 쓰면 레벨을 출력할 수 있는데 킹부터 시작해라, 그리고 사원번호 = 관리자 번호 인 것으로 연결해라

예제 1. 위의 결과를 다시 출력하는데 BLAKE를 포함해서 BLAKE의 팀원들이 다 출력되지 않게 하시오 !

```
SELECT LEVEL, empno, ename, mgr
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr AND ename != 'BLAKE'
```

예제 2. 맨위의 결과를 다시 출력하는데 서열이 깨지지 않는 상태에서 월급이 높은 사원순으로 출력되게 하시오 !

```
SELECT LEVEL, empno, ename, mgr
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr
  ORDER siblings BY sal DESC;

SELECT RPAD(' ', LEVEL*2 ) || ename, sal
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr
  ORDER siblings BY sal DESC;
```

※이렇게 쓰면 게이트에서 계층을 좀더 잘 볼 수 있다

예제 3. 계층형 질의문과 짝꿍 함수인 sys_connect_by 함수를 이용해서 아래의 결과를 출력하시오 !

```
SELECT ename, SYS_connect_by_path(ename, '/') AS path
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr;
```

```
KING                /KING
JONES               /KING/JONES
SCOTT               /KING/JONES/SCOTT
ADAMS               /KING/JONES/SCOTT/ADAMS
FORD                /KING/JONES/FORD
SMITH               /KING/JONES/FORD/SMITH
```

□□
■ 13장. 기타 데이터베이스 객체

★ 임시 테이블

"데이터를 영구히 저장하지 않고 임시로 저장할 때 사용하는 테이블"

예 : 지금 잠깐 데이터 분석을 위해 사용할 테이블인데 나중에 필요없는 데이터인 경우에 사용하기 유용하다.

• 임시 테이블 의 종류 2가지
  1. On commit delete rows : commit을 하면 데이터가 사라진다!
  2. On commit preserve rows : 세션이 종료가 되면 데이터가 사라진다!

1. on commit delete rows 테스트

```
CREATE GLOBAL temporary TABLE emp_temp1
 (empno NUMBER(10),
  ename VARCHAR2(20),
  sal   NUMBER(10) )
  ON COMMIT DELETE ROWS ;

INSERT INTO emp_temp1 VALUES (1111, 'SCOTT', 3000);
INSERT INTO emp_temp1 VALUES (2222, 'ALLEN', 3000);
INSERT INTO emp_temp1 VALUES (3333, 'JONES', 3000);

SELECT * FROM emp_temp1;
1111    SCOTT   3000
2222    ALLEN   3000
3333    JONES   3000

COMMIT;
        ※여기서 사라질거임
SELECT * FROM emp_temp1;
        ※ 데이터가 다 사라짐
```

2. on commit preserve rows 테스트

```
CREATE GLOBAL temporary TABLE emp_temp2
 (empno NUMBER(10),
  ename VARCHAR2(20),
  sal   NUMBER(10) )
  ON COMMIT preserve ROWS ;

INSERT INTO emp_temp2 VALUES (1111, 'SCOTT', 3000);
INSERT INTO emp_temp2 VALUES (2222, 'ALLEN', 3000);
INSERT INTO emp_temp2 VALUES (3333, 'JONES', 3000);

SELECT * FROM emp_temp2;
1111    SCOTT   3000
2222    ALLEN   3000
3333    JONES   3000

COMMIT;

SELECT * FROM emp_temp2;
        ※ 데이터가 사라지지 않았음!
나갔다가 들어와서 select했더니 데이터 사라짐 !
```

★ 외부 테이블

External Table 실습예제

1. emp1.txt 편집(d:Wemp1.txt)

```
SMITH,101,2001/03/15
JOHN,102,2002/04/15
SMITH,101,2001/03/15
JOHN,102,2002/04/15
SMITH,101,2001/03/15
JOHN,102,2002/04/15
SMITH,101,2001/03/15
JOHN,102,2002/04/15
SMITH,101,2001/03/15
JOHN,102,2002/04/15
SMITH,101,2001/03/15
JOHN,102,2002/04/15
```

2. Directory 생성

```
SQL> connect   heaven/heaven

SQL> create directory emp_dir as 'c:W';
                              ↑          ↑
                    디렉토리 이름  디렉토리 위치
```

3. External table 생성

```
 SQL> create table ext_emp
         (emp_id number(3),
           emp_name varchar2(10),
           hiredate date)
           organization external          ------>      외부테이블 엔진을 쓰겠다
          (type oracle_loader             ------>      외부 테이블 엔진을 결정
           default directory emp_dir      ------>      디렉토리가 emp_dir 이다
           access parameters
           (records delimited by newline  ------>      행을 엔터로 구분하겠다
            fields terminated by ","      ------>      컬럼을 , 로 구분하겠다
           (emp_name char,                ------>      (외부 텍스트 설명) emp_name 은 문자형
           emp_id char,                   ------>       (외부 텍스트 설명) emp_id 는 문자형
           hiredate date "yyyy/mm/dd") )  ------>       (외부 텍스트 설명) hiredate 는 날짜형
            location ('emp1.txt') );      ------>      (외부 텍스트 설명) 외부 텍스트 파일 이름
```

4. Select & DML 테스트

```
SQL> select * from ext_emp;

    EMP_ID EMP_NAME            HIREDATE
---------- ------------------- --------
       101 SMITH               01/03/15
       102 JOHN                02/04/15
       101 SMITH               01/03/15
       102 JOHN                02/04/15
       101 SMITH               01/03/15
       102 JOHN                02/04/15
       101 SMITH               01/03/15
       102 JOHN                02/04/15
       101 SMITH               01/03/15
```

```
       102 JOHN                 02/04/15
       101 SMITH                01/03/15

    EMP_ID EMP_NAME             HIREDATE
---------- -------------------- --------
       102 JOHN                 02/04/15

12 rows selected.
```

에러 날 경우 외부파일 저장된 곳으로 가서 f10 ---- 도구 ---- 폴더옵션 ----- 보기 ---- 알려진 파일 형식의 파일 확장자명 숨기기 체크해제

☆ 문자집합

1. 단일 바이트 문자집합 (7bit, 8bit [ANSI]) ---> 영문을 담는 문자집합
2. 유니코드 문자집합 (UTF-8) ---> 한글, 한자, 중국어, 일본어를 담는 문자집합
   참고 : http://nuli.navercorp.com/sharing/blog/post/1079940

★ 외부 테이블로 가능한 작업들

1. View 생성 가능
2. 다른 테이블과 조인이 가능

★ 외부 테이블의 단점 !!!

1. 인덱스가 안만들어 진다!

★ flashback query

• 새로운 데이터베이스에 접속하는 방법

```
C:WUsersWAdministrator>set ORACLE_SID=orcl

C:WUsersWAdministrator>sqlplus / as sysdba

SQL> select instance_name
  2    from v$instance;

INSTANCE_NAME
----------------
orcl
```

★ 데이터 분석가에게 필요한 flashback 기술 정리

"데이터를 과거로 되돌리는 기능"

1. flashback drop
2. flashback Query
3. flashback table

★flashback drop

"recycle이 다되기 전에 drop 한 것들을 되돌려주는 기능"

```
SQL> drop table emp;

테이블이 삭제되었습니다.

SQL> show recyclebin
==========================================================
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
EMP              BIN$cyENOCPwReat6n76v0nJvw==$0 TABLE        2018-11-02:10:31:00
==========================================================
```

```
 * 휴지통에서 복원 작업
SQL> flashback table emp to before drop;

플래시백이 완료되었습니다.
```

문제 481. emp테이블을 ctas로 백업하시오 !

```
create table emp_backup
as
select *
  from emp;
```

★ flashback Query

"과거(지우기 전)의 데이터를 확인하는 쿼리문"

```
SQL> delete from emp;

14 행이 삭제되었습니다.

SQL> commit;

커밋이 완료되었습니다.

select * from emp
  as of timestamp to_timestamp('2018/11/02:10:30:00','rrrr/mm/dd:hh24:mi:ss');
```

★ flashback table

```
1. table 을 flashback table 이 가능하도록 설정한다
        alter table emp enable row movement;

2. 위에서 확인한 flashback query 시간데로 테이블을 되돌린다.
        flashback table emp to timestamp
          to_timestamp ( '2018/11/02:10:30:00','rrrr/mm/dd:hh24:mi:ss');

3. flashback 이 잘 되었는지 확인한다.
        select * from emp;

4. 커밋을 한다.
        commit;
```

문제 482. dept 테이블과 salgrade테이블을 delete하고 commit하시오 !

```
delete from dept;
delete from salgrade;
commit;
```

문제 483. dept 테이블과 salgrade테이블을 flashback해서 복구하시오 !

```
select * from dept
  as of timestamp to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');

select * from salgrade
  as of timestamp to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');

alter table dept enable row movement;
alter table salgrade enable row movement;
```

```
        flashback table dept to timestamp
          to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');

        flashback table salgrade to timestamp
          to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');
```

* 골든 타임 확인 하는 방법

```
show parameter undo_retention

        SQL> connect  sys/oracle@localhost:1522/pdborcl as sysdba
        연결되었습니다.

        SQL> show parameter undo

        NAME                                 TYPE        VALUE
        ------------------------------------ ----------- --------
        temp_undo_enabled                    boolean     FALSE
        undo_management                      string      AUTO
        undo_retention                       integer     900        <----- 복구 가능 시간 15분!
        undo_tablespace                      string      UNDOTBS1

        alter system set undo_retention=2000;


grant dba to scott;

alter user sh account unlock;

alter user sh identified by sh;

alter user hr account unlock;

alter user sh identified by hr;



SQL> exit

C:WUsersWAdministrator>set ORACLE_SID=xe

C:WUsersWAdministrator>sqlplus heaven/heaven
```

□□

■ 14장. 데이터 딕셔너리를 사용한 객체 관리 기수

" 내가 가지고 있는 테이블이나 인덱스가 뭐가 있는지 확인하기 위해 "

- 내가 가지고 있는 테이블 리스트 조회
  select table_name from user_tables;

- 데이터 베이스에 있는 모든 테이블 리스트 조회
  select table_name from dba_tables;

1. User_xxx : 내가 만들고 내가 소유한
2. All_xxx : sork 만든 객체 + 다른 유저가 권한 준 객체
3. dba_xxx : 데이터베이스에 있는 모든 객체

- Database 의 객체 5가지
  1. Table : user_tables, all_tables, dba_tables
  2. View : user_views, all_views, dba_views
  3. Index : user_indexes, all_indexes, dba_indexes
  4. Sequence : user_sequences, all_sequences, dba_sequences
  5. Synonym : user_synonyms, all_synonyms, dba_synonyms

- 데이터 분석을 하려면 테이블 정의서를 확인해야한다.
  ○ 테이블 정의서는 테이블 생성할 때 이 테이블에 대해 설명해놓은 문서이다.
  select * from user_tab_comments; 이걸 보면 테이블 정의서에 대한 내용을 확인할 수 있다.

- Emp테이블에 대해서 데이터 분석을 하기 용이하도록 emp테이블에 대한 정보를 database에 기록하시오 !
  comment on table emp is '사원에 대한 정보가 있는 테이블로서 사원번호, 이름, 월급,
  직업, 커미션, 입사일, 관리자번호, 부서번호에 대한 정보가 있습니다.';

- 사원 테이블의 컬럼에 대한 주석도 남기시오 !
  comment on column emp.empno is
  '사원번호 입니다.';

※ select table_name from user_tables; 현재 생성된 테이블 리스트 현황
※ select * from user_views; 현재 생성된 뷰 리스트 현황
※ select sequence_name from user_sequences; 현재 생성된 수열 리스트 현황
※ select index_name from user_indexes; 현재 생성된 인덱스 리스트 현황
※ SELECT * FROM session_privs; 내가 가지고 있는 권한 리스트 현황

※ select synonym_name from dba_synonyms; 현재 생성된 public synonym을 확인하는 방법

□□

■ 15장. 대형 데이터 집합 조작

- 목차
  1. INSERT 문에서의 서브쿼리
  2. 다중 INSERT문 4가지
  3. MERGE 문
  4. FLASHBACK QUERY

★ insert 문에서의 서브쿼리

```
insert into emp (empno, ename, sal)    <------- 서브쿼리 가능
  values (1111, 'scott', 3000);                  <------ 서브쿼리 가능

SQL>    insert into (select empno, ename, sal, deptno
  2                    from emp
  3                    where deptno = 10
  4                    with check option )
  5       values (&empno, '&ename', &sal, &deptno);
Enter value for empno: 7788
Enter value for ename: scott
Enter value for sal: 3000
Enter value for deptno: 10
old   5:          values (&empno, '&ename', &sal, &deptno)
new   5:          values (7788, 'scott', 3000, 10)

1 row created.
```

★다중 insert 문 4가지 (p 164)

1. 무조건 insert

```
예 : insert all
        into emp1000(empno, ename, sal, deptno)
        into emp2000( empno, ename, sal, deptno)
        into emp3000( empno, ename, sal, deptno)
        select empno, ename, sal,deptno
        from emp;
```

2. 조건부 insert

```
예 :insert all
     when deptno = 10 then
     into emp1000
     when deptno = 20 then
     into emp2000
     when deptno = 30 then
     into emp3000
     select * from emp;
```

3. 조건부 insert first

"첫번째 조건을 우선 처리하고 나머지 조건을 쿼리에 넣어줌"

```
예 : insert first
       when age >= 30 then
       into emp2_sk
       when lower(telecom) = 'lg' then
       into emp2_lg
       when lower(telecom) = 'kt' then
       into emp2_kt
       select * from emp2;
```

4. pivoting insert

예 :
```
insert all
       into crime_age2 values(local, type, sum_tot, 'under_6',under_6)
       into crime_age2 values(local, type, sum_tot, 'under_12',under_12)
       into crime_age2 values(local, type, sum_tot, 'under_15',under_15)
       into crime_age2 values(local, type, sum_tot, 'under_20',under_20)
       into crime_age2 values(local, type, sum_tot, 'under_30',under_30)
       into crime_age2 values(local, type, sum_tot, 'under_40',under_40)
       into crime_age2 values(local, type, sum_tot, 'under_50',under_50)
       into crime_age2 values(local, type, sum_tot, 'under_60',under_60)
       into crime_age2 values(local, type, sum_tot, 'over_60', over_60)
     select local, type, sum_tot, under_6,under_12,under_15,under_20,
            under_30,under_40,under_50,under_60,over_60
       from crime_age;
```

(3권 162 페이지의 그림 참고!)

"다중 insert문은 여러개의 테이블에 하나의 데이터를 동시에 입력하는 SQL문"

★ merge 문 (p 180)

" insert, update, delete 를 한번에 수행하는 명령어"

예제 1. emp 테이블로 emp5000 테이블을 생성하고 emp5000테이블의 데이터 절반을 지우시오!

```
create table emp5000
as
select * from emp;

delete from emp5000 where rownum <=8;

commit;

SQL> select count(*) from emp5000;

  COUNT(*)
----------
         9

update emp5000
  set sal = 0;

commit;
```

예제 2 . emp테이블의 있는 데이터를 emp5000에 merge하는데 emp 테이블과 emp5000에 양쪽에 다 존재하는 사원들은 emp5000 의 사원들의 월급을 emp테이블의 사원의 월급으로 변경하고 그렇지 않고 emp테이블과 emp5000에 양쪽에 다 존재하지 않고 emp 에만 존재하는 사원들은 emp5000에 입력하시오 !

```
merge into emp5000 e5
using emp e
on (e5.empno = e.empno)
when matched then
 update set e5.sal = e.sal
when not matched then
insert (e5.empno, e5.ename, e5.job, e5.mgr, e5.hiredate, e5.sal, e5.comm, e5.deptno)
values (e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm, e.deptno);
```

예제 3. 문제 497번 상황을 똑같이 만들고 emp테이블에서 emp5000테이블에 있는 사원 한명을 지우시오!

merge 문 3가지 수행

1. emp의 월급으로 -----> emp5000의 월급 갱신
2. emp -----> emp5000에 insert
   (emp5000에 없는 사원들의 데이터만 입력)
3. emp테이블에는 존재하지 않는데 emp5000에만 있는 사원들은 지우시오 !

```
delete from emp where ename = 'FORD';

merge into emp5000 a
  USING
    (SELECT e.*, 'N' AS del
       from emp e
     union all
     select e5.*, 'Y' AS del
       from emp5000 e5
       WHERE e5.empno NOT IN (SELECT EMPNO FROM emp) ) b
  ON(a.empno = b.empno)
  when matched then
    update set a.sal = b.sal
    delete where ( b.del = 'Y' )
  when not matched then
    insert (empno, ename, sal, job, hiredate, mgr, comm, deptno)
    values(b.empno, b.ename, b.sal, b.job, b.hiredate, b.mgr, b.comm, b.deptno) ;
```

※ using절에서 delete 했는데 이경우 emp5000의 데이터를 지운 것이므로
마지막에 insert할 때 FORD는 emp5000, using 절의 서브쿼리문 어디에도
존재하지 않는다

□□

■ 16장, 다른 시간대에서의 데이터 관리
"앞에서 배우지 않았던 다른 날짜 함수들"

★ extract 함수 (p 214)

예제 : 오늘 날짜에서 년도만 추출하시오 !

```
select extract ( year from sysdate )
  from dual;
```

```
========================
EXTRACT(YEARFROMSYSDATE)
------------------------
                    2018
========================
```

예제 1. 오늘날짜에서 달을 추출하시오 !

```
select extract ( month from sysdate )
  from dual;
EXTRACT(MONTHFROMSYSDATE)
-------------------------
                       11
```

예제 2. 입사한 년도, 입사한 년도별 토탈월급을 출력하시오 !

```
select extract ( year from hiredate ), sum(sal)
     from emp
     group by extract ( year from hiredate );
```

```
====================================
EXTRACT(YEARFROMHIREDATE)   SUM(SAL)
------------------------- ----------
                     1982       4300
                     1983       1100
                     1980        800
                     1981      19825
====================================
```

★ tz_offset 함수
"영국의 그리니치 천문대를 기준으로 각 나라의 시간대를 확인하는 함수"

예 : 우리나라의 시간대로 확인하시오 !

```
select * from v$timezone_names
  where lower(tzname) like '%seoul%';

select tz_offset ('Asia/Seoul')
  from dual;
```

```
==============
TZ_OFFSET('ASI
--------------
+09:00
==============
```

★ to_timestamp 함수 (p 218)

- flashback query시에 활용
- flashback table시에 활용

예제 1. 지금부터 10분전에 emp테이블의 king의 월급이 얼마였는지 확인하시오 !

```
update emp
  set sal = 0
  where ename = 'KING';

commit;

select ename, sal
  from emp
  as of timestamp (systimestamp - interval '10' minute)
  where ename = 'KING';
```

```
==============================
ENAME                      SAL
-------------------- ----------
KING                       5000
==============================
```

★ to_yminterval 함수

예 : 오늘부터 1년 2개월 후의 날짜가 어떻게 되는가?

```
select sysdate + to_yminterval('01-02')
  from dual;
```

```
========
SYSDATE+
--------
20/01/06
========
```

예제 2. 위의 문제를 to_yminterval 사용하지 않고 해결하시오 !

```
select add_months(sysdate, 14)
  from dual;
```

```
========
ADD_MONT
--------
20/01/06
========
```

★ to_dsinterval (days ~ seconds)

예 : 지금부터 100일 10시간 뒤의 날짜가 어떻게 되는가 ?

```
select sysdate + to_dsinterval('100-10:00:00')
  from dual;
```

□□

■ 18장. 정규식 지원

* 정규식 함수란 ? (p 268)
"문자열 데이터의 간단한 패턴 또는 복잡한 패턴을 검색하고 조작할 수 있게 해주는 함수"

regexp -----> regular expression (정규 표현식)

* 정규식 함수의 종류 (p 270)

1. regexp_like

```
select ename, sal
  from emp
  where regexp_like(ename, 'EN|IN');
```

※ 설명 : 정규표현식에서 ' | ' 는 '또는'을 의미한다 !

```
select first_name
  from employees
  where first_name like 'St%'
    and first_name like '%en';
```

정규표현식

```
select first_name
  from employees
  where regexp_like ( first_name, '^Ste(.)+en$' );
```

※ 설명 (p 273 ~ 274)
- ^ : 문자열의 시작
- $: 문자열의 끝
- ( ) : 괄호로 묶은 식은 한단위로 나타냄
- . : 문자자리 한개
- (.)+ : 한번이상 발생한 수 일치

```
select ename
  from emp2
  where regexp_like ( ename, '^(김|이)(.)진$');
```

성씨가 김씨 또는 이씨로 시작하고 이름의 끝글자가 '진'인 학생의 이름을 출력

---

2. regexp_replace

```
select ename, regexp_replace(sal, '[0-3]', '*')
  from emp;
```

0~3까지를 *로 출력

```
select regexp_replace ( phone_number , '₩.', '-' )
  from employees;
```

※ 설명 : 백슬래시(₩)를 사용하여 일반적으로 메타 문자로 처리되는 문자를 검색합니다.

3. regexp_instr

```
        select street_address, regexp_instr ( street_address, '[[:alpha:]]')
          from locations;
                locations 테이블에 street_address데이터에서 알파벳으로 시작되는 부분의 자리 위치 출력
        ------------------------------------------------------------------------
        select street_address, regexp_instr ( street_address, '[^[:alpha:]]')
          from locations;
                알파벳이 아닌 곳의 위치번호 출력
```

4. regexp_substr

```
        select m_name, regexp_substr ( m_name, '[^-]+', 1, 2) 가수
          from music
          WHERE INSTR ( m_name, '-') != 0;
                ※ 하이픈(-)을 기준으로 문자열을 덩어리로 나눠서 첫번째 글자부터 (-)찾아서
                   두 부분로 쪼갠후 2번째 문자열을 출력
```

5. regexp_count

```
        select sum(regexp_count (LOWER(win_text), ' elsa ') ) elsa_cnt
          from winter_kingdom;
                문제 533. 겨울왕국 대본에는 elsa가 몇번 나오는지 regexp_count로 카운트
```

```
========================================
 * 12c 로 접속

C:WUsersWstu>set ORACLE_SID=orcl

C:WUsersWstu>sqlplus  / as sysdba

alter session set container=pdborcl;

alter  database  pdborcl  open;

connect scott/tiger@localhost:1522/pdborcl

create  table  employees
as
select * from hr.employees;

 * DB링크 걸기 (xe에서 해)
create database link link_12c
connect to scott
identified by tiger
using 'localhost:1522/pdborcl';
        우리가 필요한 정보
        |여삼빌딩|10층|아이티윌|

select * from employees@link_12c

(한번 가져와서 그냥 만들어 버려라 [데이터 이행을 이런식으로 한다] )
CREATE TABLE employees
as
SELECT * FROM employees@link_12c;

SELECT * FROM employees;
========================================
```

POSIX Operators in Oracle SQL Regular Expressions

. : The expression a.b matches the strings abb, acb, and adb, but does not match acc.

+ : The expression a+ matches the strings a, aa, and aaa, but does not match ba or ab.

* : The expression ab*c matches the strings ac, abc, and abbc, but does not match abb or bbc.

? : The expression ab?c matches the strings abc and ac, but does not match abbc or adc.

{m} : The expression a{3} matches the string aaa, but does not match aa.

{m, } : The expression a{3,} matches the strings aaa and aaaa, but does not match aa.

{m, n} : The expression a{3,5} matches the strings aaa, aaaa, and aaaaa, but does not match aa or aaaaaa.

[char...] : The expression [abc] matches the first character in the strings all, bill, and cold, but does not match any characters in doll.

[^char...] : The expression [^abc]def matches the string xdef, but not adef, bdef, or cdef.
The expression [^a-i]x matches the string jx, but does not match ax, fx, or ix.

[alt1 | alt2] : The expression a|b matches the character a or b.

(expr) : The expression (abc)?def matches the strings abcdef and def, but does not match abcdefg or xdef.

₩n : The expression (abc|def)xy₩1 matches the strings abcxyabc and defxydef, but does not match abcxydef or abcxy The expression ^(.*)₩1$ matches a line consisting of two adjacent instances of the same string.

₩ :The expression abc₩+def matches the string abc+def, but does not match abcdef or abccdef. (escape)

^ : The expression ^def matches the substring def in the string defghi but not in the string abcdef.

$:The expression def$ matches the substring def in the string abcdef but not in the string defghi.

[:class:] : The expression [:upper:]+, which specifies one or more consecutive uppercase characters, matches the substring DEF in the string abcDEFghi, but does not match any substring in abcdefghi.

[.element.] : The expression [.ch.], which specifies the collating element ch, matches ch in the string chabc, but does not match any substring in cdefg.
The expression [a-[.ch.]] specifies the range from a through ch.

[=char=] : The expression [[=n=]], which specifies characters equivalent to n in a Spanish locale, matches both N and ñ in the string El Niño.

□□

★ 기본 튜닝!!!!

- 블럭 개수 보는 코드 (sqlplus)

```
set autot traceonly STATISTICS;

db block gets
consistent gets       <<<<------확인해 숫자가 적으면 튜닝된거!
physical reads
```

- sqlplus에서 실행계획 보는법

```
set autot traceonly explain
```

※ SORT라는 말이 있으면 안좋은 것이다 !

1. where 절에 될 수 있으면 좌변을 가공하지 마라!

2. 집합연산자 남발 보다는 그룹함수로 해결하라 !

3. order by 절을 쓰는 것 보다는 가급적 인덱스를 이용하는게 속도개선에 효과가 있다.

