

##1020 1021

2020년 10월 20일 화요일 오후 5:34

1021

1. 오라클 설치시 주의사항

- 컨테이너 database 생성 체크박스를 반드시 해제할 것
- 시작시 폴더 2개를 생성 후 설치시작
- oracle2=오라클 설치파일 / oracle3=오라클이 설치될 디렉토리

2. SQL이 무엇인지? : 구조적 질의 언어 : database에서 데이터를 검색하고 조작하는 언어

- Database의 종류 : oracle, mssql, mysql, postgresSQL, Maria db
- mssql : 소셜커머스, 게임회사, : 데이터를 빠르게 생성, 제거
- mysql : 오라클 하부 : 가장 중요한 데이터는 oracle에 저장하고,
- 상대적으로 덜 중요한 데이터는 mysql, postgresSQL, Maria db에 저장한다
ex) 아프리카 채팅창, 방송채팅창 등

3. SQL을 왜 배우는지? : 의미있는 정보를 얻어내는 검색

- 원하는 정보를 얻기 위해서
- 고객의 needs를 파악
- 물류회사면 재고현황 파악
- 게임회사면 게임의 흥행 여부에 대한 원인 파악

4. SQL의 종류

- Query
- DML문
- DDL문
- DCL문
- TCL문

5. 기본 SELECT문의 종류

- 001 테이블에서 특정 열(column) 선택하기
- select ename, sal from emp;
- 해석 : select(검색/조회/선택), ename(이름), sal(월급), from(~참조), emp(테이블 이름) ;(실행해라)
여기서 ename, sal 등은 컬럼(특정 열)이다.
- 여기서부터 1021내용학습 / 1020자료 날아감 π

1021

1. SQL을 왜 배워야 하는가?

- 1) 데이터를 검색하는데 파이썬의 판다스를 이용해도 되는데 왜 굳이 SQL을 해야하는가?
- 1-2) 답 : 회사의 중요한 데이터는 다 database에 저장되어 있고 대용량 데이터라서 대용량 데이터에서 데이터를 검색하는 검색시간이 오래걸리기 때문. 따라서 생각없이 SQL을 작성할 경우 검색시간이 너무 오래 걸려서 빠르게 결과를 볼 수

없음. 그래서 항상 검색시간을 고려해서 SQL을 작성해야 한다. 그런데 oracle은 버전이 업그레이드 될수록 점점 인공지능화 되어지고 있어 SQL을 자체적으로 튜닝을 한다. 튜닝:SQL검색이 더 원활하게 돌아가도록 자체적 수정하는것

2. SQL을 실습 위주로 배우는 이유?

- 1) 일머리를 배우기 위해서이다. 공부머리와 일머리가 있는데, 여러 자격증을 취득하고 책을 읽고, 자격증이 있어도 업무의 효율과는 큰 상관관계가 없다. 일머리는 본인이 직접 SQL을 작성해서 데이터를 검색해 낼 수 있어야 한다.
- 2) 일머리 = 커뮤니케이션 능력도 포함된다

3. 1021복습

- 1) 기본 select문의 4개의 절
 - select 컬럼명(ename, sal 등)
 - from 테이블명(emp 등)
 - where 검색조건
 - order by 정렬할 컬럼명
- 2) 코딩순서 : select, from, where, order by
- 3) 실행순서 : from, where, select, order by
- 4) 연결 연산자 : || : 문자열로 결과를 출력하는 일이 종종 있기 때문에 사
ex)김인수님의 통장 총 잔고는 39,000 원 입니다.
- 5) 컬럼 별칭 : 결과로 출력될 컬럼의 이름을 변경하고 싶을 때 사용
ex) ename을 '이름' 이라고 나오게 하고 싶을 때
>>ename as 이름
- 6) 비교연산자 : >, <, >=, <=, !=, <>, ^=
- 6-2) 기타 비교연산자 (빨간색은 1022진도)
 - between A and B
 - like
 - is null
 - in

##1022

2020년 10월 22일 목요일 오전 10:17

1021진도 이어서

1. 기타 비교연산자 : between A and B, like, is null, in

1) like 비교연산자의 키워드 2가지

- % : 와일드카드 : 해당 자리에 뭐가 와도 상관없고 자릿수는 제한X
- _ : 언더바 : 해당 자리에 뭐가 와도 상관없고 자릿수는 1개

★

37) 이름의 첫 번째 철자가 A로 시작하는 사원들의 이름과 월급을 출력하시오!

```
select ename, sal
```

```
from emp
```

```
where ename like 'A%';<<
```

※ 위 %가 특수문자가 아닌 기타비교연산자 중에 하나인 like 를 사용해야 한다
만약 like가 아닌 '='을 사용할 경우 이름이 'A%'인 사원을 뽑는 SQL이 되어버린다

38) 이름이 SCOTT인 사원의 이름과 월급과 직업을 출력하시오!

```
select ename, sal, job
```

```
from emp
```

```
where ename = 'SCOTT';<<
```

39) 이름의 끝에서 두 번째 철자가 T인 사원들의 이름과 월급을 출력하시오!

```
select ename, sal
```

```
from emp
```

```
where ename like '%T_';<<
```

013 비교 연산자 배우기 4(IS NULL)

1. NULL 값을 조회할 때 사용하는 연산자가 is null 입니다.

1) Null : 데이터가 없는 상태 또는 알 수 없는 값(unknown)

2) 예제 : 이름과 월급과 커미션과 직업을 출력하시오!

```
>>select ename, sal, comm, job
```

```
from emp;<<
```

2-2) SALESMAN을 제외한 job을 갖고 있는 전원이 comm은 null(없음)

3) comm = null 로는 조회할 수 없다. 왜냐하면 null은 알 수 없는 값이므로 비교 연산자인 = 으로는 조회할 수 없다. 기타 비교 연산자인 is null 을 사용하여야 함(40P에 대한 설명)

★P

40) 커미션이 null인 직원들의 이름과 커미션을 출력하시오

```
select ename, comm  
from emp  
where comm is null;<<
```

※이때, 'is' 대신 '=' 을 사용하면 결과값이 출력되지 않는다.

41) 커미션이 null 이 아닌 직원들의 이름과 커미션을 출력하시오!

```
select ename, comm  
from emp  
where comm is not null;<<
```

42) mgr(관리자의 직원번호)이 null 인 직원의 이름과 직업을 출력하시오!

```
select ename, job  
from emp  
where mgr is null;<<
```

※ KING(PRESEIDENT=사장)은 관리자 번호(mgr)가 없음

43) 직원번호, 이름, 관리자 번호(mgr) 을 출력하시오

```
select empno, ename, mgr  
from emp;<<
```

014 비교 연산자 배우기 5(IN)

1. 개념 : where 절의 검색조건에서 여러개의 행을 비교할 때는 in을 사용

1) 예제 : 직원번호가 7788 7902인 직원들의 직원번호와 이름을 조회하시오

```
select empno, ename  
from emp  
where empno in (7788, 7902);<<
```

1-2) 설명 : = 연산자는 하나의 값만 비교 가능하다. 여러개의 값을 비교할 경우 in을 사용해야 한다.

★P

44) 직업이 SALESMAN, ANALYST 인 직원들의 이름과 직업을 출력하시오!

```
select ename, job  
from emp  
where job in ('SALESMAN', 'ANALYST');<<
```

※ 문자열의 경우 싱글 쿼테이션 마크(')로 둘러주어야 정상출력됨

45) 직업이 SALESMAN, ANALYST 가 아닌 직원들의 이름과 직업을 출력하시오!

```
select ename, job
```

```
from emp
```

```
where job not in ('SALESMAN', 'ANALYST');<<
```

※ 부정문은 is 와 달리 in의 경우 in 앞에 작성해 준다(영어문법구조와 동일함)

//Q1) in 비교연산자와 같이 like 절도 두 개의 조건? 으로 검색을 하고 싶을 땐 어떻게 해야할까?

```
select ename, job
```

```
from emp
```

```
where job like ('SALES%', 'AN%')<< 이렇게 하면 결과도출 안됨.
```

in 비교연산자를 사용하면 'XX' 을 포함한 부분만 도출할 수 없음. 여기서 XX 는 column에 포함된 완전한 data가 아닌 그 데이터의 일부분임

답변 : 이때는 3행에 where job like 'SALES%' and job like 'AN%'; 을 써서 결과값 출력이 가능하다 (and, or)

46) 직업이 SALESMAN 이 아닌 직원들의 이름과 월급과 직업을 출력하시오!

```
select ename, sal, job
```

```
from emp
```

```
where job not in 'SALESMAN';<<
```

※not in 대신 != (연산자)도 가능함

Q2) 복수도 가능할까?

```
select ename, sal, job
```

```
from emp
```

```
where job not in ('SALESMAN', 'CLERK');<<
```

이렇게 출력하면 복수작업도 가능하다. = 연산자 대신 in() 을 사용하면 복수항에 대한 검색조건이 가능한 것을 확인할 수 있다.

47) 위의 결과를 다시 출력하는데 월급이 높은 직원부터 출력하시오!

```
select ename, sal, job
```

```
from emp
```

```
where job != 'SALESMAN'
```

```
order by sal desc;<<
```

※ scott 창에서 emp와 dept 테이블 사용하기

emp12 테이블을 만든 scott 유저 창에서 emp와 dept 테이블을 생성

- script를 다 복사

- developer (scott)에 복사 후 실행

48) 우리반 테이블에서 이름과 나이를 출력하는데 나이가 높은 학생부터 출력하시오!

```
select ename, age
```

```
from emp12
```

```
order by age desc;<<
```

49) 이름과 나이와 주소를 출력하는데 30살 이상인 학생들만 출력하시오!

```
select ename, age, address
from emp12
where age >= 30;<<
```

50) 성씨가 김씨인 학생들의 이름과 통신사를 출력하시오!

```
select ename, telecom
from emp12
where ename like '김%';<<
```

51) 전공에 통계를 포함하고 있는 학생들의 이름과 전공을 출력하시오

```
select ename, major
from emp12
where major like '%통계%';<<
```

※ like 연산자를 사용할 때 특정 단어를 포함하고 있는 데이터를 검색하려면 %XX% 커맨드를 입력하면 됨. 와일드카드(%) 응용

52) 우리반에 gmail 을 사용하는 학생들의 이름과 메일을 출력하시오!

```
select ename, email
from emp12
where email like '%gmail%';<<
```

53) 나이가 27 에서 34 사이인 학생들의 이름과 나이를 출력하시오!

```
select ename, age
from emp12
where age between 27 and 34;<<
```

2>>select ename, age

```
from emp12
where age >= 27 and age <= 34;<<
```

54) 나이가 27에서 34 사이가 아닌 학생들의 이름과 나이를 출력하시오!

```
select ename, age
from emp12
where age not between 27 and 34;<<
```

55) 주소가 경기도인 학생들의 이름과 나이와 주소를 출력하시오!

```
select ename, age, address
from emp12
```

where address like '%경기도%';<<

56) 통신사가 sk, lg 인 학생들의 이름과 통신사를 출력하시오!

```
select ename, telecom
from emp12
where telecom in ('sk', 'lg');<<
```

57) 서울에서 사는 학생들의 이름과 나이와 전공을 출력하는데 나이가 높은 학생부터 출력하시오!

```
select ename, age, major
from emp12
where address like '%서울%'
order by age desc;<<
```

58) 이메일이 gmail 이 아닌 학생들의 이름과 이메일을 출력하시오!

```
select ename, email
from emp12
where email not like '%gmail%';<<
```

59) 아래와 같이 결과가 출력되게 하시오! (나이는 높은 순서대로 출력하시오)

김주원 학생의 나이는 44세 입니다.

권세원 학생의 나이는 36에 입니다.

...

```
select ename || ' 학생의 나이는 ' || age || '세 입니다.'
```

```
from emp12
```

```
order by age desc;<<
```

※ 이때 1행에 as 'XX' 를 추가하면 XX coloumn 명으로 결과출력 가능함

015 논리 연산자 배우기(AND, OR, NOT)

1. 오라클 연산자의 종류 3가지

1) 산술 연산자 : *, / , + , -

2) 비교 연산자 : >, <, >=, <=, =, !=, <>, ^=

기타 비교연산자 : between A and B

like

is null

in

3) 논리연산자 : and, or, not

2. 예제

1) 직업이 SALESMAN 이고 월급이 1200 이상인 직원들의 이름과 월급과 직업을 출력하시오!

```
select ename, sal, job
```

```
from emp
```

```
where job = 'SALESMAN' and sal >= 1200;<<
```

1-2) A and B 에서 A가 아닌 AA로 검색할 경우 false 로 판명됨

A를 그대로 검색한 경우 true 로 판명

1-3) 위 예제의 경우 둘 다 true 이므로 결과 출력됨

2)직업이 SALESMAN 이거나 월급이 1200 이상인 직원들의 이름과 월급과 직업을 출력하시오!

```
select ename, sal, job
```

```
from emp
```

```
where job = 'SALESMAN' or sal >= 1200;<<
```

3. 논리연산자

1) and

- true and true 면 true 여서 결과 출력됨

- false and true / true and false 면 false 이므로 결과 출력X

2) or

- false or true / true or false 면 true 이므로 결과 가 출력됨

3)not

- not like

- not in

- is not null

- not between A and B

★P

60) 직업이 SALESMAN 이거나 또는 ANALYST 인 직원들의 이름과 월급과 직업을 출력하시오!

```
select ename, sal, job
```

```
from emp
```

```
where job = 'SALESMAN' or job = 'ANALYST';<<
```

※in 구문으로 표현하면..

```
select ename, sal, job
```

```
from emp
```

```
where job in ('SALESMAN', 'ANALYST');<<
```

61) 성이 김씨, 이씨가 아닌 학생들의 이름을 출력하시오!

```
select ename
```

```
from emp12
```

```
where ename not like '김%' and ename not like '이%';<<
```

62) 이메일 gmail 과 naver 가 아닌 학생들의 이름과 이메일을 출력하시오!

```
select ename, email
```

```
from emp12
```


where email not like '%gmail%' and email not like '%naver%';<<

016 대소문자 변환 함수 배우기(UPPER, LOWER, INITCAP)

1. 정의 : 함수(function)는 어떤 특정 기능을 구현해 놓은 코드의 집합

1) 입력값 -- 함수 -- 출력값

2. 함수사용이유 : 함수를 이용하면 좀 더 복잡한 데이터 검색이 가능

1) ex) 영화 겨울왕국에는 elsa 가 많이 나올까 anna 가 많이나올까?

2) 우리반 학생들이 제일 많이 쓰는 통신사가 어디인가?