□□

문제 1. 이름과 월급과 직업과 부서번호를 출력하시오!
SQL> select ename, sal, job, deptno from emp;
SQL> set pages 4000 출력되는 결과의 세로 길이
SQL> show pages 현재 페이지의 길이를 보여줌
pagesize 4000

문제 2. 사원번호, 이름, 직업, 월급, 커미션을 출력하시오.
SQL> select empno, ename, job, sal, comm from emp;

문제 3. 직업만 출력하시오!
SQL> select job from emp;

문제 4. 직업을 출력하는데 중복을 제거해서 출력하시오!
SQL> select disjint job from emp; Distinct 중복제거 키워드

문제 5. 부서번호를 출력하는데 중복을 제거해서 출력하시오!
SQL> select distinct deptno from emp;

문제 6. 사원 태이블의 모든 컬럼과 데이터를 다 출력하시오!
SQL> select * from emp; * --> asterisk (asta)

문제 7. 관리자의 사원번호를 출력하는데 중복을 제거해서 출력하시오! (오늘의 마지막 문제)
SELECT DISTINCT mgr FROM emp;

문제 8. 사원번호, 이름과 월급과 커미션을 출력하시오!
SELECT empno, ename, sal, comm FROM emp;

문제 9. 사원테이블 전체를 다 검색하시오 !
SELECT * FROM emp;

문제 10. 이름, 월급, 커미션, 월급 + 커미션을 출력하시오!
SELECT ename, sal, comm, sal+nvl(comm,0) FROM emp;

문제 11. 이름과 월급을 출력하는데 연결 연산자를 사용해서 연결해서 출력하시오!
SELECT ename || sal FROM emp;

문제 12. 이름과 월급을 출력하는데 아래와 같이 문자열로 출력되게 하시오 !
SCOTT의 월급은 3000 입니다.
SELECT ename||'의 월급은 '||sal||' 입니다.' FROM emp;

문제 13. 아래와 같이 결과를 출력하시오!
SCOTT의 "월급" 은 3000 입니다.
SELECT ename||' ''월급''은 '||sal||'입니다.'FROM emp;

문제 14. dept 테이블의 구조를 확인하시오 ! (책 페이지 번호 79)
Desc dept
Select * from dept;

문제 15. 이름과 월급과 직업을 아래와 같이 출력하시오!
Employee name Salary Job
SCOTT 3000 ANAIYST
SELECT ename AS "employee name", sal AS "salary", job AS "job" FROM emp;

문제 16. 커미션이 300인 사원의 이름과 커미션과 직업을 출력하시오.
```
SELECT ename, comm, job FROM emp WHERE comm = 300;
```

문제 17. 이름이 SCOTT인 사원의 이름과 월급을 출력하시오!
```
SELECT ename, sal FROM emp WHERE ename = 'SCOTT';
```

문제 18. 직업이 SALESMAN인 사원들의 이름과 직업을 출력하시오!
```
SELECT ename, job FROM emp WHERE job = 'SALESMAN';
```

문제 19. 부서번호가 10번인 사원들의 이름과 월급과 직업과 부서번호를 출력하시오!
```
SELECT ename, sal, job, deptno from emp where deptno = 10;
```

문제 20. 1981년 11월 17일에 입사한 사원의 이름과 입사일을 출력하시오!
```
select ename, HIREDATE from emp where hiredate = '81/11/17';    에러
Select * from emp where hiredate = to_date('81/11/17','RR/MM/DD');
```

문제 21. (점심시간 문제) 월급이 1500 이상인 사원들의 이름과 월급을 출력하시오!
```
select ename, sal from emp where sal >= 1500;
```

문제 22. 나이가 27살인 학생들의 이름과 나이와 전공을 출력하시오.
```
select ename, age, major from emp2 where age = 27;
```

문제 23. sk 텔레콤을 사용하는 학생들의 이름과 나이와 텔레콤을 출력하시오!
```
select ename, age, telecom from emp2 where telecom = 'sk' or telecom = 'SK';
```

문제 24. 30살 이상인 학생들의 이름과 나이를 출력하시오 !
```
select ename, age from emp2 where age>=30;
```

문제 25. 컴퓨터 공학과인 학생들의 이름과 전공을 출력하시오.
```
select ename, major from emp2 where major = '컴퓨터공학과';
```

문제 26. 통계학과인 학생들의 이름과 전공을 출력하시오,
```
select ename, major from emp2 where major = '통계학과';
```

문제 27. 월급이 1000에서 3000사이인 사원들의 이름과 월급을 출력하시오!
```
select ename, sal from emp where sal between 1000 and 3000;
```

문제 28. 나이가 25에서 30살 사이가 아닌 학생들의 이름과 나이를 출력하시오!
```
select ename, age from emp2 where age > 30 or age < 25;
select ename, age from emp2 where age not  between 25 and 30;
```

문제 29. 전공에 통계가 포함되어져 있는 학생들의 이름과 전공을 출력하시오!
```
select ename, major from emp2 where major like '%통계%';
```
※ 설명 : like : ~처럼, ~ 일것 같은

문제 30. 서울에서 사는 학생들의 이름과 주소를 출력하시오!
```
select ename, address from emp2 where address like '%서울%';
```

문제 31. 이름에 첫 글자가 S로 시작하는 사원들의 이름을 출력하시오!
```
select ename from emp where ename like 'S%';
```

문제 32. 이름의 끝글자가 진으로 끝나는 학생들의 이름을 출력하시오.
```
select ename from emp2 where ename like '%진';
```

문제 33. 이름의 두번째 글자가 '혜' 자인 학생들의 이름을 출력하시오.
```
select ename from emp2 where ename like '_혜%';
```

문제 34. 이름에 세번째 철자가 L인 사원들의 이름을 출력하시오!
```
select ename from emp where ename like '__L%';
```

==========
```
insert into emp(empno, ename, sal)
values(2121,'A%B', 3400);
commit;

select ename from emp where ename like '_%%';
```
==========

문제 35. 이름의 두번째 철자가 %인 사원의 이름을 출력
```
select ename from emp where ename like '_m%%' escape 'm';
```

==========
```
insert into emp(empno, ename, sal) values (1343,'A%%B',4000);
commit;
```
==========

문제 36. 이름의 두번째 철자도 %이고 세번째 철자도 %인 사원들의 이름을 출력하시오!
```
select ename from emp where ename like '_a%a%%' escape 'a';
```

문제 37. 통계관련 학과가 아닌 학생들의 이름과 전공을 출력하시오!
```
select ename, major from emp2 where major not like '%통계%';
```

문제 38. 네이버 메일을 사용하는 학생들의 이름과 이메일을 출력하시오!
```
select ename, email from emp2 where email like '%naver%' or email like '%NAVER%';
```

문제 39. 사원번호가 7788번인 사원의 사원번호와 이름을 출력하시오!
```
select empno, ename from emp where empno=7788;
```

문제 40. 사원번호가 7788,7902, 7369번인 사원의 사원번호와 이름을 출력하시오!
```
select empno, ename from emp where empno in(7788,7902, 7369);
```

문제 41. 직업이 SALESMAN, ANALYST인 사원들의 이름과 직업을 출력하시오!
```
select ename, job from emp where job in ('SALESMAN', 'ANALYST');
```

문제 42. 부서번호가 10번, 20번이 아닌 사원들의 이름과 부서번호를 출력하시오!
```
select ename, deptno from emp where deptno not in (10,20);
```

문제 43. 커미션이 null인 사원들의 이름과 커미션을 출력하시오!
```
select ename, comm from emp where comm is null;
```

문제 44. 커미션이 null이 아닌 사원들의 이름과 커미션을 출력하시오!
```
select ename, comm from emp where comm is not null;
```

문제 45. 직업이 SALESMAN이고 월급이 1000 이상인 사원들의 이름과 월급과 직업을 출력하시오
```
select ename, sal, job from emp where job='SALESMAN' AND SAL >=1000;
```

문제 46. 직업이 SALESMAN 이거나 CLERK이고 월급이 1000 이상인 사원들의 이름과 월급과 직업을 출력하시오!
```
select ename, sal, job from emp where (job='SALESMAN' or job='CLECK') and sal >= 1000;
```

문제 47. 통신사가 lg 이거나 kt 이고 나이가 25 이상인 학생들의 이름과 나이와 통신사를 출력하시오!
```
select ename, age, telecom from emp2 where (telecom='kt' or telecom='lg') and age>=25;
```

문제 48. (마지막 문제) 아래와 같이 결과를 출력하시오!
엄한솔 학생은 27살이고 전공이 프랑스어학과이며 sk 통신사를 사용하고 있습니다.
```
select ename||' 학생은'||age||'살이고 전공이'||major||'이며'||telecom||' 통신사를 사용하고
있습니다.' from emp2;
```

문제 49. 우리반 테이블에서 이름과 나이를 출력하는데 나이가 높은학생부터 출력하시오!
```
select ename, age from emp2 order by age desc;
```

문제 50. 직업이 SALESMAN인 사원들의 이름과 직업과 입사일을 출력하는데 최근에 입사한 사원부터 출력하시오!
```
select ename, job, HIREDATE from emp where JOB = 'SALESMAN' ORDER BY    HIREDATE DESC;
```

문제 51. 통계학과인 학생들의 이름과 나이와 전공을 출력하는데 나이가 높은 학생부터 출력하시오!
```
select ename, age, major            3
  from emp2                                  1          실행순서
  where major like '%통계%'       2
  order by age desc;                    4
```

문제 52. 이름, 직업, 월급을 출력하는데 직업은 abcd 순으로 정렬해서 출력하고, 직업이 abcd순서로 정렬되어 있는 것을 기준으로 월급은 높은 순으로 출력하시오!
```
select ename, job, sal
   from emp
   order by job asc, sal desc;

select ename, job, sal
   from emp
   order by 2 asc, 3 desc;
                        결과가 같음
```

문제 53. 월급이 1000에서 3000사이인 사원들의 이름과 월급을 출력하는데 월급이 높은 사원부터 출력하시오!
```
select ename, sal
   from emp
   where sal between 1000 and 3000
   order by sal desc;
```

★ 서울시 물가 데이터 테이블 생성후 데이터 입력
카페 머신러닝 데이터 탭 6번 서울시 대형마트와 전통시장 물가 데이터

문제 54. 남대문 시장에서 파는 물건과 가격을 출력하는데 가격이 높은 것부터 출력하시오!
```
select a_name, a_price
   from price
   where m_name like '남대문시장%'
   order by a_price desc;
```

문제 55. m_name(판매처)을 중복제거해서 출력하시오!
```
select distinct m_name
   from price;
```

문제 56. 신세계 백화점에서 파는 물건이름과 가격과 단위(a_unit)을 출력하는데 가격이 높은 것 부터 출력하시오!
```
select a_name, a_price, a_unit
   from price
   where m_name like '%신세계%'
   order by a_price desc;
```

문제 57. 이름과 월급을 출력하는데 이름을 출력할 때 소문자로 출력하시오!

```
select lower(ename), sal
        from emp;

select lower(ename), sal
        from emp
  where lower (ename)= 'scott';
                이것도됨
```

문제 58. 우리반 테이블에서 네이버 이메일을 사용하는 학생들의 이름과 이메일을 출력하시오!

```
select ename, email
   from emp2
   where lower(email) like '%naver%';
                        대소문자를 구분하지않고 검색할 수 있음!
```

문제 59. 이름을 출력하는데 이름의 첫번째 철자는 대문자로 출력하고 나머지는 소문자로 출력하시오!

```
select initcap(ename)
   from emp;
```

문제 60. 이름의 첫번째 철자를 출력하는데 소문자로 출력하시오!

```
select lower(substr(ename,1,1))
   from emp;                                                    함수는 중첩이 가능하다!
```

문제 61. (점심시간 문제) 아래의 SQL의 결과를 initcap 쓰지말고 upper, lower, substr, || 를 이용해서 출력하시오!

```
select initcap(ename)
   from emp;

select substr(upper(ename),1,1)||substr(lower(ename),2,99)
   from emp;
```

문제 62. 이름, 이메일, 이메일 철자의 개수를 출력하는데 이메일 철자의 개수가 많은 학생부터 출력하시오!

```
select ename, email, length(email)
   from emp2
   order by length(email) desc;
```

문제 63. 이메일에서 @이전철자들만 출력하시오!

```
select substr(email,1,instr(email,'@')-1)
    from emp2;
```

문제 64. 이메일에서 아래의 철자만 출력하시오!

```
select substr(email, instr(email,'@')+1,instr(email,'.')-instr(email,'@')-1)
   from emp2;
```

이거 넣어보자

```
insert into emp(empno, ename, sal)
   values(2929,'JACK     ',4500);

commit;
```

문제 65. 이름이 JACK인 사원의 이름과 월급을 출력하시오!

```
select ename, sal
   from emp
   where trim(ename) ='JACK';
```

문제 66. 이름과 월급을 출력하는데 숫자 0을 *로 출력하시오 !
```
select ename, replace(sal,0,'*')
   from emp;
```

문제 67. 이름과 월급을 출력하는데 월급을 출력할 때에 숫자 0부터 3까지는 *로 출력되게 하시오!
```
select ename, regexp_replace(sal,'[0-3]','*')
   from emp;
```

문제 68. 이름, 나이, 나이가 짝수이면 0을 출력하고 나이가 홀수이면 1을 출력하시오!
```
select ename, mod(age,2)
   from emp2;
```

문제 69. 위의 SQL에 replace함수를 써서 아래와같이 결과를 출력하시오! (난이도 상)
```
select ename, replace(replace(mod(age,2),'0','짝수'),'1','홀수')
   from emp2;
```

문제 70. 이름, 입사한 날짜부터 오늘까지 총 몇일 근무했는지 출력하시오!
```
select ename, sysdate - hiredate
from emp;
```

문제 71. 이름, 내가 태어난 날부터 오늘까지 총 몇일 살았는지 출력하시오!
```
select ename, round(sysdate - birth)
   from emp2
   where ename = '정성호';
```

문제 72. 이름, 내가 태어난 날부터 오늘까지 총 몇 달 살았는지 출력하시오!
```
select ename, round(months_between(sysdate, birth))          ##앞이 최근 뒤가 나중
   from emp2
   where ename = '정성호';
```

문제 73. 위의 결과를 다시 출력하는데 개월수가 높은 학생부터 출력하시오!
```
select ename, round(months_between(sysdate, birth))
   from emp2
   order by round(months_between(sysdate, birth)) desc;
```

문제 74. 오늘날짜에서 100일 뒤의 날짜를 출력하시오!
```
select sysdate + 100
   from dual;
```

문제 75. 오늘날짜에서 100달뒤의 날짜를 출력하시오!
```
select add_months(sysdate,100)
   from dual;
```

문제 76. 오늘날짜에서 앞으로 돌아올 금요일의 날짜를 출력하시오!
```
select next_day(sysdate,'금요일')
   from dual;
```

문제 77. 오늘날짜에서 앞으로 100달뒤에 돌아올 월요일의 날짜를 출력하시오!
```
select next_day(add_months(sysdate, 100),'월요일')
   from dual;
```

문제 78. 이번달의 마지막 날짜를 출력하시오!
```
select last_day(sysdate)
   from dual;
```

문제 79. 오늘의 무슨요일인지 출력하시오!
```
select to_char(sysdate, 'day')        # day자리에 어떤 포멧을 줄 수 있음! (p 172)
   from dual;
```

문제 80. 내가 무슨요일에 태어났는지 확인하시오!

```
select to_char(birth,'day')
   from emp2
   where ename = '정성호';
```

문제 81. 이름, 태어난 요일을 출력하는데 일 월 화 수 목 금 토 순으로 출력하시오!

```
select ename, to_char(birth,'day')
   from emp2
   order by to_char(birth,'d') asc;
```

문제 82. (오늘의 마지막 문제) 이름, 태어난 요일을 출력하는데 월화수목금토일 순으로 출력하시오!

```
select ename, to_char(birth,'day')
   from emp2
   order by replace(to_char(birth,'d'),1,8) asc;
```

문제 83. 이름과 입사일, 입사한 년도를 출력하는데 입사한 년도를 아래와 같이 4자리로 출력하시오!

SCOTT 1981/12/11 1981

```
select ename, hiredate, to_char(hiredate,'RRRR')
   FROM EMP;
```

날짜를 문자로 변환하는데 문자로 변환할 때 년도 4자리로 출력해라~

문제 84. 현재 세션의 날짜 포멧을 확인하시오!

```
select * from nls_session_parameters;
```

문제 85. 81년 11월 17일에 입사한 사원의 이름과 입사일을 출력하시오 !

```
select ename, hiredate
   from emp
   where hiredate = to_date('81/11/17','rr/mm/dd');

select ename, hiredate
   from emp
   where hiredate = to_date('81/11/17','yy/mm/dd');
```

결과 출력이 안됨

문제 86. 1993년도에 태어난 학생들의 이름과 생일을 출력하는데 두가지 방법으로 수행하시오!

1. To_char

```
select ename, birth
   from emp2
   where to_char(birth, 'rrrr') = '1993';          # 문자 = 문자 형식으로 만들어 줘야함
```

(' ' 들어감)

2. substr

```
select ename, birth
   from emp2
   where substr(birth,-2,2) = 93;       # -2는 뒤에서 출력할 때 -3 -2 -1 0 1 2 3
```

문제 87. 12월달이 생일인 학생들의 이름과 생일을 출력하시오!

```
select ename, birth
   from emp2
   where to_char(birth, 'mm') = '12';
```

문제 88. ww와 iw의 차이를 알아내시오 ~

ww : 요일에 관계 없이 7일을 기준으로 주차를 구분

iw : 요일(월화수목금토일 순)을 기준으로 주차를 구분 이따가 다시 해보자

※ 설명 : iw는 iso 표준에 따른 연의 주를 의미(한주의 시작을 iso 표준은 월요일로 본다)

Ww는 7일씩 끊어서 주를 표시한다.

문제 89. 81년 11월 17일에 입사한 사원의 이름과 입사일을 출력하시오!

```
select ename, hiredate
   from emp
   where hiredate = to_date('81/11/17', 'rr/mm/dd');
```

문제 90. 81년 11월 17일에 입사한 사원들의 이름과 입사일을 출력하는데의 아래의 공란을 채우시오!

```
select ename, hiredate
   from emp
   where                                                    ='81/11/17';

select ename, hiredate
   from emp
   where to_char(hiredate,'rr/mm/dd') ='81/11/17';
```

튜닝후 :

```
select ename, hiredate
   from emp
   where hiredate = to_date('81/11/17', 'rr/mm/dd');
```

문제 91. 아래의 SQL을 튜닝하시오!

```
select ename, sal
   from emp
   where sal*12=36000;
```

튜닝후 :

```
select ename, sal
   from emp
   where sal = 36000/12;
```

문제 92. 아래의 SQL을 튜닝하시오!

```
select ename, job
   from emp
   where substr(job,1,5) = 'SALES';
```

튜닝후 :

```
select ename, job
   from emp
   where job like 'SALES%';
```

문제 93. (점심시간 문제) 1994년에 태어났고 나 통신사를 사용하는 학생들의 이름과 나이와 통신사를 출력하는데 생일일 빠른 학생부터 출력하고 컬럼명이 한글로 이름, 나이, 통신사라고 출력되게 하시오!

```
select ename as 이름, age as 나이, telecom as 통신사
   from emp2
   where to_char(birth, 'rrrr') = '1994' and lower(telecom) = 'sk'
   order by birth asc;
```

문제 94. 이름과 커미션을 출력하는데 커미션이 null인 사원들은 0으로 출력하시오

```
select ename, nvl(comm,0)
   from emp ;
```

문제 95. 이름과 커미션을 출력하는데 커미션이 null인 사원들은 no comm이란 글씨로 출력되게 하시오!

```
select ename, nvl(to_char(comm), 'no comm')
   from emp;                        문자          문자
```

문제 96. 이름, 통신사, 통신요금을 출력하는데 통신사가 sk면 50000이 출력되게 하고 lg면 60000 이 출력되게 하고 kt면 55000이 출력되게 하시오!

```
select ename, telecom, decode(lower(telecom), 'sk',50000,'lg',60000,'kt',55000)
   from emp2;
```

문제 97. 이름, 입사한 년도(4자리), 보너스를 출력하는데 입사한 년도가 1981년도면 보너스를 9000으로 출력하고 나머지년도는 다 0으로 출력하시오!

```
select ename, to_char(hiredate,'rrrr'), decode(to_char(hiredate,'rrrr'),'1981',9000,0)
   from emp;
```

문제 98. 이름, 나이, 등급을 출력하는데 나이가 30살 이상이면 A 등급, 나이가 28살 이상이면 B등급, 나이가 25살 이상이면 C등급, 나머지는 D등급으로 출력하시오!

```
"decode 함수는 등호 비교만 가능하다!!!!"
select ename, age,case when age >= 30 then 'A'
                      when age >= 28 then 'B'
                      when age >= 25 then 'C'
                      else '0' end as 등급
   from emp2;
```

※ case의 형식 : case when ○ then ○ else ○ end

문제 99. 이름, 월급과 보너스를 출력하는데 월급이 3000 이상인 사원들은 보너스를 자기의 월급의 10%로 출력되게 하고 월급이 2000 이상인 사원들은 보너스를 자기의 월급의 30%로 출력되게 하고 나머지 월급은 그냥 0을 출력하시오!

```
select ename, sal, case when sal >= 3000 then sal*0.1
                        when sal >= 2000 then sal*0.3
                        else 0 end as 보너스
   from emp;
```

문제 100. decode를 중첩해서 아래의 문제를 해결하시오
이름과 부서번호, 직업, 보너스를 출력하는데 부서번호가 20번 이면서 직업이 CLECK인 사원들은 보너스를 9000으로 출력하고 부서번호가 20번 이면서 직업이 ANALYST인 사원들은 보너스를 1200을 출력하고 나머지 사원들은 보너스를 0으로 출력하시오!

```
select ename, deptno, job, decode(deptno, 20, decode(job,'CLECK',  9000,
                                                      'ANALYST',1200,0),0) as 보너스
   from emp;
```

문제 101. 직업이 SALESMAN인 사원들 중에서의 최대월급을 출력하시오!

```
select max(sal) AS 최대월급
   from emp
   where job = 'SALESMAN';
```

문제 102. 통계학과인 학생들 중에서 가장 나이가 많은 학생의 나이를 출력하시오!

```
select max(age)
   from emp2
   where major like '%통계%';
```

문제 103. 부서번호가 20번인 사원들 중에서의 최대월급을 출력하시오!

```
select max(sal)
   from emp
   where deptno =20;

select deptno, max(sal)
   from emp
   where deptno =20
   group by deptno;
```

문제 104. 통신사, 통신사별 최대나이를 출력하시오!

```
select lower(telecom), max(age)
   from emp2
   group by lower(telecom);
```

문제 105. 전공, 전공별 최대나이를 출력하시오!

```
select major, max(age)
   from emp2
   group by major;
```

문제 106. 위의 결과를 다시 출력하는데 전공을 가나다라 순으로 출력하시오

```
select major, max(age)
   from emp2
   group by major
   order by major asc;
```

문제 107. 위의 결과를 다시 출력하는데 통계학과는 제외하고 출력하시오!

```
select major, max(age)
   from emp2
   where major not like '%통계%'
   group by major
   order by major;
```

문제 108. 직업, 직업별 최소월급을 출력하시오!

```
select job, min(sal)
   from emp
   group by job;
```

문제 109. 위의 결과를 다시 출력하는데 최소월급이 높은것부터 출력하시오!

```
select job, min(sal)
   from emp
   group by job
   order by min(sal) desc;
```

문제 110. 위의 결과를 다시 출력하는데 직업이 SALESMAN은 제외하고 출력하시오!

```
select job, min(sal)
   from emp
   where job != 'SALESMAN'
   group by job
   order by min(sal) desc;
```

※ 설명 : 같지 않다 3가지 --> !=, <>, ^=

문제 111. 직업이 SALESMAN 인 사원들 중에서 가장 먼저 입사한 사원의 입사일을 출력하시오!

```
select min(hiredate)
   from emp
   where job = 'SALESMAN';
```

문제 112. 커미션의 평균값을 출력하시오!

```
select avg(comm)
   from emp;
```

문제 113. 위의 문제를 다시 해결하는데 nvl 함수로 null 처리를 해서 4로 나누는게 아니라 14로 나눠지게 하시오!

```
select avg(nvl(comm,0))
   from emp;
```

문제 114. 통신사, 통신사별 평균 나이를 출력하시오!

```
select telecom, round(avg(age))
   from emp2
   group by telecom;
```

문제 115. 위의 결과를 다시 출력하는데 평균나이가 높은것부터 출력하시오!

```
select telecom, round(avg(age))
   from emp2
   group by telecom
   order by avg(age) desc;

select telecom, round(avg(age)) 평균
   from emp2
   group by telecom
   order by 평균  desc;
```

문제 116. 위의 결과를 다시 출력하는데 lg는 제외하고 출력하시오!

```
select telecom, round(avg(age))
   from emp2
   where lower(telecom) != 'lg'
   group by telecom
   order by 2 desc;
```

문제 117. 직업, 직업별 평균월급을 출력하시오!

```
select job, round(avg(sal))
   from emp
   group by job;
```

문제 118. (마지막 문제) 직업별 평균월급 중에서 최소값을 출력하시오!

```
select min(round(avg(sal)))
   from emp
   group by job;
```

문제 119. 직업, 직업별 평균월급을 출력하는데 직업별 평균 월급이 높은것부터 출력하시오!

```
select job, avg(sal)
   from emp
   group by job
   order by 2 desc;
```

문제 120. 위의 결과를 다시 출력하는데 직업별 평균월급이 3000이상인 것만 출력하시오!

```
select job, avg(sal)
   from emp
   where avg(sal) >= 3000
   group by job
   order by 2 desc;

ORA-00934: group function is not allowed here
      ※ group 함수로 검색 조건을 주는것을 where 절에 줄 수 없다.
        group 함수로 검색 조건을 줄 때는 having 절을 사용해야 한다.
      select job, avg(sal)
        from emp
        group by job
        having avg(sal) >= 3000
        order by 2 desc;
```

문제 121. 직업, 직업별 토탈월급을 출력하는데 직업이 'SALESMAN'은 제외하고 출력하고 직업별 토탈월급이 4000 이상인 것만 출력하고 직업별 토탈월급이 높은것부터 출력하는데 직업별 토탈월급을 출력할 때에 천단위를 부여하시오!

```
select job, to_char(sum(sal),'999,999')
   from emp
   where job not like 'SALESMAN'         not like는 악성이 될 수 있다!
   group by job
   having sum(sal) >= 4000
   order by sum(sal) desc;
```

문제 122. 직업이 SALESMAN인 사원들의 인원수를 출력하시오!

```
select count(*)
   from emp
   where job = 'SALESMAN';
```

문제 123. 직업, 직업별 인원수를 출력하시오 !

```
select job, count(*)
   from emp
   group by job;
```

문제 124. 통신사, 통신사별 인원수를 출력하는데 통신사별 인원수가 5명 이상인것만 출력하시오!

```
select lower(telecom), count(*)
   from emp2
   group by lower(telecom)
   having count(*) >= 5;
```

문제 125. 커미션만 카운트 하시오 !

```
select count(comm)
   from emp;                          그룹함수는 null 값을 무시한다.
```

문제 126. 사원 테이블의 직업의 종류는 몇가지가 있는가?

```
select count(distinct job)
   from emp;
```

문제 127. 우리반의 전공의 종류는 몇가지 있는가?

```
select count(distinct major)
   from emp2;
```

문제 128. 입사한 년도 (4자리), 입사한 년도 (4자리) 별 인원수를 출력하시오!

```
select to_char(hiredate,'rrrr'), count(*)
   from emp
   group by to_char(hiredate,'rrrr');
```

문제 129. 통계관련학과 prouping
전공과 전공별 인원수를 출력하는데 통계학과를 정보통계 보험수리학과를 통계학과로 카운트해서 출력하시오(생각해야 할 문제)

```
      select major
         from (select ename, major
                  from emp2
                  where major like '%통계%' )        제대로 풀어보자
         group by major;
```

문제 130. 부서번호, 부서번호별 토탈월급을 출력하시오!

```
select deptno, sum(sal)
   from emp
   group by deptno;
```

문제 131. 부서번호, decode를 이용해서 부서번호가 10번이면 월급이 출력되게 하고 10번이 아니면 null이 출력되게 하시오 !

```
select deptno, decode(deptno, 10, sal)
   from emp;
```

문제 132. 직업, 직업별 토탈월급을 세로로 출력하시오!

```
select job, sum(sal)
   from emp
   group by job;
```

문제 133. 직업, 직업별 토탈월급을 가로로 출력하시오!

```
select sum(decode(job, 'SALESMAN', sal)) as SALESMAN,
       sum(decode(job, 'CLERK', sal)) as CLERK,
       sum(decode(job, 'PRESIDENT', sal)) as PRESIDENT,
       sum(decode(job, 'MANAGER', sal)) as MANAGER,
       sum(decode(job, 'ANALYST', sal)) as ANALYST
   from emp;
```

문제 134. 입사한 년도, 입사한 년도별 인원수를 가로로 출력하시오 !

```
select count(decode(to_char(hiredate,'rrrr'),1980,1)) as "1980",
       count(decode(to_char(hiredate,'rrrr'),1981,1)) as "1981",
       count(decode(to_char(hiredate,'rrrr'),1982,1)) as "1982",
       count(decode(to_char(hiredate,'rrrr'),1983,1)) as "1983"
   from emp;
```

문제 135. 아래와 같이 결과를 출력하시오 ~

```
프랑스어학과    1       0       0
컴퓨터공학과    0       1       0
보건행정학과    0       0       1
통계학과        1       0       3
치기공학과      1       0       0
분자생물학      0       1       0
```

```
       select major,
              count(decode(lower(telecom), 'sk', 1)) as sk,
              count(decode(lower(telecom), 'lg', 1)) as lg,
              count(decode(lower(telecom), 'kt', 1)) as kt
          from emp2
          group by major;
```

문제 136. (생각해야 할 문제) 위의 문제를 다시 해결하는데 정보통계보험수리학과를 통계학과에 포함시켜서
출력되게 하시오

```
select major,
       count(decode(lower(telecom), 'sk', 1)) as sk,
       count(decode(lower(telecom), 'lg', 1)) as lg,
       count(decode(lower(telecom), 'kt', 1)) as kt
   from (select decode(major,'정보통계보험수리학과', '통계학과',major) as major, telecom
            from emp2)
   group by major;
```

문제 137. 직업, 직업별 최대월급, 직업별 최소월급, 직업별 평균월급을 출력하시오!

```
select job, max(sal), min(sal), avg(sal)
   from emp
   group by job;
```

문제 138. 통신사, 통신사별 최대나이, 통신사별 최소나이, 통신사별 평균나이, 통신사별 인원수를 출력하시오!

```
select lower(telecom), max(age), min(age), avg(age), count(*)
   from emp2
   group by lower(telecom);
```

문제 139. 직업, 직업별 토탈월급을 출력하시오.(세로)

```
select job, sum(sal)
   from emp
   group by job;

SALESMAN        5600
CLERK                   4150
PRESIDENT       5000
MANAGER 8275
ANALYST 6000
                          <-------- 전체 토탈 월급 !
```

문제 140. 아래와 같이 직업, 직업별 토탈월급을 출력하는데 전체 토탈월급이 맨 아래에 출력되게 하시오!

```
select job, sum(sal)
   from emp
   group by rollup(job);
```

문제 141. 아래의 결과를 출력하는데 맨 아래쪽에 전체 인원수가 나오게 출력하시오!

```
                sk      lg      kt
프랑스어학과    1       0       0
컴퓨터공학과    0       1       0
보건행정학과    0       0       1
통계학과        1       0       3
치기공학과      1       0       0
분자생물학      0       1       0
select nvl(major,'전체인원수'),
       count(decode(lower(telecom), 'sk', 1)) as sk,
       count(decode(lower(telecom), 'lg', 1)) as lg,
       count(decode(lower(telecom), 'kt', 1)) as kt
   from emp2
   group by rollup(major);
```

문제 142. 직업, 직업별 토탈월급을 출력하는데 전체 토탈월급이 이번에는 맨위에 출력되게 하시오!

```
select nvl(job,'전체토탈월급'), sum(sal)
   from emp
   group by cube(job);
```

문제 143. 아래와 같이 결과를 출력하시오!

```
                sk      lg      kt
전체인원수      13      5       9
프랑스어학과    1       0       0
컴퓨터공학과    0       1       0
보건행정학과    0       0       1
통계학과        1       0       3
치기공학과      1       0       0
분자생물학      0       1       0
select nvl(major,'전체인원수'),
       count(decode(lower(telecom), 'sk', 1)) as sk,
       count(decode(lower(telecom), 'lg', 1)) as lg,
       count(decode(lower(telecom), 'kt', 1)) as kt
   from emp2
   group by cube(major);
```

문제 144. 나이, 통신사와 인원수를 아래와 같이 출력하시오!

	Sk	lg	kt
25	1	0	1
30	1	0	0
28	2	1	0
26	3	1	3
31	2	0	0
24	0	0	1

```
select age,
       count(decode(lower(telecom), 'sk', 1)) as sk,
       count(decode(lower(telecom), 'lg', 1)) as lg,
       count(decode(lower(telecom), 'kt', 1)) as kt
   from emp2
   group by age;
```

문제 145. 위의 결과를 아래와 같이 출력하시오!

	23	24	25	26	27	28	30	31	32	33	40
sk	0	0	1	3	3	2	1	2	0	1	0
lg	0	0	0	1	1	1	0	0	1	0	1
kt	1	1	1	3	3	0	0	0	0	0	0

문제 146. 아래의 결과를 pivot문으로 구현하시오!

	23	24	25	26	27	28	30	31	32	33	40
sk	0	0	1	3	3	2	1	2	0	1	0
lg	0	0	0	1	1	1	0	0	1	0	1
kt	1	1	1	3	3	0	0	0	0	0	0

```
select *
   from (select lower(telecom), age from emp2)
   pivot (count(*) for age in (23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 40));
```

문제 147. 살인을 유발시키는 가장 큰 원인이 무엇인가?

```
select *
   from crime_cause2
   where crime_type = '살인'
   order by cnt desc;
```

문제 148. 범죄유형, 범죄유형별 총 건수 (토탈값)을 출력하는데 범죄유형별 총 건수가 높은것부터 출력하시오!

```
select crime_type, sum(cnt)
   from crime_cause2
   group by crime_type
   order by 2 desc;
```

문제 149. 살인이 일어나는 장소와 건수를 출력하는데 건수가 높은것부터 출력하시오 !

```
select c_loc, cnt
   from crime_loc2
   where crime_type = '살인'
   order by cnt desc;
```

문제 150. 아파트에서 일어나는 범죄유형과 그 건수를 출력하는데 그 건수가 높은것부터 출력하시오 !

```
select crime_type, cnt
   from crime_loc2
   where c_loc = '아파트'
   order by cnt desc;
```

문제 151. 대학이름, 등록금을 출력하는데 등록금이 높은 순서대로 출력하시오 !

```
select university, college_fee
   from univ
   order by college_fee desc;
```

문제 152. 아래와 같이 결과를 출력하시오(마지막문제, 댓글로 달아주세요)

	10	20	30	토탈값
ANALYST	0	6,000	0	6,000
CLERK	1,300	1,900	950	4,150
MANAGER	2,450	2,975	2,850	8,275
PRESIDENT	5,000	0	0	5,000
SALESMAN	0	0	5,600	5,600
토탈값	8,750	10,875	9,400	29,025

```
select nvl(job,'토탈값'), to_char(sum(decode(deptno, 10, sal,0)), '999,999') as "10",
                          to_char(sum(decode(deptno, 20, sal,0)), '999,999') as "20",
                          to_char(sum(decode(deptno, 30, sal,0)), '999,999') as "30",
                          to_char(sum(sal), '999,999') as 토탈값
   from emp
   group by rollup(job);
```

문제 153. 이름, 나이, 순위를 출력하는데 순위가 나이가 높은 순서대로 순위를 부여하시오 !

```
select ename, age, rank() over ( order by age desc ) 순위
   from emp2;
```

문제 154. 위의 결과를 다시 출력하는데 전공이 심리학과 학생들로 제한해서 출력하시오 !

```
select ename, age, rank() over ( order by age desc ) 순위
   from emp2
   where major = '심리학과';
```

문제 155. 1981년도에 입사한 사원들의 이름과 입사일과 순위를 출력하는데 먼저 입사한 사원순으로 순위를 부여하시오 !

튜닝전 :

```
select ename, hiredate, rank() over ( order by hiredate asc ) 순위
   from emp
   where to_char( hiredate, 'rrrr') = '1981';
```

튜닝후 :

```
select ename, hiredate, rank() over ( order by hiredate asc ) 순위
   from emp
   where hiredate between to_date('81/01/01', 'rr/mm/dd')
                      and to_date('81/12/31', 'rr/mm/dd');
```

※ 도스든 sqlgate든 날짜를 인식하게 하려면 to_date를 꼭 쓰자!

문제 156.이름과 월급, 순위를 출력하는데 순위가 월급이 높은 사원부터 출력하시오 !
(순위가 같은 순위가 있어도 다음 순위로 바로 이어지게 하시오 !)

```
select ename, sal, rank() over ( order by sal desc ) 순위
   from emp;
        KING    5000    1
        SCOTT   3000    2
        FORD    3000    2
        JONES   2975    4       <------- 3등으로 나오게 해볼까?
select ename, sal, dense_rank() over ( order by sal desc) 순위
   from emp;
```

문제 157. 월급이 2975인 사원은 전체 사원들중에 월급의 순위가 어떻게 되는가?

```
select rank(2975) within group ( order by sal desc ) 순위
   from emp;
```

문제 158. 81년 11월 17일에 입사한 사원은 사원 테이블에서 몇번째로 입사한 사원인가?

```
select rank(to_date('81/11/17', 'rr/mm/dd')) within group ( order by hiredate desc) 순위
   from emp;
```

문제 159. 부서번호, 이름 , 월급, 순위를 출력하는데 순위를 출력할 때 부서번호별로 각각 월급이 높은 순서대로 순위를 출력하시오 !

```
select deptno, ename, sal, dense_rank() over (partition by deptno
                                                order by sal desc ) 순위
   from emp;
```

※ partition은 group by 랑은 다른것이다 분석함수 쓸 때 쓰는 옵션 키워드

문제 160. 통신사, 이름, 나이, 순위를 출력하는데 순위가 통신사별로 각각 나이가 높은 순서대로 순위를 부여하시오!

```
select lower(telecom), ename, age, dense_rank() over ( partition by lower(telecom)
                                                        order by age desc) 순위
   from emp2;
```

문제 161. 입사한 년도, 입사한 년도별로 입사한 사원들의 이름을 가로로 출력하시오 !

```
select to_char( hiredate, 'rrrr' ), listagg(ename, ', ') within group (order by ename)
   from emp
   group by to_char( hiredate, 'rrrr' );
```

문제 162. 통신사, 학생이름을 출력하는데 아래와 같이 나이도 같이 출력되게 하고 나이가 높은 순서대로 출력되게 하시오!

kt 김준구(27), 안우용(27), 이상엽(27), 김건휘(26), 이서영(26), 이소진(26), 주소현(25), 신선혜(24), 김혜진(23)
lg 허석우(40), 김용식(32), 정지엽(28), 최재혁(27), 정성호(26)
sk 김진철(33), 김용원(31), 오세희(31), 안혜진(30), 박태균(28), 장보겸(28), 김준하(27), 엄한솔(27), 이후림(27), 서일(26), 임혜진(26), 최원형(26), 유이수(25)

```
select lower(telecom), listagg(ename||'('||age||')',', ') within group ( order by age desc)
   from emp2
   group by lower(telecom);
```

문제 163. 위의 결과를 다시출력 하는데 등급이 1등급인 사원들만 출력하시오!

```
select *
   from (select ename, sal, ntile(4) over ( order by sal desc) 등급
   from emp)
   where 등급 = 1 ;
```

문제 164. 통신사, 이름, 나이, 순위(나이)를 출력하는데 통신사별로 순위가 1등인 학생들만 출력하시오!

```
select * from
(select lower(telecom), ename, age, dense_rank() over (partition by lower(telecom) order by age desc) 순위
   from emp2)
where 순위 = 1 ;
```

문제 165. 통신사, 이름, 나이, 순위를 출력하는데 순위가 통신사별로 각각 나이가 높은 순서대로 출력하시오!

```
select lower(telecom), ename, age, dense_rank() over (partition by lower(telecom) order by age desc ) 순위
from emp2;
```

문제 166. 아래와 같이 결과를 출력하시오 ! (점심시간 문제)
kt 김준구(2), 김건휘(3), 안우용(3), 이상엽(3), 이서영(3), 김혜진(4), 신선혜(4),
lg 김용식(1), 허석우(1), 정지엽(2), 최재혁(2), 정성호(3)
sk 김용원(1), 김진철(1), 박태균(1), 안혜진(1), 오세희(1), 김준하(2), 엄한솔(2),

```
select lower_telecom, listagg (ename||'('||등급||')', ', ')
                        within group ( order by 등급 asc)
   from ( select lower(telecom) as lower_telecom, ename,
                ntile(4) over ( order by age desc) 등급
           from emp2)
   group by lower_telecom;
```

문제 167. 토탈월급을 출력하시오!

```
select sum(sal)
   from emp;
```

문제 168. 이름, 월급, 사원테이블 전체의 토탈월급을 출력하시오!

```
select ename, sal, sum(sal) over () 전체토탈
   from emp;
```

문제 169. 이름, 월급, 사원 테이블의 월급의 평균값을 출력하시오 !

```
select ename, sal, round(avg(sal) over ()) 평균월급
   from emp;
```

문제 170. 이름, 월급, 사원테이블의 월급의 평균값을 출력하는데 자기의 월급이 사원 테이블의 월급의 평균값보다 더 큰 사원들만 출력하시오 !

```
select *
   from  (select ename, sal, round(avg(sal) over ()) 평균월급
   from emp)
where sal >= 평균월급;
```

문제 171. 부서번호, 부서번호별 평균월급을 출력하시오 !

```
select deptno, round(avg(sal))
   from emp
   group by deptno;
```

문제 172. 부서번호, 이름, 월급, 자기가 속한 부서번호의 평균월급을 출력하시오 !

```
select deptno, ename, sal,
       round(avg(sal) over (partition by deptno)) 부서평균
   from emp;
```

문제 173. 위의 결과를 다시 출력하는데 자기의 월급이 자기 부서의 평균월급보다 높은 사원들만 출력하는데 그들의 직업이 무엇인지 확인하시오 !

```
select *
   from (select deptno, ename, sal, job,
       round(avg(sal) over (partition by deptno)) 부서평균
   from emp)
   where sal >= 부서평균;
```

문제 174. 이름, 직업, 자기가 속한 직업의 인원수를 출력하시오 !

```
select ename, job, count(*) over ( partition by job) 인원수
   from emp;
```

문제 175. 사원번호, 이름, 월급, 월급의 누적치를 출력하시오 !

```
select deptno, ename, sal, sum(sal) over (order by empno rows
                                          between unbounded preceding
                                          and current row) 누적치
   from emp;
```

문제 176. 위의 결과를 다시 출력하는데 부서번호, 사원번호, 이름, 월급, 월급의 누적치를 출력하는데
월급의 누적치가 부서번호 별 각각 월급의 누적치를 출력하시오!.

```
select deptno, ename, sal, sum(sal) over (partition by deptno order by empno rows
 between unbounded preceding and current row) 누적
  from emp;
```

문제 177. 위의 결과를 다시 출력하는데 부서번호, 이름, 월급, 월급의 그 전행, 월급의 그 다음행을
출력하는데 부서번호로 각각 나오게 하시오!

```
select deptno, ename, sal, lead(sal, 1) over (partition by deptno order by empno) as
        lead,
                              lag(sal, 1)  over (partition by deptno order by empno) as lag
   from emp;
```

문제 178. 최대 월급을 받는 사원의 이름과 월급을 출력하시오.

```
select ename, max_sal
   from(select ename, sal, max(sal) over () max_sal
   from emp)
   where sal >= max_sal;
```

문제 179. 이름, 부서위치를 출력하시오 !

```
select ename, loc
   from emp, dept
   where 이름과 부서위치를 출력할 수 있을 만큼 emp와 dept가 서로 어떤 연결고리가
         있다는 증거를 보여줘야 한다.

select ename, loc
   from emp, dept
   where emp.deptno = dept.deptno;     <-----강한 연결고리
```

문제 180. 위의 결과를 다시 출력하는데 부서위치가 DALLAS 인 사원들만 출력하시오 !

```
select ename, loc
   from emp, dept
   where emp.deptno = dept.deptno
     and dept.loc = 'DALLAS';
```

문제 181. 직업이 SALESMAN인 사원들의 이름과 직업과 부서위치를 출력하시오 !

```
select ename, job, loc
   from emp, dept
   where emp.deptno = dept.deptno
     and job = 'SALESMAN';
```

문제 182. 월급이 1000에서 3000사이인 사원들의 이름과 월급과 부서명과 직업을 출력하시오 !

```
select ename, sal, dname, job
   from emp, dept
   where emp.deptno = dept.deptno
     and sal between 1000 and 3000;
```

문제 183. 위의 결과에서 deptno도 같이 출력해 보시오!

```
select ename, sal, dname, job, emp.deptno
   from emp, dept
   where emp.deptno = dept.deptno
     and sal between 1000 and 3000;

select emp.ename, emp.sal, dept.dname, emp.job, emp.deptno
   from emp, dept
   where emp.deptno = dept.deptno
     and emp.sal between 1000 and 3000;

select e.ename, e.sal, d.dname, e.job, e.deptno
   from emp e, dept d                          <----------------------테이블 별칭
   where e.deptno = d.deptno
     and e.sal between 1000 and 3000;
```

문제 185. 이름이 KING인 사원의 이름과 월급과 직업과 부서위치를 출력하시오!

```
select e.ename, e.sal, d.loc
   from emp e, dept d
   where e.deptno = d.deptno
     and e.ename = 'KING';
```

문제 186. 부서위치, 부서위치별 토탈월급을 출력하시오!

```
select d.loc, sum(e.sal)
   from emp e, dept d
   where e.deptno = d.deptno
   group by d.loc;
```

문제 187. 위의 결과를 다시 출력하는데 토탈 월급이 9000 이상인 것만 출력하시오 !

```
select *
   from(select d.loc, sum(e.sal) sumsal
   from emp e, dept d
   where e.deptno = d.deptno
   group by d.loc)
   where sumsal >= 9000;
```

문제 188. 직업이 SALESMAN이고 월급이 1000 이상인 사원들의 이름과 월급과 직업과 부서위치를 출력하시오 !

```
select e.ename, e.sal, d.loc
   from emp e, dept d
   where e.deptno = d.deptno
     and e.job = 'SALESMAN'
     and e.sal >=1000;
```

문제 189. 부서위치, 이름, 월급, 월급에 대한 순위를 출력하시오 ! (월급에 대한 순위는 월급이 높은 순서)

```
select d.loc, e.ename, e.sal, dense_rank() over (order by e.sal desc)
   from emp e, dept d
   where e.deptno = d.deptno
```

문제 190. 위의 결과를 다시 출력하는데 순위를 부서위치별로 각각 출력되게 하시오 !

```
select d.loc, e.ename, e.sal, dense_rank() over (partition by d.loc
                                                  order by e.sal desc)
   from emp e, dept d
   where e.deptno = d.deptno;
```

문제 191. 부서위치, 부서위치별로 속한 사원들의 이름을 가로로 출력하시오!

```
CHICAGO         ALLEN, BLAKE, JAMES, MARTIN, TURNER, WARD
DALLAS                ADAMS, FORD, JONES, SCOTT, SMITH
NEW YORK        CLARK, KING, MILLER
```

```
select d.loc, listagg (e.ename, ', ') within group (order by e.ename)
   from emp e, dept d
   where e.deptno = d.deptno
   group by d.loc;
```

문제 192. 부서위치, 부서위치별 토탈월급을 출력하는데 전체 토탈월급이 맨 아래쪽에 출력되게 하시오!

```
select nvl(d.loc,'토탈월급 :'), sum(e.sal) 토탈월급
   from emp e, dept d
   where e.deptno = d.deptno
   group by rollup(d.loc);
```

문제 193. 그럼 위의 문제를 다시 아래와 같이 결과가 출력되게 하시오 !

```
토탈월급 :      29025
DALLAS          10875
CHICAGO         9400
NEW YORK        8750
```

```
select nvl(d.loc,'토탈월급 :'), sum(e.sal) 토탈월급
   from emp e, dept d
   where e.deptno = d.deptno
   group by cube(d.loc);
```

문제 194. 부서위치, 부서위치별 토탈월급, 부서위치별 최대 월급, 부서위치별 평균월급, 부서위치별 인원수를 출력하시오 !

```
select d.loc, sum(e.sal), max(e.sal), round(avg(e.sal)), count(*)
   from emp e, dept d
   where e.deptno = d.deptno
   group by d.loc;
```

문제 195. (오늘의 마지막 문제) 아래와 같이 결과를 출력하시오! (토탈월급)
(카페 댓글로 올려~)

```
NEW YORK            DALLAS          CHICAGO
-------------           ----------          -----------
    8750                10875               9400
```

```
답 1
select sum( decode (d.loc, 'NEW YORK' , e.sal)) "NEW YORK",
       sum( decode (d.loc, 'CHICAGO'  , e.sal)) "CHICAGO",
       sum( decode (d.loc, 'DALLAS'   , e.sal)) "DALLAS"
   from emp e, dept d
   where e.deptno = d.deptno;
답 2
select *
   from (select d.loc dloc, e.sal esal
   from emp e, dept d
   where e.deptno = d.deptno)
   pivot(sum(esal) for dloc IN ('NEW YORK', 'CHICAGO', 'DALLAS')) ;
```

문제 196. (복습) DALLAS에서 근무하는 사원들의 이름과 월급과 부서위치를 출력하는데 월급이 높은 사원부터 출력하시오!

```
select e.ename, e.sal, d.loc
   from emp e, dept d
   where e.deptno = d.deptno
     and d.loc = 'DALLAS'
   order by sal desc;
```

문제 197. 부서테이블 전체를 조회하시오 !

```
select * from dept;
```

문제 198. 이름과 부서위치를 출력하는데 출력안된 부서위치가 무엇인지 확인하시오!

```
select e.ename, d.loc
   from emp e, dept d
   where e.deptno = d.deptno
   order by d.loc;
```

※ 설명 : emp 테이블에 40번 부서번호인 사원이 없음, 그래서 BOSTON 이 출력이 안됨

문제 199. 이름과 부서위치를 출력하는데 출력안된 부서위치도 같이 출력하시오 !

```
select e.ename, d.loc
   from emp e, dept d
   where e.deptno (+) = d.deptno
   order by d.loc;
```

※ (+) : outer join sign 모자란 쪽에 쓰면 플러스해서 양변을 맞춰주는 느낌(컬럼 오른쪽에 붙여야 함!

```
====================================
insert into emp( empno, ename, deptno)
  values(9293,'JANE', 70 ) ;

commit;
====================================
```

문제 200. 이름과 부서위치를 출력하는데 사원 테이블에는 존재하는데 부서 테이블에는 존재하지 않는 데이터도 같이 출력하시오 !

```
select e.ename, d.loc
   from emp e, dept d
   where e.deptno = d.deptno (+);
```

문제 201. 부서위치, 부서위치별 토탈 월급을 출력하시오 ! BOSTON도 출력하고 토탈월급이 높은 것부터 출력하시오!

```
select d.loc, sum(e.sal)
   from emp e, dept d
   where e.deptno (+) = d.deptno
   group by d.loc
   order by sum(e.sal) desc nulls last;
```

문제 202. 위의 결과를 다시 출력하는데 토탈월급이 낮은것부터 출력하시오 !

```
select d.loc, sum(e.sal)
   from emp e, dept d
   where e.deptno (+) = d.deptno
   group by d.loc
   order by sum(e.sal) asc nulls first;
```

문제 203. 이름, 월급, 등급(grade) 을 출력하시오 !(emp, salgrade를 조인해서 출력)

```
select e.ename, e.sal, s.grade
   from emp e, salgrade s
   where e.sal between s.losal and s.hisal;
```

문제 204. 등급(grade), 등급(grade) 별로 해당하는 사원들의 이름을 가로로 출력하시오!

```
1       ADAMS, JAMES, SMITH
2       MARTIN, MILLER, WARD
3       ALLEN, TURNER
4       BLAKE, CLARK, FORD, JONES, SCOTT
5       KING
select s.grade, listagg(e.ename, ', ') within group (order by e.ename)
   from emp e, salgrade s
   where e.sal between s.losal and s.hisal
   group by s.grade;
```

문제 205. 이름, 월급 급여등급(grade), 부서위치를 출력하시오 !

```
select e.ename, e.sal, s.grade, d.loc
   from emp e, salgrade s, dept d
   where e.deptno = d.deptno                          -- 연결고리
     and e.sal between s.losal and s.hisal;      -- 연결고리
                ※ 테이블이 3개면 연결고리는 두개 써야한다!
```

문제 206. 위의 결과에서 부서위치가 CHICAGO인 사원들만 출력하시오 !

```
select e.ename, e.sal, s.grade, d.loc
   from emp e, salgrade s, dept d
   where e.deptno = d.deptno
     and e.sal between s.losal and s.hisal
     and d.loc = 'CHICAGO';                                    <-----검색조건
```

문제 207.(난이도 업) 아래와 같이 결과를 출력하시오 !

```
Grade    name
1       ADAMS(DALLAS), JAMES(CHICAGO), SMITH(DALLAS)
2       MARTIN(CHICAGO), MILLER(NEW YORK), WARD(CHICAGO)
3       ALLEN(CHICAGO), TURNER(CHICAGO)
4       BLAKE(CHICAGO), CLARK(NEW YORK), FORD(DALLAS), JONES(DALLAS), SCOTT(DALLAS)
5       KING(NEW YORK)
select s.grade, listagg(e.ename||'('||d.loc||')', ', ') within group (order by e.ename)
   from emp e, salgrade s, dept d
   where e.sal between s.losal and s.hisal
     and e.deptno = d.deptno
   group by s.grade;
```

문제 208. 사원이름, 관리자의 이름(직속상사) 를 출력하시오 !

```
select 사원.ename 사원, 관리자.ename 상사
   from emp 사원, emp 관리자
   where 사원.mgr = 관리자.empno;
```

문제 209. (점심시간 문제) 아래와 같이 결과를 출력하시오 !

```
KING(5000)      BLAKE(2850), CLARK(2450), JONES(2975)
FORD(3000)      SMITH(800)
SCOTT(3000)     ADAMS(1100)
JONES(2975)     FORD(3000), SCOTT(3000)
BLAKE(2850)     ALLEN(1600), JAMES(950), MARTIN(1250), TURNER(1500), WARD(1250)
CLARK(2450)     MILLER(1300)
```

```
select m.ename||'('|| m.sal ||')' 관리자 ,
       listagg(e.ename||'('||e.sal||')', ', ')
       within group(order by e.ename) 사원
   from emp e, emp m
   where e.mgr = m.empno
   group by m.ename, m.sal
   order by m.sal desc;
```

문제 211. 이름과 부서위치를 출력하는데 outer join을 사용해서 작성을 하고 (+) <-- outer join 사인을 양쪽에 둘다 작성해서 수행해 보시오 !

```
select e.ename, d.loc
   from emp e, dept d
   where e.deptno (+) = d.deptno (+);
```

ORA-01468: a predicate may reference only one outer-joined table

Full outer join을 사용해야한다 !! (1999 ansi 문법이다)

Ansi : american national standards institute

Oracle, mssql, mysql, postgre sql, maria 유

어느 데이터베이스에서든 다 사용할 수 있다

-----> full outer join 문법

```
select e.ename, d.loc
   from emp e full outer join dept d
   on (e.deptno = d.deptno);
```

문제 212. 이름과 부서위치를 출력하는데 on절을 사용한 조인문법으로 작성하시오 !

```
select e.ename, d.loc
   from emp e join dept d
   on( e.deptno = d.deptno );
```

문제 213. 위의 결과에서 DALLAS에서 근무하는 사원들만 출력하시오!

```
select e.ename, d.loc
   from emp e join dept d
   on ( e.deptno = d.deptno)     ------- 조인 연결 조건
   where d.loc = 'DALLAS';         -------- 검색 조건
```

※ on 절의 역할과 WHERE 절의 역할이 다르다

WHERE 대신 AND를 써도 되는데 왠만하면 WHERE 써라

문제 214. 월급이 3000 이상인 사원들의 이름과 월급과 직업과 부서명을 출력하시오 !

```
select e.ename, e.sal, e.job, d.dname
   from emp e join dept d
   on (e.deptno = d.deptno)
   where e.sal >= 3000;
```

문제 215. 이름, 월급, 등급(grade)를 출력하는데 grade가 3등급인 사원들만 출력하시오 !(emp와 salgrade 조인)

```
select e.ename, e.sal, s.grade
   from emp e join salgrade s
   on (e.sal between s.losal and s.hisal)
   where s.grade = 3;
```

문제 216. 이름, 월급, 부서위치, 급여등급(grade)을 출력하시오!
(dept----------emp----------salgrade)

```
select e.ename, e.sal, d.loc, s.grade
   from emp e join dept d     on(e.deptno = d.deptno)
              join salgrade s on (e.sal between s.losal and s.hisal);
```

※ 설명 : 위의 가운데 emp 테이블을 가장 먼저 작성해야 한다.

문제 217. 아래의 오라클 조인 문법을 1999 ansi 문법으로 변경하시오 !

```
select e.ename, d.loc
   from emp e, dept d
   where e.deptno (+) = d.deptno;          ------오라클 조인 문법

select e.ename, d.loc
   from emp e right outer join dept d     -------1999 ansi 문법
   on(e.deptno = d.deptno);
```

문제 218. 이름과 부서위치를 출력하는데 using절을 사용한 조인문법으로 수행하시오 !

```
select e.ename, d.loc
   from emp e join dept d
   using(deptno);                         -------주의사항 ! Using절에 테이블 별칭을
                                                  사용해서는 안된다.
```

문제 219. 이름과 부서위치를 출력하는데 naturl join으로 수행하시오 !

```
select e.ename, d.loc
   from emp e natural join dept d;
```

문제 220. 아래의 cross join을 오라클 조인문법으로 수행하시오 !

```
select e.ename, d.loc
   from emp e cross join dept d;

select e.ename, d.loc
   from emp e, dept d;
```

문제 221. 부서위치, 부서위치별 토탈월급을 출력하는데 부서위치가 boston도 출력되게 하고
부서위치가 chicago는 제외하고 토탈월급이 4000이상인 것만 출력하고, 토탈월급이 높은것 부터
출력하는데 null을 맨뒤에 출력되게 하시오 !

```
select d.loc, sum(e.sal)
   from emp e right outer join dept d
   using(deptno)
   where d.loc != 'CHICAGO'
   group by d.loc
   having sum(e.sal) >= 4000
   order by sum(e.sal) desc nulls last;
```

문제 222. 부서위치, 부서위치별 인원수를 출력하는데 부서위치별 인원수가 3명 이상인 것만 출력하시오 !

```
select d.loc, count(*)
   from emp e join dept d
   using(deptno)
   group by d.loc
   having count(*) >=3;
```

문제 223. 부서위치, 부서위치별 인원수를 출력하는데 아래와 같이 가로로 출력하시오 !

```
뉴욕   달라스  시카고   보스톤
3      5      6      0
select *
   from (select d.loc
   from emp e, dept d
   where e.deptno = d.deptno)
   pivot(count(*) for loc in ('NEW YORK', 'DALLAS', 'CHICAGO', 'BOSTON'));
```

문제 224. 위의 결과를 아래와 같이 출력하시오!

```
SALESMAN        0       0       4       0
CLERK           1       2       1       0
PRESIDENT       1       0       0       0
MANAGER 1       1       1       0
ANALYST         0       2       0       0
select *
   from (select d.loc, e.job
   from emp e, dept d
   where e.deptno = d.deptno)
   pivot(count(*) for loc in ('NEW YORK', 'DALLAS', 'CHICAGO', 'BOSTON'));
```

문제 225. 위의 결과를 아래와 같이 출력하시오 !

```
ANALYST         0       2       0       0
CLERK           1       2       1       0
MANAGER 1       1       1       0
PRESIDENT       1       0       0       0
SALESMAN        0       0       4       0
토탈값          3       5       6       0
select nvl(job,'토탈값'),
       count( decode ( d.loc , 'NEW YORK', 1, null)) as "NEW YORK",
       count( decode ( d.loc , 'DALLAS'  , 1, null)) as "DALLAS",
       count( decode ( d.loc , 'CHICAGO' , 1, null)) as "CHICAGO",
       count( decode ( d.loc , 'BOSTON'  , 1, null)) as "BOSTON"
   from emp e, dept d
   where e.deptno = d.deptno
   group by rollup(job);
```

문제 226. 아래의 통신사 테이블을 생성하시오 !

```
create table telecom_price
  ( telecom_id   number(10),
    telecom_name varchar2(20),
    month_price  number(10) );

insert into telecom_price values( 1, 'sk', 56000);
insert into telecom_price values( 2, 'lg', 54000);
insert into telecom_price values( 3, 'kt', 52000);
insert into telecom_price values( 4, 'cj hello', 50000);


commit;

select * from telecom_price;
```

문제 227. 이름, 나이, 통신사, 월정액(month_price)을 출력하시오 !

```
select e.ename, e.age, e.telecom, t.month_price
   from emp2 e, telecom_price t
   where lower(e.telecom) = t.telecom_name;
```

문제 228. JONES의 월급을 출력하시오 !

```
select ename, sal
   from emp
   where ename = 'JONES';
```

문제 229. JONES의 월급보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하시오 !

```
select ename, sal
   from emp                                          <<----------- main query
   where sal > (select sal
                from emp                             <<-----------subquery
                where ename = 'JONES');
                     ※ 문장이 실행되는 것은 subquery부터 실행한다.
```

문제 230. SCOTT과 같은 월급을 받는 사원들의 이름과 월급을 출력하시오 !

```
select ename, sal
   from emp
   where sal = ( select sal
                from emp
                where ename = 'SCOTT')
     and ename != 'SCOTT';
```

문제 231. 최대월급을 받는 사원의 이름과 월급을 출력하시오 !

```
select ename, sal
   from emp
   where sal >= (select max(sal)
                from emp);
```

문제 232. 서울시 물가 데이터 중 가장 가격이 높은 생필품의 이름과 가격과 파는 매장을 출력하시오 !

```
select a_name, a_price, m_name
   from price
   where a_price = ( select max(a_price) from price);
                        ※ select table_name from user_tables; 현재 생성된 테이블 현황
```

문제 233. 가정불화로 생기는 범죄중에 가장 많이 발생하는 범죄가 무엇인가?(crime_cause2)

```
select crime_type, cnt
   from crime_cause2
   where cnt = ( select max(cnt) from crime_cause2 where term = '가정불화');
```

문제 234. 병원에서 많이 발생하는 범죄유형이 무엇인지 출력하시오 ! (crime_loc2 사용)

```
select crime_type, c_loc, cnt
   from crime_loc2
   where cnt = (select max(cnt) from crime_loc2 where c_loc = '병원')
     and c_loc = '병원';
```

문제 235. 30번 부서번호에서 가장 많은 월급을 받는 사원의 이름과 그 월급을 출력하시오 !

```
select ename, sal
   from emp
   where deptno = 30
     and sal = (select max(sal) from emp where deptno = 30);
```

문제 236. car_accident.csv를 오라클 데이터베이스에 입력하시오 !