3. 함수의 종류(2가지)

1) 단일행 함수

- 문자함수 : upper, lower, initcap

- 숫자함수

- 날짜함수

- 변환함수

- 일반함수

2) 복수행 함수

- max, min, avg, sum, count, var, stddev

4. 단일행 -- 문자함수

1) upper : 대문자로 출력하는 함수

2) lower : 소문자로 출력하는 함수

3) initcap : 첫 번째 철자는 대문자로 출력하고 나머지는 다 소문자로 출력

5. 예제

1) select upper(ename), lower(ename), initcap(ename)
from emp;

★P

63)☆ 우리반 테이블에서 통신사가 sk 와 관련된 통신사이면 그 학생의 이름과 통신사를 출력하시오! 단, 정확하게 데이터가 출력되어야 함

- 대문자로 출력

select ename, upper(telecom)

from emp12

where upper(telecom) like '%SK%';<<

- 소문자로 출력

```
select ename, lower(telecom)
from emp12
where lower(telecom) like '%sk%';<<
```

※ telecom 내부의 데이터를 upper(telecom)으로 전부 대문자로 변경한 뒤
3행에서 upper(telecom)을 새로운 coulmn으로 지정하여 결과출력한 것

Qe : 63번 문제에서 원본 그대로 결과값을 산출할수 있을까? / 노가다 제외
ex) 결과값에 telecom 이 (sk, skt, SK, Sk, SkT 등)

017 문자에서 특정 철자 추출하기(SUBSTR)

1. substr

1) 예제 : select ename, substr(ename, 1, 1)
from emp12;

2) 정의 : substr(A, n1, n2) : A column 에서 data의 n1 부터 n2 개를 출력
n1 은 좌표, n2 는 자리수

3) 예제2 : 성씨가 이씨인 학생들의 이름을 출력하시오!

```
select ename, age
from emp12
where substr(ename,1,1) in ('이');<<
```

4) 참조 : substr 은 select 절, where절 다 사용가능

★P

64)☆ 성씨가 김, 이, 유 씨인 학생들의 이름과 나이를 출력하는데, like 쓰지 말고 in 과 subsrt 을 이용하여 출력하시오

```
select ename, age
from emp12
where substr(ename, 1, 1) in ('김', '이', '유');<<
```

※ select 절을 이용한 해답

```
MariaDB [orcl]> select ename, age
-> from emp12
-> where ename like '김%' or ename like '이%' or ename like '유%';
```

018 문자열의 길이를 출력하기(LENGTH)

1. 정의 : 철자의 길이를 출력하는 함수

```
1) ex) select ename, length(ename)
      from emp;
```

★P

65) 우리반 테이블에서 이메일과 이메일 철자의 길이를 출력하는데 이메일 철자의 길이가 가장 긴 것부터 출력되도록 하시오!

```
select email, length(email)
      from emp12
      order by length(email) desc;<<
```

66) emp 테이블에서 ename 을 출력하고, 그 옆에 ename 의 첫 번째 철자를 출력하시오!

```
select ename, substr(ename, 1,1)
      from emp;<<
```

67) 위 결과를 다시 출력하는데, 이름에 첫번째 철자로 출력되는 부분을 소문자로 출력하시오!

```
select ename, lower(substr(ename, 1,1))
      from emp;<<
```

※ select ename, substr(lower(ename),1,1)
 from emp; << 이렇게 해도 결과값은 동일하게 출력된다

68) ☆ ★ 아래의 결과를 initcap 쓰지 말고 upper, lower, substr, || 를 사용해서 출력하시오! 아래의 결과에 해당하는 SQL : select initcap(ename)

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

※ substr 함수에서 substr(XX, n1, n2) 중 n2 값을 지정하지 않으면 해당 data의 끝 자리까지 추출된다. 단, 가운데 값은 비울 수 없음. 이때 각각의 XX, n1, n2 는 인자라고 통칭함.

##1021

2020년 10월 21일 수요일 오전 10:07

001 테이블에서 특정 열(COLUMN) 선택하기

1. emp(사원) 테이블 컬럼 소개 (해당 아래 내용들은 다 COLUMN(컬럼)에 해당함)

- empno : 사원번호
- ename : 사원이름
- sal : 월급
- job : 직업
- comm : 커미션
- mgr : 관리자 번호
- hiredate : 입사일
- deptno : 부서번호

2. SQL 작성시 주의사항

- SQL은 대소문자는 구분하지 않는다

ex) SELECT ENAME, SAL FROM EMP; = select ename, sal from emp; // 모두 동일한 출력값을 지닌다

- SQL은 한줄로 작성하지 말고 절은 다음 라인으로 분리해서 작성한다.

ex) select절, from절 (엔터치면 '2'라는 숫자가 나오는데 거기서부터 그대로 작성하면 동일한 결과 출력)

- 들여쓰기를 사용해서 SQL의 가독성을 높인다 ('tab'키를 사용하면 편리(혹은 간격이 일정하니 이쁠듯))

ex) select ename, sal

from emp;

where

group by

having

order by

★SQL연습문제(위 "a"참조) / 앞으로 연습문제 항목은 앞에 ★로 관리(따로 문서서식 양식에 따르지 않는다)

3) emp테이블의 컬럼이 무엇무엇이 있는지 확인하시오!

>> describe emp

4) 이름, 월급, 커미션을 출력하시오!

>> select ename, sal, comm from emp;

5) 이름, 직업, 입사일, 부서번호를 출력하시오! (들여쓰기, SQL절 분리해서 작성)

>> select ename, hiredate, deptno

from emp;

6) emp 테이블에 모든 컬럼을 검색하시오!

>>select empno, ename, sal, job, comm, mgr, hiredate, deptno

2 from emp;

이때, "set lines 300" : 결과 화면에서 가로 사이즈 조절하는 명령어

"set pages 400" : 결과 화면에서 세로 사이즈 조절하는 명령어

위 두 명령어는 해당 창에서만 유효하며, 단순 명령어이고 SQL은 아니다

002 테이블에서 모든 열(COLUMN) 출력하기

1. 명령어

```
select *
```

```
from emp;
```

- * : asterisk라고 하며 모든 컬럼을 다 선택할 때 사용함

★exercise

7) dept 테이블의 모든 컬럼을 출력하시오

```
>> select*
```

```
from dept;
```

※ dept는 부서 테이블이고 컬럼(deptno, dname, loc)은 3개가 있으며 다음과 같다

- deptno : 부서번호

- dname : 부서명

- loc : 부서위치

003 컬럼 별칭을 사용하여 출력되는 컬럼명 변경하기

1. 컬럼별칭은 컬럼명 대신에 다른 컬럼명을 지정할 때 사용하는 문법

- ex) select ename as 이름, sal as 월급

```
from emp;
```

- 컬럼명 뒤에 as 를 쓰고 컬럼별칭을 작성

- 결과출력시 컬럼별칭이 출력

- as는 생략 가능

- 원본의 결과값은 변경 x

★practice

8) 이름과 입사일과 부서번호를 출력하는데 컬럼명이 한글로

이름, 입사일, 부서번호로 출력되게 하시오!

```
>>select ename as 이름, hiredate as 입사일, deptno as 부서번호
```

```
2 from emp;
```

9) 이름과 월급을 출력하는데 컬럼명이 아래와 같이 출력되게 하시오

Employee name, Salary (대소문자 구분)

```
>>select ename as "Employee name", sal as "Salary"
```

```
2 from emp;
```

※ 컬럼별칭에 대소문자를 구분하고 싶거나 공백 문자나 특수문자를 넣으려면 양쪽에 더블 쿼테이션 마크를 둘러 주어야 함("")

004 연결 연산자 사용하기(||)

1. 연결 연산자는 두 컬럼의 데이터를 연결해서 출력하는 연산자

- ex) select ename || sal
from emp;
- 연결연산자 '||'는 엔터키 위 원화표시(shift) 누른 후 입력가능
- ex) select ename || '의 월급은 ' || sal
from emp;
- **ed : error** 발생시 'ed'명령어를 입력하여 메모장을 불러와 오류가 난 부분을 수정하여 저장 후 닫기
후 **/+enter**키를 누르면 다시 SQL명령어 수행 가능

★exercise

- 10) 아래와 같이 결과를 출력하시오!
KING의 직업은 PRESIDENT 입니다.
SCOTT의 직업은 ANALYST 입니다.
... 14명이 나오게 출력하시오
>>select ename ||'의 직업은'|| job ||' 입니다.'
2 from emp;

11) 위의 결과에서 출력되는 컬럼명을 사원정보라는 한글로 출력되게 하시오!

- >> select ename ||'의 직업은 ' ||job||'입니다.' as 사원정보
2 from emp;

005 중복된 데이터를 제거해서 출력하기(DISTINCT)

1. **distinct** 키워드를 컬럼명 앞에 작성하고 실행하면 중복된 데이터를 제거하고 출력할 수 있다

- ex) select job
from emp; >>중복데이터 출력됨

select distinct job
from emp; >>중복된 데이터 삭제완료

★practice

- 12)부서번호를 출력하는데 중복을 제거해서 출력하시오
>>select distinct deptno
2 from emp;

2. coloum 두 개를 쓰면 중복제거가 되지 않음

- ex) select distinct deptno, sal, ename
from emp; >>제거안됨

006 데이터를 정렬해서 출력하기(ORDER BY)

1. order by 절은 데이터를 정렬하는 절이고 select 문장에 맨 마지막에 기술함

- ex)select ename, sal

2 from emp

3 order by sal asc;

정렬할 컬럼명 정렬방법 : asc : 낮은값부터 높은값 순으로(오름차순)

desc : 높은값부터 낮은값 순으로(내림차순)

★pracrice

13) 이름과 월급을 출력하는데 월급이 높은 사원부터 출력하시오!

>>select ename, sal

2 from emp

3 order by sal desc;

14) 이름과 입사일을 출력하는데 최근에 입사한 사원부터 출력하시오!

>>select ename, hiredate

2 from emp

3 order by hiredate desc;

☆점심시간문제(카페에 답글 후 검사받고 식사)

- 이름과 월급과 부서번호를 출력하는데 부서번호가 10번, 20번, 30번 순으로 출력되게하고, 컬럼명이 한글로 이름, 월급, 부서번호로 출력되게 하시오!

>>select ename as 이름, sal as 월급, deptno as 부서번호

2 from emp

3 order by deptno asc;

2. order by 절에 컬럼을 여러개를 작성할 수 있다.

- ex) select ename, deptno, sal

from emp

order by deptno asc, sal desc;

- ex) select ename, deptno, sal

from emp

order by 2 asc, 3 desc; >>해도 동일한 결과값 출력된다

★practice

15) 이름과 직업과 입사일을 출력하는데 직업은 ABC 순으로 정렬해서 출력하고,

직업을 ABCD 순으로 정렬한 것을 기준으로 입사일을 먼저 입사한 사원부터 출력되게 하시오!

>>select ename, job, hiredate

2 from emp

3 order by 2 asc, 3 asc;<<

007 WHERE절 배우기 1(숫자 데이터 검색)

1. "where 절을 사용하면 특정 조건에 대한 데이터만 선별해서 출력할 수 있다"

- ex) select ename, sal
 from emp
 where sal = 3000;

★practice

16) 사원번호가 7788번인 사원의 사원번호와 이름과 월급을 출력하시오!

```
>>select empno, ename, sal  
2        from emp  
3            where empno = 7788;<<
```

17) 30번 부서번호에서 근무하는 사원들의 이름과 월급과 부서번호를 출력하시오!

```
>>select ename, sal, deptno  
2        from emp  
3            where deptno = 30;<<
```

※'where'은 'order by'와 달리 숫자 넘버로 지칭되지 않는다

18) 위의 결과를 다시 출력하는데 월급이 높은 사원부터 출력하시오!

```
>>select ename, sal, deptno  
2        from emp  
3            where deptno = 30  
4            order by 3 desc;<<
```

※절의 순서 : select > from > where > order by

19) 부서번호가 20번인 사원들의 이름과 입사일과 부서번호를 출력하는데 최근에 입사한 사원부터 출력하시오!

```
>>select ename, hiredate, deptno  
2        from emp  
3            where deptno = 20  
4            order by 2 desc<<
```

008 WHERE절 배우기 2(문자와 날짜 검색)

1. where 절로 데이터를 검색할 때 숫자와는 다르게 문자는 양쪽에 싱글 쿼테이션 마크를 둘러줘야 합니다.

- ex) select ename, sal
 from emp
 where ename = 'SCOTT';

- SQL은 대소문자를 구분하지 않으나 데이터는 대소문자를 구분합니다
- 파이선과 다르게 싱글 쿼테이션 마크만인 경우 정상적으로 값을 출력한다
- **싱글 쿼테이션(') 마크 안에 있는 데이터는 문자 또는 날짜이다 라고 oracle에게 인식시키는 것**

★practice

20) 직업이 SALESMAN 인 직원들의 이름과 월급과 직업을 출력하시오!

```
>>select ename, sal, job
2   from emp
3   where job = 'SALESMAN';<<
```

21) 81년 11월 17일에 입사한 직원의 이름과 입사일을 출력하시오

```
>>select ename, hiredate
2   from emp
3   where hiredate = '81/11/17';<<
```

※날짜검색을 할 때는 년도/월/일 순으로 검색을 하면 되는데,
나라마다 순서가 다를 수 있다. 서구권은 일/월/년도 순이다

009 산술 연산자 배우기(*, /, +, -)

1. 예제

- ex) select ename, sal, sal+3000
from emp;

★practice

22) 이름과 연봉(sal*12)을 출력하는데 컬럼명을 한글로 이름, 연봉으로 출력하시오!

```
>>select ename 이름, sal*12 연봉
2   from emp;<<
※as는 생략 가능함
```

23) 위의 결과를 다시 출력하는데 연봉이 36000 인 직원들의 이름과 연봉을 출력하시오!

```
>>select ename 이름, sal*12 연봉
2   from emp
3   where sal*12 = 36000<<
```

※3행의 sal*12을 '연봉'으로 작성하면 err가 발생하는데, 그 이유는 실행 순서 때문이다.

- oracle 내부 실행순서는 [from, where, select]
- 코딩순서는 [select, from, where]
- 'order by'절은 항상 맨 마지막 순서이다

24) 이름과 연봉을 출력하는데 연봉이 높은 직원부터 출력하시오!

```
>>select ename, sal*12 연봉
2   from emp
3   order by '연봉' desc;
```

※원래 '연봉'대신 sal*12 을 넣어도 나오지만 order by절은 항상 마지막 순서이므로 연봉도 가능함
산술연산자는 항상 계산에 유의해야함 ex)금융권은 잘못된 SQL로 인한 피해도 크다

25) 아래의 SQL에서 더하기가 먼저 실행되게 하시오

```
select ename, sal+200*2
      from emp;
>>select ename, (sal+200)*2
      from emp;<<
```

010 비교 연산자 배우기 1(> , < , >= , <= , = , != , <> , ^=)

1. (> , < , >= , <= , = , != , <> , ^=)

같지 않다(3개 다 동일한 의미)

★practice

26) 커미션이 150 이상인 직원들의 이름과 커미션을 출력하시오!

```
>>select ename, comm
      2   from emp
      3   where comm >= 150;<<
```

27) 월급이 3000 이상인 직원들의 이름과 월급을 출력하시오!

```
>>select ename, sal
      2   from emp
      3   where sal >= 3000;<<
```

28) 직업이 SALESMAN이 아닌 직원들의 이름과 직업을 출력하시오!

```
>>select ename, job
      2   from emp
      3   where job != 'SALESMAN';<<
```

※더블 쿼테이션 마크(")는 oracle전체에서 딱 하나의 케이스에서만 사용하는데, 컬럼별칭 사용할때 공백문자, 특수문자, 대소문자를 구분해서 컬럼별칭을 출력하고자 할때만 사용한다. 그 이외의 경우는 싱글 쿼테이션 마크(')를 사용한다.

29) 월급이 2400 이하인 직원들의 이름과 월급을 출력하는데 월급이 높은 직원부터 출력하시오!

```
>>select ename, sal
      2   from emp
      3   where sal <= 2400
      4   order by sal desc;<<
```

30) 월급이 1000 에서 3000 사이인 직원들의 이름과 월급을 출력하시오!

```
>>select ename, sal
```

```
2   from emp
3   where sal between 1000 and 3000;<<
```

※이때, 3행은 [where sal >= 1000 and sal <= 3000;] 로 대체할 수 있다

31) 월급이 1000에서 3000 사이가 아닌 직원들의 이름과 월급을 출력하시오!

```
>>select ename, sal
2   from emp
3   where sal not between 1000 and 3000;<<
```

32) 1981년도에 입사한 직원들의 이름과 입사일을 출력하시오

```
>>select ename, hiredate
2   from emp
3   where hiredate between '81/1/1' and '81/12/31';<<
```

012 비교 연산자 배우기 3(LIKE)

1. 예제

- 이름의 첫글자가 S로 시작하는 직원들의 이름을 출력하시오!
- select ename
- 2 from emp
- 3 where ename like 'S%';
- like : ~일 것 같은 이라는 영어 뜻처럼 이름의 첫 번째 철자가 S로 시작할 것 같은과 같은 의미
- % : 와일드 카드로, 이 자리에 뭐가 와도 관계없다. 그 철자의 개수가 몇 개이든 상관 없다는 의미
- 또한, %가 위에서 언급한 역할을 하려면 'like'와 함께 쓰여야 한다.

★P

33) 이름의 끝 글자가 T로 끝나는 직원들의 이름을 출력하시오!

```
>>select ename
2   from emp
3   where ename like '%T';<<
```

34) 81년도에 입사한 직원들의 이름과 입사일을 출력하는데 between and 사용하지 말고 like로 하시오!

```
>>select ename, hiredate
2   from emp
3   where hiredate like '81%';<<
```

35) 이름의 두 번째 철자가 M인 직원들의 이름을 출력하시오!

```
>>select ename
   from emp
   where ename like '_M%';<<
```

※like와 같이 쓸 수 있는 키워드가 2개가 있음

- % : 와일드카드 : 이 자리에 어떤것이나, 수량에 관계없음
- _ : 언더바 : 이 자리에 어떤 것이 와도 관계없으나 자릿수는 하나이다(%와 차이점)

36) 이름의 세 번째 철자가 A 인 사원들의 이름을 출력하시오!

```
>>select ename
```

```
from emp
```

```
where ename like ' __A%';<<
```

##1023

2020년 10월 23일 금요일 오전 9:34

1. 1022 이어서 / substr 함수의 응용예제

1) 예제 : select substr('smith', 1, 3)

from dual;

1-2) dual : select 절에 있는 함수의 값을 보기위한 가상의 테이블

1-3) 3행에서 dual 대신 emp 를 사용하면 함수개수만큼 나옴(14개)

1-4) s m i t h

1 2 3 4 5 >> 위 1)에서 결과는 smi 로 출력된다

1-5) 만약 어떤 data 의 마지막에서 n번째 철자부터 n2개를 출력하고자 한다면

substr('XX', -n, n2) 의 형태로 가능하다

★P

69) 사원 테이블에서 이름을 출력하고 그 옆에 이름의 끝철자를 출력하는데 끝 철자를 소문자로 출력하시오!

select ename, lower(substr(ename, -1,1))

from emp;<<

019 문자에서 특정 철자의 위치 출력하기(INSTR)

1. 정의 : instr('XYX', 'Y') 로 표현되며 XYX 문자열 중 Y가 몇 번째 자리에 있는지 출력해주는 함수.

XYX 대신 column 을 활용함(보통의 경우)

1) 예제 : select instr('smith', 'm')

from dual;

2) 이 함수는 두 번째 인자의 자리수를 출력한다

3) 예제2 : 우리반 테이블에서 이메일을 출력하고 그 옆에 이메일에서 @ 가 몇 번째 자리에 있는지 출력하시오

select email, instr(email, '@')

from emp12;

4) 예제3 : 우리반 테이블에서 이메일에서 @ 앞까지의 철자를 잘라내시오!

select substr(email, 1, instr(email, '@')-1)

from emp12;

★P

70)☆ 이메일을 출력하고 그 옆에 이메일의 도메인을 출력하시오!

도메인 : @ 우측부터 .com 전까지

>>1 length 함수 이용

select email, substr(email, instr(email, '@')+1,

length(email)-4-instr(email, '@'))

from emp12;<<

```
>>2 length 함수 이용X
select email, substr(email, instr(email, '@')+1,
instr(email, '.',-1)-instr(email, '@')-1)
from emp12;<<
```

020 특정 철자를 다른 철자로 변경하기(REPLACE)

1. Replace : relace('XX', 'm1','m2') >> replace(컬럼명, 대체전 문자, 대체후 문자)
m1 문자를 m2 문자로 치환한다

```
1)예제 : select replace('smith', 'm', 'k')
from dual;
```

★P

71) 사원 테이블에서 이름과 월급을 출력하는데 월급을 출력할 때 숫자 0을 * 로 출력하시오

```
select ename, replace(sal, 0, '*')
from emp;<<
```

※dual 은 함수 설명할때만 사용하는것

```
select replace('smith', 'm', 'k')
from dual;<< 보통 여기서 'smith' 자리에 column 이 들어감
```

72) 우리반 테이블에서 이름을 출력하고 그 옆에 이름에 두 번째 철자를 출력하시오!

```
유연수 연
송종미 종
```

...

```
select ename, substr(ename, 2,1)
from emp12;<<
```

※ 병원 등 전광판에 환자명이 유*수 로 나오도록 SQL을 짜보자

```
select replace(ename, substr(ename,2,1), '*')
from emp12;<<
```

73) 아산병원의 전광판을 구현하시오

```
select replace(ename, substr(ename,2,1), '*')
from emp12;<<
```

74) 남궁솔미 데이터를 입력하고 남궁*미 로 출력되게 위의 SQL을 다시 작성하시오! 데이터 입력방법은 바탕화면에 '남궁솔미 data' 파일 복사 후 실행(drag all)

```
select replace(ename, substr(ename, -2,1), '*')
```

from emp12;<< 이 경우 외자는 성씨가 * 로 바뀜. 현재진도로 수행불가
이건 다음주에

021 특정 철자를 N개 만큼 채우기(LPAD, RPAD)

1. 필요한 이유 : 항상 고정된 자릿수를 보장하기 위해서 필요

2. 문법

1) Lpad(컬럼명, 전체 자릿수, 채워넣을 값) : 좌측에 3인자 채우기

- 예제 : select sal, lpad(sal, 10, '*')

from emp;

- 데이터 값은 변하지 않고 좌측에 3번째 인자값을 채워넣는 함수

2) Rpad(컬럼명, 전체 자릿수, 채워넣을 값) : 우측에 3인자 채우기

- 예제 : select sal, rpad(sal, 10, '*')

from emp;

- Lpad 와 동일하며 3번째 인자가 우측에 채워진다

3) 만약 2번째 인자에 data 자릿수보다 적은 값을 입력한 경우 data 의 앞 자리부터 출력된다. >>

select sal, lpad(sal, 1, '*')

from emp; << 결과 보면 이해됨

4) pad : '채워넣다' 라는 의미를 지님

022 특정 철자 잘라내기(TRIM, RTRIM, LTRIM)

1. 공백 잘라내기 : 공백때문에 데이터 검색이 안되는 경우가 종종 있기 때문에 trim 함수를 자주 사용함

2. trim(coloumn) : trim(address) 을 실행하면 address 양쪽 공백을 제거한다

1) ltrim : 왼쪽공백제거

2) rtrim : 오른쪽 공백제거

3) trim : 양쪽 공백제거

3. SQL 튜닝

select ename, address

from emp12

where trim(address) like '경기도%';<<

에서 3행은 악성 SQL 이다(느린 속도)

★P

75) 경기도에 사는 학생의 이름과 주소를 출력하시오! 단, 와일드 카드를 양쪽에 사용하지 말고 한쪽에만 사용해서 출력하시오(이 경우 속도 저하)

select ename, address

from emp12

where trim(address) like '경기도%';<<

76)☆ 정보통계학과가 전공인 학생의 이름과 나이와 전공을 출력하시오!

```
select ename, age, major
from emp12
where trim(major) like '정보통계학과';<<
```

023 반올림해서 출력하기(ROUND)

1. 숫자함수의 종류

- 1) **round : 반올림 함수**
- 2) **trunc : 잘라내서 버리기 함수(024)**
- 3) **mod : 나눈 나머지값을 출력하기 함수(025)**

2. round

1) 예제 : select round(786.567, 2)
from dual;

2) 7 8 6 . 5 6 7 >> 첫 번째 data
-3 -2 -1 0 1 2 3 >> 두 번째 인자 자리수

3) **round(n1, p1) : 숫자 n1의 p1 자리수까지 반올림하시오**

3-2) 이 때, 두 번째 인자 p2 는 숫자 data가 소수점 좌측은 - 부호를 사용하고
소수점 우측에는 자연수를 사용한다. 소수점은 0 이다.

4) 두 번째 인자에 -1 을 입력하면 해당 자리에서 반올림한다

5) 두 번째 인자에 1 을 입력하면 소수점 첫 번째 자리에 그 이전값이 반올림된다

6) 4)와 5) 는 소수점 좌우측의 성질이 다른점을 알 수 있다.

7) 2번째 인자에 값을 지정하지 않으면 자동으로 0으로 설정된다

★P

77) 우리반 나이의 평균값을 출력하시오!

>>평균값 구하는 SQL

```
select avg(age)
from emp12;
```

79) 77)의 결과를 반올림해서 소수점 이후는 안나오게 하시오!

```
select round(avg(age), 0)
from emp12; <<
```

024 숫자를 버리고 출력하기(TRUNC)

1. trunc

- 1) `select trunc(785.657, 2)`
`from dual;`
- 2) 7 8 6 . 5 6 7 >> 첫 번째 data
-3 -2 -1 0 1 2 3 >> 두 번째 인자 자리수
- 3) 소수점 이후로는 지정된 자리 이후부터 버림(1,2,3..)
- 4) 소수점 이전으로는 지정된 자리를 포함해서 버림(-1,-2,-3...

025 나눈 나머지 값 출력하기(MOD)

1. mod

- 1) `select mod(10,3)`
`from dual;`
- 2) `mod(n1, p1)` : n1 을 p1 으로 나눈 나머지
- 3) 활용 : 홀수/짝수 구분시 주로 사용
- 4) `select mod(24,2), mod(25,2)`
`from dual;`

★P

79) 우리반에서 나이가 짝수인 학생들의 이름과 나이를 출력하시오!

```
select ename, age
from emp12
where mod(age, 2) = 0;<
```

026 날짜 간 개월 수 출력하기(MONTHS_BETWEEN)

1. 날짜 data 개념

- 1) 날짜 - 숫자 : 날짜
- 2) 날짜 + 숫자 : 날짜
- 3) 날짜 - 날짜 : 숫자

2. data 로 확인

- 1) 오늘 날짜 확인
`select sysdate`
`from dual;`
- 2) 오늘 날짜
`select sysdate-1`
`from dual;`
- 3) 내일
`select sysdate + 1`
`from dual;`

4) 날짜 - 날짜

```
select ename, sysdate - hiredate  
from emp;
```

★P

80) 아래 결과에서 소수점 이하는 만나오게 반올림 하시오

```
select ename, sysdate - hiredate  
from emp; << 아래 결과  
select ename, round(sysdate - hiredate, 0)  
from emp; <<
```

※ 0은 제외 가능

81) 이름, 입사한 날짜부터 오늘까지 총 몇 주 근무했는지 출력하시오! (정수로)

```
select ename, round((sysdate-hiredate)/7, 0)  
from emp; <<
```

82) 이름, 입사한 날짜부터 오늘까지 총 몇 달 근무했는지 출력하시오

```
select ename, round(months_between(sysdate, hiredate), 0)  
from emp; <<
```

※ **month_between(최신날짜, 옛날날짜)**

- 날짜와 날짜 사이의 개월수를 출력
- **oracle** 에서 사전 준비되어 있는 함수

83) 아래와 같이 결과를 출력하시오!

KING 은 467 달을 근무했습니다.

...

```
select ename || ' 은 ' || round(months_between(sysdate,hiredate))  
|| ' 달을 근무했습니다.'  
from emp; <<
```

3. 날짜함수 정리

1) **months_between(최신날짜, 옛날날짜)**

1-2) **months_between(n1, n2)** : (n1-n2) 인 달의 차이를 소수점까지 계산됨

2) **add_months**

3) **next_day**

4) **last_day**

027 개월 수 더한 날짜 출력하기(ADD_MONTHS)

1. add_months(최신날짜, p1)

1) 오늘날짜에서 100달 뒤에 돌아오는 날짜가 몇일인가?

```
select add_months(sysdate, 100)
      from dual;
```

2) p1 은 개월 수

028 특정 날짜 뒤에 오는 요일 날짜 출력하기(NEXT_DAY)

1. next day

1) 오늘부터 앞으로 바로 돌아올 월요일의 날짜를 출력하시오!

```
select next_day(sysdate, '월요일')
      from dual;
```

★P

84) 오늘 날짜에서 100달 뒤에 돌아오는 목요일의 날짜를 출력하시오!

```
select next_day(add_months(sysdate,100), '목요일')
      from dual;<<
```

029 특정 날짜가 있는 달의 마지막 날짜 출력하기(LAST_DAY)

1. last_day(sysdate) : sysdate 가 특정 날짜가 된다

```
1) select sysdate, last_day(sysdate)
      from dual;
```

2) 1)을 실행하면 10월의 마지막 날짜가 출력된다

※ 함수 정리

1. 단일행 함수 : 문자함수, 숫자함수, 날짜함수, 변환함수, 일반함수

1) 문자함수 : upper, lower, initcap, substr, instr, replace,
trim/rtrim/ltrim

lpad/rpad,

2) 숫자함수 : round, trunc, mod

3) 날짜함수 : months_between, add_months, next_day, last_day

2. 복수행 함수 : max, min, avg, count, sum

(그룹함수)

※ 해외 작업의 경우 : 현재 내가 접속한 세션의 날짜 형식을 확인

```
select *
      from nls_session_parameters;
```

1) nls : national language support

2) 위 SQL 을 실행하면 oracle 에서 제공하는 parameter 을 확인할 수 있다

3) NLS_DATE_FORMAT RR/MM/DD (RR:연도 MM:월 DD:일) >>korea

4) NLS_DATE_FORMAT DD/MM/RR >> america

★P

85) 81/11/17 일에 입사한 사원의 이름과 입사일을 출력하시오!

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

※ 미국의 오라클 환경으로 날짜 형식을 변경해본다

alter session set nls_date_format = 'DD/MM/RR';

2. 날짜 형식

1) 년도 : RRRR, YYYY, RR, YY

- RR :

- YY :

1-2) alter session set nls_date_format = 'RR/MM/DD';

1-3) alter session set nls_date_format = 'YY/MM/DD';

1-4) select ename, hiredate

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

1-5) 설명 : 1-3)을 적용하고 1-4)를 실행하면 값이 출력되지 않음.

이것은 RR 와 YY 가 서로 다르기 때문

RR	YY
81	81
1981 / 2081	1981 / 2081
2020 >>> 1981	2020 >>> 2081
현재연도에서 가장 가까운 연도를 선택	현재세기로 인식

2) 월 : MM, MON

- MON : janurary, feb~

3) 일 : DD

4) 시간 : HH, HH24

- HH24 : 23, 24 등

5) 분 : MI

6) 초 : SS

7) 요일 : day, dy, d

7-2) select ename, hiredate, to_char(hiredate, 'day'), to_char(hiredate, 'dy'),
to_char(hiredate, 'd')

```
from emp;  
7-3) day
```

030 문자형으로 데이터 유형 변환하기(TO_CHAR)

1. 정의 : 숫자형 데이터를 문자형으로 변환하거나, 날짜형 데이터를 문자형으로 변환할 때 사용하는 함수

1) 예제 : 오늘이 무슨 요일인지 출력하고 싶다면?

```
select to_char(sysdate, 'day')  
from dual;
```

2) to_char(날짜, 'XX') : 날짜를 XX 의 문자로 출력할 수 있다

3) char 은 character(캐릭터) 약어

★P

86) 이름, 입사일, 입사한 요일을 출력하시오!

```
select ename, hiredate, to_char(hiredate, 'day')  
from emp;<<
```

87) 수요일에 입사한 직원들의 이름과 입사일을 출력하시오!

```
select ename, to_char(hiredate, 'day')  
from emp  
where to_char(hiredate, 'day') = '수요일';<<
```

88) 내가 무슨 요일에 태어났는지 확인하시오!

```
select to_char(to_date('94/05/14', 'RR/MM/DD'), 'DAY')  
from dual;<<
```

※ 88)번 설명 : to_date 함수로 변환해주어야 한다

to date 함수는 날짜형으로 데이터 유형을 변환시켜준다

89)☆ 이름, 입사한 요일을 출력하는데, 입사한 요일이 월화수목금토일 순으로 정렬되어서 출력되게 하시오!