SQLgate-->도구-->데이터가져오기-->scott 유저-->car_accident-->시작--> 반드시 자동 열매핑

```
create table car_accident
  ( a_year  number(10),
    a_loc   varchar2(20),
    a_loc2  varchar2(100),
    a_cnt   number(10),
    a_type  varchar2(20) );
```

문제 237. (오늘의 마지막 문제) 서울시에서 교통사고가 가장 많이 일어나는 지역이 어디인지 알아내시오 !

```
SELECT a_loc2, a_cnt, a_loc
   FROM CAR_ACCIDENT
   WHERE a_cnt = (SELECT max(a_cnt)
                  FROM CAR_ACCIDENT)
     AND a_loc = '서울';
```

문제 238. SMITH 보다 높은 월급을 받는 사원들의 이름과 월급을 출력하는데 월급이 높은 사원부터 출력하시오!

```
SELECT ename, sal
  FROM Emp
  WHERE sal >= (SELECT sal FROM EMP WHERE ename ='SMITH')
  ORDER BY sal DESC;
```

문제 239. KING에게 보고하는 사원들의 이름을 출력하시오 ! (KING의 직속 부하사원들)

```
SELECT ename
  FROM EMP
  WHERE mgr = (SELECT empno
                FROM EMP
                WHERE ename = 'KING');

SELECT e.ename
  FROM EMP e, EMP m
  WHERE e.mgr = m.empno
    AND m.ENAME = 'KING';
```

문제 240. DALLAS에 있는 부서번호가 무엇인지 출력하시오!

```
SELECT DEPTNO
  FROM DEPT
  WHERE LOC = 'DALLAS';
```

문제 241. DALLAS에 있는 부서번호에서 근무하는 사원들의 이름과 월급을 출력하시오!

```
SELECT ename, sal
  FROM EMP
  WHERE deptno = (SELECT DEPTNO FROM DEPT WHERE loc = 'DALLAS');
                ※ SUBQUERY의 테이블과 메인쿼리의 테이블이 달라도 상관 없다
```

문제 242. 직업이 SALESMAN인 사원들과 월급이 같은 사원들의 이름과 월급을 출력하시오 !

```
SELECT ename, sal
  FROM EMP
  WHERE sal = (SELECT sal FROM EMP WHERE job = 'SALESMAN');
                ※ row가 4개나 되서 오류남
SELECT ename, sal
  FROM EMP
  WHERE sal in (SELECT sal FROM EMP WHERE job = 'SALESMAN');
                ※ 이게 multiple row subquery
```

문제 243. 월급이 1000에서 2000 사이인 사원들과 같은 직업을 갖는 사원들의 이름과 직업과 월급을 출력하시오 !

```
SELECT ename, job, sal
  FROM EMP
  WHERE job IN (SELECT job FROM EMP WHERE sal BETWEEN 1000 AND 3000);
```

문제 244. 직업이 SALESMAN인 사원들과 월급이 같지 않은 사원들의 이름과 월급을 출력하시오!

```
SELECT ename, sal
  FROM EMP
  WHERE sal NOT IN (SELECT sal FROM EMP WHERE job = 'SALESMAN');
```

문제 245. 관리자인 사원들의 이름을 출력하시오 !
(자기 밑에 직속부하 한명이라도 있는 사원들)

```
SELECT ename
  FROM EMP
  WHERE empno IN (SELECT mgr FROM EMP);
```

문제 246. 관리자가 아닌 사원들의 이름을 출력하시오 !

```
SELECT ename
  FROM EMP
  WHERE empno NOT IN (SELECT nvl(mgr,0) FROM EMP);
```

문제 247. 커미션이 null 인 사원들과 같은 직업을 갖는 사원들의 이름과 직업을 출력하시오 !

```
SELECT ename, job
  FROM EMP
  WHERE job IN (SELECT job FROM EMP WHERE comm IS null);
```

문제 248. 직업이 ANALYST 인 사원들의 커미션과 같은 커미션을 받는 사원들의 이름과 커미션을 출력하시오!

```
SELECT ENAME, COMM
  FROM EMP
  WHERE NVL(COMM, -1) IN ( SELECT NVL(COMM, -1) FROM EMP WHERE JOB = 'ANALYST');
```

문제 249. 직업이 SALESMAN인 사원들의 월급중에서 가장 많은 월급보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하시오 !

```
SELECT ename, sal
  FROM EMP
  WHERE sal > (SELECT max(sal) FROM EMP WHERE job = 'SALESMAN');
                ※ 보통 이렇게 함

SELECT ename, sal
  FROM EMP
  WHERE sal >ALL (SELECT max(sal) FROM EMP WHERE job = 'SALESMAN');
                ※ >all 활용
                   >all : 이 모든 값보다 큰 월급을 찾아라 (해석 : 멀티를 싱글로 ??)
```

문제 250. 부서번호가 30번인 사원들중에서의 가장 작은 월급보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하시오 !

```
SELECT ename, sal
  FROM EMP
  WHERE sal > (SELECT min(sal) FROM EMP WHERE deptno = 30);

SELECT ename, sal
  FROM EMP
  WHERE sal >any (SELECT sal FROM EMP WHERE deptno =30);
                ※ 수직선으로 나타내면 가장 작은것 보다 큰 것들 모두
```

문제 251.(점심시간 문제) DALLAS에서 월급이 2등인 사원과 같은 월급을 받는 사원의 이름과 월급을 출력하시오 !

```
SELECT e.ename, e.sal
  FROM EMP e, (SELECT ENAME, DENSE_RANK() OVER (ORDER BY E.SAL DESC) 순위
          FROM EMP E, DEPT D
         WHERE E.DEPTNO = D.DEPTNO
           AND D.LOC = 'DALLAS') r
  WHERE e.sal = r.sal
    AND 순위 = 2;
```

문제 252. 직업이 SALESMAN인 사원들과 월급이 같고 커미션도 같은 사원들의 이름과 월급과 커미션을 출력하시오.

```
1. Non pair wise 방식
SELECT ename, sal, COMM
  FROM EMP
  WHERE sal IN (SELECT sal FROM EMP WHERE job = 'SALESMAN')
    AND NVL(COMM, -1) IN (SELECT NVL(comm,-1) FROM EMP WHERE job ='SALESMAN');
                훨씬 더 많은 데이터를 가지고 옴
                (내 생각에 WHERE절에 AND가 OR과 비슷하게 연산이 된다는 거 같아
                서브쿼리 각각의 집합에 둘다 속하기만 하면 출력되는거 같아)
                UPDATE EMP SET sal = 1250, COMM = 300 WHERE ename = 'KING';
                COMMIT;   해보면 알수있음

2. Pair wise 방식
SELECT ename, sal, comm
  FROM EMP
  WHERE (sal, NVL(comm,-1)) IN (SELECT sal, NVL(comm,-1) FROM EMP WHERE job =
'SALESMAN');
```

문제 253. 통계학과인 학생들과 나이가 같고 통신사가 같은 학생들의 이름과 나이와 전공과 통신사를 출력하시오 !

```
1. Non pairwise
SELECT ename, age, major, lower(telecom)
  FROM EMP2
  WHERE age IN ( SELECT age
                 FROM EMP2
                 WHERE major like '%통계%')
    AND LOWER(telecom) IN (SELECT LOWER(telecom)
                           FROM EMP2
                           WHERE major like '%통계%');
2. Pair wise
SELECT ename, age, major, lower(telecom)
  FROM EMP2
  WHERE (age, LOWER(telecom)) IN (SELECT age, LOWER(telecom)
                                  FROM EMP2
                                  WHERE major LIKE '%통계%');
            ※ 성능의 문제가 아니라 결과가 다르게 나옴
```

문제 254. 사원테이블의 토탈월급을 출력하시오 !

```
SELECT SUM(sal) FROM EMP;
```

문제 255. 이름, 월급, 사원 테이블의 토탈월급을 출력하시오 !(분석함수 이용 x)

```
튜닝전 :
SELECT ename, sal, (SELECT SUM(sal) FROM EMP) 토탈월급
  FROM EMP;
튜닝 후 :
SELECT ename, sal, SUM(sal) OVER () 토탈월급
  FROM EMP;
```

문제 256. 이름, 월급, 사원테이블 전체의 토탈월급, 사원테이블 전체의 최대월급, 사원테이블 전체의 최소월급, 사원테이블 전체의 평균월급을 출력하시오 !

```
SELECT ename, sal, (SELECT SUM(sal) FROM emp),
                   (SELECT max(sal) FROM emp),
                   (SELECT min(sal) FROM emp),
                   (SELECT avg(sal) FROM emp)
  FROM EMP;
       추적결과
       db block gets   0
       consistent gets 53
       physical reads  0

SELECT ename, sal, SUM(sal) OVER(),
                   MAX(sal) OVER(),
                   MIN(sal) OVER(),
                   AVG(sal) OVER()
  FROM EMP;
       추적결과
       db block gets   0
       consistent gets 41
       physical reads  0
```

문제 257. 부서번호, 이름, 월급, 자기가 속한 부서번호의 토탈월급을 출력하시오 !

```
튜닝전 :
SELECT e.deptno, e.ename, e.sal, (SELECT SUM(s.sal)
                                   FROM EMP s
                                   WHERE s.deptno=e.deptno) 부서토탈
  FROM EMP e;



튜닝후 :
SELECT deptno, ename, sal,
       SUM(sal) OVER ( PARTITION BY deptno) 부서토탈
  FROM EMP;
```

문제 258. 사원번호, 이름, 월급, 월급의 누적치를 출력하시오 !

```
튜닝전 :
SELECT empno, ename, sal, ( SELECT SUM(sal)
                            FROM EMP
                            WHERE empno BETWEEN (select MIN(empno)
                                                 FROM EMP ) AND e.empno ) 누계
 FROM EMP e
 ORDER BY empno;



튜닝후 :
SELECT empno, ename, sal, SUM(sal) OVER (ORDER BY empno ROWS
                                         BETWEEN unbounded preceding
                                         AND CURRENT row) 누적치
  FROM EMP;
```

문제 259. 부서번호, 이름, 월급, 순위를 출력하는데 (순위가 부서번호별로 각각 월급이 높은 사원에 대한 순위임)

```
SELECT deptno, ename, sal,
       RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) 순위
  FROM EMP;
```

문제 260. 위의 결과를 다시 출력하는데 각 부서번호 별로 순위가 1위인 사원들만 출력하시오 !

```
SELECT *
  FROM (SELECT deptno, ename, sal,
               RANK() OVER (PARTITION BY deptno ORDER BY sal DESC) 순위
          FROM EMP)
  WHERE 순위 = 1;
```

문제 261. 부서번호, 이름, 월급, 자기가 속한 부서번호의 평균월급을 출력하시오 !

```
SELECT deptno, ename, sal,
       ROUND(AVG(sal) OVER (PARTITION BY deptno)) 부서평균월급
  FROM EMP;
```

문제 262. 위의 결과를 다시 출력하는데 자기의 월급이 자기 부서의 평균월급보다 더 큰 사원들만 출력하시오!

```
SELECT *
  FROM (SELECT deptno, ename, sal,
               ROUND((sal) OVER (PARTITION BY deptno)) 부서평균월급
          FROM EMP)
  WHERE sal > 부서평균월급;
```

문제 263. 위의 결과를 분석함수를 이용하지 않고 수행하시오 !

튜닝 전 :

```
SELECT e.deptno, e.ename, e.sal, 부서평균월급
  FROM EMP e, (SELECT deptno, round(SUM(sal)/COUNT(*)) 부서평균월급
                 FROM EMP GROUP BY deptno) a
  WHERE E.deptno = a.deptno
    AND e.sal >= a.부서평균월급;
```

튜닝 후 :

```
SELECT *
  FROM (SELECT deptno, ename, sal,
               ROUND(AVG(sal) OVER (PARTITION BY deptno)) 부서평균월급
          FROM EMP)
  WHERE sal > 부서평균월급;
```

문제 264. 직업, 직업별 토탈월급을 출력하시오 !

```
SELECT job, SUM(sal)
  FROM EMP
  GROUP BY job;
```

문제 265. 위의 결과를 다시 출력하는데 직업이 SALESMAN의 토탈월급보다 더 큰것만 출력하시오 !

```
SELECT job, SUM(sal)
  FROM EMP
  GROUP BY job
  HAVING SUM(sal) > ( SELECT SUM(sal) FROM EMP WHERE job = 'SALESMAN');
```

문제 266. 직업, 직업별 토탈월급을 출력하는데 직업별 토탈월급들의 평균값보다 더 큰것만 출력하시오 !

```
SELECT job, SUM(sal)
  FROM EMP
  GROUP BY job
  HAVING SUM(sal) > (SELECT AVG(sum(sal)) FROM EMP GROUP BY job);
```

문제 267. 이름, 월급, 자기의 직속상사(관리자), 직속상사의 월급을 출력하시오 !(self join)

```
SELECT e.ename, e.sal, m.ename 관리자, m.sal 관리자
  FROM EMP e, EMP m
  WHERE e.mgr = m.empno;
```

문제 268. 위의 결과를 다시 출력하는데 자기의 관리자보다 더 많은 월급을 받는 사원들만 출력하시오 !

```
SELECT e.ename, e.sal, m.ename 관리자, m.sal 관리자
  FROM EMP e, EMP m
  WHERE e.mgr = m.empno
    AND e.sal > m.sal;
```

문제 269. (오늘의 마지막 문제)

이름, 입사일, 자기의 관리자 (직속상사)의 이름, 자기의 관리자 (직속상사) 의 입사일을 출력하는데 자기의 직속상사보다 먼저 입사한 사원들만 출력하시오 !

```
SELECT e.ename, e.hiredate, m.ename 상사이름, m.hiredate 상사입사일
  FROM EMP e, EMP m
  WHERE e.mgr = m.empno
    AND e.hiredate < m.hiredate;
```

문제 264.직업, 직업별 토탈월급을 출력하시오 !

```
SELECT job, SUM(sal)
  FROM EMP
  GROUP BY job;
```

문제 265. 사원테이블의 토탈월급을 출력하시오 !

```
SELECT SUM(sal)
  FROM EMP;
```

문제 266. 위의 두개의 결과를 아래와 같이 하나로 합쳐서 출력하시오 !

```
JOB             SUM(SAL)
SALESMAN        5600
CLERK           4150
PRESIDENT       5000
MANAGER 8275
ANALYST         6000
                29025
```

```
SELECT job, SUM(sal)
  FROM EMP
  GROUP BY job
  UNION ALL
SELECT TO_CHAR(null) AS job, SUM(sal)
  FROM EMP;
```

※ 작성시 유의사항 !

1. 위아래의 컬럼의 개수와 데이터 타입이 맞아야 한다
2. Order by 절은 맨 아래의 쿼리에만 사용할 수 있다.

문제 267. 위의 SQL의 결과를 아래와 같이 출력하시오 !

```
                29025
ANALYST         6000
CLERK                4150
MANAGER 8275
PRESIDENT       5000
SALESMAN        5600
```

```
SELECT TO_CHAR(null) AS job, SUM(sal)
  FROM EMP
  UNION ALL
  SELECT job, SUM(sal)
  FROM EMP
  GROUP BY job;
```

문제 268. 통신사, 통신사별 인원수를 출력하는데 맨 아래쪽에 전체 인원수를 출력하시오 !
(union all 이용해서)

```
SELECT lower(telecom), COUNT(*)
  FROM EMP2
  GROUP BY LOWER(telecom)
   UNION ALL
     SELECT NULL AS telecom, COUNT(*)
       FROM EMP2;
```

문제 269. 위의 결과를 아래와 같이 통신사를 abcd순으로 출력하시오 !

```
kt      9
lg      5
sk      13
        27
```

```
SELECT lower(telecom) AS telecom, COUNT(*)
  FROM EMP2
  GROUP BY LOWER(telecom)
   UNION ALL
     SELECT NULL AS telecom, COUNT(*)
       FROM EMP2
       ORDER BY telecom ASC;
```

문제 270. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT lower(telecom) AS telecom, COUNT(*)
  FROM EMP2
  GROUP BY LOWER(telecom)
   UNION ALL
     SELECT NULL AS telecom, COUNT(*)
       FROM EMP2
       ORDER BY telecom ASC;
```

튜닝후 :

```
SELECT LOWER(telecom), COUNT(*)
  FROM EMP2
  GROUP BY ROLLUP(LOWER(telecom));
```

문제 271. 부서번호, 부서번호별 평균월급을 출력하는데 맨 아래쪽에 전체 평균월급도 출력하시오 !

```
SELECT deptno, avg(sal)
  FROM EMP
  GROUP BY deptno
    UNION ALL
SELECT NULL AS deptno, avg(sal)
  FROM EMP;
```

문제 272. 위의 결과를 아래와 같이 다시 출력하시오 !

```
10      2916.6666666666666666666666666666666666667
20      2175
30      1566.6666666666666666666666666666666666667
        2073.2142857142857142857142857142857714286
```

튜닝전 :

```
SELECT deptno, avg(sal)
  FROM EMP
  GROUP BY deptno
    UNION ALL
SELECT NULL AS deptno, avg(sal)
  FROM EMP
  ORDER BY deptno ;
```

튜닝후 :

```
SELECT deptno, AVG(sal)
  FROM EMP
  GROUP BY ROLLUP(deptno);
```

문제 274. 부서테이블에는 존재하는 부서번호인데 사원테이블에는 존재하지않는 부서번호를 출력하시오 !

```
SELECT deptno
  FROM DEPT
  MINUS
SELECT deptno
  FROM EMP;
```

```
Dept     -     emp
10               10
20               20
30               30          =      40 이 나옴
40
```

문제 275. 사원테이블에는 존재하는 부서번호인데 부서 테이블에는 존재하지 않는 부서번호를 출력하시오 !

```
SELECT deptno
  FROM EMP
  MINUS
SELECT deptno
  FROM DEPT;
```

문제 276. (점심시간 문제) 아래의 grouping sets 의 결과를 union all 로 구현하시오 !

```
SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY GROUPING sets((deptno,job),(deptno),());
```

답 :

```
SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY deptno, job
UNION ALL
SELECT deptno, NULL AS job, SUM(sal)
  FROM EMP
  GROUP BY deptno
UNION ALL
SELECT NULL AS deptno, NULL AS job, SUM(sal)
  FROM EMP
  ORDER BY deptno, job;
```

문제 277. 부서번호, 부서번호별 토탈월급을 출력하는데 맨 아래쪽에 전체 토탈월급을 출력하시오 !
(grouping sets 를 활용)

```
SELECT deptno, SUM(sal)
  FROM EMP
  GROUP BY GROUPING sets ( (deptno), ());
```

※ grouping sets 가 rollup과 다른점은 grouping된 결과를 마음대로 지정할 수 있다라는 것이다.

문제 278. 부서번호, 직업, 부서번호별 직업별 토탈월급을 출력하시오!

```
SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY deptno, job
  ORDER BY deptno, job;
```

문제 279. 위의 결과를 grouping sets로 구현하시오 !

```
SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY GROUPING sets( (deptno,job))
  ORDER BY deptno, job;
```

문제 280. 위의 결과에서 맨 아래쪽에 전체 토탈 웕브을 하나 출력하시오 !

```
SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY GROUPING sets( (deptno,job), ())
  ORDER BY deptno, job;
```

문제 281. 아래와 같이 결과를 출력하시오 !

```
Ename   sum(sal)
SCOTT   3000
SMITH   800
TURNER  1500
WARD    1250
        29025
        SELECT ename, SUM(sal)
          FROM EMP
          GROUP BY GROUPING sets ( (ename),());
```

문제 282. 위의 결과를 레포팅함수 사용하지 말고 union all 을 사용해서 출력하시오!

```
SELECT ename, sal
  FROM EMP
UNION ALL
SELECT NULL AS ename,SUM(sal)
  FROM EMP;
```

문제 283. 아래의 union all의 결과를 grouping sets로 구현하시오 !

```
SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY deptno, job
  UNION ALL
SELECT deptno, TO_CHAR(null) AS job, SUM(Sal)
  FROM EMP
  GROUP BY deptno
  ORDER BY deptno, job;

SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY GROUPING sets ( (deptno,job), (deptno) )
  ORDER BY deptno, job;
```

문제 284. 아래의 sql 결과를 union all 구현하시오 !

```
SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY GROUPING sets ( (deptno,job), (deptno), () );

SELECT deptno, job, SUM(sal)
  FROM EMP
  GROUP BY deptno, job
UNION ALL
SELECT deptno, NULL AS job, SUM(sal)
  FROM EMP
  GROUP BY deptno
UNION ALL
SELECT NULL AS deptno, NULL AS job, SUM(sal)
  FROM EMP
  ORDER BY deptno, job;
```

문제 285. 아래와 SQL과 같은 결과를 GROUPING SETS 로 출력하시오 !

```
SELECT deptno, TO_CHAR(null) AS job, SUM(sal)
  FROM EMP
  GROUP BY deptno
UNION ALL
SELECT TO_NUMBER(null), job, SUM(sal)
  FROM EMP
  GROUP BY job;

SELECT DEPTno, job, SUM(sal)
 FROM EMP
 GROUP BY GROUPING sets ( (deptno), (job));
```

문제 266. 아래의 결과를 출력하시오 !

```
ENAME   JOB             SAL
SCOTT   ANALYST         3000
SMITH   CLERK           800
MARTIN  SALESMAN        1250
MILLER  CLERK           1300
TURNER  SALESMAN        1500
                        29025
SELECT ename, job, SUM(sal) AS sal
  FROM EMP
  GROUP BY GROUPING sets ( (ename,job), () );
```

문제 287. 아래의 결과를 출력하시오 !

```
20      CLERK           1900
20      ANALYST         6000
20      MANAGER 2975
20      부서토탈 :      10875
30      CLERK           950
30      MANAGER 2850
30      SALESMAN        5600
30      부서토탈 :      9400
        전체토탈                29025
```

답1

```
SELECT deptno,
       CASE WHEN deptno IS NULL AND
                 nvl(job,'부서토탈 : ') = '부서토탈 : '
            THEN '전체토탈'
            ELSE nvl(job,'부서토탈 : ')
            END AS sal,
       SUM(sal)
  FROM EMP
  GROUP BY GROUPING sets ((deptno,job), deptno, ())
```

답2

```
select deptno, decode(deptno, null, '전체토탈 : ', nvl(job, '부서토탈 : ')),
 sum(Sal)
  from emp
  group by rollup(deptno, job);
```

문제 288. 아래의 데이터를 emp 테이블에 입력하시오 !

```
사원번호        9345
사원이름        jane
월급            4600
입사일  2018년 10월 23일
```

```
INSERT INTO EMP (empno, ename, sal, hiredate)
  VALUES (9345, 'JANE', 4600, TO_DATE('2018/10/23', 'RRRR/MM/DD'));
                ※ 날짜 입력할 때 TO_DATE 쓰는게 주의할 사항 !
```

문제 289. 아래의 데이터를 입력하시오 !

```
사원번호        3485
사원이름        JONE
월급    5600
입사일  오늘날짜
```

```
INSERT INTO EMP (EMPNO, ENAME, SAL, HIREDATE)
  VALUES (3485, 'JONE', 5600, SYSDATE );
                ※ 시 분 초 까지 다 들어가 버린다
```

문제 290. 오늘 입사한 사원의 이름과 입사일을 출력하시오 !

```
SELECT ename, hiredate
  FROM EMP
  WHERE hiredate = TO_DATE('18/10/24', 'rr/mm/dd');

SELECT ename, hiredate
  FROM EMP
  WHERE hiredate = sysdate;
```

※ sysdate를 이용해서 데이터를 입력했다면 둘다 검색 결과가 없음 !

검색 결과가 나오게 한번 만들어 보자

튜닝전 :

```
SELECT ename, hiredate
  FROM EMP
  WHERE TO_CHAR(hiredate, 'rrrr/mm/dd') = '2018/10/24';
```

튜닝 후 :

```
SELECT ename, hiredate
  FROM EMP
  WHERE hiredate BETWEEN TO_DATE('18/10/24', 'rr/mm/dd') AND SYSDATE;
```

문제 291. (오늘의 마지막 문제) 아래의 데이터를 입력하고 이름이 null이 아니고 공백도 아닌 사원들의 이름과 월급을 출력하시오 !

내 답(틀림) :

```
SELECT ename, sal
  FROM EMP
  WHERE ename IS NOT NULL
    AND ename NOT LIKE '% %';
```

정답 :

```
select ename, sal
   from emp
   where trim(ename) is not null;
```

문제 292. 아래의 데이터를 사원테이블에 입력하시오 !

사원번호 3821
사원이름 Biff
월급 4500
입사일 오늘날짜
부서번호 30
직업 SALESMAN

```
INSERT INTO EMP (EMPNO, ename, sal, hiredate, deptno, job)
     VALUES (3821, 'Biff', 4500, TO_DATE('18/10/25', 'rr/mm/dd'), 30, 'SALESMAN');

COMMIT;       <------- 데이터를 DATABASE에 저장 하겠다.
```

문제 293. 아래의 데이터를 사원테이블에 입력하시오 !

사원번호 2912
사원이름 Anneena
월급 5000
커미션 Null
직업 ANALYST
부서번호 20

```
INSERT INTO EMP ( empno, ename, sal, comm, job, deptno)
   VALUES ( 2912, 'Anneena', 5000, null, 'ANALYST', 20);
```

```
DELETE FROM EMP WHERE hiredate IS NULL;
DELETE FROM EMP WHERE ename IN ('JANE', 'JONE', 'Biff');
```

문제 294. 부서번호가 10번인 사원들의 커미션을 9000 으로 수정하시오 !
```
UPDATE EMP SET COMM =9000 WHERE deptno = 10;
```

문제 295. SCOTT의 월급을 4700으로 변경하고 커미션을 5600으로 변경하시오 !
```
UPDATE EMP SET SAL = 4700, COMM = 5600 WHERE ENAME = 'SCOTT';
```
※ 콤마를 쓰면 한 개의 레코드의 여러 자료를 변경할 수 있음 !

문제 296. 월급이 1000 에서 3000 사이인 사원들의 부서번호를 50번으로 변경하시오 !
```
UPDATE EMP SET DEPTNO = 50 WHERE SAL BETWEEN 1000 AND 3000;
```

```
ROLLBACK;
```
커밋 이후부터 지금까지 한 모든 작업을 되돌린다 !

문제 297. sk의 월정액을 60000원으로 변경하시오 ! (telecom_price 테이블)
```
UPDATE TELECOM_PRICE SET month_price = 60000 WHERE telecom_name = 'sk';
```

```
ROLLBACK;
```

문제 298. 직업이 SALESMAN인 사원들을 삭제하시오!
```
DELETE FROM EMP WHERE JOB = 'SALESMAN';
```

```
ROLLBACK;
```

```
DELETE FROM EMP;

COMMIT;

ROLLBACK;
```

문제 299. 우리반 테이블의 data를 전부 삭제하고 commit 하시오 !
```
DELETE FROM EMP2;

COMMIT;
```

문제 300. 백업 받은 emp2_backup2 테이블의 데이터를 emp2 테이블로 로드하시오 !
```
Emp2_backup2 ----------------> emp2
                               data
INSERT INTO EMP2
SELECT * FROM EMP2_backup2;

COMMIT;
```
※ 백업 과 복구를 쿼리로 할 수 있음!

문재 301. 아래와 같이 emp2_backup3라는 테이블을 생성하시오 !
(데이터는 가져오지 않고 테이블 구조만 가져오는 문법)
```
CREATE TABLE emp2_backup3 AS
  SELECT * FROM EMP2 WHERE 1=2;
```
※ 1=2가 false이니까 테이블 구조만 가져오게 됨!

문제 302. emp2_backup3 테이블에 우리반 테이블에 통계학과와 심리학과 학생들의 데이터를 모두 입력하시오 !
```
INSERT INTO emp2_backup3
SELECT * FROM EMP2 WHERE major LIKE '%통계%'
                      OR major LIKE '심리학과';
```

문제 303. JONES보다 더 많은 월급을 받는 사원들의 커미션을 7000으로 변경하시오 !

```
UPDATE EMP SET comm = 7000
 WHERE sal > (SELECT sal FROM EMP WHERE ename = 'JONES');
```

문제 304. 관리자인 사원들의 월급을 9000 으로 변경하시오 (자기일에 직속부하가 한명이라도 있는 사원들)

```
UPDATE EMP SET sal = 9000 WHERE empno IN (SELECT mgr FROM emp);
```

※ update문 의 where 절에도 subquery 가능!

문제 305. 관리자가 아닌 사원들의 월급을 2000 으로 변경하시오 !
(자기 밑에 직속부하가 한명도 없는 사원들)

```
UPDATE EMP SET sal = 2000 WHERE empno NOT IN ( SELECT NVL(mgr,-1) FROM EMP);
```

문제 306. SMITH의 월급을 KING의 월급으로 변경하시오 !

```
UPDATE EMP SET SAL = (SELECT SAL FROM EMP WHERE ENAME = 'KING')
  WHERE ENAME = 'SMITH';
```

※update문의 set 절에도 subquery 가능!

문제 307. 이름과 부서위치를 출력하시오 !

```
SELECT e.ename, d.loc
  FROM EMP e, DEPT d
  WHERE e.deptno = d.DEPTNO;
```

문제 308. 사원 테이블에 sal2 라는 컬럼을 추가하시오 !