>>일요일 데이터 넣기 :

```
insert into emp(empno, ename, hiredate)  
values(1234, 'JACK', to_date('82/01/10', 'RR/MM/DD'));  
commit;<<
```

>>답안

```
select ename, to_char(hiredate, 'day')  
from emp  
order by to_char(hiredate-1, 'd') asc;<<
```

##1026

2020년 10월 26일 월요일 오전 9:33

1. 1023 이어서

★P 손풀기

90) 직업이 SALESMAN 인 직원들의 이름과 월급과 직업을 출력하는데 월급이 높은 직원부터 출력하시오!

```
select ename, sal, job
from emp
where job = 'SALESMAN'
order by 2 desc;<<
```

※ order by 절은 coding 과 실행 순서가 가장 마지막이다

91) 직업이 SALESMAN 이 아닌 직원들의 이름과, 입사일과, 직업을 출력하는데 최근에 입사한 직원부터 출력하시오!

```
select ename, hiredate, job
from emp
where job != 'SALESMAN'
order by 2 desc;<<
```

※ 3 select 컬럼명
1 from 테이블명
2 where 검색조건
4 order by 정렬할 컬럼명
좌측의 숫자는 실행순서이다

92) 월급이 1000에서 3000 사이인 직원들의 이름과 월급을 출력하시오!

```
select ename, sal
from emp
where sal between 1000 and 3000;<<
```

93) 이름을 출력하고 그 옆에 이름의 첫번째 철자만 출력하는데 소문자로 출력하시오!

```
select ename, substr(lower(ename), 1,1)
from emp;<<
```

94) 우리반 테이블에서 이름과 이메일을 출력하고 그 옆에 이메일에서 @ 가 몇번째 철자인지 출력하시오!

```
select ename, email, instr(email, '@')
FROM EMP12;<<
```

95) 이름과 입사일, 입사한 년도를 4자리로 출력하시오!

```
select ename, hiredate, to_char( hiredate, 'rrrr')  
from emp;<<
```

※ 95) 부연설명

년도 : RRRR, RR, YYYY, YY

달 : MM, MON

시간 : DD

분 : HH, HH24

초 : SS

요일 : DAY, DY, D

※ to_char : 날짜와 숫자 데이터를 문자 데이터로 변환해주는 함수

96) 11월에 입사한 직원들의 이름과 입사일을 출력하시오!

```
select ename, hiredate  
from emp  
where to_char(hiredate,'MM') = '11';<<
```

97) 96)을 to_char 을 이용하지 않고 substr 로 수행하시오!

```
select ename, hiredate  
from emp  
where substr(hiredate, 4, 2) = 11;<<
```

031 날짜형으로 데이터 유형 변환하기(TO_DATE)

1. 정의 : 날짜로 형 변환하는 함수

1) 예제 : select ename, hiredate
from emp
where hiredate = to_date('81/11/17', 'RR/MM/DD');
↓
날짜형 데이터 유형 = 날짜

>>2 select ename, hiredate
from emp
where hiredate = '81/11/17';

where sal = 3000;
↓
숫자형 데이터 유형 = 숫자

2) 날짜형 데이터를 검색할 때는 반드시 to_date 함수를 사용할 것을 권장
즉, 1) 예제의 첫번째 SQL 로 검색하는 것이 바람직함

★P

98) 1981년도에 입사한 직원들의 이름과 입사일을 출력하시오

>>1 to_char

```
select ename, hiredate
from emp
where to_char(hiredate, 'RRRR') = '1981';<<
```

>>2 to_date

```
select ename, hiredate
from emp
where hiredate between to_date('81/01/01', 'RR/MM/DD')
and to_date('81/12/31', 'RR/MM/DD');<<
```

※ where 절에 to_char 함수를 사용하면 검색속도가 저하되기 때문에 98) <<2 의 SQL 을 사용하는 것이 더 효율적이다.

※ to_number 함수

1) 숫자로 형 변환하는 함수

2) 예시 : select ename, sal

```
from emp
where sal = 3000;
```

↓ ↓
숫자형 숫자형

3) 자료가 숫자형인 것을 어떻게 알 수 있을까?

- desc emp;<<

이름	널?	유형
-----	-----	-----
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

- number : 숫자형
- varchar2 : 문자형
- date : 날짜형

4) 2)의 3행에서 3000 대신 '3000' 을 해도 oracle 은 값이 동일하게 출력됨

4-2) 다른 database 소프트웨어에서는 err 발생

4-3) oracle 에서는 내부적으로 '3000' 의 값을 문자형에서 숫자형으로 변환

4-4) 이때, '3000' 으로 작성하면 검색속도가 느려지기 때문에 **데이터 유형을 맞추어 주는 것이 검색속도 향상에 큰 도움이 된다**

5) 4-3)수행 oracle 내부 수행과정을 보려면 SQL 앞에 set autot on 작성후 SQL과 함께 수행한 뒤 스크립트를 보면 확인할 수 있다

★P

99)☆★ 1981년도에 입사한 직원들의 이름과 입사일과 입사한 년도를 출력하는데 가장 최근에 입사한 직원부터 출력하시오! 단 where 절에 to_char 을 사용X

```
select ename, hiredate, to_char( hiredate, 'RRRR')
from emp
where hiredate between to_date( '81/01/01', 'RR/MM/DD')
and to_date('81/12/31', 'RR/MM/DD')
order by 2 desc;<<
```

>>to_char 을 사용

```
select ename, hiredate, to_char( hiredate, 'RRRR')
from emp
where to_char( hiredate, 'RRRR') = '1981'
order by 2 desc;<<
```

※ 변환함수 : to_char, to_number, to_date

※ to_number

```
1) select ename, sal
from emp100
where sal = '2000'<< sal이 문자 data 일 때
```

2) where 절에 검색조건을 사용할 때 주의할 사항은 문자 컬럼의 데이터를 검색할 땐 문자로 검색하고 숫자 컬럼의 데이터를 검색할 때는 숫자로 검색해야 한다. 만약 이를 뒤섞어서 사용할 경우 oracle 상에서의 err 는 발생하지 않으나 검색성능이 저하된다.

2-2) 추가로, where column 에서column 을 함수로 가공하는 경우에도 검색속도 저하가 일어난다. (최소화하기)

※ 일반함수

- 1) nvl
- 2) decode
- 3) case

033 NULL 값 대신 다른 데이터 출력하기(NVL, NVL2)

1. nvl 함수

1) 정의 : null 값 대신에 다른값을 출력하고 싶을 때 사용하는 함수

2) 예제 : 이름, 월급, 커미션, 월급+커미션을 출력하시오!

```
select ename, sal, comm, sal+comm
```

from emp;<< 이 경우 sal+comm column 에 null 값이 생성된다

2-2) select ename, sal, comm, nvl(comm, 0)

from emp;<<

2-3) select ename, sal, comm, sal+nvl(comm, 0)

from emp;<< 이 경우 월급+커미션이 정상출력된다

3) null 값 : 데이터가 없는 상태 또는 알 수 없는 값

4) nvl 함수로 다른값으로 대체해서 출력하는 것이지 실제로 테이블의 데이터가 0으로 변경되는 것은 아니다

5) nvl(column, 'n2') : column 의 null 에 해당하는 값을 n2 로 출력, 그외값은 그대로 출력

★P

100) 이름, 커미션을 출력하시오

```
select ename, comm
```

from emp;<< null 값 생성

101) 이름, 커미션을 출력하는데 커미션이 null 인 직원들은 no comm 이라는 글씨로 출력되게 하시오!

```
select ename, nvl( to_char( comm), 'no comm' )
```

from emp;<<

※ 101)은 숫자형 데이터 comm 을 to_char 함수를 이용해서 데이터 타입을 문자형으로 맞추어 출력해서 err 을 해결할 수 있다. 미사용시 err 발생

102) 커미션이 null 인 직원의 이름과 커미션을 출력하시오!

>>nvl 사용

```
select ename, comm
```

from emp

```
where nvl(comm, -1) = -1;<<
```

↓

null 대신에 -1 로 변경하겠다

>>is null 사용

```
select ename, comm
```

from emp

where comm is null;<<

034 IF문을 SQL로 구현하기 1(DECODE)

1. decode : if 문을 SQL 로 구현할 때 사용하는 함수

1) 예제 : select ename, sal, deptno,
 decode(deptno, 10, 5600,
 20, 3400,
 0) as 보너스
 from emp;

2) 문법 : decode(column, c1, n1,
 c2, n2,
 d)

2-2) 의미 : column 내 data 가 c1 이면 n1을, c2 면 n2 를, 그 외 data는 d를 출력

★P

103) 이름, 월급, 직업, 보너스를 출력하는데 보너스가 직업이 SALESMAN 이면 4500 을 출력하고 직업이 ANALYST 면 2400 을 출력하고 나머지 직업은 0을 출력하시오!

```
select ename, sal, job, decode(
    job, 'SALESMAN', 4500, 'ANALYST', '2400', 0) as 보너스
from emp;<<
```

104) 이름, 입사한 년도를 4자리로 출력하시오!

```
select ename, to_char(hiredate, 'RRRR')
from emp;<<
```

105) 이름, 입사한 년도, 보너스를 출력하는데 보너스가 입사한 년도가 1980 년이면 5000을 출력하고, 1981 년이면 4000을 출력하고 나머지 년도는 0을 출력하시오!

```
select ename, to_char(hiredate, 'RRRR'),
       decode( to_char(hiredate, 'RRRR'),'1980', 5000,
              '1981', 5000, 0) 보너스
from emp;<<
```

106) 이름, 월급, 보너스를 출력하는데 보너스가 월급이 4000 이상이면 500을 출력하고 월급이 2000 이상 4000 미만이면 300을 출력하고 나머지 월급 직원들은 그냥 0을 출력하시오!

```
select ename, sal, case when sal >= 4000 then 500
                        when sal >= 2000 then 300
                        else 0 end as 보너스
from emp;<<
```

※ decode, case

1) decode 는 등호(=) 비교만 가능하기 때문에 106) 과 같이 부등호 비교를 하려면 case 문을 사용해야 한다.

case 은 등호, 부등호 비교 둘 다 가능

2) 문법 : case when column >= c1, n1
 column >= c2, n2
 else 'XX' end

3) case 문은 실행순서가 순차적이다

107) 이름, 월급, 부서번호, 보너스를 출력하는데 보너스가 부서번호가 10번이면 500을, 부서번호가 20번이면 300을, 나머지 부서번호면 0을 출력하시오!

```
select ename ,sal, deptno,  
       case when deptno = 10 then 500  
            when deptno = 20 then 300  
            else 0 end as 보너스  
from emp;<<
```

108) 우리반 테이블에서 이름을 출력하고 그 옆에 보너스를 출력하는데 이름의 철자가 3글자이면 보너스를 7000을 출력하고 이름의 철자가 2글자이면 보너스를 5000을 출력하고 이름의 철자가 4글자이면 보너스를 4000을 출력하시오!

```
select ename, case when length(ename) = 4 then 4000  
                  when length(ename) = 3 then 7000  
                  when length(ename) = 2 then 5000  
                  else 0 end as 보너스  
from emp12;<<
```

>>2 부등호 사용

```
select ename, case when length(ename) >= 4 then 4000  
                  when length(ename) >= 3 then 7000  
                  else 7000 end as 보너스  
from emp12;<<2
```

109) 우리반 테이블로 이름 세글자로만 이름의 가운데 글자를 * 로 출력하시오!

```
select replace( ename, substr(ename,2,1), '*')  
from emp12;<<
```

- 이름의 철자의 갯수가 3글자와 2글자는 아래의 SQL로 수행

```
select replace( ename, substr(ename,2,1), '*')  
from emp12;<<
```

- 이름의 철자의 갯수가 4글자면 아래의 SQL로 수행

```
select replace( ename, substr(ename, -2, 1), '*')
```

```
from emp12;
```

110) 우리반 테이블의 이름의 철자의 갯수와 관계없이 일괄적으로 이름이 *이 아래와 같이 출력되게 하시오

```
>>>>
```

```
남궁*미
```

```
허*
```

```
김*비
```

```
...
```

```
select case when length(ename) <= 3 then
            replace(ename, substr(ename,2,1), '*')
        when length(ename) = 4 then
            replace( ename, substr(ename,-2,1), '*')
        else null end
from emp12;<<
```

111) emp(사원) 테이블에서 이름을 출력하고 입사한 요일을 출력하는데 입사한 요일이 월화수목금토일로 순으로 출력하시오!

```
select ename, to_char(hiredate, 'day')
from emp
order by to_char(hiredate-1, 'd') asc;<<
```

>>decode나 case 이용

```
select ename, to_char(hiredate, 'day')
from emp
order by decode( to_char(hiredate, 'd'),'1', '8',
                '2', '1',
                '3', '2',
                '4', '3',
                '5', '4',
                '6', '5',
                '7', '6') asc;<<
```

>>replace 이용

```
select ename, to_char(hiredate, 'day')
from emp
order by replace( to_char(hiredate, 'd'), 1, 8) asc;<<
```

※ 함수 종류

1) 단일행 함수

2) 복수행 함수 : max, min, avg, sum, count

036 최대값 출력하기(MAX)

1. max

```
1) select max(sal)
   from emp;
```

★P

112) 직업이 SALESMAN 인 사원들의 최대월급을 출력하시오!

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

113) 우리반에서 최소 나이인 학생의 나이를 출력하시오!

```
select min(age)
   from emp12;<<
```

114) 통신사가 sk 인 학생들 중에서 최대 나이인 학생의 나이를 출력하시오!

```
select max(age)
   from emp12
  where upper(telecom) in ('SK', 'SKT');<<
```

115) ☆ 30번 부서번호의 최대 월급을 출력하시오!

```
select max(sal)
   from emp
  where deptno = 30;<<
```

※ 115) 에서 1행에 select deptno, max(sal) 을 입력하면 err 발생

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

group by deptno;<< 4행에 group by 작성하면 출력 가능

※ group by deptno 을 하면 여러 개 나오려는 deptno 를 grouping 해준다

116) ☆ 직업, 직업별 최대월급을 출력하는데 직업이 SALESMAN 만 출력하시오!

```
select job, max(sal)
```

```
from emp
where job = 'SALESMAN'
group by job;<<
```

★★★Practice 10.26 - 10.28

문제1. 부서번호, 부서번호별 최대월급을 출력하는데 부서번호별 최대월급이 높은것 부터 출력하시오 !

```
select deptno, max(sal)
from emp
where deptno is not null
group by deptno
order by 2 desc;<<
```

문제2. 직업과 직업별 최대월급을 출력하는데 직업이 SALESMAN 은 제외하고 출력하시오 !

```
select job, max(sal)
from emp
where job != 'SALESMAN'
group by job;<<
```

문제3. 입사한 년도(4자리), 입사한 년도별 최소 월급을 출력하시오 !

```
select to_char(hiredate, 'RRRR') 입사년도, min(sal) 최소월급
from emp
group by to_char(hiredate, 'RRRR');<<
```

문제4. 입사한 년도(4자리), 입사한 년도별 최소 월급을 출력하는데 입사한 년도별 최소월급이 높은 사원부터
출력하시오 !

```
select to_char(hiredate, 'RRRR') 입사년도, min(sal) 최소월급
from emp
group by to_char(hiredate, 'RRRR')
order by 2 desc;<<
```

문제5. 통신사, 통신사별 최대 나이를 출력하시오 ! (결과가 아래와 같이 출력 되어야 합니다.)

sk 36

lg 44

kt 31

...

>>1 decode 사용

```
select decode( lower(telecom), 'sk', 'sk',
```

```

        'skt', 'sk',
        'lg', 'lg',
        'kt', 'kt',
        null ) 통신사
        , max(age) 최고령자
from emp12
where telecom is not null
group by decode( lower(telecom), 'sk', 'sk',
                'skt', 'skt',
                'lg', 'lg',
                'kt', 'kt',
                null );<<

```

>>2 case 이용

```

select case when lower(telecom) = 'sk' then 'sk'
        when lower(telecom) = 'skt' then 'sk'
        when lower(telecom) = 'lg' then 'lg'
        when lower(telecom) = 'kt' then 'kt'
        end 통신사, max(age) 최고령자
from emp12
where telecom is not null
group by case when lower(telecom) = 'sk' then 'sk'
        when lower(telecom) = 'skt' then 'sk'
        when lower(telecom) = 'lg' then 'lg'
        when lower(telecom) = 'kt' then 'kt'
        end;<<

```

>>3 substr 활용

```

select lower(substr(telecom,1,2)), max(age)
from emp12
where telecom is not null
group by lower(substr(telecom,1,2));

```


##1029

2020년 10월 29일 목요일 오전 9:50

이어서

※ 데이터 전처리 : 코딩에서 실제로 중요한 부분

1) SQL, python, 판다 등이 숙달되면 좀 편해지는 부분이 있음

★P 몸풀기(~121)

117) 직업이 ANALYST 인 직원들의 최대월급을 출력하시오!

```
select max(sal)
from emp
where job = 'ANALYST';<<
```

118) 직업과 직업별 최대월급을 출력하시오!

```
select job, max(sal)
from emp
where sal is not null
group by job;<<
```

119) 위의 결과를 다시 출력하는데, 직업별 최대월급이 높은것부터 출력하시오!

```
select job, max(sal)
from emp
where sal is not null
group by job
order by 2 desc;<<
```

120) 부서번호, 부서번호별 최대월급을 출력하는데 부서번호별 최대월급이 높은것부터 출력하시오!

```
select deptno, max(sal)
from emp
where sal is not null
group by deptno
order by 2 desc;<<
```

※ order by 절의 옵션

- 1) 120)의 문제 4행에서 order by 2 desc nulls last; 으로 대체하고 3행을 삭제하면 NULL 값이 맨 아래에 정렬된다
- 2) last 대신 first 입력시 맨 앞에 null 값 정렬
- 3) 정리 : order by n1 desc nulls last // order by n1 desc nulls first

※ to_char 를 이용해서 숫자를 문자로 형변환하기

1) EX) select sal, to_char(sal)

from emp;

1-2) 출력값을 보면 sal 은 오른쪽 정렬, to_char(sal)은 왼쪽정렬

2) **select sal, to_char(sal, '999,999')**

from emp;

2-2) to_char(숫자형 column, '999,999') : sal 의 숫자를 문자로 출력하되,

to_char(숫자형 column, '999,999') 문법에 따라 문자 포맷에 맞는 문자로 출력

2-3) 9는 자리수를 의미하고 이 자리에 0~9 사이의 숫자중에 어떤 숫자가 와도 관계가 없지만 자리수는 한자리 여야 한다는 의미

121) 이름과 연봉(sal*12)을 출력하는데 연봉을 출력할 때 천단위와 백만단위가 표시되게 하시오!

```
select ename, to_char( sal*12, '999,999,999')
```

```
from emp;<<
```

122) 위의 결과를 다시 출력하는데 컬럼명을 한글로 연봉이라고 하고 연봉이 높은 사원부터 출력하시오!

```
select ename, to_char( sal*12, '999,999,999') 연봉
```

```
from emp
```

```
order by sal*12 desc nulls last;<<
```

123) 직업과 직업별 최대 월급을 출력하는데 직업별 최대월급을 출력할 때 천단위콤마(,) 가 출력되게 하시오!

```
select job, to_char( max(sal), '999,999')
```

```
from emp
```

```
where sal is not null
```

```
group by job;<<
```

124) 직업, 직업별 최소월급을 출력하시오!

```
select job, min(sal)
```

```
from emp
```

```
where sal is not null
```

```
group by job;<<
```

125) 위의 결과를 다시 출력하는데 직업이 abcd 순서로 출력하시오!

```
select job, min(sal)
```

```
from emp
```

```
where sal is not null
```

```
group by job
```

```
order by 1 asc;<<
```

126) 위 결과에서 직업이 SALESMAN 은 제외하고 출력하시오

```
select job, min(sal)
from emp
where job != 'SALESMAN'
group by job
order by 1 asc;<<
```

※ 126)문제에서 null 값은 비교대상이 안 되므로 출력되지 않는다. 즉 SALESMAN이 아닌 job 을 data 와 대조하는 과정에서 null 값은 자동으로 출력되지 않게 된다

037 최소값 출력하기(MIN)

127) 우리반에서 최소 나이를 출력하시오!

```
select min(age)
from emp12;<<
```

※ 만약 이름도 같이 출력하려면 지금까지 배운 내용으로는 불가.
이때는 '서브쿼리'를 사용해야 정상적인 값을 출력할 수 있다.

128) 서울에서 사는 학생중에 최소 나이를 출력하시오!

```
select min(age)
from emp12
where address like '서울%';<<
```

※ 와일드카드(%) 를 양쪽에 사용시 성능이 느려지므로 불필요시 사용X
특히, 앞쪽에 있는 경우 검색성능 저하가 크다. (뒤쪽에 있는 경우는 크게 X)

129) 입사한 년도(4자리)를 출력하고 입사한 년도별 최소월급을 출력하시오

```
select to_char(hiredate, 'RRRR'), min(sal)
from emp
group by to_char(hiredate, 'RRRR');<<
```

038 평균값 출력하기(AVG)

1. avg : 평균값 출력 가능

★P

130) 우리반 나이의 평균값을 구하시오

```
select avg(age)
  from emp12;<<
```

131) 사원 테이블의 월급의 평균값을 출력하시오!

```
select avg(sal)
  from emp;<<
```

※ 131)번의 avg 는 월급을 다 더한후 14로 나눔

132) 커미션의 평균값을 구하시오!

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

※ 132) 경우는 커미션을 다 더한 후 4로 나눈다.

이때 그룹함수는 null 값을 무시한다.

133) 위 결과를 다시 출력하는데 4로 나누지 않고 14로 나누게 하시오!

null 값을 0으로 출력하게 하는 함수를 이용하면 된다. 내 자료의 경우 jack 이 두명이라 값은 좀 달라질 수 있음

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

※ null 값을 다 0으로 변경했기 때문에 4로 나누지 않고 16(jack 두명) 으로 나눔

134) 위 결과에서 소수점 이하는 안나오게 반올림하시오

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

135) 직업, 직업별 평균월급을 출력하시오!

```
select job, avg( nvl(sal,0) )
  from emp
 where job is not null
 group by job;<<
```

136) 통신사, 통신사별 평균나이를 출력하시오!

```
select decode( lower(telecom), 'skt', 'sk', lower(telecom) ) ,
       avg(age)
  from emp12
 group by decode( lower(telecom), 'skt', 'sk', lower(telecom) );<<
```

※ decode, case 문 정리

1) decode(column, n1, c1, n2, c2, p)

1-2) column 의 data 가 n1 이면 c1을, n2 이면 c2를 출력하고 그외에는 p 를 출력하시오 (단, p 자리에는 column 도 가능함)

2) case when A then B

A2 then B2

else C end

137) ☆ 전공, 전공별 평균나이를 출력하는데 전공이 ㄱ, ㄴ, ㄷ ... 순으로 출력되게 하시오

```
select ltrim(major), avg(age)
```

```
from emp12
```

```
group by ltrim(major)
```

```
order by 1 asc;<<
```

039 토탈값 출력하기(SUM)

1. sum : 토탈값을 출력하는 함수

1) 예제 : 사원 테이블의 총 월급을 출력하시오

```
select sum(sal)
```

```
from emp;<<
```

★P

138) 직업, 직업별 토탈월급을 출력하는데 직업별 토탈월급이 높은것부터 출력하시오!

```
select job, sum(sal)
```

```
from emp
```

```
group by job
```

```
order by 2 desc;<<
```

139) 위 결과에서 직업이 SALESMAN 은 제외하고 출력하시오!

```
select job, sum(sal) as total
```

```
from emp
```

```
where job != 'SALESMAN'
```

```
group by job
```

```
order by 2 desc;<<
```

140) 위의 결과를 다시 출력하는데 토탈월급이 6000 이상인 것만 출력하시오!

```
select job, sum(sal) as total
```

```
from emp
```

```

where job != 'SALESMAN'
group by job
having sum(sal) >= 6000
order by 2 desc;<<

```

※☆☆ Having 절

140) 문제에서 having 절을 추가해야 답을 구할 수 있다

Group 함수로 조건을 줄 때는 where 절에 사용하면 안되고 having 절에 사용해야 한다. where 절에는 그룹함수를 사용하지 않은 일반적인 검색조건을 사용

※ select 문의 6가지 절

	코딩순서	실행순서
select column	1	5
from	2	1
where	3	2
group by	4	3
having	5	4
order by	6	6

select 검색할 컬럼명

from 검색할 테이블명

where 검색조건

group by 그룹핑할 컬럼

having 그룹함수로 검색조건줄 때 사용

order by 정렬할 컬럼명

141) 직업, 직업별 토달월급을 출력하는데 직업이 SALESMAN 은 제외하고 출력하고 직업별 토달월급이 6000 이상 인 것만 출력하고 직업별 토달월급이 높은것부터 출력하시오!

```

select job, sum(sal)
from emp
where job != 'SALESMAN'
group by job
having sum(sal) >= 6000
order by 2 desc;<<

```

※ having 절에 where 절 조건에 해당하는 조건을 넣어도 값은 출력된다.

그러나 이 경우 검색속도가 느려진다. 아래 예시 SQL 에서 4행의 and 이후의 sql 은 2행 이후 where 절을 삽입하고 141)번과 같이 작성해야 한다.

```

select job, sum(sal)
from emp

```

```
group by job
having sum(sal) >= 6000 and job != 'SALESMAN'
order by 2 desc;
```

142) 통신사, 통신사별 토털나이를 출력하는데 skt 는 제외하고 출력하고 통신사별 토털나이가 100살 이상인 데이터만 출력하고 통신사별 토털나이가 높은것부터 출력하시오!

```
select lower(telecom), sum(age)
from emp12
where lower(telecom) != 'skt'
group by lower(telecom)
having sum(age) >= 100
order by 2 desc;<<
```

143) 위 문제를 다시 푸는데, 이번에는 skt 를 sk 에 포함시켜서 출력하시오!
>>1 decode 활용

```
select decode(lower(telecom), 'skt', 'sk', 'sk', 'sk', lower(telecom))
, sum(age)
from emp12
group by decode(lower(telecom), 'skt', 'sk', 'sk', 'sk', lower(telecom))
having sum(age) >= 100
order by 2 desc;<<
```

>>2 substr 활용

```
select substr(lower(telecom), 1, 2), sum(age)
from emp12
group by substr(lower(telecom), 1, 2)
having sum(age) >= 100
order by 2 desc;<<2
```

144) 입사한 년도(4자리), 입사한 년도별 토털월급을 출력하는데 토털월급을 출력할 때 천단위 표시가 출력되게 하시오!

```
select to_char(hiredate, 'RRRR'), to_char(sum(sal), '999,999')
from emp
group by to_char(hiredate, 'RRRR');<<
```

040 건수 출력하기(COUNT)

1. count : 건수를 출력하는 함수

1) 예제 : 사원 테이블의 인원수를 출력하시오

```
select count(empno)
  from emp;
```

2) 아래의 sql 은 하나의 행을 개수로 다 카운터함

select count(*)

from emp;

2-2) 특정 column 에 null 값이 있을 수 있으므로 count(*) 이 안전성이 높음

2-3) 특별한 경우가 아니면 count(*) 을 사용하자

3) 그룹함수는 null 값을 무시하므로 아래값은 4 가 출력된다

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

※ 그룹함수는 null 값을 무시한다 >> count 에도 적용됨

★P

145) 직업, 직업별 인원수를 출력하시오!

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

※ 부모키 컬럼

↑

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

하지만, empno 자리에 * 쓰는게 제일 확실하다

146) 나이, 나이별 인원수를 출력하시오! (emp12)

```
select age, count(*)
  from emp12
 group by age;<<
```

147) 위 결과를 다시 출력하는데 나이별 인원수가 높은 것부터 출력하시오!

```
select age, count(*)
  from emp12
 group by age
 order by 2 desc;<<
```

148) 위 결과를 다시 출력하는데 나이별 인원수가 2명 이상인 것만 출력하시오!

```
select age, count(*)
  from emp12
 group by age
```


having count(*) >= 2

order by 2 desc;<<

149) 통신사, 통신사별 인원수를 출력하시오!

```
select lower( substr(telecom, 1,2) ), count(*)  
from emp12  
group by lower( substr(telecom, 1,2) )<<
```

150) 이름, 이메일의 도메인만 출력하시오!

```
select ename, substr( email, instr(email, '@')+1,  
length(email)-instr(email, '@')-4 )  
from emp12;<<
```

151) 이메일 도메인, 이메일 도메인별 인원수를 출력하시오!

```
select lower(substr( email, instr(email, '@')+1,  
length(email)-instr(email, '@')-4 )), count(*)  
from emp12  
group by lower(substr( email, instr(email, '@')+1,  
length(email)-instr(email, '@')-4 ));<<
```

152) 주소, 주소별 인원수를 아래와 같이 출력하시오!

서울시 10

경기도 1

...

```
select substr(ltrim(address),1,3), count(*)  
from emp12  
group by substr(ltrim(address),1,3)<<
```

데이터 전처리 후 다시 시도

041 데이터 분석 함수로 순위 출력하기 1(RANK)

1. 데이터 분석 함수 : 데이터 분석을 용이하게 하기 위해서 제공하는 함수

그 중 rank 는 순위를 출력하는 함수

1) 예제 : 이름, 월급, 월급에 대한 순위를 출력하시오

```
select ename, sal, rank() over (order by sal desc) as 순위  
from emp;
```

1-2) rank() : 순위를 매기다

over : 확장하다

(...) : 확장의 조건

★P

153) 이름, 나이, 순위를 출력하는데 순위가 나이가 높은 순서대로 순위를 출력하시오!

```
select ename, age, rank() over (order by age desc) 나이순  
from emp12;<<
```

154) 직업, 이름, 월급, 순위를 출력하는데 직업별로 각각 월급이 높은 순서대로 순위가 출력되게 하시오!

```
select job, ename, sal, rank() over( partition by job  
                                order by sal desc ) 순위  
from emp;<<
```

※ partition by 는 group by 하고 혼동하면 안된다.

partition by 는 데이터 분석함수에서 괄호 안에 쓰는 문법으로

partition by job 이라고 하면 직업별로 각각 파티션해서 나누겠다는 뜻이다.

154) 에서 아래의 뜻은

**rank() over(partition by job >>> 직업별로 파티션해서
order by sal desc) >>> 월급이 높은순서대로 순위를 출력하겠다**

155) 부서번호, 이름, 월급, 입사일, 순위를 출력하는데 순위가 부서번호별로 각각 먼저 입사한 사원 순으로 순위가 부여되게 하시오!

```
select deptno, ename, sal, hiredate,  
       rank() over( partition by deptno  
                   order by hiredate asc) 순위  
from emp;<<
```

156) 통신사, 이름, 나이, 순위를 출력하는데 통신사별로 각각 나이가 높은 학생순으로 순위를 부여하시오!

```
select substr(lower(telecom),1,2), ename, age,  
       rank() over( partition by substr(lower(telecom),1,2)  
                   order by age desc) 순위  
from emp12;<<
```

042 데이터 분석 함수로 순위 출력하기 2(DENSE_RANK)

1. 예제

```
select substr(lower(telecom),1,2), ename, age,  
       dense_rank() over( partition by substr(lower(telecom),1,2)  
                          order by age desc) 순위
```

from emp12;

★P

158) 이름, 월급, 순위를 출력하는데 dense_rank 를 써서 순위가 1,2,3,4,5.. 등 전부 출력되도록 하시오! 단, 순위는 월급이 높은 순서에 대해 구하시오

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

159) 입사한 년도(4자리), 이름, 월급, 순위를 출력하는데 순위가 입사한 년도별로 각각 월급이 높은 순서대로 순위를 출력하시오!

```
select to_char( hiredate, 'RRRR') 입사년도, ename, sal,
       dense_rank() over(partition by to_char( hiredate, 'RRRR')
                          order by sal desc) 순위
```

```
from emp;<<
```

※ rank 와 dense_rank

월급이 같아서 같은 순위가 여러개 나오면 다음 순위가 그 다음 순위가 중복된 수만큼 더해서 출력되는것이 rank 이고 중복되었더라도 바로 그 다음 순위가 나오는 것이 dense_rank 이다

160)☆ 이메일의 도메인, 이름, 나이, 순위를 출력하는데 순위가 이메일 도메인별로 각각 나이가 높은순으로 출력하시오!

```
select lower(substr( email, instr(email, '@')+1,
                    length(email) - instr(email, '@') -
                    decode(length(email)-instr(email, '.',-1), 3, 4, 2, 3, null) ) ) 도메인,
       ename, age, dense_rank() over( partition by
                                     lower(substr( email, instr(email, '@')+1,
                                     length(email) - instr(email, '@') -
                                     decode(length(email)-instr(email, '.',-1), 3, 4, 2, 3, null) ) )
                                     order by age desc) 순위
from emp12;
```

##1030

2020년 10월 30일 금요일 오전 9:44

1029R

1. 기본 select 문장

- 1) 5 select 검색할 컬럼명
- 1 from 검색할 테이블명
- 2 where 검색조건
- 3 group by 그룹핑할 컬럼
- 4 having 그룹함수로 만든 조건
- 6 order by 정렬할 컬럼명

2. 함수

- 1) 단일행 함수 : 한개의 행 >> 함수 >> 한개의 행

1-2) 단일함수의 종류

문자함수 : upper, lower, initcap, substr, instr, lpad, rpad, trim, ltrim, rtrim,

replace

숫자함수 : round, trunc, mod

날짜함수 : months_between, add_months, next_day, last_day

변환함수 : to_char, to_number, to_date

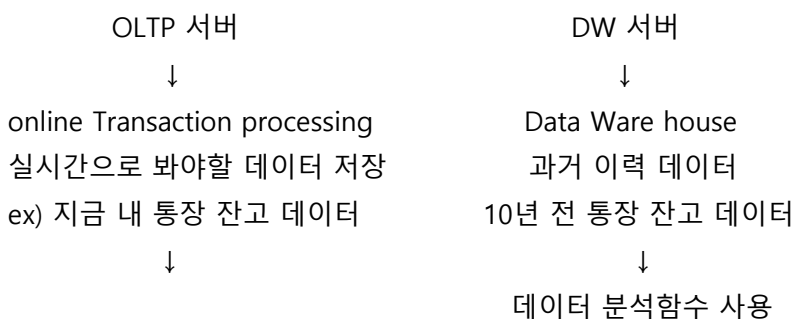
일반함수 : decode(column, n1, c1, n2, c2, d), case when A then B else C end,
nvl, nvl2

- 2) 복수행 함수 : 여러개의 행 >>> 그룹함수 >>> 한개의 행

2-2) 복수함수의 종류 : max, min, avg, sum, count

3. 데이터 분석 함수

- 1) 회사의 전산 시스템의 구조(데이터 저장 부분에 대한 서버의 종류)



★P 몸풀기~162

160) 이름, 입사일, 순위를 출력하는데 순위가 먼저 입사한 직원순으로 순위를 부여하시오!

select ename, hiredate, rank() over(order by hiredate asc) 순위

from emp;<<

161) 직업, 이름, 입사일, 순위를 출력하는데 순위가 직업별로 각각 먼저 입사한 직원순으로 순위를 부여하시오!

```
select job, ename, hiredate,  
       rank() over( partition by job  
                   order by hiredate asc) rank  
from emp;<<
```

162) 이름, 월급, 순위를 출력하는데 순위가 월급이 높은 순서대로 순위를 부여하시오!

```
select ename, sal, rank() over( order by sal desc)  
from emp;<<
```

※ 162) 의 경우 sal 이 같은 경우 공동 인수를 제외한 만큼 다음 순위에 반영되기 때문에 이를 공동순위 다음에 순차적인 순위가 나오게 하려면 rank 대신 dense_rank 를 사용하면 된다.