```
ALTER table EMP ADD sal2 number(10);
```

문제 309. 사원테이블의 sal2 를 sal 데이터로 수정하시오 !

```
UPDATE EMP SET sal2 = sal ;
```

```
Commit;
```

문제 310. 사원테이블에 emp_loc 라는 컬럼을 추가하시오 !

```
ALTER TABLE EMP ADD emp_loc VARCHAR2(20);
```

문제 311. 이름, emp_loc, 부서위치를 출력하는 부서위치의 컬럼명을 dept_loc라는 이름으로 출력하시오 !

```
SELECT e.ename, e.emp_loc, d.loc AS dept_loc
  FROM EMP e, DEPT d
  WHERE e.deptno = d.deptno;
```

문제 312. emp 테이블에 job2 라는 컬럼을 추가하고 job 의 데이터를 job2에 update하시오

```
ALTER TABLE EMP ADD job2 VARCHAR2(20);

UPDATE EMP SET job2 = job;
```

문제 313. (316 문제 먼저 하면 됨) 위의 결과에서 dept_loc의 데이터를 emp_loc에 탁! 업데이트 하시오 ~~

```
UPDATE (SELECT e.ename, e.emp_loc, d.loc AS dept_loc
          FROM EMP e, DEPT d
         WHERE e.deptno = d.deptno) SET emp_loc = dept_loc;
```

문제 314. 아래의 데이터를 emp테이블에 입력하시오 !

사원번호	8291
사원이름	Jack
월급	3000
부서번호	70

```
INSERT INTO EMP (empno, ename, sal, deptno)
   VALUES( 8291, 'jack', 3000, 70 );
```

문제 315. emp 테이블에 지금 방금 입력한 부서번호 70번 사원의 데이터를 지우시오 !

```
DELETE FROM EMP WHERE deptno = 70;
```

문제 316. (제약단원에서 배울거니까 그냥 코딩만 작성)

강제로 emp테이블에 10, 20, 30, 40 부서번호 외에는 다른 데이터가 입력 안되게 제약을 걸어라

```
- Dept테이블에 부모키를 생성
ALTER TABLE DEPT ADD
  CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno);

- Emp 테이블에 deptno에 자식키를 걸면서 dept 테이블에 deptno를 참조하라 !
ALTER TABLE EMP ADD
   CONSTRAINT emp_deptno_fk FOREIGN KEY(deptno) REFERENCEs DEPT(deptno);

- 70번 부서번호를 emp에 입력을 시도
INSERT INTO EMP (empno, ename, sal, deptno)
   VALUES( 8291, 'jack', 3000, 70 );
```

문제 317. 우리반 테이블에 month_price컬럼을 추가하시오 !

```
ALTER TABLE EMP2
  ADD month_price number(10);

컬럼삭제 :
        ALTER TABLE EMP2
          DROP COLUMN movth_price;
```

문제 318. 우리반 테이블의 telecom에 sk를 대문자로 되어져 있는 데이터를 소문자 sk 로 변경하시오 !

```
UPDATE EMP2 SET telecom = 'sk' WHERE telecom = 'SK';
```

문제 319. 아래와 같이 telecom_price를 부모 테이블로 두고 emp2 를 자식테이블로 만드시오 !

```
ALTER TABLE TELECOM_PRICE
ADD CONSTRAINT T_pk PRIMARY KEY(telecom_name);

ALTER TABLE EMP2
ADD CONSTRAINT t_fk FOREIGN KEY(telecom)
REFERENCES TELECOM_PRICE(telecom_name);
```

문제 319. 우리반 테이블의 month_price 컬럼을 telecom_price 테이블의 months_price 데이터로 변경하시오 !

```
UPDATE (SELECT e.ename, e.month_price, t.month_price months_price
   FROM EMP2 e, TELECOM_PRICE t
   WHERE e.telecom = t.TELECOM_name)
   SET month_price = months_price;
```

문제 321. SCOTT보다 월급을 많이 받는 사원들을 삭제하시오 !

```
DELETE FROM EMP WHERE sal > ( SELECT sal FROM EMP WHERE ename = 'SCOTT');
```

문제 322. ALLEN보다 늦게 입사한 사원들을 삭제하시오 !

```
DELETE FROM EMP WHERE HIREDATE >
 (SELECT HIREDATE FROM EMP WHERE ENAME = 'ALLEN');
```

문제 323. DALLAS에서 근무하는 사원들을 삭제하시오 !

```
DELETE FROM EMP WHERE deptno = ( SELECT deptno FROM DEPT WHERE loc = 'DALLAS');
```

문제 324. KING에게 보고하는 사원들을 삭제하시오 !

```
DELETE FROM EMP WHERE mgr = (SELECT empno FROM EMP WHERE ename = 'KING');
```

```
ROLLBACK;
```

문제 325. 사원테이블에 emp_loc 컬럼을 추가하시오 !

```
ALTER TABLE EMP
  ADD emp_loc VARCHAR2(20);
```

문제 326. merge 문을 이용해서 emp_loc컬럼을 dept 테이블의 loc 컬럼의 데이터로 update 하시오 !

```
MERGE INTO EMP e        Emp 테이블을 머지(합치다)할거다
  USING DEPT d  Dept 테이블을 이용해서
  ON (e.deptno = d.deptno)      연결조건
  WHEN matched THEN     만약 매치가 된다면
  UPDATE SET e.emp_loc = d.loc; 이렇게 업데이트해라!
        ※ merge가 진짜 중요해 현업에서 !!!
```

문제 327. emp테이블에 dname 컬럼을 추가하고 dept테이블의 dname데이터로 merge하시오!

```
ALTER TABLE EMP
  ADD dname VARCHAR2(20);

MERGE INTO EMP e
  USING DEPT d
  ON(e.deptno = d.deptno)
  WHEN matched THEN
  UPDATE SET e.dname = d.dname;
```

문제 328. emp 테이블과 salgrade테이블을 조인해서 이름과 월급과 grade(급여등급)을 출력하시오 !

```
SELECT e.ename, e.sal, s.grade
  FROM EMP e, SALGRADE s
  WHERE e.sal BETWEEN s.losal AND s.hisal;
```

문제 329. emp테이블에 grade컬럼을 추가하시오 !

```
ALTER TABLE EMP
  ADD grade NUMBER (10);
```

문제 330. emp 테이블에 grade컬럼을 salgrade 테이블의 grade 컬럼의 데이터로 변경하시오 !(merge문 사용)

```
MERGE INTO EMP e
  USING SALGRADE s
  ON ( e.sal BETWEEN s.losal AND s.hisal)
  WHEN matched THEN
  UPDATE SET e.grade = s.grade;

MERGE INTO EMP e
  USING (SELECT distinct e.sal, s.grade
  FROM EMP e, SALGRADE s
  WHERE e.sal BETWEEN s.losal AND s.hisal) s
  ON ( e.sal = s.sal)
  WHEN matched THEN
  UPDATE SET e.grade = s.grade;
```

※이런 방식으로 할거면 참고테이블에 중복제거 해줘야 함 !

문제 331. 부서번호, 부서번호별 토탈월급을 출력하시오 !

```
SELECT deptno, SUM(sal)
  FROM EMP
  GROUP BY deptno;
```

문제 332. 부서 테이블에 sumsal 이라는 컬럼을 추가하시오 !

```
ALTER TABLE DEPT
  ADD sumsal number(10);
```

문제 333. 부서 테이블에 sumsal 에 해당 부서번호의 토탈월급으로 값을 갱신하시오 ! (merge문 사용)

```
MERGE INTO DEPT d
  USING ( SELECT deptno, SUM(sal) sumsal
            FROM EMP
            GROUP BY deptno) e
  ON (d.deptno = e.deptno)
  WHEN matched THEN
  UPDATE SET d.sumsal = e.sumsal;
```

문제 334. 부서테이블에 deptno_cnt라는 컬럼을 추가하시오 !

```
ALTER TABLE DEPT
  ADD deptno_cnt number(10);
```

문제 335. 부서번호, 부서번호별 인원수를 출력하시오 !

```
SELECT deptno, COUNT(*)
  FROM EMP
  GROUP BY deptno;
```

문제 336. 부서 테이블에 deptno_cnt라는 컬럼에 해당 부서번호와 인원수로 값을 갱신하시오!

```
MERGE INTO DEPT d
  USING ( SELECT deptno, COUNT(*) cnt
            FROM EMP
            GROUP BY deptno) e
  ON ( d.deptno = e.deptno)
  WHEN matched THEN
  UPDATE SET d.deptno_cnt = e.cnt
```

문제 337. (오늘의 마지막 문제) 사원테이블에는 존재하지 않는데 부서테이블에만 존재하는 부서번호에 대한 데이터를 부서테이블에서 지우시오 !

```
DELETE TABLE DEPT WHERE deptno NOT IN (SELECT NVL(deptno,-1) FROM emp);
```

문제 338. 아래의 상황에서는 락이 발생할까 발생하지 않을 까?

```
A세션   B세션
        1. COMMIT;
                2. COMMIT;
        3. UPDATE EMP
        SET SAL = 9500
        WHERE ENAME = 'KING';           4. UPDATE EMP
                        SET SAL = 5000
                WHERE ENAME = 'SMITH'
```

답 : 락안걸림

문제 338. 아래의 상황에서는 락이 발생할까 발생하지 않을 까?

```
A세션   B세션
        1. COMMIT;
                2. COMMIT;
        3. UPDATE EMP
        SET SAL = 7000
        WHERE ENAME = 'ALLEN';          4. UPDATE EMP
                        SET DEPTNO = 20
                WHERE ENAME = 'ALLEN'
```

답 : 락이 걸림!

문제 340. ALLEN의 월급보다 많은 월급을 받는 사원들의 월급을 SMITH의 월급으로 변경하시오!

```
UPDATE EMP SET sal = (SELECT sal FROM EMP WHERE ename = 'SMITH')
  WHERE sal > (SELECT sal FROM EMP WHERE ename = 'ALLEN');
```

문제 341. JONES 와 같은 직업을 갖는 사원들의 커미션을 MARTIN의 커미션으로 변경하시오 !

```
UPDATE EMP SET COMM = (SELECT COMM FROM EMP WHERE ENAME = 'MARTIN')
  WHERE JOB = (SELECT JOB FROM EMP WHERE ENAME = 'JONES');
```

문제 342. emp 테이블에 loc 컬럼을 추가하시오 !

```
ALTER TABLE EMP
 ADD loc VARCHAR2(20);
```

문제 343. 지금 emp 테이블에 추가한 loc컬럼을 해당 사원의 부서 위치로 값을 갱신하시오 !

• 방법 3가지

§ Merge (튜닝된 SQL)
§ Update절의 서브쿼리문 (튜닝된 SQL)
§ 상호관련 서브쿼리문을 이용한 update문 (악성 SQL)

```
1. MERGE INTO EMP e
USING DEPT d
ON (e.deptno = d.deptno)
WHEN matched THEN
UPDATE SET e.loc = d.loc;

3. UPDATE EMP e
SET loc = (SELECT loc FROM DEPT WHERE deptno = e.deptno );
```

※ 메인쿼리문의 속성이 서브쿼리에서 참조를 한다면 메인쿼리부터 실행되서 한번에 업데이트가 되지 않고 참조속성의 첫레코드를 가져와서 서브쿼리를 진행하고 둘, 셋 레코드도 순서대로 업데이트가 된다 그렇기 때문에 속도가 느리지만 에러가 나지 않고 느리지만 결과가 나온다 !

문제 344. (점심시간 문제) 사원테이블에 grade컬럼을 추가하고 salgrade 테이블의 grade(등급)으로 값을 갱신하시오 ! (자신의 월급에 맞는 급여등급으로 갱신)

```
ALTER TABLE EMP
  ADD grade number(10);

UPDATE EMP e SET grade = (SELECT grade
                            FROM SALGRADE s
                           WHERE e.sal BETWEEN s.losal AND s.hisal);
```

문제 345. emp03 데이터를 2건 정도 입력하시오 !

```
INSERT INTO EMP03
 VALUES ( 1993, 'wjdtjdgh', 6000, 'dataanalyst', TO_DATE('1993/03/02', 'rrrr/mm/dd'));

INSERT INTO emp03
 VALUES ( 1995, 'rlawjddks', 5000, 'teacher', to_date('1995/12/25', 'rrrr/mm/dd'));
```

문제 346. 아래의 테이블을 생성하시오 !

테이블명 : emp04
컬럼명 : 이름, 나이, 주민등록번호, 생일, 주소, 핸드폰번호

```
CREATE TABLE emp04 ( ename VARCHAR2(20),
                     age   NUMBER(10),
                     code  varchar2(20),
                     birth DATE,
                     address VARCHAR2(100),
                     phone VARCHAR2(20) );
```

문제 348. 부도예측 데이터3.csv를 오라클 데이터베이스에 로드하시오 !
1. Table 생성
2. SQL GATE툴로 로드

창업 폐업 건수 로드

문제 349. 치킨집 폐업건수가 가장 높은 연도의 그 폐업건수를 출력하시오 !

```
SELECT years, col_4
  FROM end_cnt
  WHERE col_4 = (SELECT MAX(col_4) FROM end_cnt);
```

문제 350. 2006년도에 가장 폐업을 많이 한 업종은 무엇인가?

답 :

```
CREATE TABLE end_cnt2
as
SELECT * FROM end_cnt
unpivot( bbb FOR aaa IN (
HAIR,
RES,
SUSI,
CHI,
COFFEE,
KOREAN,
HOF));

SELECT * FROM end_cnt2 WHERE bbb=(SELECT MAX(bbb) FROM end_cnt2
WHERE years = 2006);
```

문제 351. (마지막 문제) 년도, 업종, 폐업 건수, 순위를 출력하는데 순위가 1위만 출력하시오 !

```
SELECT *
  FROM ( SELECT years, aaa, bbb, RANK() OVER (ORDER BY bbb desc) AS 순위
           FROM End_cnt2)
        WHERE 순위 = 1;
```

"테이블의 컬럼을 추가, 삭제, 변경할 때 사용하는 명령어 "

문제 352. 사원테이블의 사원번호, 이름, 월급, 직업, 부서번호를 가지는 EMP09라는 테이블을 생성하시오 !

```
CREATE TABLE EMP09
  AS SELECT empno, ename, sal, job, deptno
       FROM EMP;
```

문제 353. emp09에 hiredate 를 추가하시오 ! (컬럼추가)

```
Alter table emp09
  add hiredate date ;
```

컬럼명 데이터 타입

문제 354. emp09에 hiredate 에 emp 테이블의 hiredate로 값을 갱신하시오 !

```
MERGE INTO EMP09 e
USING EMP d
ON( e.empno = d.empno )
WHEN matched THEN
UPDATE SET e.hiredate = d.hiredate;
```

문제 355. emp09에 hiredate에 emp테이블의 hiredate로 값을 갱신하는데 상호관련 서브쿼리인 update문으로 수행하시오 ! (악성 sql)

```
UPDATE EMP09 a
  SET hiredate = (SELECT hiredate FROM EMP e WHERE a.empno = e.empno );
```

문제 356. emp09에 loc컬럼을 추가하고 해당 사원의 부서위치로 값을 갱신하시오 !

```
ALTER TABLE EMP09
  ADD loc VARCHAR2(20);

MERGE INTO EMP09 e
USING DEPT d
ON (e.deptno = d.deptno)
WHEN matched THEN
UPDATE SET e.loc = d.loc;
```

문제 357. (select 문) 이름, 나이 순위를 출력하시오!

```
SELECT ename, age, RANK() OVER (ORDER BY age desc)
  FROM EMP2;
```

문제 358. 우리반 테이블에 순위(rnk)라는 컬럼을 추가하고 나이에 대한 순위로 값을 갱신하시오 !

```
ALTER TABLE EMP2
  ADD rnk NUMBER(10);

MERGE INTO emp2 e
USING (SELECT ename, age, RANK() OVER (ORDER BY age desc) rnk
  FROM EMP2) d
ON ( e.ename = d.ename )
WHEN matched THEN
UPDATE SET e.rnk = d.rnk;
```

문제 359. 우리반 테이블에 empno 컬럼을 추가하고 번호를 1 ~ 27로 갱신하시오 !

```
UPDATE emp2 SET empno = ROWNUM;
```

문제 360. 우리반 테이블에 empno를 맨 앞에 추가해서 만들려면 어떻게 해야 하는가?

☆컬럼 삭제

```
ALTER TABLE emp2
  DROP COLUMN empno;

CREATE TABLE emp2_backup7
AS SELECT ROWNUM AS empno, e.*
     FROM EMP2 e;

DROP TABLE emp2;
```

문제 362. 이상엽 학생 데이터를 우리반 테이블에 한건 더 입력하시오 !

```
INSERT INTO emp2
  SELECT * FROM EMP2 WHERE ename = '이상엽';
```

문제 363. 우리반 테이블의 empno를 다시 번호 순서대로 1 ~ 28 까지 갱신하시오 !

```
UPDATE emp2 SET empno = ROWNUM;
```

문제 364. 우리반 테이블에 rnk컬럼을 삭제하고 다시 rnk 컬럼을 추가한 후에 해당 학생의 나이에 대한 순위로 값을 갱신하시오 !

```
MERGE INTO emp2 e
USING (SELECT empno, ename, DENSE_RANK() OVER (ORDER BY age desc) rnk
         FROM EMP2) e2
ON (e.empno = e2.empno)
WHEN matched THEN
UPDATE SET e.rnk = e2.rnk;
```

위의 SQL을 merge 문이 아니라 상호 관련 update문으로 수행하려면 어떻게 해야 할까?(점심시간 문제)

```
update emp2 e set rnk = (select rank() over (order by age desc) dd
  from emp2 d where e.empno = d.empno);
```

문제 365. 순위(rnk) 컬럼의 데이터를 전부 null로 변경하시오 !

```
UPDATE emp2 SET rnk = NULL;
```

문제 366. 진철의 rnk 컬럼의 값을 갱신하시오 ! (진철이만 갱신하시오)

```
MERGE INTO emp2 e
USING (SELECT empno, ename, DENSE_RANK() OVER (ORDER BY age desc) rnk
         FROM EMP2) e2
ON (e.empno = e2.empno)
WHEN matched THEN
UPDATE SET e.rnk = e2.rnk WHERE ename = '김진철';

UPDATE emp2 SET rnk = (SELECT rnk
   FROM ( SELECT ename, RANK() OVER (ORDER by age desc) rnk FROM emp2)
   WHERE ename = '김진철')
 WHERE ename = '김진철';
```

문제 368. 우리반 테이블의 주소 컬럼의 데이터 타입이 무엇이고 그 길이가 어떻게 되는지 확인하시오 !

```
DESC emp2;
```

문제 369. 우리반 테이블의 주소 컬럼의 길이를 varchar2(100) ---> varchar2(200)으로 변경하시오 !

```
ALTER TABLE emp2
  MODIFY address VARCHAR2(200);
```

문제 370. 다시 varchar2(200)에서 varchar2(100)으로 줄여보시오

```
ALTER TABLE emp2
  MODIFY address VARCHAR2(100);
```

※ 만약 100 넘어가는 데이터가 있으면 줄여지지 않는다.

문제 371. 사원 테이블에 sal 컬럼을 감추시오 !

```
ALTER TABLE EMP
 SET UNUSED COLUMN sal;
```

※ 다시 used 한다고 나오지 않는다.
당장 drop하면 DB가 느려지기 때문에 unused를 이용해서 컬럼을 숨기고 속도를 유지하면서 나중에 drop을 밤 10시쯤 한다.

```
ALTER TABLE EMP
   DROP UNUSED COLUMN;
```

문제 372.아래의 테이블을 생성하는데 사원번호에 primary key 제약을 걸어서 생성하시오 !

테이블명 : emp10

컬럼명 : empno, ename, sal

```
CREATE TABLE emp10 (empno NUMBER(10) PRIMARY KEY,
                    ename VARCHAR2(10),
                    sal   NUMBER(10) );
```

문제 373. emp10 테이블에 아래의 데이터를 입력하시오 !

사원번호	사원이름	월급
7788	scott	3000
7566	smith	2500
7788	allen	5000

```
INSERT INTO EMP10
  VALUES ( 7788, 'scott', 3000 );

INSERT INTO emp10
  VALUES (7566, 'smith', 2500);

INSERT INTO emp10
  VALUES (7788, 'allen', 5000);
```

ORA-00001: unique constraint (HEAVEN.SYS_C007001) violated

※ 사원번호에 primary key가 설정되어 있어서 중복된 데이터와 null이 입력이 안되는 것이다 !

문제 374. 아래의 테이블을 생성하는데 ename에 unique제약을 걸어서 생성하시오 !

```
CREATE TABLE emp20 (empno NUMBER(10),
                    ename VARCHAR2(20) UNIQUE,
                    sal   NUMBER(10) );
```

- 테이블의 제약 확인

```
SELECT *
  FROM USER_constraints
  WHERE TABLE_name = 'EMP20' ;   <---- EMP20 대문자로 작성

SELECT *
  FROM user_cons_columns
  WHERE TABLE_name = 'EMP20';
```

※컬럼명 확인

문제 375. emp20테이블에 ename 에 null을 중복해서 입력해보시오 !

```
INSERT INTO emp20(empno, ename, sal)
  VALUES (1111, NULL, 3000);

INSERT INTO emp20 (empno, ename, sal)
  VALUES (2222,NULL,4000);
```

※ null은 값을 알 수 없는 값이기 때문에 비교할 수 없어서 중복 입력된다.

문제 376. 아래의 테이블을 생성하는데 부서위치가 서울, 부산, 대전만 입력되도록 체크제약을 걸어서 생성하시오 !

테이블명 Emp50

컬럼명 ename
Sal
loc

```
CREATE TABLE emp50 ( empno NUMBER(10),
                     ename VARCHAR2(20),
                     sal   NUMBER(10),
                     loc   VARCHAR2(20) CHECK ( loc IN ( '서울', '부산', '대전' )));

INSERT INTO emp50
  VALUES (2122, 'scott', 4000, '광주')
```

ORA-02290: check constraint (HEAVEN.SYS_C007003) violated

※ 체크 제약이 걸려있어서 광주는 입력이 안된다 !

문제 377. 아래의 테이블을 생성하는데 loc에 제약을 걸어서 생성하시오 !
테이블명 : Dept100
컬럼명 Deptno
Loc <------NEW YORK, DALLAS, CHICAGO, BOSTON
Dname

```
CREATE TABLE dept100
 ( deptno NUMBER(10),
   loc    VARCHAR2(20) CHECK (loc IN ('NEW YORK', 'DALLAS', 'CHICAGO', 'BOSTON')),
   dname  VARCHAR2(10));
```

문제 378. dept의 모든 데이터를 dept100에 입력하시오 !

```
INSERT INTO dept100 ( deptno, loc, dname )
  SELECT deptno, loc, dname
    FROM DEPT;
```

※순서가 다르기 때문에 * 를 쓸수 없다. 이거 좀 중요하다

문제 379. dept100 테이블의 loc컬럼의 길이를 현재의 길이의 두배로 변경하시오 !

```
ALTER TABLE dept100
  MODIFY LOC VARCHAR2(40);
```

문제 380. 아래의 테이블을 생성하시오 !
테이블명 : Emp700
컬럼명 EMPNO
ENAME
SAL <-- 0 ~ 9000 사이의 데이터만 입력/수정 되도록 CHECK 제약을 거시오 !

```
CREATE TABLE emp700
 ( empno NUMBER(10),
   ename VARCHAR2(20),
   sal   NUMBER(10)  CHECK ( sal BETWEEN 0 AND 9000));
```

문제 381. 아래의 테이블을 생성하시오 !
테이블명 : Emp800
컬럼명 Empno
Ename <-- 성씨가 김씨, 이씨, 박씨만 입력되게 하시오 !
sal

```
CREATE TABLE EMP800
 ( empno NUMBER(10),
   ename VARCHAR2(20) CHECK (ename LIKE '김%' or ename LIKE '이%' OR ename like '박%'),
   sal   NUMBER(10) );

CREATE TABLE emp801
 ( empno NUMBER(10),
   ename VARCHAR2(20) CHECK (SUBSTR (ename, 1 , 1) IN ('김','이','박') ),
   sal  NUMBER(10));
```

문제 382. dept 테이블과 똑같은 테이블 구조와 데이터를 갖는 dept900테이블을 생성하는데 deptno에
primary key 제약을 걸어서 생성하시오 !

```
Emp900  --------------------dept900
(자식)                          (부모)
Deptno                        deptno
10                               10
20                               20
20                               30
10                               40
30
10
'
'
'
```

```
CREATE TABLE dept900
 ( deptno PRIMARY KEY,
   dname ,
   loc )
AS SELECT deptno, dname, LOC FROM DEPT;
```

문제 383. emp테이블과 똑같은 구조와 데이터를 갖는 emp900 테이블을 생성하는데 deptno에 foreign key
제약을 걸고 dept900에 deptno를 참조하겠다라고 해서 만드시오 !

```
CREATE TABLE emp900
AS SELECT * FROM EMP;
```

※ 테이블 구조와 데이터만 가져오고 제약은 가져오지 않는다.

```
ALTER TABLE emp900
  ADD CONSTRAINT emp900_deptno_fk FOREIGN KEY(deptno) REFERENCES DEPT900(deptno);
```

문제 384. 사원테이블에 empno에 primary key제약을 거시오 !

```
ALTER TABLE EMP
  ADD CONSTRAINT emp_empno_ky primary KEY(empno);
                        제약이름(마음대로 줘도 되나 의미있게 줘야 나중에 삭제할 때 쉽다)

CREATE TABLE EMP 434
 ( empno NUMBER(10) CONSTRAINT emp434_empno_pk PRIMARY KEY,
   ename VARCHAR2(20) );
```

※ create 를 할때는 이렇게 제약이름을 줄 수 있다.

문제 385. 우리반 테이블에 empno에 primary key제약을 거시오 !

```
ALTER TABLE EMP2
  ADD CONSTRAINT emp2_empno_pk PRIMARY KEY(empno);
```

문제 386. 사원테이블의 ename에 unique제약을 거시오 !

```
ALTER TABLE EMP
  ADD CONSTRAINT emp_ename_un UNIQUE(ename);
```

※ 현제 테이블에 중복된 데이터가 없기 때문데 걸리는 것이다.

문제 387. 우리반 테이블에 ename 에 unique 제약을 거시오 !

```
ALTER TABLE emp2
  ADD CONSTRAINT emp2_ename_un UNIQUE(ename);
```

ORA-02299: cannot validate (HEAVEN.EMP2_ENAME_UN) - duplicate keys found

문제 388. 이상엽 데이터를 하나 지우고 다시 제약을 거시오 !

```
DELETE FROM EMP2 WHERE empno = 28;

ALTER TABLE emp2
  ADD CONSTRAINT emp2_ename_un UNIQUE(ename);
```

문제 389. 아래의 데이터를 입력하시오 !

```
INSERT INTO emp2
  SELECT * FROM emp2
  WHERE ename = '이상엽';
```

문제 390. 이상엽 데이터를 한 개를 지우시오!

답1

```
ALTER TABLE emp2
  ADD rnum NUMBER(10);

UPDATE emp2 SET rnum = ROWNUM;

DELETE FROM EMP2 WHERE rnum = 28;

ALTER TABLE emp2
  DROP COLUMN rnum;
```

문제 391. 우리반 테이블에 나이 컬럼에 제약을 거는데 20살 ~ 55살 까지만 입력/수정 될 수 있도록 check 제약을 걸으시오!

```
ALTER TABLE emp2
  ADD CONSTRAINT emp2_age_ck CHECK ( age BETWEEN 20 AND 55 );
```

문제 392. 우리반 테이블에 통신사 컬럼에 check 제약을 거는데 sk, lg, kt, cj hello만 입력/수정 되게끔 체크제약을 거시오 !

```
ALTER TABLE emp2
  ADD constraint emp2_telecom_ck CHECK ( telecom IN ('sk', 'lg', 'kt', 'cj hello'));
```

문제 393. 이메일을 입력/수정할때에 @와 .이 있어야 입력 / 수정 되게끔 제약을 거시오 !

```
ALTER TABLE emp2
  ADD CONSTRAINT emp2_email_ck CHECK ( email LIKE '%@%.%');
```

- Rowid 해당 row의 유니크한 물리적 주소

  AAAE5OAABAAALCxAAC

  File번호 + block번호 + row 번호

  답2

```
DELETE FROM EMP WHERE ROWID = 'AAAE5OAABAAALCxAAN';
```

문제 391. 숫자 1부터 100까지를 출력하는 쿼리를 작성하시오 !

```
SELECT ROWNUM FROM dual CONNECT
 BY ROWNUM < 101;
```

※개체명 질의

문제 392. 위의 쿼리의 결과를 ctas로 number100이라는 테이블러 생성하시오!

```
CREATE TABLE number100
AS SELECT ROWNUM AS rnum FROM dual CONNECT
by ROWNUM <101;
```

문제 392. dept 테이블과 number100 을 cross조인 해보시오 !

```
SELECT d.deptno, d.loc, d.dname FROM DEPT d, number100 n;
```

문제 393. 이상엽 데이터를 100건 중복해서 출력하시오 !

```
SELECT e.* FROM EMP2 e, number100 WHERE ename = '이상엽';
```

문제 394. emp2테이블에 이상엽 데이터를 100건 입력하시오 !

```
INSERT INTO EMP2
SELECT e.* FROM EMP2 e, number100 WHERE ename = '이상엽';

INSERT INTO EMP2
SELECT * FROM EMP2 WHERE ename = '장보겸';

INSERT INTO EMP2
SELECT e.* FROM EMP2 e, number100 WHERE ename = '이상엽';
```

문제 395. 중복된 이상엽 데이터를 지우고 한건만 남겨두시오 !