163) 월급이 2975 는 순위가 몇위인가?

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

※ 163) 설명

rank(a) 가로안에 a 는 특정 값의 순위를 출력할 때 사용

within : ~이내에

>> **월급이 높은순서대로 정렬한 그룹 이내에서 2975가 몇위인지 출력**

164) 우리반에서 34 나이의 순위를 출력하시오! (단, 나이가 높은순으로)

```
select dense_rank(34) within group(order by age desc) rank  
from emp12;<<
```

165) 81년 11월 17일에 입사한 직원은 직원 테이블에서 몇번째로 입사한 직원인가?

```
select dense_rank( to_date('81/11/17', 'RR/MM/DD'))  
       within group( order by hiredate desc)  
from emp;<<
```

※ 165) 에서 to_date 를 사용하지 않아도 값은 출력되지만 사용하는게 확실함

043 데이터 분석 함수로 등급 출력하기(NTILE)

1. ntile : 등급을 출력하는 함수

1) 예제 : **select ename, sal, ntile(4) over(order by sal desc) 등급
from emp;**

1-2) 월급이 높은 순으로 정렬한 데이터를 4등급으로 나누겠다는 의미

1-3) 등급 정의

상위 1~25% : 1등급

상위 26~50% : 2등급

상위 51%~75% : 3등급

상위 76%~100% : 4등급

2) ntile(n) n에 따라 등급이 달라진다

★P

166) 이름, 나이, 등급을 출력하는데 등급을 7등급으로 나눠서 출력하시오!

단, 나이가 높은 순서대로 등급을 나누시오!

```
select ename, age, ntile(7) over( order by age desc) as rank
from emp12;<<
```

167) 직업, 이름, 월급, 등급을 출력하는데 직업별 각각 등급이 3등급으로 나눠지게 하시오! 단, 등급은 월급이 높은 순서대로의 등급.

```
select job, ename, sal, ntile(3) over( partition by job
                                     order by sal)
from emp;<<
```

044 데이터 분석 함수로 순위의 비율 출력하기(CUME_DIST)

1. cume_dist : 순위에 대한 비율을 출력하는 데이터 분석함수

1) 예제 : **select ename, sal,**

cume_dist() over(order by sal desc) 비율

from emp;

2) **cume_dint() over(condition)**

★P

168) 아래 결과에서 소수점 세번째까지만 출력되도록 반올림하시오!

```
select ename, sal,
       cume_dist() over( order by sal desc) 비율
from emp;
```

>>

```
select ename, sal,
       round( cume_dist() over( order by sal desc), 3) 비율
from emp;<<
```

045 데이터 분석 함수로 데이터를 가로로 출력하기(LISTAGG)

1. listagg : 데이터를 가로로 출력하는 함수

1) 예제 :

```
select deptno,  
       listagg(ename, ',') within group( order by ename asc) 이름  
from emp  
group by deptno;
```

1-2) listagg(column, '/') : column 을 가로로 출력하는데 / 으로 data 구분

1-3) within group(condition) : ~조건으로 정렬하겠다

1-4) 이름을 abcd 순으로 정렬하여 가로로 출력하는데 data 구분은 ','로 하겠다

2) listagg 는 다른 분석함수와 달리 group by 절이 필요함

★P

169) 직업, 직업별로 해당하는 직원들의 이름을 가로로 출력하시오!

```
select job,  
       listagg(ename, ',') within group( order by ename asc) 이름  
from emp  
group by job;<<
```

170) 아래와 같이 결과를 출력하시오! 이름옆에 월급이 출력되도록

SALESMAN MARTIN(3000),ALLEN(5009),TURNER,WARD(239)

CLERK JAMES,SMITH,ADAMS,MILLER

ANALYST FORD,SCOTT

MANAGER BLAKE,CLARK,JONES

PRESIDENT KING(300)

>>

```
select job,  
       listagg(ename|| '(' ||sal|| ')', ',')  
       within group( order by ename asc) 이름  
from emp  
group by job;<<
```

171) 나이, 나이별로 해당하는 학생들의 이름을 가로로 출력하시오!

```
select age,  
       listagg(ename, ',') within group( order by ename asc)  
from emp12  
group by age;<<
```

172) ☆ 통신사를 출력하고 통신사별로 해당하는 학생들의 이름을 출력하는데 이름옆에 나이도 같이 출력되게 하고 나이가 높은 학생순으로 출력되게 하시오!

```

kt 한결(31), 김소라(29), ...
lg 김주원(44), 김정민(28), ...
sk 권세원(36), ...
> >
select telecom,
       listagg(ename|| '(' || age || ')', ',')
       within group(order by age desc) 이름
from emp12
group by telecom;< <

```

046 데이터 분석 함수로 바로 전 행과 다음 행 출력하기(LAG, LEAD)

1. lag : **lag(column, n1)** : column의 행-n1 행의 data를 출력하는 함수
(1)예제 : select ename, sal, lag(sal, 1) over (order by sal asc) as 전행
from emp;
2. lead : **lead(column, m1)** : column의 행+m1 행의 data를 출력하는 함수
(1)예제 : select ename, sal, lead(sal, 1) over (order by sal asc) as 다음행
from emp;

★P

173) 이름, 입사일, 바로 전에 입사한 사원의 입사일, 바로 다음에 입사한 사원의 입사일을 출력하시오!

```

select ename, hiredate,
       lag(hiredate, 1) over ( order by hiredate asc) 전행,
       lead(hiredate, 1) over ( order by hiredate asc) 다음행
from emp;< <

```

047 COLUMN을 ROW로 출력하기 1(SUM+DECODE)

1. DW(data ware house) 서버에서 많이 사용하는 함수
2. **sum+decode : column 을 row(열) 로 출력하는 함수**
1) 예제 : 부서번호, 부서번호별 토달월급을 출력하시오!

```

select deptno, sum(sal)
from emp
group by deptno; > > 이 경우 세로로 출력

```

deptno	sum(sal)
30	9400
10	8750

1-2) 1)을 가로로 출력하시오

```
10      20      30      <<< 컬럼명
8750    10870    9400    <<< 데이터
```

2) 예제2 : 부서번호, 부서번호가 10번이면 월급이 출력되게 하고 아니면 0이 출력되게 하시오!

```
select deptno, decode( deptno, 10, sal, 0) as deptno10
from emp
```

3) 예제3 : 예제2 결과에서 부서번호 컬럼은 안나오게 하시오

```
select decode( deptno, 10, sal, 0)
from emp
```

4) 예제4 : 예제3 결과에서 출력된 14개 데이터를 다 sum 하시오!

```
select sum(decode( deptno, 10, sal, 0))
from emp
```

5) 예제5 : 예제4 의 컬럼명을 별칭을 써서 숫자 10으로 변경하시오!

```
select sum(decode( deptno, 10, sal, 0)) as "10"
from emp
```

※ 오라클에서 더블 콤레이션 마크를 사용해야 하는 경우]

컬럼별칭 사용시 특수문자, 공백문자, 대소문자 구분, 숫자를 사용시

6) 아래와 같이 20번과 30번도 그 옆에 출력하시오!

```
select sum( decode(deptno, 10, sal, 0) ) as "10",
       sum( decode(deptno, 20, sal, 0) ) as "20",
       sum( decode(deptno, 30, sal, 0) ) as "30"
from emp;
```

★P

174) 통신사, 통신사별 토털나이를 출력하시오! (세로출력)

```
select telecom, sum(age)
from emp12
group by telecom;< <
```

175) 이번에는 아래와 같이 가로로 출력하시오

```
sk      lg      kt
322     126     411
```

>>

```
select sum( decode( telecom, 'sk', age, 0) ) as "sk",  
       sum( decode( telecom, 'lg', age, 0) ) as "lg",  
       sum( decode( telecom, 'kt', age, 0) ) as "kt"  
from emp12;<<
```

176) 아래의 SQL 두개는 결과가 같을까? **SQL 튜닝**

>> 결과는 같음(2200)

[1] select sum(comm) from emp; >> null 값이 아닌 4건만 다 더했다

[2] select sum(nvl(comm,0)) from emp; >> null 값을 0 으로 치환 후 sum 작업
시 연산에 포함되었다

[2] 보다 [1]의 SQL 성능이 좋다

이런 튜닝이 가능한 이유는 **그룹함수의 연산 시 null 값은 연산에 포함X** 때문

177) 아래의 SQL을 튜닝하시오!

튜닝전

```
select sum( decode( telecom, 'sk', age, 0) ) as "sk",  
       sum( decode( telecom, 'lg', age, 0) ) as "lg",  
       sum( decode( telecom, 'kt', age, 0) ) as "kt"  
from emp12;
```

튜닝후>>

```
select sum( decode( telecom, 'sk', age, null) ) as "sk",  
       sum( decode( telecom, 'lg', age, null) ) as "lg",  
       sum( decode( telecom, 'kt', age, null) ) as "kt"  
from emp12;<<
```

※ 튜닝 후 값 decode 에서 null 자리에 공란으로 두어도 null 로 연산함
decode(column, n1, c2) 으로 하면 if not 은 default , 즉 null 이 된다

178) 직업, 직업별 토탈월급을 출력하시오(세로로 출력)!

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

179) 직업, 직업별 토탈월급을 출력하시오(가로로 출력)!

```
select sum( decode( job, 'ANALYST', sal, null)) as "ANALYST",  
       sum( decode( job, 'CLERK', sal, null)) as "CLERK",  
       sum( decode( job, 'MANAGER', sal, null)) as "MANAGER",  
       sum( decode( job, 'PRESIDENT', sal, null)) as "PRESIDENT",
```

```
sum( decode( job, 'SALESMAN', sal, null)) as "SALESMAN"
from emp;<<
```

※ 179) 에서 sum 값 없이 출력하면 아래와 같다

	ANALYST	CLERK	MANAGER	PRESIDENT	SALESMAN
1	(null)	(null)	(null)	5000	(null)
2	(null)	(null)	2850	(null)	(null)
3	(null)	(null)	2450	(null)	(null)
4	(null)	(null)	2975	(null)	(null)
5	(null)	(null)	(null)	(null)	1250
6	(null)	(null)	(null)	(null)	1600
7	(null)	(null)	(null)	(null)	1500
8	(null)	950	(null)	(null)	(null)
9	(null)	(null)	(null)	(null)	1250
10	3000	(null)	(null)	(null)	(null)
11	(null)	800	(null)	(null)	(null)
12	3000	(null)	(null)	(null)	(null)
13	(null)	1100	(null)	(null)	(null)
14	(null)	1300	(null)	(null)	(null)

여기에 앞에 sum 을 둘러주면 first coulumn을 column 내 data 를 다 더한 값을 출력해 준다

048 COLUMN을 ROW로 출력하기 2(PIVOT)

1. pivot : 세로를 가로로 출력하는 함수

1) 예 : **select ***
from (select deptno, sal from emp)
pivot (sum(sal) for deptno in (10, 20, 30));

1-2) 결과값

```
10      20      30
8750    10875    9400
```

1-3) 2행의 **select deptno, sal from emp**

- 사원테이블에서 부서번호와 월급만 가져온다
- 이때, emp 테이블명만 딱 쓸 수 없음
- 결과를 출력하기 위해서 필요 컬럼만 선별해서 사용해야 함 (unless err)
- 3행의 pivot(A) 에서 A 에 필요한 것들을 from 으로 참조하는 것

1-4) 3행의 **pivot(sum(sal) for deptno in (10, 20, 30));**

- 해석 : **sum(sal)**을 출력 // **for** [어떤 토탈월급을 출력할 것인가?] //
deptno [부서번호를 위한] // **in(10,20,30)**[10,20,30 번의 부서번호]
//pivot[해당값을 가로로 출력하겠다]

★P

180) 통신사, 통신사별 토탈나이를 가로로 출력하시오! (pivot 문 사용)

>> 결과값

'sk'	'lg'	'kt'
322	126	411

select *

```
from( select telecom, age from emp12)
pivot( sum(age) for telecom in ('sk','lg', 'kt') );<<
>>2 세로로 출력하려면
```

```
select telecom, sum(age)
from emp12
group by telecom;
>>3 'sk' 가 아닌 sk 로 출력하려면 3행만 수정
pivot( sum(age) for telecom in ('sk' as "sk", 'lg', 'kt') )
```

181) 위의 결과를 토달나이가 아니라 평균나이가 나오게 하시오

```
select *
from( select telecom, age from emp12)
pivot( avg(age) for telecom in ('sk' as "sk",
                                'lg' as "lg",
                                'kt' as "kt") );<<
```

049 ROW를 COLUMN으로 출력하기(UNPIVOT)

1.