```
DELETE FROM (SELECT * FROM EMP2 WHERE ename = '이상엽')
  WHERE ROWID NOT IN ( SELECT MIN(rowid) FROM EMP2);
```

문제 396. emp900 테이블에 emp900_deptno_fk를 삭제하시오 !

```
ALTER TABLE EMP900
  DROP CONSTRAINT emp900_deptno_fk;
```

※제약이름으로 제약을 삭제할 수 있다 !

문제 397. scott 이 가지고 있는 모든 제약을 다 삭제하시오 !

```
SELECT ' alter table ' || TABLE_name || ' drop  constraint ' || CONSTRAINT_name || ';'
  FROM USER_cons_columns;
```

※ 연결연산자로 쿼리내용을 만들어서 삭제

문제 398. 10번 부서번호인 사원들의 월급을 전부 0으로 변경하고 아래의 쿼리를 실행해보시오 !

```
SELECT * FROM deptno_avg WHERE sal > 부서평균;

UPDATE emp SET sal = 0 WHERE deptno = 10;
```

※아까랑 결과가 달라진다, 그래서 테이블로 만들면서 하는것은 좋은 방법이 아니다.

Demobld 다시 돌리세요~

문제 399. 이름, 나이, 전공, 순위를 출력하는 view를 생성하시오 (순위는 나이가 높은 순서에 대한 순위임)

View 이름 : emp2_v

```
CREATE VIEW emp2_v
AS SELECT ename, age, major, RANK() OVER (ORDER by age desc) 순위
  FROM EMP2;
```

문제 400. 우리반에서 나이가 1워부터 3위까지만 이름, 나이와 전공을 출력하시오 !

```
SELECT * from emp2_v WHERE 순위 BETWEEN 1 AND 3;
```

문제 401. 겨울왕국 대본을 입력할 테이블 winter_kingdom라는 테이블을 아래와 같이 생성하고 겨울왕국 스크립트를 입력하시오 !

카페에 올린 겨울왕국 대본을 입력하시오!

메뉴---> 도구 ---> 데이터 가져오기 ---> winter_kindom 선택 ---> 시작버튼 ---> text 선택 ---> 다음 ---> 가져올 파일 선택----> 구분식별자 (필드 구분 기호를 없음, 오른쪽 텍스트 한정자 없음)---> 첫행부터 시작 ---> 다음 ---> 그 다음은 똑같다---> 소스에 (대상이 필드1로 선택) ---> 다음 --> 실행

문제 402. 겨울왕국 대본을 단어별로 쪼개서 결과를 출력하시오 !

```
select regexp_substr(lower(win_text), '[^ ]+',1,1) aaa1 from  winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,2) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,3) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,4) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,5) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,6) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,7) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,8) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,9) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,10) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,11) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,12) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,13) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,14) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,15) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,16) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,17) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,18) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,19) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,20) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,21) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,22) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,23) aaa1 from winter_kingdom
    union all
 select regexp_substr(lower(win_text), '[^ ]+',1,24) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,25) aaa1 from winter_kingdom
```

```
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,26) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,27) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,28) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,29) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,30) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,31) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,32) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,33) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,34) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,35) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,36) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,37) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,38) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,39) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,40) aaa1 from WINTER_KINGDOM;
```

문제 403. 위의 쿼리를 view로 만들고 view를 쿼리하는데 겨울왕국 대본의 단어, 단어별 건수를 출력하는데 그 건수가 높은것부터 출력하시오 ! (점심시간 문제)

```
CREATE VIEW winter_v
AS
select regexp_substr(lower(win_text), '[^ ]+',1,1) aaa1 from  winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,2) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,3) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,4) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,5) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,6) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,7) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,8) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,9) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,10) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,11) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,12) aaa1 from winter_kingdom
```

```
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,13) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,14) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,15) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,16) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,17) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,18) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,19) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,20) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,21) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,22) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,23) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,24) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,25) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,26) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,27) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,28) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,29) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,30) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,31) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,32) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,33) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,34) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,35) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,36) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,37) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,38) aaa1 from winter_kingdom
  union all
 select regexp_substr(lower(win_text), '[^ ]+',1,39) aaa1 from winter_kingdom
   union all
 select regexp_substr(lower(win_text), '[^ ]+',1,40) aaa1 from WINTER_KINGDOM;

SELECT * FROM winter_v;
UNT(*)
```

```
SELECT aaa1, CO
  FROM WINTER_v
  WHERE aaa1 IS NOT null
  GROUP BY aaa1
  ORDER BY COUNT(*) DESC;
```

문제 404. 셜록홈즈 대본을 오라클 데이터 베이스에 입력하고 셜록홈즈에서 가장 많이 나오는 단어가 무엇인지 1위부터 10위까지 출력하시오 !

```
select COUNT(regexp_substr(lower(sl_text), '[^ ]+',1,187)) aaa1 from sherlock;
       ※문장길이 확인

CREATE TABLE sherlock
 ( sl_text VARCHAR2(4000) );
CREATE VIEW sherlock_v
AS
select regexp_substr(lower(sl_text), '[^ ]+',1, 1      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 2      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 3      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 4      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 5      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 6      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 7      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 8      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 9      ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 10     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 11     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 12     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 13     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 14     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 15     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 16     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 17     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 18     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 19     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 20     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 21     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 22     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 23     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 24     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 25     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 26     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 27     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 28     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 29     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 30     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 31     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 32     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 33     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 34     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 35     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 36     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 37     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 38     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 39     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 40     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 41     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 42     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 43     ) aaa1 from sherlock union all
```

```
select regexp_substr(lower(sl_text), '[^ ]+',1, 44     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 45     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 46     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 47     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 48     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 49     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 50     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 51     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 52     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 53     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 54     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 55     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 56     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 57     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 58     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 59     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 60     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 61     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 62     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 63     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 64     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 65     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 66     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 67     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 68     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 69     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 70     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 71     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 72     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 73     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 74     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 75     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 76     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 77     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 78     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 79     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 80     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 81     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 82     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 83     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 84     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 85     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 86     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 87     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 88     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 89     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 90     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 91     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 92     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 93     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 94     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 95     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 96     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 97     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 98     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 99     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 100    ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 101    ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 102    ) aaa1 from sherlock union all
```

```
select regexp_substr(lower(sl_text), '[^ ]+',1, 103     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 104     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 105     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 106     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 107     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 108     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 109     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 110     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 111     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 112     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 113     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 114     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 115     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 116     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 117     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 118     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 119     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 120     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 121     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 122     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 123     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 124     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 125     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 126     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 127     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 128     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 129     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 130     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 131     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 132     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 133     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 134     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 135     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 136     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 137     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 138     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 139     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 140     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 141     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 142     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 143     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 144     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 145     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 146     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 147     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 148     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 149     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 150     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 151     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 152     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 153     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 154     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 155     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 156     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 157     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 158     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 159     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 160     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 161     ) aaa1 from sherlock union all
```

```
select regexp_substr(lower(sl_text), '[^ ]+',1, 162     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 163     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 164     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 165     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 166     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 167     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 168     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 169     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 170     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 171     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 172     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 173     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 174     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 175     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 176     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 177     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 178     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 179     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 180     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 181     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 182     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 183     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 184     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 185     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 186     ) aaa1 from sherlock union all
select regexp_substr(lower(sl_text), '[^ ]+',1, 187     ) aaa1 from sherlock;

CREATE VIEW sherlock_v_c
AS
SELECT aaa1, COUNT(*) cnt
 FROM sherlock_v
 WHERE aaa1 IS NOT null
 GROUP BY aaa1
 ORDER BY COUNT(*) desc;

SELECT *
  FROM ( SELECT aaa1, cnt, RANK() OVER (ORDER BY cnt desc) 순위
           FROM sherlock_v_c)
  WHERE 순위 BETWEEN 1 AND 10;
```

문제 405. 겨울왕국 대본에서 anna라는 단어가 몇번 나오는가?

```
SELECT COUNT(*)
  FROM WINTER_v
  WHERE aaa1 LIKE 'anna%';

SELECT COUNT(*)
  FROM WINTER_v
  WHERE aaa1 LIKE 'elsa%';
```

문제 406. 겨울왕국 대본에는 긍정적인 단어가 많은가 부정적인 단어가 많은가 ?

1. 긍정 사전 테이블 생성

```
CREATE TABLE POSITIVE
(p_text VARCHAR2(2000) );

CREATE VIEW winter_p
as
SELECT aaa1
  FROM winter_v
  WHERE LOWER(aaa1) IN (SELECT LOWER(p_text) FROM positive);

SELECT COUNT(*) FROM WINTER_p;
```

2. 부정 사전 테이블 생성

```
CREATE TABLE nagative
(n_text VARCHAR2(2000) );

CREATE VIEW winter_n
as
SELECT aaa1
  FROM winter_v
  WHERE LOWER(aaa1) IN (SELECT LOWER(n_text) FROM nagative);

SELECT COUNT(*) FROM winter_n;
```

문제 407. 부서테이블에서 부서번호를 출력하는데 사원테이블에 있는 부서번호만 출력하시오 !

```
SELECT deptno
  FROM DEPT
  WHERE deptno IN (SELECT deptno FROM emp);
```

문제 408. 사원이름, 부서위치를 출력하는 view를 생성하시오 (뷰 이름 : emp303)

```
CREATE VIEW emp303
AS
SELECT e.ename, d.loc
  FROM EMP e, DEPT d
  WHERE e.deptno = d.deptno;
```

※ 생성된 뷰는 e. D. 이 안붙는다
※ 생성된 뷰는 복합뷰 이다.

문제 409. 위의 emp303을 수정하는데 KING의 부서위치를 SEOUL로 변경하시오 !

```
UPDATE EMP303 SET LOC = 'SEOUL' WHERE ENAME = 'KING';
```

ORA-01779: cannot modify a column which maps to a non key-preserved table
※복합 뷰 이기 때문에 수정이 불가능 할 수도 있어서 안된다.
원래는 바꿀 수 있는데 바꾸려니 찝찝한거임 그래서 확실하게 정해줘라

```
ALTER TABLE DEPT
  ADD CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno);

UPDATE EMP303 SET ENAME = 'AAA' WHERE ENAME = 'KING';
```

※ 이건 된다.

문제 410. 직업, 직업별 토탈월급을 출력하는 view를 생성하시오 (view 이름 : dept_sumsal)

```
CREATE VIEW dept_sumsal
AS
SELECT job, SUM(sal) sumsal     <--------view 생성시 컬럼별칭을 줘야 한다.(그룹함수)
  FROM EMP
  GROUP BY job;
```

문제 411. dept_sumsal 뷰를 수정하는데 job이 MANAGER의 토탈월급을 2000으로 수정하시오 !

```
UPDATE dept_sumsal SET sumsal = 2000 WHERE job = 'MANAGER';
        ORA-01732: data manipulation operation not legal on this view
        ※안됨
```

문제 412. scott이 가지고 있는 view리스트를 확인하시오 !

```
SELECT * FROM USER_views;
```

문제 413. 직업이 SALESMAN 인 사원들의 사원번호, 이름, 직업, 월급을 출력하는 뷰를 아래와 같이 생성하시오.

```
CREATE VIEW emp45
as
SELECT empno, ename, job, sal
  FROM EMP
  WHERE job = 'SALESNAM'
  WITH CHECK option;

UPDATE emp45 SET sal = 0 WHERE ename = 'ALLEN';

UPDATE EMP45
  SET JOB = 'MANAGER'
  WHERE ENAME = 'ALLEN';
```

문제 414. 사원번호, 이름, 월급, 직업을 출력하는 뷰를 생성하는데 월급을 9000 이상으로는 수정 못하게 하는 뷰를 생성하시오 !

```
CREATE VIEW emp414
AS
SELECT empno, ename, sal, job
  FROM EMP
  WHERE SAL < 9000
  WITH CHECK OPTION;
```

문제 415. 직업이 SALESMAN인 사원들의 사원번호, 이름, 월급, 직업을 출력하는 뷰를 생성하는데 뷰 전체 데이터를 수정, 삭제, 입력을 못하게 하시오 !

```
CREATE VIEW EMP525
AS
SELECT empno, ename, sal, job
  FROM EMP
  WITH READ ONLY;
```

문제 416. 지금 방금 만든 view 를 삭제하시오 !

```
DROP VIEW emp525;
```

문제 417. scott이 가지고 있는 모든 view를 다 삭제하시오 !

```
SELECT * FROM USER_views;

SELECT 'drop view '||VIEW_name||';' FROM user_views;

drop view DEPTNO_AVG;
drop view DEPTNO_AVG2;
drop view DEPT_SUMSAL;
drop view EMP2_V;
drop view EMP303;
drop view EMP414;
drop view EMP45;
drop view SHERLOCK_V;
drop view SHERLOCK_V_C;
drop view WINTER_N;
drop view WINTER_P;
drop view WINTER_V;
```

문제 418. 아래의 테이블을 생성하고 아래의 테이블에 번호를 1번부터 1000번 까지 입력하시오 !

```
CREATE SEQUENCE seq1;

CREATE TABLE emp418
(empno NUMBER(10) );

BEGIN FOR i IN 1 .. 100 loop
INSERT INTO emp418 VALUES(seq1.nextval);
END LOOP;
END;
/

SELECT * FROM emp418;
```

문제 419. scott이 가지고 있는 sequence를 확인하시오 !

```
SELECT sequence_name FROM user_sequences;
```

문제 420. scott 이 가지고 있는 sequence를 삭제하시오 !

```
DROP SEQUENCE seq1;
```

문제 421. 사원테이블에 이름에 인덱스를 거시오 !

```
CREATE INDEX emp_ename
  ON EMP(ename);
```

※설명 : 사원 이름을 조회할 때 검색속도를 높이기 위한 인덱스(목차)가 만들어졌음, emp_ename ,인덱스는 abcd… 순으로 정렬이 되어서 만들어졌음.

문제 422. EMP_ENAME이 ABCD순으로 정렬이되어서 사원이름을 저장했는지 확인하시오 !

```
SELECT ename, ROWID
  FROM EMP
  WHERE ename > '  ';
```

※ 인덱스의 구조 : 1. 컬럼값 + rowid로 구성
2. 컬럼값이 내림차순으로 정렬이 되어있다.
Where 절에 검색조건에서 인덱스를 엑세스하기 때문에 저게 들어간다.

튜닝전 :

```
SELECT ename, sal FROM EMP ORDER BY ename ASC;
```

튜닝후 :

```
SELECT ename, sal FROM EMP WHERE ename > '  ';
```

문제 423. (오늘의 마지막 문제) 월급에 인덱스를 걸고 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal FROM EMP ORDER BY sal ASC;

        0  db block gets
        8  consistent gets
        0  physical reads
```

튜닝후 :

```
SELECT ename, sal FROM EMP WHERE sal > -1;

        0  db block gets
        4  consistent gets
        0  physical reads
```

문제 424. 아래의 SQL을 튜닝하시오 ! (order by 절 없이 이미 데이터가 정렬되어있는 인덱스에서 읽어오게끔 튜닝하시오 !)

튜닝전 :

```
SELECT ename, sal
  FROM EMP
  ORDER BY sal DESC;
```

튜닝전 :

```
SELECT /*+ index_desc(emp emp_sal) */ ename, sal
  FROM EMP
  WHERE sal >-1 ;
```

※ 힌트를 써야 한다.

• 힌트(hint) ? 오라클에게 어떠한 데이터를 보여달라고 하는것은 select 문이고 오라클에게 지금 수행하는 select 문을 어떻게 실행해달라고 하는것이 힌트이다.

예 : 커피를 주문할 때 카페라떼 주세요 ~ (SQL)
카페라떼를 주시는데요, 원드는 하우스 블랜드로 갈아주시고 우유는 120도로 데워주시고 시럽을 2번 반 넣어주세요 ! (힌트)

```
Select /*+ 힌트 */ 컬럼명, …..

Select /*+ index_desc(테이블명 컬럼명) */ 컬럼명, ……
```

• 힌트의 종류

1. Index_desc ----> 인덱스를 descending(역순) 으로 읽어라 !
2. Index_asc ------> 인덱스를 ascending (순방향) 으로 읽어라 !

Emp_sal의 인덱스의 모습 (컬럼값 + rowid)

```
SELECT sal, ROWID
  FROM EMP
  WHERE sal > 0;
```

문제 425. 이름과 월급을 출력하는 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal
  FROM EMP
  ORDER BY ename DESC;
```

튜닝후 :

```
SELECT /*+ index_desc (emp emp_ename)*/ ename, sal
  FROM EMP
  WHERE ename > '  ';
```

문제 426. 아래의 SQL을 튜닝하시오 ! (튜닝을 위해 인덱스도 알아서 거시오 !)

튜닝전 :

```
SELECT ename, job
  FROM EMP
  ORDER BY job ASC;
```

튜닝후 :

```
CREATE INDEX emp_job on EMP(job);

SELECT /*+ index_asc(emp emp_job)*/ ename, job
  FROM EMP
  WHERE job > '  ';
```

문제 427. 이름이 SCOTT인 사원의 이름과 월급을 인덱스를 통해서 테이블을 조회하는 과정을 기술하시오 !

```
SELECT ename, sal
  FROM EMP
  WHERE ename = 'SCOTT';
        인덱스------------------------------>테이블
        (emp_sal)                                                    (emp)
SELECT ename, ROWID     SELECT rowid, ename
  FROM EMP        FROM EMP;
  WHERE ename >= '  ';
        AAAE6FAABAAALCxAAA      AAA
AAA     AAAE6FAABAAALCxAAA      AAAE6FAABAAALCxAAB      BLAKE
AAA     AAAE6FAABAAALCxAAO      AAAE6FAABAAALCxAAC      CLARK
ADAMS   AAAE6FAABAAALCxAAM      AAAE6FAABAAALCxAAD      JONES
ADAMS   AAAE6FAABAAALCxAAa      AAAE6FAABAAALCxAAE      MARTIN
ALLEN   AAAE6FAABAAALCxAAF      AAAE6FAABAAALCxAAF      ALLEN
ALLEN   AAAE6FAABAAALCxAAT
```

문제 428. 아래의 쿼리문이 어떻게 인덱스를 통해서 테이블의 데이터를 조회하는지 그림으로 설명하시오 !

```
SELECT ename, sal
  FROM EMP
  WHERE sal = 3000;

        Emp_sal 인덱스
SELECT sal, ROWID       SELECT rowid, sal
  FROM EMP        FROM EMP e;
  WHERE sal >=0;
        AAAE6FAABAAALCxAAA      0
0       AAAE6FAABAAALCxAAN      AAAE6FAABAAALCxAAB          2850
0       AAAE6FAABAAALCxAAT      AAAE6FAABAAALCxAAC          0
800     AAAE6FAABAAALCxAAK      AAAE6FAABAAALCxAAD          2975
800     AAAE6FAABAAALCxAAY      AAAE6FAABAAALCxAAE          1250
950     AAAE6FAABAAALCxAAH      AAAE6FAABAAALCxAAF          0
```

Sqplus에서 실행계획 보는법

```
 set autot traceonly explain
```

문제 429. 아래의 sql의 블록의 개수를 비교하시오 ! (실행계확과 블록의 개수를 확인하시오 !)

튜닝전 :

```
SELECT /*+ full(emp) */ ename, sal
  FROM EMP
  WHERE ename = 'SCOTT';

db block gets   0
consistent gets 41
physical reads  0

-------------------------------------------
| Id  | Operation         | Name | Rows  | By
-------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |
|*  1 |  TABLE ACCESS FULL| EMP  |     1 |
-------------------------------------------
```

튜닝후 :

```
SELECT /*+ INDEX(EMP EMP_ENAME) */ ENAME, SAL
FROM EMP
WHERE ENAME = 'SCOTT';

db block gets   0
consistent gets 40
physical reads  0

-------------------------------------------
| Id  | Operation                   | Name
-------------------------------------------
|   0 | SELECT STATEMENT            |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP
|*  2 |   INDEX RANGE SCAN          | EMP_ENA
-------------------------------------------
```

문제 430. 아래의 SQL 을 튜닝하시오 ! (튜닝전과 튜닝후의 실행계획을 각각 보시오 !)

튜닝전:

```
SELECT ename, sal
  FROM EMP
  WHERE sal * 12 = 36000;

db block gets   0
consistent gets 41
physical reads  0

-----------------------------------
| Id  | Operation         | Name |
-----------------------------------
|   0 | SELECT STATEMENT  |      |
|*  1 |  TABLE ACCESS FULL| EMP  |
-----------------------------------
```

튜닝후 :

```
SELECT ename, sal
  FROM EMP
  WHERE sal = 36000/12;

db block gets   0
consistent gets 40
```

```
physical reads  0

-------------------------------------------
| Id  | Operation                   | Name    |
-------------------------------------------
|   0 | SELECT STATEMENT            |         |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP     |
|*  2 |   INDEX RANGE SCAN          | EMP_SAL |
-------------------------------------------
```

문제 431. 아래의 SQL을 튜닝하시오 ! (튜닝전후의 실행계획을 각각 확인하시오 !)

튜닝전 :

```
SELECT ename, job, sal
  FROM EMP
  WHERE SUBSTR(job, 1, 5) = 'SALES' ;

---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
|*  1 |  TABLE ACCESS FULL| EMP  |
---------------------------------
```

튜닝후 :

```
SELECT ename, job, sal
  FROM EMP
  WHERE job LIKE 'SALES%';

-------------------------------------------
| Id  | Operation                   | Name    |
-------------------------------------------
|   0 | SELECT STATEMENT            |         |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP     |
|*  2 |   INDEX RANGE SCAN          | EMP_JOB |
-------------------------------------------
```

문제 432. 아래의 SQL을 튜닝하시오 !

```
CREATE INDEX emp_hiredate ON EMP(hiredate);
```

튜닝전 :

```
SELECT ename, sal, hiredate
  FROM EMP
  WHERE TO_CHAR(hiredate, 'rr/mm/dd') = '81/11/17';
---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
|*  1 |  TABLE ACCESS FULL| EMP  |
---------------------------------
```

튜닝후 :

```
SELECT ename, sal, hiredate
  FROM EMP
  WHERE hiredate = TO_DATE('81/11/17', 'rr/mm/dd');
-------------------------------------------------
| Id  | Operation                   | Name        |
-------------------------------------------------
|   0 | SELECT STATEMENT            |             |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP         |
|*  2 |   INDEX RANGE SCAN          | EMP_HIREDATE |
-------------------------------------------------
```

문제 433. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal, job
  FROM EMP
  WHERE ename || sal = 'SCOTT3000';
---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
|*  1 |  TABLE ACCESS FULL| EMP  |
---------------------------------
```

튜닝후 :

```
SELECT ename, sal, job
  FROM EMP
  WHERE ename = 'SCOTT' AND sal = 3000;
-----------------------------------------------
| Id  | Operation                   | Name      |
-----------------------------------------------
|   0 | SELECT STATEMENT            |           |
|*  1 |  TABLE ACCESS BY INDEX ROWID| EMP       |
|*  2 |   INDEX RANGE SCAN          | EMP_ENAME |
-----------------------------------------------
```

※ ename, sal 모두 인덱스가 있다. 두개중에 좋은 인덱스(emp_ename)를 선택해서 사용함!

왜냐하면 emp_sal 은 두건이 나오지만 emp_ename은 한건이라 간단하기 때문이다 !

월급을 인덱스로 사용하고 싶다면 ?

```
SELECT /*+ index(emp emp_sal) */ ename, sal, job
  FROM EMP
  WHERE ename = 'SCOTT' AND sal = 3000;
```

문제 434. 아래의 sql을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal
  FROM EMP
  WHERE sal LIKE '30%';
---------------------------------
| Id  | Operation         | Name |
---------------------------------
|   0 | SELECT STATEMENT  |      |
|*  1 |  TABLE ACCESS FULL| EMP  |
---------------------------------
```

※ 좌변이 전혀 가공되지 않았는데 인덱스를 타지않았지만 실행계획을 보면
1 - filter(TO_CHAR("SAL") LIKE '30%') 로 변경되어 검색이 되었다

Like는 문자형 데이터를 위한 검색 명령어이다.

숫자가 우선순위가 더 높기 때문에 문자를 숫자로 변경해줘야 하는데 %를 숫자로 못바꾸니까 숫자를 문자로 변경해준다.

튜닝후 : 쿼리문에서 like를 주로 사용할 것 같은 컬럼은 처음부터 문자로 만들었어야 한다. (모델링)

위의 경우의 해결방법은 함수기반 인덱스를 생성하는 것이다 !

```
CREATE INDEX emp_sal_func
  ON EMP( TO_CHAR(sal) );
-------------------------------------------------
| Id  | Operation                   | Name        |
-------------------------------------------------
|   0 | SELECT STATEMENT            |             |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP         |
|*  2 |   INDEX RANGE SCAN          | EMP_SAL_FUNC |
-------------------------------------------------
```

문제 435. 아래의 데이터를 입력하고 아래의 데이터를 조회하는 쿼리를 튜닝된 SQL로 작성하시오 !

```
INSERT INTO emp(empno, ename, sal)
  VALUES ( 2912, '  biff  ', 3000 );

COMMIT;
```

작성해야 할 쿼리 : 이름이 biff인 사원의 이름과 월급을 조회하시오 !

```
CREATE index emp_ename_func
 ON EMP(TRIM(ename));

SELECT ename, sal
  FROM EMP
  WHERE TRIM(ename) = 'biff';
```

```
-----------------------------------------------------
| Id  | Operation                    | Name          |
-----------------------------------------------------
|   0 | SELECT STATEMENT             |               |
|   1 |  TABLE ACCESS BY INDEX ROWID| EMP           |
|*  2 |   INDEX RANGE SCAN           | EMP_ENAME_TRIM |
-----------------------------------------------------
```

※ 설명 : 그냥 trim함수를 쓰게 되면 좌변이 가공되어서 인덱스를 엑세스하지 못하고 full table scan 하게 되므로 함수기반 인덱스를 생성해야 한다 !

문제 436. (점심시간 문제) 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, hiredate
  FROM EMP
  ORDER BY hiredate DESC;
```

튜닝후 :

```
SELECT /*+ index_desc(emp emp_hiredate) */ ename, hiredate
  FROM EMP
  WHERE hiredate > TO_DATE('1901/01/01', 'rrrr/mm/dd');
```

```
-----------------------------------------------------
| Id  | Operation                     | Name         |
-----------------------------------------------------
|   0 | SELECT STATEMENT              |              |
|   1 |  TABLE ACCESS BY INDEX ROWID | EMP          |
|*  2 |   INDEX RANGE SCAN DESCENDING| EMP_HIREDATE |
-----------------------------------------------------
```

※order by 절을 사용 안하고 인덱스를 통해서 정렬될 결과를 보려면 반드시 where 절에 해당 인덱스 컬럼이 존재해야 한다.

문제 437. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal, job
  FROM EMP
  WHERE job = 'SALESMAN'
  ORDER BY sal DESC;
```

튜닝후 :

```
SELECT /*+ index_desc(emp emp_sal) */ ename, sal, job
  FROM EMP
  WHERE job = 'SALESMAN'
    AND sal > -1;
```

문제 438. 아래의 SQL을 튜닝하시오 (그룹함수 사용하지 말고 결과를 출력하시오 !)

튜닝전 :

```
SELECT MAX(sal)
  FROM EMP;
-------------------------------------------
| Id  | Operation                  | Name    |
-------------------------------------------
|   0 | SELECT STATEMENT           |         |
|   1 |  SORT AGGREGATE            |         |
|   2 |   INDEX FULL SCAN (MIN/MAX)| EMP_SAL |
-------------------------------------------
```

※ SORT라는 말이 있으면 안좋은 것이다 !

튜닝후 :

```
SELECT /*+ index_desc(emp emp_sal) */ sal
  FROM EMP
  WHERE sal >= 0 AND ROWNUM = 1
-----------------------------------------------
| Id  | Operation                    | Name    |
-----------------------------------------------
|   0 | SELECT STATEMENT             |         |
|*  1 |  COUNT STOPKEY               |         |
|*  2 |   INDEX RANGE SCAN DESCENDING| EMP_SAL |
-----------------------------------------------
```

문제 439. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT MAX(sal)
  FROM EMP
  WHERE job = 'SALESMAN';
```

튜닝후 :

```
SELECT /*+ INDEX_DESC(emp emp_sal) */ sal
  FROM EMP
  WHERE job = 'SALESMAN' AND sal >=0 AND ROWNUM = 1;
-----------------------------------------------
| Id  | Operation                    | Name    |
-----------------------------------------------
|   0 | SELECT STATEMENT             |         |
|*  1 |  COUNT STOPKEY               |         |
|*  2 |   TABLE ACCESS BY INDEX ROWID | EMP     |
|*  3 |    INDEX RANGE SCAN DESCENDING| EMP_SAL |
-----------------------------------------------
```

문제 440. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, hiredate
  FROM EMP
  WHERE hiredate = (SELECT MAX(hiredate) FROM emp);
```

튜닝후 :

```
SELECT /*+ index_desc(emp emp_hiredate) */ ename, hiredate
  FROM EMP
  WHERE hiredate IS NOT NULL AND ROWNUM = 1
-------------------------------------------------
| Id  | Operation                    | Name       |
-------------------------------------------------
|   0 | SELECT STATEMENT             |            |
|*  1 |  COUNT STOPKEY               |            |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP        |
|*  3 |    INDEX FULL SCAN DESCENDING| EMP_HIREDATE |
-------------------------------------------------
```

문제 441. 아래의 SQL을 튜닝하시오 !

튜닝전 :

```
SELECT ename, sal
  FROM EMP
  WHERE sal = (SELECT max(sal) FROM emp);
```

튜닝후 :

```
SELECT /*+ index_desc(emp emp_sal) */ ename, sal
  FROM EMP
  WHERE sal >=0 AND ROWNUM = 1;
```

```
-----------------------------------------------
| Id  | Operation                     | Name    |
-----------------------------------------------
|   0 | SELECT STATEMENT              |         |
|*  1 |  COUNT STOPKEY                |         |
|   2 |   TABLE ACCESS BY INDEX ROWID | EMP     |
|*  3 |    INDEX RANGE SCAN DESCENDING| EMP_SAL |
-----------------------------------------------
```

문제 442. 우리반 테이블에서 이름과 나이와 전공을 출력하는데 order by절 이용하지 않고 인덱스를 통해서 정렬되게 인덱스를 걸고 SQL을 작성하시오 ! (나이가 높은 순서대로)

```
CREATE INDEX emp2_age
  ON EMP2(age);

SELECT /*+ index_desc(emp2 emp2_age) */ ename, age, major
  FROM EMP2
  WHERE age > 0;
```

```
-----------------------------------------------
| Id  | Operation                     | Name     |
-----------------------------------------------
|   0 | SELECT STATEMENT              |          |
|   1 |  TABLE ACCESS BY INDEX ROWID | EMP2     |
|*  2 |   INDEX RANGE SCAN DESCENDING| EMP2_AGE |
-----------------------------------------------
```

문제 443. SCOTT이 가지고 있는 인덱스 리스트를 조회하시오 !

```
SELECT INDEX_NAME
  FROM USER_INDEXES;
```

문제 444. 사원테이블에 월급에 걸린 sal인덱스를 삭제하시오 !

```
DROP INDEX EMP_SAL;
```

문제 445. scott이 가지고 있는 모든 인덱스를 다 삭제하시오 !

```
SELECT 'drop index ' || index_name || ';' FROM USER_indexes;

drop index EMP2_AGE;
drop index DEPT_DEPTNO_PK;
drop index EMP_JOB;
drop index EMP_HIREDATE;
drop index EMP_SAL_FUNC;
drop index EMP_ENAME_FUNC;
drop index EMP_ENAME;
```

문제 446. employee 시너님을 삭제하시오 !

```
DROP SYNONYM employee;
```

문제 447. 내가 가지고 있는 권한중에 create user 권한이 있는지 확인하시오 !

```
SELECT PRIVILEGE
  FROM session_privs
  WHERE PRIVILEGE LIKE 'CREATE USER';
```

문제 448. smith라는 유저를 생성하는데 패스워드를 oracle이라고 해서 생성하시오 !

```
CREATE USER smith
  IDENTIFIED BY oracle;

SQL> grant connect to smith;
SQL> connect smith/oracle
```

문제 449. smith 유저에서 아래의 테이블을 생성하시오 !

테이블명 emp05

컬럼명 empno
ename
sal

```
Create table emp05
  ( empno number(10),
    ename varchar2(20),
    sal      number(10));
```

문제 450. smith유저에게 create table 권한을 부여하시오

```
SQL> connect heaven/heaven
SQL> grant create table to smith;
```

문제 451. smith 유저에게 아래의 테이블을 생성하고 데이터를 insert하면 입력 될까요 ?

테이블명 emp05

컬럼명 empno
ename
sal

```
Create table emp05
  ( empno number(10),
    ename varchar2(20),
    sal      number(10));
    ORA-01950: no privileges on tablespace 'SYSTEM'
```

※집을 지을 수 있는데 땅이 없어서 못하는 것임

문제 452. scott유저로 접속해서 smith유저에게 오라클 데이터베이스의 땅을 자유롭게 이용할 수 있는 권한을 부여하시오!

```
SQL> grant unlimited tablespace to smith;
```

문제 453. scott유저로 접속해서 smith유저에게 scott유저의 emp 테이블을 select 할 수 있는 권한을 부여하시오 !

```
SQL> grant select on emp to smith;
SQL> connect smith/oracle
SQL> select * from heaven.emp;
```

※ 회사가면 scott유저 말고 smith유저를 줄거다
그래서 유저이름 다 치면서 조회하는거는 불편하다
그래서 synonym을 만들어라

문제 454. scott유저에서 smith유저가 scott.emp로 테이블을 조회하지 않도록 emp라는 시너님을 public으로 생성하시오 !

```
SQL> connect heaven/heaven

SQL> create public synonym emp for heaven.emp;

SQL> connect smith/oracle

SQL> select * from emp;
```

※ 퍼블릭을 주게되면 모든 유저기 이 시너님을 사용할 수 있다 !

문제 455. smith 유저에게 emp2테이블을 아래와 같이 조회할 수 있는 권한을 주시오 !

```
SQL> connect smith/oracle
SQL> select * from emp2;
```

답 :

```
SQL> connect heaven/heaven
Connected.
SQL> create public synonym emp2 for heaven.emp2;
```

문제 456. 아래와 같은 상황을 만드시오 !

Allen과 king을 만들고 allen과 king에게 connect 할 수 있는 권한을 부여하고 아래와 같이 도스창 3개를 여시오 ~

```
Scott -----------> allen --------------> king
SQL> connect heaven/heaven

SQL> create user allen
  2  identified by allen;

SQL> create user king
  2  identified by king;

SQL> grant connect to allen;
SQL> grant connect to king;
```

문제 457. scott이 allen에게 emp table 을 select 할 수 있는 권한을 부여하시오 !

```
SQL> grant select on emp to allen;
```

문제 458. allen이 king에게 scott.emp 테이블을 select 할 수 있는 권한을 부여하시오 !

```
SQL> grant select on emp to king;
grant select on emp to king
                       *
ERROR at line 1:
ORA-01031: insufficient privileges
```

※권한이 없다면서 실행이 안된다 !

```
Scott -----------> allen --------------> king
• Grant select on emp to allen with grant option;
        1. Emp 테이블을 select 할 수 있는 권한
        2. Emp 테이블을 select 할 수 있는 권한을 남에게 줄 수 있는 권한
SQL> grant select on emp to king;

Grant succeeded.
```

문제 459. 위와 같은 상황에서 scott이 다시 allen에게 주었던 emp테이블을 select 할 수 있는 권한을 뺏으면 king은 어떻게 되겠는가 ?

```
Scott -----X-----> allen --------------> king
Grant select on emp to allen with grant option;
SQL> revoke select on emp from allen;
```

※이렇게 하면 king도 select 할 수 있는 권한을 같이 뺏긴다 !

문제 460. start with의 데이터를 BLAKE로 변경해서 출력하시오 !

```
SELECT LEVEL, empno, ename, mgr
  FROM EMP
  START WITH ename = 'BLAKE'
  CONNECT BY PRIOR empno = mgr;
```

문제 461. 다시 KING부터 전체사원을 출력하는데 BLAKE는 제외하고 출력하시오 !

```
SELECT LEVEL, empno, ename mgr
  FROM EMP
  WHERE ename != 'BLAKE'
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr;
```

문제 462. 위의 결과를 다시 출력하는데 BLAKE를 포함해서 BLAKE의 팀원들이 다 출력되지 않게 하시오 !

```
SELECT LEVEL, empno, ename, mgr
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr AND ename != 'BLAKE'
```

문제 463. 다시 전체를 출력하는데 월급이 높은 사원부터 출력하시오 !

```
SELECT LEVEL, empno, ename, mgr
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr
  ORDER BY sal DESC;
```

※그냥 다 섞여서 나와버림

문제 464. 위의 결과를 다시 출력하는데 서열이 깨지지 않는 상태에서 월급이 높은 사원순으로 출력되게 하시오 !

```
SELECT LEVEL, empno, ename, mgr
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr
  ORDER siblings BY sal DESC;

SELECT RPAD(' ', LEVEL*2 ) || ename, sal
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr
  ORDER siblings BY sal DESC;
```

※이렇게 쓰면 게이트에서 계층을 좀더 잘 볼 수 있다

문제 465. 계층형 질의문과 짝꿍 함수인 sys_connect_by 함수를 이용해서 아래의 결과를 출력하시오 !

```
SELECT ename, SYS_connect_by_path(ename, '/') AS path
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr;
```

```
KING                /KING
JONES               /KING/JONES
SCOTT               /KING/JONES/SCOTT
ADAMS               /KING/JONES/SCOTT/ADAMS
FORD                /KING/JONES/FORD
SMITH               /KING/JONES/FORD/SMITH
BLAKE               /KING/BLAKE
ALLEN               /KING/BLAKE/ALLEN
WARD                /KING/BLAKE/WARD
MARTIN              /KING/BLAKE/MARTIN
TURNER              /KING/BLAKE/TURNER
```

문제 466. 아래와 같이 결과를 출력하시오 ! (오늘의 마지막 문제)

```
KING    KING
JONES   KING/JONES
SCOTT   KING/JONES/SCOTT
ADAMS   KING/JONES/SCOTT/ADAMS
FORD    KING/JONES/FORD
SMITH   KING/JONES/FORD/SMITH
BLAKE   KING/BLAKE
ALLEN   KING/BLAKE/ALLEN
WARD    KING/BLAKE/WARD
MARTIN  KING/BLAKE/MARTIN
TURNER  KING/BLAKE/TURNER
JAMES   KING/BLAKE/JAMES
CLARK   KING/CLARK
MILLER  KING/CLARK/MILLER

SELECT ename, ltrim(SYS_connect_by_path(ename, '/'),'/') AS path
  FROM EMP
  START WITH ename = 'KING'
  CONNECT BY PRIOR empno = mgr;
```

문제 467. resource 라는 role 이 가지고 있는 권한이 무엇인지 확인하시오 !

```
SELECT * FROM ROLE_sys_privs
  WHERE ROLE = 'RESOURCE';
        RESOURCE        CREATE SEQUENCE
        RESOURCE        CREATE TRIGGER
        RESOURCE        CREATE CLUSTER
        RESOURCE        CREATE PROCEDURE
        RESOURCE        CREATE TYPE
        RESOURCE        CREATE OPERATOR
        RESOURCE        CREATE TABLE
        RESOURCE        CREATE INDEXTYPE
```

문제 468. CONNECT 라는 ROLE에는 어떠한 권한이 있는지 확인하시오!

```
SELECT * FROM ROLE_sys_privs
  WHERE ROLE = 'CONNECT';
        CONNECT CREATE SESSION
```

문제 469. DBA 라는 role에는 들어있는 권한이 총 몇 개인지 확인하시오 !

```
SELECT COUNT(*) FROM role_sys_privs
  WHERE ROLE = 'DBA';
          COUNT(*)
        ----------
               202
```

문제 470. king과 allen 유저를 삭제하시오 !

```
DROP USER king CASCADE;
DROP USER allen CASCADE;
```

문제 471. jones 라는 유저를 만들고 scott의 emp 테이블을 select, insert, update, delete 할 수 있는 권한을 부여하시오 !

```
CREATE USER jones
  IDENTIFIED BY tiger;

GRANT CONNECT, RESOURCE TO jones;

GRANT SELECT, INSERT, UPDATE, DELETE  ON EMP TO jones;

GRANT ALL ON EMP TO jones;
 위에꺼랑 같은거
```

문제 472. jones가 가지고 있는 권한들이 무엇인지 확인하시오 !

1. System권한 확인 (객체를 만들 수 있는 권한)

```
SQL> connect jones/tiger

SQL> SELECT * FROM session_privs;

PRIVILEGE
--------------------------------
CREATE SESSION
UNLIMITED TABLESPACE    땅덩어리를 사용할 수 있는 권한
CREATE TABLE
CREATE CLUSTER
CREATE SEQUENCE
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE
```

2. 객체 권한 확인 (특정 데이터를 엑세스 할 수 있는 권한)

★★★Select * from user_tab_privs;★★★

문제 473. 저번기수 학생들 데이터 (emp3.csv)파일을 링크 걸어서 쿼리할 외부 테이블을 생성하시오 !

```
create directory emp_dir2 as 'c:W';

create table ext_emp2
         (empno number(10),
          ename varchar2(20),
          age number(10),
          birth DATE,
          major VARCHAR2(80),
          email VARCHAR2(50),
          mobile VARCHAR2(50),
          address VARCHAR2(100),
          telecom VARCHAR2(10))
          organization external
         (type oracle_loader
          default directory emp_dir
          access PARAMETERS
          (records delimited by newline
           fields terminated by ","
          (empno char,
           ename char,
           age CHAR,
           birth DATE "yyyy/mm/dd",
           major char,
           email CHAR,
           mobile CHAR,
           address CHAR,
           telecom char))
           location ('emp3.txt') )
           REJECT LIMIT UNLIMITED;

SELECT * FROM ext_emp2;
```

문제 474. 전 기수 학생들 데이터인 외부 테이블 ext_emp2로 view를 생성하는데 이름과 나이와 전공만 출력하는 view를 ext_view라는 이름으로 생성하시오 !

```
CREATE VIEW ext_view
AS
SELECT ename, age, major
  FROM ext_emp2;
```

※ 만들어 짐

문제 475. 우리반(emp2) 테이블과 전기수 학생들 테이블 (ext_emp2)을 통신사를 연결고리 컬럼으로 해서 조인해서 8기 학생 이름, 9기 학생 나이, 7기 학생 이름, 7기학생 나이 를 출력하시오 !

```
SELECT e.ename, e.age, d.ename, d.age
  FROM EMP2 e, ext_emp2 d
  WHERE LOWER(e.telecom) = LOWER(d.telecom);
```

문제 476. 전기수 학생들 테이블인 ext_emp2 의 ename에 인덱스를 생성하시오 !

```
CREATE INDEX ext_emp2_ename
ON ext_emp2(ename);
```

ORA-30657: operation not supported on external organized table

※ 안된다 !!

문제 477. 우리반 테이블인 emp2에 대한 테이블 정보를 데이터베이스에 저장하시오 !

```
comment on table emp2 is ' 우리반 학생들에 대한 정보가 있는 테이블로서 이름, 나이, 생일,
                           전공, 메일, 통신사, 주소에 대한 정보가 있습니다.';

select * from user_tab_comments;
```

EMP2 TABLE
우리반 학생들에 대한 정보가 있는 테이블로서 이름, 나이, 생일, 전공, 메일, 통신사, 주소에 대한 정보가 있습니다.

문제 478. 사원 테이블의 컬럼에 대한 주석도 남기시오 !

```
comment on column emp.empno is
'사원번호 입니다.';

select * from user_col_comments
where table_name = 'EMP';
```

EMP EMPNO
사원번호 입니다.

문제 479. 위의 insert 문을 다시 작성하는데 월급이 0~9000 사이의 데이터만 입력되게끔 insert문을 작성하시오 !

```
insert into emp (select empno, ename, sal, deptno)
                 from emp
                 where sal between 0 and 9000
                 with check option)
  values ( &empno, '&ename', &sal, &deptno );
```

문제 480. (오늘의 마지막 문제) 7기 학생들의 나이의 평균과 8기 학생들의 나이의 평균을 아래와 같이 출력하시오 !

```
select '7기' as 기수, round(avg(age)) as 나이평균값 from ext_emp2
union all
select '8기' as 기수, round(avg(age)) as 나이평균값 from emp2;
```

문제 481. emp테이블을 ctas로 백업하시오 !

```
create table emp_backup
as
select *
  from emp;
```

문제 482. dept 테이블과 salgrade테이블을 delete하고 commit하시오 !

```
delete from dept;
delete from salgrade;
commit;
```

문제 483. dept 테이블과 salgrade테이블을 flashback해서 복구하시오 !

```
select * from dept
  as of timestamp to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');

select * from salgrade
  as of timestamp to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');

alter table dept enable row movement;
alter table salgrade enable row movement;

flashback table dept to timestamp
  to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');

flashback table salgrade to timestamp
  to_timestamp('2018/11/02:11:20:00','rrrr/mm/dd:hh24:mi:ss');
```

```
grant dba to scott;

alter user sh account unlock;

alter user sh identified by sh;

alter user hr account unlock;

alter user sh identified by hr;
```

```
SQL> exit

C:WUsersWAdministrator>set ORACLE_SID=xe

C:WUsersWAdministrator>sqlplus heaven/heaven
```

문제 484. 아래의 테이블을 생성하고 아래의 테이블에 emp 테이블에서 empno, ename, sal, deptno를 입력하시오 !

```
create table emp1000
as
select *
  from emp
  where 1=2;

create table emp2000
as
select *
  from emp
  where 1=2;

create table emp3000
as
select *
  from emp
  where 1=2;
```

```
insert into emp1000(empno, ename, sal, deptno)
  select empno, ename, sal, deptno
  from emp;

insert into emp2000(empno, ename, sal, deptno)
  select empno, ename, sal, deptno
  from emp;

insert into emp3000(empno, ename, sal, deptno)
  select empno, ename, sal, deptno
  from emp;
```

문제 485. 위의 문제를 다시 해결하는데 이번에는 emp1000, emp2000, emp3000에 다중 insert문을 이용해서 한번에 입력되게 하시오 ! (p 162)

```
insert all
  into emp1000(empno, ename, sal, deptno)
  into emp2000( empno, ename, sal, deptno)
  into emp3000( empno, ename, sal, deptno)
  select empno, ename, sal,deptno
  from emp;
```

문제 486. 조건부 insert를 이용해서 p 170 페이지를 잘 보고
emp 테이블에서 부서번호 10번은 emp1000에 입력
부서번호 20번은 emp2000에 입력
부서번호 30번은 emp3000에 입력되게
하시오 ! (점심시간 문제)
컬럼은 전체 컬럼을 다 입력하게 하시오 !

```
insert all
  when deptno = 10 then
  into emp1000
  when deptno = 20 then
  into emp2000
  when deptno = 30 then
  into emp3000
  select * from emp;

  select * from emp1000;
  select * from emp2000;
  select * from emp3000;
```

문제 487. 우리반 테이블의 구조를 담는 테이블 3개를 아래와 같이 만드시오

```
emp2_sk
emp2_lg
emp2_kt

create table emp2_sk
as
select * from emp2 where 1=2;

create table emp2_lg
as
select * from emp2 where 1=2;

create table emp2_kt
as
select * from emp2 where 1=2;
```

문제 488. 조건부 다중 insert문을 이용해서 우리반 테이블의 데이터중에 sk는 emp2_sk로
lg는 emp2_lg로 kt는 emp2_kt로 데이터를 이행하시오 !

```
insert all
  when lower(telecom) = 'sk' then
  into emp2_sk
  when lower(telecom) = 'lg' then
  into emp2_lg
  when lower(telecom) = 'kt' then
  into emp2_kt
select * from emp2;
```

문제 489. 위의 테이블 3개를 다 truncate시키고 다시 데이터를 입력하는데 나이가 30대 이상이면
통신사와 상관없이 emp2_sk 에 입력하고 나머지 20대인 학생들 중에서 통신사가 lg이면
emp2_lg 에 입력하고 통신사가 kt면 emp2_kt에 입력하시오 (p 172 ~ 173 확인)

```
truncate table emp2_sk;
truncate table emp2_lg;
truncate table emp2_kt;

insert first
  when age >= 30 then
  into emp2_sk
  when lower(telecom) = 'lg' then
  into emp2_lg
  when lower(telecom) = 'kt' then
  into emp2_kt
  select * from emp2;
```

만들어라

```
create table order2
( ename  varchar2(10),
  bicycle  number(10),
  camera   number(10),
  notebook  number(10) );
```

```
insert  into  order2  values('SMITH', 2,3,1);
insert  into  order2  values('ALLEN',1,2,3 );
insert  into  order2  values('KING',3,2,2 );
```

```
commit;
```

문제 490. unpivot문을 이용해서 order2를 아래와 같이 출력하시오 !

```
ENAME                AAA                    BBB
-------------------- ---------------- ----------
SMITH           BICYCLE               2
SMITH           CAMERA                3
SMITH           NOTEBOOK              1
ALLEN           BICYCLE               1
ALLEN           CAMERA                2
ALLEN           NOTEBOOK              3
KING            BICYCLE               3
KING            CAMERA                2
KING            NOTEBOOK              2
```

```
select * from order2
  unpivot( ITEM for CNT in ( BICYCLE,CAMERA, NOTEBOOK));
```

문제 491. 아래의 지역별 범죄현황 데이터를 오라클 db에 입력하시오 !

```
create table crime_age
( local varchar2(100),
type varchar2(20),
sum_tot number(10),
under_6 number(10),
under_12 number(10),
under_15 number(10),
under_20 number(10),
under_30 number(10),
under_40 number(10),
under_50 number(10),
under_60 number(10),
over_60 number(10),
unkonwn number(10),
gender varchar2(20),
year number(10) );
```

문제 492. crime_age테이블을 unpivot해서 아래의 결과를 출력하시오 !

```
답1
select * from crime_age
  unpivot ( cnt for age in (under_6,
                            under_12,
                            under_15,
                            under_20,
                            under_30,
                            under_40,
                            under_50,
                            under_60,
                            over_60));

답2
select * from (SELECT LOCAL,
                        type,
                        sum_tot,
                        under_6,
                        under_12,
                        under_15,
                        under_20,
                        under_30,
```

```
                            under_40,
                            under_50,
                            under_60,
                            over_60,
                            gender
                        FROM crime_age)
  unpivot ( cnt for age in (under_6,
                            under_12,
                            under_15,
                            under_20,
                            under_30,
                            under_40,
                            under_50,
                            under_60,
                            over_60));
```

문제 493. 2014년도에 살인의 피해자인 나이대, 나이대별 건수를 출력하시오 !

```
select age, SUM(cnt) from crime_age
  unpivot ( cnt for age in (under_6,
                            under_12,
                            under_15,
                            under_20,
                            under_30,
                            under_40,
                            under_50,
                            under_60,
                            over_60))
WHERE TYPE = '살인'
GROUP BY age
ORDER BY SUM(cnt) DESC;
```

문제 494. pivoting insert 문을 이용해서 나이대에 대한 정보를 테이블의 data로 만드시오.

```
create table crime_age2
 ( local varchar2(100),
   type varchar2(20),
   sum_tot number(10),
   age_type varchar2(20),
   cnt number(10) );

insert all
  into crime_age2 values(local, type, sum_tot, 'under_6',under_6)
  into crime_age2 values(local, type, sum_tot, 'under_12',under_12)
  into crime_age2 values(local, type, sum_tot, 'under_15',under_15)
  into crime_age2 values(local, type, sum_tot, 'under_20',under_20)
  into crime_age2 values(local, type, sum_tot, 'under_30',under_30)
  into crime_age2 values(local, type, sum_tot, 'under_40',under_40)
  into crime_age2 values(local, type, sum_tot, 'under_50',under_50)
  into crime_age2 values(local, type, sum_tot, 'under_60',under_60)
  into crime_age2 values(local, type, sum_tot, 'over_60', over_60)
select local, type, sum_tot, under_6,under_12,under_15,under_20,
       under_30,under_40,under_50,under_60,over_60
  from crime_age;
```

문제 495. 서울시내에서 살인이 가장 많이 일어나는 지역이 어디인지 지역이름, 건수, 순위를 출력하시오 !

```
SELECT LOCAL, sum(cnt),DENSE_RANK() OVER (ORDER BY sum(cnt) desc) AS 순위
  FROM crime_age2
  WHERE TYPE = '살인'
  GROUP BY local;
```

문제 496. emp 테이블로 emp5000 테이블을 생성하고 emp5000테이블의 데이터 절반을 지우시오!

```
create table emp5000
as
select * from emp;

delete from emp5000 where rownum <=8;

commit;

SQL> select count(*) from emp5000;

  COUNT(*)
----------
         9

update emp5000
  set sal = 0;

commit;
```

문제 497. emp테이블의 있는 데이터를 emp5000에 merge하는데 emp 테이블과 emp5000에 양쪽에 다 존재하는
사원들은 emp5000 의 사원들의 월급을 emp테이블의 사원의 월급으로 변경하고 그렇지 않고
emp테이블과 emp5000에 양쪽에 다 존재하지 않고 emp 에만 존재하는 사원들은 emp5000에 입력하시오!

```
merge into emp5000 e5
using emp e
on (e5.empno = e.empno)
when matched then
 update set e5.sal = e.sal
when not matched then
insert (e5.empno, e5.ename, e5.job, e5.mgr, e5.hiredate, e5.sal, e5.comm, e5.deptno)
values (e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm, e.deptno);
```

문제 498. 문제 497번 상황을 똑같이 만들고 emp테이블에서 emp5000테이블에 있는 사원 한명을 지우시오!

```
merge 문 3가지 수행
1. emp의 월급으로 -----> emp5000의 월급 갱신
2. emp            -----> emp5000에 insert
        (emp5000에 없는 사원들의 데이터만 입력)
3. emp테이블에는 존재하지 않는데 emp5000에만 있는 사원들은 지우시오 !

delete from emp where ename = 'KING';

merge into emp5000 e5
using emp e
on (e5.empno = e.empno)
when matched then
 update set e5.sal = e.sal
 delete where e5.ename not in e.ename
when not matched then
```

```
insert (e5.empno, e5.ename, e5.job, e5.mgr, e5.hiredate, e5.sal, e5.comm, e5.deptno)
values (e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm, e.deptno);
```
이거말고 다른 답

문제 499. 오늘날짜에서 달을 추출하시오 !

```
select extract ( month from sysdate )
  from dual;
EXTRACT(MONTHFROMSYSDATE)
-------------------------
                       11
```

문제 500. 입사한 년도, 입사한 년도별 토탈월급을 출력하시오 !

```
select extract ( year from hiredate ), sum(sal)
     from emp
     group by extract ( year from hiredate );
```

```
====================================
EXTRACT(YEARFROMHIREDATE)   SUM(SAL)
------------------------- ----------
                     1982       4300
                     1983       1100
                     1980        800
                     1981      19825
====================================
```

문제 501. 지금부터 10분전에 emp테이블의 king의 월급이 얼마였는지 확인하시오 !

```
update emp
  set sal = 0
  where ename = 'KING';

commit;

select ename, sal
  from emp
  as of timestamp (systimestamp - interval '10' minute)
  where ename = 'KING';
```

```
==============================
ENAME                       SAL
-------------------- ----------
KING                       5000
==============================
```

문제 502. 위의 문제를 to_yminterval 사용하지 않고 해결하시오 !

```
select add_months(sysdate, 14)
  from dual;
```

```
========
ADD_MONT
--------
20/01/06
========
```

문제 503. 사원테이블의 평균월급을 출력하시오.

```
select avg(sal)
  from emp;
```

```
==========
  AVG(SAL)
----------
1716.07143
==========
```

문제 504. 사원번호, 이름, 월급, 사원테이블의 평균월급을 출력하시오 !

```
select empno, ename, sal, avg(sal) over ()
  from emp;
```

```
=============================================================
     EMPNO ENAME                      SAL AVG(SAL)OVER()
---------- -------------------- ---------- --------------
      7839 KING                          0     1716.07143
      7698 BLAKE                      2850     1716.07143
      7782 CLARK                      2450     1716.07143
      7566 JONES                      2975     1716.07143
      7654 MARTIN                     1250     1716.07143
      7499 ALLEN                      1600     1716.07143
      7844 TURNER                     1500     1716.07143
      7900 JAMES                       950     1716.07143
      7521 WARD                       1250     1716.07143
      7902 FORD                       3000     1716.07143
      7369 SMITH                       800     1716.07143
      7788 SCOTT                      3000     1716.07143
      7876 ADAMS                      1100     1716.07143
      7934 MILLER                     1300     1716.07143
=============================================================
```

문제 505. 위의 결과를 데이터 분석함수를 이용하지 말고 수행하시오 !

```
select empno, ename, sal, (select avg(sal) from emp)
  from emp;
```

```
=============================================================
     EMPNO ENAME                      SAL AVG(SAL)OVER()
---------- -------------------- ---------- --------------
      7839 KING                          0     1716.07143
      7698 BLAKE                      2850     1716.07143
      7782 CLARK                      2450     1716.07143
      7566 JONES                      2975     1716.07143
      7654 MARTIN                     1250     1716.07143
      7499 ALLEN                      1600     1716.07143
      7844 TURNER                     1500     1716.07143
      7900 JAMES                       950     1716.07143
      7521 WARD                       1250     1716.07143
      7902 FORD                       3000     1716.07143
      7369 SMITH                       800     1716.07143
      7788 SCOTT                      3000     1716.07143
      7876 ADAMS                      1100     1716.07143
      7934 MILLER                     1300     1716.07143
=============================================================
```

문제 506. 사원번호, 이름, 월급, 사원테이블의 토탈월급,
                              사원테이블의 최대월급,
                              사원테이블의 최소월급,
                              사원테이블의 평균월급을
  출력하시오 !

답1
```
select empno, ename, sal, sum(sal) over (),
                          max(sal) over (),
                          min(sal) over (),
                          avg(sal) over ()
  from emp;
```

답2
```
select empno, ename, sal, (select sum(sal) from emp) 토탈,
                          (select max(sal) from emp) 최대,
                          (select min(sal) from emp) 최소,
                          (select avg(sal) from emp) 평균
  from emp;
```

EMPNO	ENAME	SAL	SUM(SAL)OVER()	MAX(SAL)OVER()	MIN(SAL)OVER()	AVG(SAL)OVER()
7839	KING	0	24025	3000	0	1716.07143
7698	BLAKE	2850	24025	3000	0	1716.07143
7782	CLARK	2450	24025	3000	0	1716.07143
7566	JONES	2975	24025	3000	0	1716.07143
7654	MARTIN	1250	24025	3000	0	1716.07143
7499	ALLEN	1600	24025	3000	0	1716.07143
7844	TURNER	1500	24025	3000	0	1716.07143
7900	JAMES	950	24025	3000	0	1716.07143
7521	WARD	1250	24025	3000	0	1716.07143
7902	FORD	3000	24025	3000	0	1716.07143
7369	SMITH	800	24025	3000	0	1716.07143
7788	SCOTT	3000	24025	3000	0	1716.07143
7876	ADAMS	1100	24025	3000	0	1716.07143
7934	MILLER	1300	24025	3000	0	1716.07143

```
          0  db block gets
         16  consistent gets
          0  physical reads
```

```
select empno, ename, sal, (select sum(sal), max(sal), min(sal), avg(sal) from emp)
  from emp;
```

```
1행에 오류:
ORA-00913: too many values
```

※ 스칼라 서브쿼리의 특징 !
"스칼라 서브쿼리는 하나의 값만 리턴한다."