050 데이터 분석 함수로 누적 데이터 출력하기(SUM OVER)

1. sum over : 데이터를 누적해서 합계하는 데이터 분석 함수

1) 예제 : 사원번호, 이름, 월급, 월급의 누적치를 출력하시오!

```
select empno, ename, sal, sum(sal) over( order by empno asc) 누적치
from emp;
```

2) **sum(coulmn) over(condition)**

★P

182) 이름, 나이, 나이의 누적치를 출력하시오!

```
select ename, age, sum(age) over(order by ename asc) 누적치
from emp12;<<
```

183) 직업, 이름, 월급, 월급의 누적치를 출력하는데 직업별로 각각 월급의 누적치가 출력되게 하시오!

```
select job, ename, sal, sum(sal) over( partition by job
```

order by sal asc) 누적치

from emp;<<

184) 통신사, 이름, 나이, 나이의 누적치를 출력하는데 나이의 누적치가 통신사별로 각각 누적되어서 출력되게 하시오!

```
select telecom, ename, age, sum(age) over( partition by telecom
                                         order by telecom) 누적치
```

from emp12;<<

※ sum(column) over(condition) 에서 뒷부분 조건인 order by ~ 에서 ~ 해당하는 컬럼 내 data 값이 같으면 바로 누적된 값이 출력이 된다. 그리고 이것을 바로잡는 SQL 은 따로 존재하지 않는다

<t	김홍비	24	24
<t	장수진	25	74
<t	이성원	25	74
<t	허혁	26	126
<t	정다희	26	126
<t	황나현	27	180
<t	김예린	27	180

051 데이터 분석 함수로 비율 출력하기(RATIO_TO_REPORT)

1. ratio_to_report : 자기의 column 이 column 중에서의 비율이 어떻게 되는지 확인하는 함수

1) 예제 : **select ename, sal, ratio_to_report(sal) over() as 비율**
from emp
where job = 'SALESMAN';

1-2) 자신의 월급이 전체 월급에서 차지하는 비율을 출력하는 함수

2) **ratio_to_report(column) over(condition)**

052 데이터 분석 함수로 집계 결과 출력하기 1(ROLLUP)

1. rollup : 집계한 결과를 맨 아래쪽에 출력하고 싶을 때 사용하는 함수

1) 예제 : 부서번호, 부서번호별 토달월급을 출력하시오! (세로로 출력)

```
select deptno, sum(deptno)
```

```
from emp
```

```
group by deptno;
```

1-2) 3행을 group by rollup(deptno);

1-3) 1-2)를 적용하면 row 4 에 합계가 생성된다

2) 예제 : 직업, 직업별 토달월급을 출력하시오(세로)!

```
select job, sum(sal)
```

```
from emp
```

```

group by job;
2-2) 위 결과에서 직업별 토달월급의 합계를 맨 아래에 출력하시오!
select job, sum(sal)
from emp
group by rollup(job);
3) group by rollup(column) : column은 select 절에서 복수행 함수와 엮이는
column으로 해 주어야 함

```

053 데이터 분석 함수로 집계 결과 출력하기 2(CUBE)

1. cube : 집계 결과를 위쪽에 출력하는 함수

```

1) 예제 : select job, sum(sal)
from emp
group by cube(job);
1-2) rollup 과 반대로 cube는 합을 data row에서 가장 위쪽에 출력

```

★P

185) 통신사, 통신사별 토달나이를 출력하는데 맨 위에 전체 토달나이가 출력되게 하시오!

```

select telecom, sum(age)
from emp12
group by cube(telecom);<<

```

186) 입사한 년도(4자리), 입사한 연도별 토달월급을 출력하는데 맨 위에 전체 토달 월급을 출력하시오!

```

select to_char(hiredate, 'RRRR') 입사년도, sum(sal)
from emp
group by cube(to_char(hiredate, 'RRRR'));<<

```

187)☆ 입사한 년도(4자리), 입사한 연도별 토달월급을 출력하시오!

>>1 세로출력

```

select to_char(hiredate, 'RRRR') 입사년도, sum(sal) 토달월급
from emp
group by to_char(hiredate, 'RRRR');<<

```

>>2 가로출력(sum+decode)

```

select sum( decode( to_char(hiredate, 'RRRR'), '1980', sal, null)) as "1980",
       sum( decode( to_char(hiredate, 'RRRR'), '1981', sal, null)) as "1981",
       sum( decode( to_char(hiredate, 'RRRR'), '1982', sal, null)) as "1982",
       sum( decode( to_char(hiredate, 'RRRR'), '1983', sal, null)) as "1983"
from emp;

```

>>3 가로출력(pivot)

##1102

2020년 11월 2일 월요일 오전 10:14

1031R

1. 기본select 문장 6가지절

select
from
where
group by
having 그룹함수 검색조건
order by

2. 함수

단일행함수 : 문자, 숫자, 날짜, 변환, 일반

복수행함수 : max, min, avg, sum, count

그룹함수의 특징 : null 값 무시,

where 절의 조건이 거짓이어도 결과를 리턴한다

ex) select sum(sal)
from emp
where 1=2;

때문에 where 절 작성에 주의해야 함

3. 데이터 분석함수

- 1) rank : 순위출력
- 2) dense_rank : 순위출력
- 3) ntile : 등급 출력
- 4) cume_dist : 비율출력
- 5) listagg : 데이터를 가로로 출력
- 6) sum(column) over(문법) : 누적 데이터 출력
- 7) ratio_to_report : 비율출력
- 8) rollup : 집계 결과를 맨 아래에 출력
- 9) cube : 집계 결과를 맨 위에 출력

데이터 분석함수를 이용하면 우리가 현업에서 필수로 검색해야하는 데이터를 긴 SQL로 작성하지 않고 간단한 함수를 이용해서 볼 수 있다.

오라클 버전의 히스토리

8 >> 8i >> 9i >> 10g >> 11g >> 12c >> 18c >> 19c

i : internet

g : grid : 성능이 보통인 여러개의 컴퓨터를 붙여서 마치 성능이 아주 좋은 큰 서버처럼 운영하는 기술

c : cloud

★

188) 직업, 직업별 토달월급을 출력하시오!


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

189) 위 결과를 다시 출력하는데 맨 아래에 전체 토탈월급이 출력되게 하시오!

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

190) 아래의 job 맨 아래 null 로 나오는 부분에 토탈값 이라고 한글로 null 대신 출력하시오!

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

191) 위 결과를 다시 출력하는데 컬럼명이 아래와 같이 출력되게 하시오! (컬럼명 job)

```
select nvl(job, '토탈값') as job, sum(sal)
  from emp
 group by rollup(job);
```

192) 위의 결과를 다시 출력하는데 아래와 같이 토탈월급 부분에 천단위를 부여하시오

```
select nvl(job, '토탈값') as job, to_char(sum(sal), '999,999')
  from emp
 group by rollup(job);
```

054 데이터 분석 함수로 집계 결과 출력하기 3(GROUPING SETS)

grouping sets : 집계결과를 출력하는 데이터 분석 함수

ex) select deptno, sum(sal)

```
  from emp
 group by grouping sets( deptno, ( ) );
```

>> **grouping sets(column, ()) : column 별로 집계한 것과 ()전체로 집계한 것**
grouping sets((column1, column2), (column2), ()) 의 형태도 가능함

이때, () 을 생략하면 sum(sal)의 전체 합산값이 나오지 않음

★

193) 직업, 직업별 토탈월급을 출력하는데 맨 아래에 전체 토탈월급이 출력되게 하시오!

grouping sets 를 이용

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

194)부서번호와 직업을 출력하고 그 옆에 부서번호별 직업별 토탈월급을 출력하시오!

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

※ select에 그룹함수와 함께 나열한 컬럼들은 반드시 group by 절에 명시해줘야 에러가 나지 않고 출력이 가능하다

195)부서번호와 직업을 출력하고 그 옆에 부서번호별 직업별 토탈월급을 출력하고 동시에 부서번호별 토탈월급도 중간 중간 출력되게 하시오!

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

196) 위 결과가 맨 아래에 전체 토탈월급이 출력되게 하시오!

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

197) 직원번호, 직원이름, 월급을 출력하는데 맨 아래에 전체 토탈월급을 출력하시오!

```
select empno, ename, sum(sal)
  from emp
 group by grouping sets( (empno, ename), () );
```

※직원번호는 중복되지 않기 때문에 grouping sets 함수의 괄호안에 grouping 결과로 넣어도 그냥 자기 월급이 출력된다. 때문에 sum(sal) 로 sql을 작성해도 각각의 sal을 출력가능

grouping sets 의 () 은 전체 토탈월급을 출력. 또한 grouping sets 는 반드시 그룹함수와 동행

198) 우리반 테이블에서 통신사, 이름, 나이를 출력하는데 중간중간 통신사별 토탈나이가 출력되게 하시오!

```
select telecom, nvl(ename, '**'), sum(age)
  from emp12
 group by grouping sets( (telecom, ename), (telecom), () );
```

※어법정리 grouping sets

grouping sets((column1, column2), (column1), ())

이때 select 절에는 항상 그룹함수가 존재해야 하며 이 sql은 그 그룹함수에 대해 그룹핑한 결과를 나타내 준다. 괄호 첫 칸에는 그룹함수를 제외한 모든 column이 포함되며 다음 인자에는 원하는 column의 중간합산값을 출력 가능하다. 3번째 인자에는 ()을 입력하여 그룹함수 전체에 대한 값을 산출할 수 있다. 이때, 194)처럼 괄호를 사용하지 않을 경우에는 () 인자를 포함할 수 없으며 다른 예제들 처럼 row가 추가되면서 지정된 column에 대한 그룹함수의 결과값이 출력되지 않는다.

199)입사한 년도(4자리), 입사한 년도별 토탈월급을 출력하는데 전체 토탈월급이 맨 아래에 출력되게 하시오!

>>1 grouping sets 사용

```
select to_char(hiredate, 'RRRR') as R, sum(sal)
  from emp
 group by grouping sets( (to_char(hiredate, 'RRRR')), () );
```

>>2 rollup 사용

```
select to_char(hiredate, 'RRRR') as R, sum(sal)
  from emp
 group by rollup(to_char(hiredate, 'RRRR'));
```

200) 위 결과를 다시 출력하는데 (아래와 같이) 천 단위를 부여해서 출력하시오!

```
select to_char(hiredate, 'RRRR') as R, to_char( sum(sal), '999,999') as sum
  from emp
 group by grouping sets( (to_char(hiredate, 'RRRR')), () );
```

201) 입사한 년도(4자리), 부서번호, 입사한 년도별 부서번호별 토탈월급을 출력하시오!

```
select to_char(hiredate, 'RRRR'), deptno, sum(sal)
  from emp
 group by grouping sets( (to_char(hiredate, 'RRRR'), deptno) );
```

☆202) 입사한 년도(4자리), 부서번호, 입사한 년도(4자리)별 부서번호별 토탈월급을 출력하는데 중간중간 입사한 년도(4자리)별 토탈월급이 출력되게 하고 맨 아래에 전체 토탈월급이 출력되게 하시오!

```
select to_char(hiredate, 'RRRR')입사년도, deptno 직원번호, sum(sal)
  from emp
 group by grouping sets( (to_char(hiredate, 'RRRR'), deptno), to_char(hiredate, 'RRRR'), () );
```

※ 이때 null 값이 뜨지 않게 하려면 select이 있는 1행에 nvl 함수를 column들에 적용하면 됨

055 데이터 분석 함수로 출력 결과 넘버링 하기(ROW_NUMBER)

row_number : 출력결과의 행 앞에 숫자를 차례대로 부여하는 함수

ex) select row_number() over(order by empno), empno, ename
from emp;

row_number() over(condition) 을 사용하면 출력되는 행의 맨 위의 3개의 행만, 혹은 하나의 행을 검색하고자 할 때 유용하게 활용가능. 인라인뷰(from 절의 서브쿼리) 를 배운 뒤 활용

ROW_NUMBER() OVER(ORDER BY EMPNO)	EMPNO	ENAME
1	7369	SMITH
2	7499	ALLEN
3	7521	WARD
4	7566	JONES
5	7654	MARTIN
6	7698	BLAKE
7	7782	CLARK
8	7788	SCOTT
9	7839	KING
10	7844	TURNER
11	7876	ADAMS
12	7900	JAMES
13	7902	FORD
14	7934	MILLER

★

203) 직업이 SALESMAN 인 직원들의 이름과 직업을 출력하는데 row_number 함수를 사용해서 맨 앞에 번호를 차례대로 부여하시오! 단, 이름순으로 정렬하시오

```
select row_number() over(order by empno asc) 번호, ename, job  
from emp  
where job = 'SALESMAN';
```

※order by 절 이후에 정렬하고 싶은 column을 기입하여 결과를 출력할 수 있다. 또한 column이후 asc 나 desc 를 따로 지정하지 않으면 default(기본값)은 asc 로 출력된다. row_number 은 그 자체로 column으로 출력된다

204) 직업이 SALESMAN 인 직원들의 이름과 월급과 직업을 출력하는데 맨 앞에 Row_number함수를 사용하여 번호가 부여되게 하시오! 단, 월급이 높은순서대로 출력

```
select row_number() over(order by sal desc) 번호, ename, sal, job  
from emp  
where job = 'SALESMAN';
```

049 ROW를 COLUMN으로 출력하기(UNPIVOT)

pivot 문 : 세로 >> 가로

unpivot 문 : 가로 >> 세로

unpivot 문 활용을 위해서는 table을 하나 만들어야 함(가로테이블 data)

```
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;
```

첫째 단락은 테이블 생성 스크립트

둘째 단락은 데이터 입력 스크립트

commit 은 실행문

ex1)

```
select *
  from order2
  unpivot( 건수 for 물품 in (BICYCLE, CAMERA, NOTEBOOK) );
```

	ENAME	물품	건수
1	SMITH	BICYCLE	2
2	SMITH	CAMERA	3
3	SMITH	NOTEBOOK	1
4	ALLEN	BICYCLE	1
5	ALLEN	CAMERA	2
6	ALLEN	NOTEBOOK	3
7	KING	BICYCLE	3
8	KING	CAMERA	2
9	KING	NOTEBOOK	2

※위 SQL에서 건수와 물품은 작성자 임의대로 이름을 명명해도 되며 그대로 column명으로 출력이 된다. 또한 BICYCLE, CAMERA, NOTEBOOK 은 양쪽에 싱글 쿼테이션 마크를 둘러주지 않아도 잘 출력된다. (싱글 쿼테이션 마크를 달면 err 발생함)

ex2) select *

from order2;

	ENAME	BICYCLE	CAMERA	NOTEBOOK
1	SMITH	2	3	1
2	ALLEN	1	2	3
3	KING	3	2	2

ex3) select *

```
from ( select deptno, sal from emp)
pivot ( sum(sal) for deptno in (10, 20, 30) );
```

	10	20	30
1	8750	10875	9400

part3 중급 SQL

056 출력되는 행 제한하기 1(ROWNUM)

rownum : 출력되는 행의 개수를 제한할 수 있다

ex) select rownum, empno, ename, sal

from emp;

	ROWNUM	EMPNO	ENAME	SAL
1	1	7839	KING	5000
2	2	7698	BLAKE	2850
3	3	7782	CLARK	2450
4	4	7566	JONES	2975
5	5	7654	MARTIN	1250
6	6	7499	ALLEN	1600
7	7	7844	TURNER	1500
8	8	7900	JAMES	950
9	9	7521	WARD	1250
10	10	7902	FORD	3000
11	11	7369	SMITH	800
12	12	7788	SCOTT	3000
13	13	7876	ADAMS	1100
14	14	7834	MILLER	1300

※ row_number() 함수와는 다르게 인라인뷰(from절의 서브쿼리) 를 사용하지 않고도 '위 3개의 행만 가져와라' 와 같은 데이터 검색이 가능하다

ex) select rownum, empno, ename, sal

from emp

where rownum <= 3;

※ 대용량 테이블의 데이터가 있는 테이블의 내용을 살짝 보고 싶다면 rownum을 사용해서 조금의 데이터만 살펴 보는 것이 바람직하다. * 을 사용해서 전체 데이터 열람시 과부하 발생

★

205) 직업이 SALESMAN 인 사원들의 데이터 중 2개만 출력하시오! (사원이름, 직업, 월급)

select rownum, ename, job, sal

from emp

where job = 'SALESMAN' and rownum <= 2;

206) 위 결과에서 1건만 출력되게 하시오

```
select rownum, ename, job, sal
from emp
where job = 'SALESMAN' and rownum <= 1;
```

※ <= 대신 = 도 가능, 단 열람하고자 하는 data 가 1개일 경우에만 해당. 2개 이상인 경우에는 자료가 출력되지 않는다. rownum은 항상 부등호 비교와 같이 사용해야 함.

ex) 만약 우리 건물앞에 강남역을 가는 미니버스가 있는데 앞에 광장의 사람들에게 이 버스에 탈 사람 중 제일 먼저 탈 사람은 타세요 라고 했을 때 제일 먼저 달려간 사람이 타면 된다. 그런데 이 버스에 2번째로 탈 사람 타세요 라고 했을 때는 다 머뭇거리고 못탈 것이다. 누군가 한명이 먼저 가야 두번째로 갈 수 있다. **rownum은 무조건 1을 알아야 그 다음을 알 수 있게 설계되어있다. select 절에 꼭 rownum 추가할 필요는 없다**

```
ex) select rownum, ename, job, sal
from emp
where rownum between 2 and 4;
```

이 예제의 값을 보고 싶으면 인라인 뷰를 사용해야 한다. 위 값은 출력x

207) 우리반 테이블의 데이터를 가져오는데 위의 3건만 가져와서 출력하시오!

```
select *
from emp12
where rownum <= 3;
```