```
select empno, ename, sal,
       (select sum(sal) || max(sal) || min(sal) || avg(sal) from emp)
  from emp;
```

```
=============================================================================================
EMPNO ENAME                      SAL (SELECTSUM(SAL)||MAX(SAL)||MIN(SAL)||AVG(SAL)FROMEMP)
----- ------------------- --------- ------------------------------------------------------
 7839 KING                        0 24025300001716.071428571428571428571428571428571428571
 7698 BLAKE                    2850 24025300001716.071428571428571428571428571428571428571
 7782 CLARK                    2450 24025300001716.071428571428571428571428571428571428571
 7566 JONES                    2975 24025300001716.071428571428571428571428571428571428571
 7654 MARTIN                   1250 24025300001716.071428571428571428571428571428571428571
 7499 ALLEN                    1600 24025300001716.071428571428571428571428571428571428571
 7844 TURNER                   1500 24025300001716.071428571428571428571428571428571428571
 7900 JAMES                     950 24025300001716.071428571428571428571428571428571428571
 7521 WARD                     1250 24025300001716.071428571428571428571428571428571428571
 7902 FORD                     3000 24025300001716.071428571428571428571428571428571428571
 7369 SMITH                     800 24025300001716.071428571428571428571428571428571428571
 7788 SCOTT                    3000 24025300001716.071428571428571428571428571428571428571
 7876 ADAMS                    1100 24025300001716.071428571428571428571428571428571428571
 7934 MILLER                   1300 24025300001716.071428571428571428571428571428571428571
=============================================================================================
```

문제 507.(점심시간 문제) 위의 SQL에 substr을 사용해서 아래와 같이 결과가 출력되게 하시오 !

```
=================================================
EMPNO ENAME      SAL  토탈       최대    최소     평균
----- -------- ----- ----- --------- -------- -------
 7839 KING      5000  29025       5000      800      2073
 7698 BLAKE     2850  29025       5000      800      2073
 7782 CLARK     2450  29025       5000      800      2073
 7566 JONES     2975  29025       5000      800      2073
 7654 MARTIN    1250  29025       5000      800      2073
 7499 ALLEN     1600  29025       5000      800      2073
 7844 TURNER    1500  29025       5000      800      2073
 7900 JAMES      950  29025       5000      800      2073
 7521 WARD      1250  29025       5000      800      2073
 7902 FORD      3000  29025       5000      800      2073
 7369 SMITH      800  29025       5000      800      2073
 7788 SCOTT     3000  29025       5000      800      2073
 7876 ADAMS     1100  29025       5000      800      2073
 7934 MILLER    1300  29025       5000      800      2073
=================================================
```

답1 (문제 506의 답2의 튜닝 후)

```
select empno, ename, sal, substr(an,1 ,5) 토탈,
                          substr(an,6 ,4) 최대,
                          substr(an,10,3) 최소,
                          substr(an,13,4) 평균
  from (select empno, ename, sal,
               (select sum(sal) ||
                       max(sal) ||
                       min(sal) ||
                       avg(sal)
                  from emp) an
          from emp);
```

답2 (문제 506의 답2의 튜닝 후)

```
select empno, ename, sal, trim(substr(an,1 ,10)) 토탈,
                          trim(substr(an,11,10)) 최대,
                          trim(substr(an,21,10)) 최소,
                          trim(substr(an,31,10)) 평균
```

```
  from (select empno, ename, sal,
              (select rpad(sum(sal),10,' ') ||
                      rpad(max(sal),10,' ') ||
                      rpad(min(sal),10,' ') ||
                      round(avg(sal))
                 from emp) an
          from emp);

        0  db block gets
        7  consistent gets
        0  physical reads
```

문제 508. 직업이 SALESMAN인 사원들의
이름, 월급, 직업, 직업이 SALESMAN인 사원들의 최대월급,
직업이 SALESMAN인 사원들의 최소월급,
직업이 SALESMAN인 사원들의 토탈월급,
직업이 SALESMAN인 사원들의 평균월급
을 출력하시오 !

답1

```
select ename, sal, job,
       (select max(sal) from emp where job = 'SALESMAN'),
       (select min(sal) from emp where job = 'SALESMAN'),
       (select sum(sal) from emp where job = 'SALESMAN'),
       (select avg(sal) from emp where job = 'SALESMAN')
  from emp
  where job = 'SALESMAN';

        0  db block gets
       16  consistent gets
        0  physical reads
```

답2

```
select ename, sal, job, max(sal) over (partition by job) 최대,
                        min(sal) over (partition by job) 최소,
                        sum(sal) over (partition by job) 토탈,
                        avg(sal) over (partition by job) 평균
  from emp
  where job = 'SALESMAN';

        0  db block gets
        3  consistent gets
        0  physical reads
```

답3

```
select ename, sal, job, trim(substr(an,1,10)) 최대,
                        trim(substr(an,11,10)) 최소,
                        trim(substr(an,21,10)) 토탈,
                        trim(substr(an,31,10)) 평균
  from (select ename, sal, job,
              (select rpad(max(sal),10,' ') ||
                      rpad(min(sal),10,' ') ||
                      rpad(sum(sal),10,' ') ||
                      round(avg(sal))
                 from emp
                 where job = 'SALESMAN') an
          from emp)
  where job = 'SALESMAN';

        0  db block gets
```

```
          7  consistent gets
          0  physical reads

==========================================================================================
ENAME                      SAL JOB                  최대       최소       토탈       평균
-------------------- ---------- ------------------ ---------- ---------- ---------- ----------
MARTIN                     1250 SALESMAN                 1600       1250       5600       1400
ALLEN                      1600 SALESMAN                 1600       1250       5600       1400
TURNER                     1500 SALESMAN                 1600       1250       5600       1400
WARD                       1250 SALESMAN                 1600       1250       5600       1400
==========================================================================================
```

"

예 : 직업별 인원수가 4명 이상인 직업인 사원들의 이름과 직업을 출력하시오 !

```
select ename, job
  from emp m
  where 4 <= (select count(*)
                from emp s
                where job = m.job);
```

```
=====================================
ENAME                JOB
-------------------- ------------------
MARTIN               SALESMAN
ALLEN                SALESMAN
TURNER               SALESMAN
JAMES                CLERK
WARD                 SALESMAN
SMITH                CLERK
ADAMS                CLERK
MILLER               CLERK
=====================================
```

※ 상호관련 서브쿼리는 main query 부터 실행이 된다.
메인쿼리의 컬럼을 하나씩 서브쿼리에서 읽으면서 서브쿼리가 완성되고
서브쿼리가 완성된 후 메인쿼리가 수행되어 완성되어서 출력이 된다.

문제 509. 우리반에서 나이가 같은 나이인 동료 학생이 한명도 없는 학생들의 이름과 나이를 출력하시오 !

```
select ename, age
  from emp2 m
  where 1 = (select count(*)
               from emp2 s
               where s.age = m.age);
```

```
=========================================================================
ENAME                                                              AGE
---------------------------------------------------------------- ----------
김혜진                                                               23
신선혜                                                               24
허석우                                                               40
안혜진                                                               30
김진철                                                               33
김용식                                                               32
=========================================================================
```

문제 510. 부서 테이블의 부서번호를 출력하는데 사원 테이블에 존재하는 부서번호만 출력하시오 !

```
답1
select deptno
  from dept
  where deptno in (select deptno
                     from emp);

답2
select deptno
  from dept d
  where exists (select 'A'
                  from emp e
                  where e.deptno = d.deptno);

==========
    DEPTNO
----------
        10
        30
        20
==========
```

※ exists 문은 메인 쿼리부터 수행한다.
exists 문은 메인쿼리의 데이터를 서브쿼리에서 찾을때
존재하면 더 이상 찾지 않고 멈춘다.
그래서 검색속도가 빠르다.

문제 511. 아래의 SQL을 튜닝하시오 !

```
튜닝전 :
select distinct 관리자.ename
  from emp 사원, emp 관리자
  where 사원.mgr = 관리자.empno;
      ------------------
      0  db block gets
      6  consistent gets
      0  physical reads

튜닝후 :
select ename
  from emp 관리자
  where exists (select 'A'
                  from emp 사원
                  where 사원.mgr = 관리자.empno);
      ------------------
      0  db block gets
      7  consistent gets
      0  physical reads
```

※ exists가 효과를 보려면 메인쿼리의 테이블이 더 작아야 한다.

```
============================
create table telecom_price
 (telecom_id number(10),
  telecom_name varchar2(10) );

insert into telecom_price
```

```
  values (1, 'sk');
insert into telecom_price
  values (2, 'lg');
insert into telecom_price
  values (3, 'kt');
insert into telecom_price
  values (4, 'cj hello');
==============================
```

문제 512. 우리반 테이블에서 telecom_price 존재하지 않는 통신사가 어떤건지 통신사 이름을 출력하시오 !

```
        select telecom_name
          from telecom_price t
          where not exists (select 'A'
                            from emp2 e
                            where e.telecom = t.telecom_name);

                ====================
                TELECOM_NAME
                --------------------
                cj hello
                ====================
```

```
==================================
create table telecom_price_backup
as
select * from telecom_price;

select * from telecom_price_backup;
==================================
```

문제 513. telecom_price에는 존재하는데 우리반 테이블에는 존재하지 않는 통신사를 telecom_price에서 지우시오 !

```
        delete from telecom_price t
          where not exists (select 'A'
                            from emp2 e
                            where e.telecom = t.telecom_name);

        select * from telecom_price;

                ==============================
                TELECOM_ID TELECOM_NAME
                ---------- --------------------
                         1 sk
                         2 lg
                         3 kt
                ==============================
```

```
=========
rollback;
=========
```

문제 514. 직업, 직업별 토탈월급을 출력하시오 !

```
select job, sum(sal)
  from emp
  group by job;
```

```
=============================
JOB                  SUM(SAL)
------------------ ----------
SALESMAN                 5600
CLERK                    4150
PRESIDENT                5000
MANAGER                  8275
ANALYST                  6000
=============================
```

문제 515. 직업별 토탈월급들의 평균값을 출력하시오 !

내 답

```
select round(avg(sumsal))
  from (select sum(sal) sumsal
          from emp
          group by job);
```

선생님 답

```
select avg(sum(sal))
  from emp
  group by job;
```

```
=============
AVG(SUM(SAL))
-------------
         5805
=============
```

문제 516. 직업, 직업별 토탈월급을 출력하는데 직업별 토탈월급들의 평균값보다 더 큰것만 출력하시오 !

```
select job, sum(sal)
  from emp
  group by job
  having sum(sal) > (select avg( sum(sal) )
                     from emp
                     group by job);
```

```
=============================
JOB                  SUM(SAL)
------------------ ----------
MANAGER                  8275
ANALYST                  6000
=============================
```

※ 동일한 쿼리블럭(직업별 토탈월급) 두번이상 발생 (비슷한 SQL)

문제 518. 위의 SQL을 with절로 변경하시오 !

```
with job_sumsal as (select /*+ inline */ job, sum(sal) 토탈월급
                      from emp
                      group by job)
select job, 토탈월급
  from job_sumsal
  where 토탈월급 > ( select avg(토탈월급)
                       from job_sumsal);
```

```
-------------------------------------------------------------------
| Id  | Operation                  | Name                       |
-------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                            |
|   1 |  TEMP TABLE TRANSFORMATION |                            |
|   2 |   LOAD AS SELECT           | SYS_TEMP_0FD9D6601_8A8F2 |
|   3 |    HASH GROUP BY           |                            |
|   4 |     TABLE ACCESS FULL      | EMP                        |
|*  5 |   VIEW                     |                            |
|   6 |    TABLE ACCESS FULL       | SYS_TEMP_0FD9D6601_8A8F2 |
|   7 |    SORT AGGREGATE          |                            |
|   8 |     VIEW                   |                            |
|   9 |      TABLE ACCESS FULL     | SYS_TEMP_0FD9D6601_8A8F2 |
-------------------------------------------------------------------
```

※ with절 문장은 마치 job_sumsal 이라는 테이블을 하나 만들어 놓은 효과이다.
  효력은 이 문장에서만 발생 나중에 job_sumsal 해도 검색 안됨!(TEMP TABLE TRANSFORMATION)
   왜 이게 더 빠른 SQL이냐 동일한 쿼리블럭을 select 하는 것은 같은것을 두번함
   이 쿼리는 with절에서 한번 select 하기 때문에 빠름 (거의 절반으로 시간 단축)

※ with 절의 유명한 힌트 2가지 ?
1. /*+ inline */ ---> temp 테이블 안만들겠다.
                      (with절이 아니라 그냥 서브쿼리로 수행)
2. /*+ materialize */ ---> temp 테이블을 만들겠다.

문제 519. 아래의 SQL을 with절로 변경하시오 !

(오늘의 마지막 문제)

변경 전 :

```
select d.loc, sum(sal)
  from emp e, dept d
  where e.deptno = d.deptno
  group by d.loc
  having sum(sal) > (select avg( sum(sal) )
                      from emp e, dept d
                      where e.deptno = d.deptno
                      group by d.loc )
  and    sum(sal) > (select sum( sum(sal) )/4
                      from emp e, dept d
                      where e.deptno = d.deptno
                      group by d.loc);
```

```
-------------------
 0  db block gets
18  consistent gets
 0  physical reads
```

변경 후 :

```
with loc_sumsal as(select /*+ inline */ d.loc, sum(sal) sumsal
                     from emp e, dept d
                     where e.deptno = d.deptno
                     group by d.loc)
select loc, sumsal
  from loc_sumsal
```

```
  where sumsal > (select avg(sumsal)
                  from loc_sumsal)
    and sumsal > (select sum(sumsal)/4
                  from loc_sumsal);
```

```
==================================
LOC                          SUMSAL
------------------------- ----------
DALLAS                        10875
==================================
```

```
-------------------
 0  db block gets
18  consistent gets
 0  physical reads
 0  redo size
```

문제 521. 가우스의 공식으로 1부터 100까지의 합을 SQL로 구하시오 !

```
with number100 as (select rownum as rn
                   from dual
                   connect by rownum < 101 )
select sum(rn)
from number100;

with number100 as (select rownum as rn
                   from dual
                   connect by rownum < 101 )
SELECT ( MAX(rn)+min(rn) )*max(rn)/2
  FROM NUMBER100;
```

```
==========================
(MAX(RN)+MIN(RN))*MAX(RN)/2
--------------------------
                      5050
==========================
```

문제 522. 1부터 1억까지 다 더한 숫자의 합을 SQL로 구하시오!

```
select (1+100000000)*(100000000/2) from dual;
```

* with절 힌트

1. inline ---> temp 테이블 사용 안하겠다. (다시 서브쿼리로 바꿔주는 힌트)
   temp 저장공간이 부족한데 with절 남발하면 오히려 오래걸림
2. materialize ---> temp 테이블 사용하겠다.
   temp 저장공간이 다시 충분해 지면 materialize써서 temp저장공간 사용

```
select * from dba_temp_files;
```

```
==========================
(1+100000000)*(100000000/2)
--------------------------
                5.0000E+15
==========================
```

문제 523. 이름에 EN 또는 IN을 포함하고 있는 사원들의 이름과 월급을 출력하시오 !

내가 아는 방법

```
select ename, sal
  from emp
  where ename like '%EN%' or ename like '%IN%';
```

정규 표현식

```
select ename, sal
  from emp
  where regexp_like(ename, 'EN|IN');
```

ENAME	SAL
KING	5000
MARTIN	1250
ALLEN	1600

문제 524. 전공이 심리학과, 통계학과, 프랑스어학과, 물리학과인 학생들의 이름과 전공을 출력하시오 !

내가 알던 방식

```
select ename, major
  from emp2
  where major like '%심리%' or
        major like '%통계%' or
        major like '%프랑스어%' or
        major like '%물리%' ;
```

정규표현식

```
select ename, major
  from emp2
  where regexp_like ( major, '심리|통계|프랑스어|물리' );
```

ENAME	MAJOR
서일	심리학과
엄한솔	프랑스어학과
김준하	정보통계보험수리학과
김건휘	통계학과
주소현	통계학과
임혜진	심리학과
이소진	통계학과
이서영	심리학과
안혜진	통계학과
김진철	물리학과

문제 525. 12c의 hr계정의 테이블에 접근할 수 있는 디비링크를 아래와 같이 생성하시오 !

```
create database link link_12c_hr
connect to hr
identified by hr
using 'localhost:1522/pdborcl';

select * from tab@link_12c_hr;
       ================
       TNAME
       ----------------
       REGIONS
       COUNTRIES
       LOCATIONS
       DEPARTMENTS
       JOBS
       EMPLOYEES
       JOB_HISTORY
       EMP_DETAILS_VIEW
       ================
```

문제 526. 12c의 hr 계정의 모든 테이블들과
12c의 sh 계정의 모든 테이블들을 다 11g xe의 scott 계정에 생성하시오 ! (점심시간문제)

```
================================
create DATABASE LINK link_12c_sh
CONNECT TO sh
IDENTIFIED BY sh
USING 'localhost:1522/pdborcl';

SELECT * FROM tab@link_12c_sh;
================================

  SELECT 'create table '|| tname ||
           ' as
         select * from '|| tname || '@link_12c_sh;'
  FROM tab@link_12c_sh;

create table SALES as
       select * from SALES@link_12c_sh;
create table COSTS as
       select * from COSTS@link_12c_sh;
create table TIMES as
       select * from TIMES@link_12c_sh;
create table PRODUCTS as
       select * from PRODUCTS@link_12c_sh;
create table CHANNELS as
       select * from CHANNELS@link_12c_sh;
create table PROMOTIONS as
       select * from PROMOTIONS@link_12c_sh;
create table CUSTOMERS as
       select * from CUSTOMERS@link_12c_sh;
create table COUNTRIES as
       select * from COUNTRIES@link_12c_sh;
create table SUPPLEMENTARY_DEMOGRAPHICS as
       select * from SUPPLEMENTARY_DEMOGRAPHICS@link_12c_sh;
create table SALES_TRANSACTIONS_EXT as
       select * from SALES_TRANSACTIONS_EXT@link_12c_sh;
create table DR$SUP_TEXT_IDX$I as
```

```
        select * from DR$SUP_TEXT_IDX$I@link_12c_sh;
create table DR$SUP_TEXT_IDX$K as
        select * from DR$SUP_TEXT_IDX$K@link_12c_sh;
create table DR$SUP_TEXT_IDX$R as
        select * from DR$SUP_TEXT_IDX$R@link_12c_sh;
create table DR$SUP_TEXT_IDX$N as
        select * from DR$SUP_TEXT_IDX$N@link_12c_sh;
create table DIMENSION_EXCEPTIONS as
        select * from DIMENSION_EXCEPTIONS@link_12c_sh;
create table PROFITS as
        select * from PROFITS@link_12c_sh;
create table CAL_MONTH_SALES_MV as
        select * from CAL_MONTH_SALES_MV@link_12c_sh;
create table FWEEK_PSCAT_SALES_MV as
        select * from FWEEK_PSCAT_SALES_MV@link_12c_sh;

  SELECT 'create table '|| tname ||
           ' as
        select * from '|| tname || '@link_12c_hr;'
  FROM tab@link_12c_hr;

create table REGIONS as
        select * from REGIONS@link_12c_hr;
create table COUNTRIES as
        select * from COUNTRIES@link_12c_hr;
create table LOCATIONS as
        select * from LOCATIONS@link_12c_hr;
create table DEPARTMENTS as
        select * from DEPARTMENTS@link_12c_hr;
create table JOBS as
        select * from JOBS@link_12c_hr;
create table EMPLOYEES as
        select * from EMPLOYEES@link_12c_hr;
create table JOB_HISTORY as
        select * from JOB_HISTORY@link_12c_hr;
create table EMP_DETAILS_VIEW as
        select * from EMP_DETAILS_VIEW@link_12c_hr;

select count(*) from sales;
```

문제 527. employees 테이블에서 first_name 이 steven인 사원의 first_name, email, phone_number를 출력하시오 !

```
select first_name, email, phone_number
  from employees
  where first_name like 'Steven';
```

```
=====================================
FIRST_NAME      EMAIL   PHONE_NUMBER
----------      ------- ------------
Steven          SKING   515.123.4567
Steven          SMARKLE 650.124.1434
=====================================
```

문제 528. 이름의 첫글자가 St로 시작하면서 끝글자가 en으로 끝나는 사원들의 first_name 을 출력하시오 !

```
select first_name
  from employees
  where first_name like 'St%'
    and first_name like '%en';
```

정규표현식

```
select first_name
  from employees
  where regexp_like ( first_name, '^Ste(.)+en$' );
```

```
==========
FIRST_NAME
----------
Steven
Steven
Stephen
==========
```

※ 설명 (p 273 ~ 274)
- ^ : 문자열의 시작
- $: 문자열의 끝
- ( ) : 괄호로 묶은 식은 한단위로 나타냄
- . : 문자자리 한개
- (.)+ : 한번이상 발생한 수 일치

문제 529. 성씨가 김씨 또는 이씨로 시작하고 이름의 끝글자가 '진'인 학생의 이름을 출력하시오 !

내 답

```
select ename
  from emp2
  where regexp_like ( ename, '^김(.)+진$|^이(.)+진$');
```

선생님 답

```
select ename
  from emp2
  where regexp_like ( ename, '^(김|이)(.)진$');
```

```
==========================================================
insert into emp2(ename) values('김진');

        select ename
          from emp2
          where regexp_like ( ename, '^(김|이)(.)진$');
안나옴
==========================================================
```

문제 529. 이름과 월급을 출력하는데 월급을 출력 할 때 replace함수를 이용해서 숫자 0 을 * 로 출력하시오 !

```
select ename, replace(sal, 0, '*')
  from emp;
```

```
=====================================
ENAME                REPLACE(SAL,0,'*')
-------------------- ------------------
KING                 5***
BLAKE                285*
CLARK                245*
JONES                2975
=====================================
```

문제 530. 이름과 월급을 출력하는데 월급 숫자 0~3까지를 *로 출력하시오 !

```
select ename, regexp_replace(sal, '[0-3]', '*')
  from emp;
```

```
=====================================
ENAME   REGEXP_REPLACE(SAL,'[0-3]','*')
------  -----
KING    5***
BLAKE   *85*
CLARK   *45*
JONES   *975
MARTIN  **5*
=====================================
```

문제 531. 우리반 학생의 이름을 출력하는데 아래와 같이 중간글자가 *로 출력되게 하시오 !

```
=====
ENAME
-----
서*
엄*솔
김*구
김*하
김*휘
이*림
박*균
김*원
장*겸
최*혁
주*현
=====
```

```
select replace ( ename, substr( ename, 2,1 ), '*' )
  from emp2;
```

문제 532. employees 테이블에서 phone_number를 아래와 같이 출력하시오 !

```
============
PHONE_NUMBER
------------
515-123-4567
515-123-4568
515-123-4569
590-423-4567
============
```

```
select regexp_replace ( phone_number , 'W.', '-' )
  from employees;
```

※ 설명 : 백슬래시(W)를 사용하여 일반적으로 메타 문자로 처리되는 문자를 검색합니다.

```
예제 :  SELECT REGEXP_COUNT(
        'ccacctttccctccactcctcacgttctcacctgtaaagcgtccctccctcatccccatgccccttaccctgcag
        ggtagagtaggctagaaaccagagagctccaagctccatctgtggagaggtgccatccttgggctgcagagagaggag
        aatttgccccaaagctgcctgcagagcttcaccacccttagtctcacaaagccttgagttcatagcatttcttgagtt
        ttcaccctgcccagcaggacactgcagcacccaaagggcttcccaggagtagggttgccctcaagaggctcttgggtc
        tgatggccacatcctggaattgttttcaagttgatggtcacagccctgaggcatgtaggggcgtggggatgcgctctg
        ctctgctctcctctcctgaacccctgaaccctctggctaccccagagcacttagagccag',
```

```
'gtc') AS Count
FROM dual;

==========
     COUNT
----------
         4
==========
```

문제 533. 겨울왕국 대본에는 elsa가 몇번 나오는지 regexp_count로 카운트 하시오 !

```
select sum(regexp_count (LOWER(win_text), ' elsa ') ) elsa_cnt
  from winter_kingdom;

==========
  ELSA_CNT
----------
        52
==========
```

문제 533-1. 안철수 연설문에서 국민이라는 단어가 몇번 나오는지 카운트 하시오 !

```
create table ext_pol
(poli varchar2(1000))
organization external
(type oracle_loader
default directory emp_dir2
access PARAMETERS
(records delimited by newline
fields terminated by "/"
(poli))
location ('pol.txt') )
REJECT LIMIT UNLIMITED;


select sum ( regexp_count ( poli, '국민') ) as count
  from ext_pol;

==========
     COUNT
----------
        22
==========
```

문제 534. 우리반 테이블에서 email을 출력하고 그 옆에 email에 @ 가 몇번째 자리에 있는지 출력하시오 !

```
select email, instr(email, '@')
  from emp2;
```

```
==========================================
EMAIL                    INSTR(EMAIL,'@')
-------------------      ----------------
joil4@naver.com                         6
ok7821@naver.com                        7
jungu0519@gmail.com                    10
hjshljy@gmail.com                       8
kuni4210@naver.com                      9
==========================================
```

```
==================================
select * from locations;

select street_address from locations;
==================================
```

문제 535. locations 테이블에 street_address데이터에서 알파벳으로 시작되는 부분의 자리 위치가 어떻게 되는지 확인하시오 !

```
select street_address, regexp_instr ( street_address, '[[:alpha:]]')
  from locations;
```

```
================================
STREET_ADDRESS              시작
----------------------- ---------
1297 Via Cola di Rie            6
93091 Calle della Testa         7
2017 Shinjuku-ku                6
9450 Kamiya-cho                 6
2014 Jabberwocky Rd             6
2011 Interiors Blvd             6
2007 Zagora St                  6
2004 Charade Rd                 6
147 Spadina Ave                 5
================================
```

문제 536. 위의 문제를 다시 해결하는데 알파벳이 아닌 곳의 위치번호를 출력하시오 !

```
select street_address, regexp_instr ( street_address, '[^[:alpha:]]')
  from locations;
```

```
================================
STREET_ADDRESS              시작
-----------------------     ----
1297 Via Cola di Rie            1
93091 Calle della Testa         1
2017 Shinjuku-ku                1
9450 Kamiya-cho                 1
2014 Jabberwocky Rd             1
2011 Interiors Blvd             1
2007 Zagora St                  1
================================
```

문제 537. 우리반 테이블에 이메일에 체크제약을 거는데 이메일에 @가 포함되어 있지 않으면 데이터가 입력되지 못하게 하시오 !

```
alter table emp2
  add constraint emp_email_ck check ( regexp_like ( email , '@' ));

테이블이 변경되었습니다.
```

```
==========================================
create table music( m_name varchar2(2000) );
==========================================
```

문제 538. 김용신 라디오 선곡표에서 아래와 같이 가수명만 잘라서 출력하시오 !

```
select m_name, substr ( m_name, instr(m_name, '-'), 40)
  from music
  WHERE INSTR ( m_name, '-') != 0;

select m_name, regexp_substr ( m_name, '[^-]+', 1, 2) 가수
  from music
  WHERE INSTR ( m_name, '-') != 0;
```

```
================================================================================
M_NAME                                                  가수
------------------------------------------------------  ------------------
 1. Raindrops Keep Fallin' On My Head - B.J. Thomas     B.J. Thomas
 2. Java Jive - Manhattan Transfer                      Manhattan Transfer
 3. Let There Be Love - Laura Fygi                      Laura Fygi
 4. Don't You Want Me - The Human League                The Human League
 5. Don`t Go - Yazoo                                    Yazoo
 6. At Seventeen - Janis Ian                            Janis Ian
 7. How Can I Tell Her - Lobo                           Lobo
================================================================================
```

문제 539. 김용신 라디오 선곡표에서 가장 많이 나오는 가수명과 건수와 순위를 출력하시오 !

```
CREATE VIEW music1
as
select m_name, trim(regexp_substr ( m_name, '[^-]+', 1, 2)) 가수
          from music
          WHERE INSTR ( m_name, '-') != 0;

select 가수, count(*), dense_rank() over (order by count(*) desc) 순위
  from music1
  group by 가수;
```

```
=================================
가수             COUNT(*)     순위
---------------  --------     ----
ABBA                   50        1
The Carpenters         23        2
Boney M                23        2
Michael Jackson        21        3
Bee Gees               20        4
Queen                  20        4
=================================
```

문제 540. cbs 라디오 김용신 노래 선곡표에서 가장 많이 나오는 노래명과 건수와 순위를 출력하시오 !

```
select 노래가수, count(*), dense_rank() over (order by count(*) desc) 순위
  from (select m_name, trim(substr( m_name, regexp_instr(m_name, '[[:alpha:]]'))) 노래가수
          from music
          WHERE INSTR ( m_name, '-') != 0)
  group by 노래가수;
```

노래가수	COUNT(*)	순위
Reality - Richard Sanderson	4	1
Juliet - Robin Gibb	4	1
Physical - Olivia Newton John	4	1
Funkytown - Lipps Inc.	4	1
Java Jive - Manhattan Transfer	4	1
Shake It Up - Cars	3	2
You're Still The One - Shania Twain	3	2