057 출력되는 행 제한하기 2(Simple TOP-n Queries)

simple top-n queries : order by 절까지 사용해서 검색하는 select 문의 출력되는 행의 일부를 가져올 때 사용하는 문법

```
ex) select rownum, empno, ename, sal
from emp
order by sal desc;
```

ROWNUM	EMPNO	ENAME	SAL
1	1	7839 KING	5000
2	12	7788 SCOTT	3000
3	10	7902 FORD	3000
4	4	7566 JONES	2975
5	2	7698 BLAKE	2850
6	3	7782 CLARK	2450
7	6	7499 ALLEN	1600
8	7	7844 TURNER	1500
9	14	7934 MILLER	1300
10	9	7521 WARD	1250
11	5	7654 MARTIN	1250
12	13	7876 ADAMS	1100
13	8	7900 JAMES	950
14	11	7369 SMITH	800

위 결과를 보면 order by 절이 수행되기 전에 rownum 이 수행되어 번호가 섞여있다

ex2) 사원 테이블에 월급이 높은 사원 4명만 출력하시오! 이름과 월급 출력

```
select ename, sal
from emp
where rownum <= 4
order by sal desc;
```

ENAME	SAL
1 KING	5000
2 JONES	2975
3 BLAKE	2850
4 CLARK	2450

이는 제대로 출력되지 않았다(sal이 제대로 출력되지 않은 결과) 이것은 실행 순서 때문인데, rownum 먼저 실행된 후 order by 절이 실행되었기 때문에 sal 이 높은 순서대로 4명이 아닌 rownum 이 1-4 까지 먼저 출력 후 그것을 정렬한 결과가 출력된 것으로 이해할 수 있다

ex2) 의 요구된 결과값 출력을 위해서는 top n query 를 사용하면 된다.

ex3) top n query

4 select ename, sal

1 from emp

2 order by sal desc

3 fetch first 4 rows only

fetch 는 검색으로 가져오는 데이터를 가져와라 하는 곳

first 4 rows only 는 문법 : 그중 4개의 행만 가져와라 하는 뜻

G : fetch first n rows only

★

208) 우리반에 나이가 가장 많은 학생들의 이름과 나이를 출력하는데 한 5명만 출력하시오!

select ename, age

from emp12

order by age desc

fetch first 5 rows only;

209) 직업, 직업별 토탈월급을 출력하는데 직업별 토탈월급이 높은것부터 출력하고 위쪽에 2개의 행만 출력하시오!

select job, sum(sal)

from emp

group by job

order by 2 desc

fetch first 2 rows only;

058 여러 테이블의 데이터를 조인해서 출력하기 1(EQUI JOIN)

join 을 이용하면 두개 이상의 테이블들의 컬럼들을 하나의 결과로 모아서 볼 수가 있다

ex1)

select *

from dept;

DEPTNO	DNAME	LOC
1	10 ACCOUNTING	NEW YORK
2	20 RESEARCH	DALLAS
3	30 SALES	CHICAGO
4	40 OPERATIONS	BOSTON

deptno : 부서번호 // dname : 부서명 // loc : location

ex2) 이름, 부서위치를 출력하시오! 이름은 emp table, 부서위치는 dept table

select ename, loc

from emp, dept; >> 이렇게 조회하면 14*4 = 56 row 로 출력된다(잘못출력한것)

그렇게 조회되는 이유는 사원이 어디서 근무하는지 data와 연관되어 있지 않기 때문

한편, (emp 와 dept) table 들에 공통으로 포함된 column 은 deptno 가 있다.

ex3) where절에 두 테이블의 연관성 작성

select ename, loc

from emp, dept

where emp.deptno = dept.deptno; >> 정상적으로 출력 // 3행 주목

어법형태 : where table1.column = table2.column

	ENAME	LOC
1	KING	NEW YORK
2	BLAKE	CHICAGO
3	CLARK	NEW YORK
4	JONES	DALLAS
5	MARTIN	CHICAGO
6	ALLEN	CHICAGO
7	TURNER	CHICAGO
8	JAMES	CHICAGO
9	WARD	CHICAGO
10	FORD	DALLAS
11	SMITH	DALLAS
12	SCOTT	DALLAS
13	ADAMS	DALLAS
14	MILLER	NEW YORK

★

210) 직업이 SALESMAN 인 직원들의 이름과 월급과 직업과 부서위치를 출력하시오!

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

※ 조인조건 emp.deptno = dept.deptno 을 where 절에 작성해야 조인가능

211) 월급이 2000이상인 직원들의 이름과 월급과 부서위치를 출력하시오!

```
select ename, sal, loc
      from emp, dept
      where emp.deptno = dept.deptno and sal >= 2000;
```

212) 위 결과에서 이름, 월급, 부서위치 옆에 부서번호도 같이 출력하시오!

```
select ename, sal, loc, emp.deptno
      from emp, dept
      where emp.deptno = dept.deptno and sal >= 2000;
```

※ 1행의 deptno 는 emp 와 dept 테이블 둘 다 존재하므로 emp.deptno 혹은 dept.deptno로 지정을 해 주어야 정상적으로 출력된다. 이를 하지 않을 시 err 발생.

이때, 1행을 select emp.ename, emp.sal, dept.loc, emp.deptno 와 같이 각각의 column을 table을 지정하면 검색속도가 더 향상된다. 따라서 212)을 만족하는 이상적인 SQL은

```
select emp.ename, emp.sal, dept.loc, emp.deptno
      from emp, dept
      where emp.deptno = dept.deptno and emp.sal >= 2000;
```

↓↓↓ 코드 간결하게 ↓↓↓

```
select e.ename, e.sal, d.loc, e.deptno
      from emp e, dept d
      where e.deptno = d.deptno and e.sal >= 2000;
```

※ 2행 emp 는 e 로, dept 는 d 로 별칭을 주어서 코딩을 하면 간결하게 조인문장 작성가능
이때 1행에 e 대신 emp 를 작성하면 err 발생. 이미 emp 는 e 로 별칭 지정되었기 때문

★

213) 월급이 1000에서 3000 사이인 직원들의 이름과 월급과 부서위치를 출력하시오!

```
select e.ename, e.sal, d.loc
      from emp e, dept d
      where e.deptno=d.deptno and e.sal between 1000 and 3000;
```

조인조건

검색조건

214) 사원번호가 7788, 7902, 7369번인 사원의 사원번호와 이름과 월급과 부서위치를 출력하시오!

```
select e.ename, e.sal, d.loc
```

```
from emp e, dept d
```

```
where e.deptno = d.deptno and e.empno in(7788, 7902, 7369);
```

※ 하나의 값을 비교할 때는 = 을 사용하지만 여러개의 값을 검색하여 비교시 in 사용

215) 이름의 첫 번째 철자가 S로 시작하는 사원의 이름과 월급과 부서위치를 출력하시오!

```
select e.ename, e.sal, d.loc
```

```
from emp e, dept d
```

```
where e.deptno = d.deptno and e.ename like 'S%';
```

※ 두 개 이상의 테이블을 조인(합치기)하려면 반드시 조인조건이 where절에 기술되어야 함

216) DALLAS 에서 근무하는 사원들의 이름과 직업과 부서위치를 출력하시오!

```
select e.ename, e.job, d.loc
```

```
from emp e, dept d
```

```
where e.deptno=d.deptno and d.loc = 'DALLAS';
```

217) 부서위치, 부서위치별 토탈월급을 출력하시오!

```
select d.loc, sum(e.sal) as sal
```

```
from emp e, dept d
```

```
where e.deptno=d.deptno
```

```
group by d.loc;
```

☆218) 부서위치, 부서위치별 평균월급을 출력하는데 소수점 이하는 안나오게 반올림하고 부서위치별 평균월급이 높은 것부터 출력하고 부서위치별 평균월급이 출력될때에 천단위를 부여하시오!

```
select d.loc, to_char( round(avg(e.sal) ), '999,999') as asal
```

```
from emp e, dept d
```

```
where e.deptno=d.deptno
```

```
group by d.loc
```

```
order by 2 desc;
```


##1103

2020년 11월 3일 화요일 오전 9:46

1102R

1. 기본 select 문장 6가지 절

5 select 컬럼명

1 from 테이블명

2 where 검색조건

3 group by 그룹핑할 컬럼

4 having 그룹함수만든 검색조건

6 order by 정렬할 컬럼명

2. 함수

1) 단일함수 : 문자, 숫자, 날짜, 변환, 일반

2) 복수행 함수 : max, min, avg, sum, count

현업에서 많이 보는 검색 결과들은 주로 데이터 분석함수를 이용한 결과들이 많았다.

3. 데이터 분석함수

1) rank

2) dense_rank

3) ntile

4) cume_dist

5) listagg

6) report_to_ratio

7) lag, lead

8) sum(column) over(condition) : 데이터 누적

9) rollup, cube

10) grouping sets

4. 조인(join) 문장

: 하나의 테이블에서 얻을 수 있는 정보가 아닌 여러개의 테이블에서 얻을 수 있는 정보를 하나의 결과로 보여주기 위해서 만든 문법

```
ex1) select e.ename, d.loc
      from emp e, dept d
      where e.deptno = d.deptno and d.loc = 'DALLAS';
           조인조건           검색조건
```

1103>>

★

219) 월급이 2700 이상인 직원들의 이름과 월급과 부서위치를 출력하시오!

```
select e.ename, e.sal, d.loc
      from emp e, dept d
      where e.deptno=d.deptno and e.sal >= 2700;
```

※테이블 별칭시 이점 : 1)성능향상 2)SQL 유지보수가 원활

220) 이름의 끝글자가 T 로 끝나는 직원들의 이름과 월급과 부서위치와 부서명을 출력하시오!

```
select e.ename, e.sal, d.loc, d.dname
      from emp e, dept d
      where e.deptno = d.deptno and e.ename like '%T';
```

221) 직업이 SALESMAN 이고 월급이 1200 이상인 직원들의 이름과 직업과 부서위치와 월급을 출력하시오!

```
select e.ename, e.job, d.loc, e.sal
  from emp e, dept d
 where e.deptno=d.deptno and e.job = 'SALESMAN' and e.sal >= 1200;
```

222) 부서위치, 부서위치별 토달월급을 출력하는데 DALLAS 는 제외하고 출력하시오!

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

223) 지금 출력된 결과를 다시 출력하는데 토달월급이 높은것부터 출력하시오!

```
select d.loc, sum(e.sal)
  from emp e, dept d
 where e.deptno=d.deptno and d.loc != 'DALLAS'
 group by d.loc
 order by 2 desc;
```

059 여러 테이블의 데이터를 조인해서 출력하기 2(NON EQUI JOIN)

non equi join : 조인하려는 두 개의 테이블 사이에 공통된 컬럼이 없었을 때 사용하는 조인문법

ex1) salgrade 테이블 생성 : salgrade data file 이용 다른 pc에도 적용

	GRADE	LOSAL	HISAL
1	1	700	1200
2	2	1201	1400
3	3	1401	2000
4	4	2001	3000
5	5	3001	9999

grade : 급여 등급

losal : 최소월급

hisal : 최대월급

ex2) 이름, 월급, 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;
```

	ENAME	SAL	GRADE
1	SMITH	800	1
2	JAMES	950	1
3	ADAMS	1100	1
4	WARD	1250	2
5	MARTIN	1250	2
6	MILLER	1300	2
7	TURNER	1500	3
8	ALLEN	1600	3
9	CLARK	2450	4
10	BLAKE	2850	4
11	JONES	2975	4
12	FORD	3000	4
13	SCOTT	3000	4
14	KING	5000	5

★

224) ex2)에서 등급이 3등급인 직원들만 출력하시오 !

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

※ 조인 문법의 종류

1. 오라클 조인 문법

- 1) equi join : 두 개의 테이블 사이에 공통된 컬럼을 사용
- 2) non equi join : 두 개의 테이블 사이에 공통된 컬럼이 없을 때 사용
- 3) outer join : 두 개의 테이블의 공통된 컬럼은 있으나 조인하려는 컬럼의 데이터가 서로 일치하지 않을 때 사용
- 4) self join : 자기 자신의 테이블과 조인할 때 사용

2. 1999 ansi 조인 문법

- 1) ansi : american national standard institute 의 약어

225) 급여등급(grade), 급여등급별로 해당하는 직원들의 이름을 가로로 출력하시오

GRADE	LISTAGG(ENAME, ',' WITHINGROUP ORDER BY E.ENAME ASC)
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 asc)
from emp e, salgrade s
where e.sal between s.losal and hisal
group by s.grade;
```

226) 위 결과에서 월급도 옆에 같이 나오게 하시오!

GRADE	NAME
1	ADAMS(1100), JAMES(950), SMITH(800)
2	MARTIN(1250), MILLER(1300), WARD(1250)
3	ALLEN(1600), TURNER(1500)
4	BLAKE(2850), CLARK(2450), FORD(3000), JONES(2975), SCOTT(3000)
5	KING(5000)

```
select s.grade, listagg(e.ename || '(' || e.sal || ')', ',')
within group (order by e.ename asc) name
from emp e, salgrade s
where e.sal between s.losal and hisal
group by s.grade;
```

060 여러 테이블의 데이터를 조인해서 출력하기 3(OUTER JOIN)

outer join : 조인하려는 두 테이블의 공통의 컬럼인 deptno 의 데이터가 서로 똑같이 일치하지 않을 때 조인하기 위해 사용

ex1) 사원 테이블에서 부서번호를 출력하는데 중복제거해서 출력하시오!

```
select distinct deptno
from emp;
```

ex2) 부서(dept) 테이블에서 부서번호를 출력하시오!

```
select deptno
from dept;
```

★

227) emp와 dept 를 서로 조인해서 이름과 부서위치와 부서번호를 출력하시오!

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

※ 227) 에서 40번은 emp 테이블의 deptno 에 없어서 조인이 되지 않았고 출력 x

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

EMPNO	ENAME	LOC	DEPTNO
1	KING	NEW YORK	10
2	BLAKE	CHICAGO	30
3	CLARK	NEW YORK	10
4	JONES	DALLAS	20
5	MARTIN	CHICAGO	30
6	ALLEN	CHICAGO	30
7	TURNER	CHICAGO	30
8	JAMES	CHICAGO	30
9	WARD	CHICAGO	30
0	FORD	DALLAS	20
1	SMITH	DALLAS	20
2	SCOTT	DALLAS	20
3	ADAMS	DALLAS	20
4	MILLER	NEW YORK	10
5	(null)	BOSTON	40

outer join sign (+) 는 결과로 출력될 때 데이터가 모자란 쪽에 붙어준다. emp 테이블에서는 40번 부서번호가 없고 dept 테이블에는 존재하므로 emp 테이블 쪽에 outer join sign(+)을 사용

228) 부서위치, 부서위치별 토탈월급을 출력하시오

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

229) 위 결과에서 BOSTON 도 나오게끔 조인문법을 변경하시오!

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

LOC	SUM(E SAL)
1 NEW YORK	8750
2 DALLAS	10875
3 CHICAGO	9400
4 BOSTON	(null)

※outer join은 두 개의 행에 동시에 사용하면 err 발생. 다른 방법으로 시도해야 함

※우리반 테이블과 조인하기 위한 테이블 생성(통신사 기본 요금 테이블)

TELECOM	PRICE	SERVICE_CNT
1 sk	18500	9
2 kt	17000	9
3 lg	18000	10

☆230) 우리반 테이블과 telecom_price 테이블을 조인해서 이름, 나이, 성별, 통신사, 통신사 기본요금(price)을 출력하는데 나이가 27이상인 학생들만 출력되게 하시오!

```
select e.ename, e.age, e.gender, e.telecom, t.price
  from emp12 e, telecom_price t
 where e.telecom=t.telecom and e.age >=27;
```

EMPNO	ENAME	AGE	GENDER	TELECOM	PRICE
1	이준혁	29	남	kt	17000
2	한광	31	남	kt	17000
3	전지현	35	여	sk	18500
4	심가람	29	남	sk	18500
5	유재영	28	여	sk	18500
6	김정민	28	남	lg	18000
7	김정현	29	여	kt	17000
8	김미숙	27	여	lg	18000
9	김예민	27	여	kt	17000
10	신현용	27	남	lg	18000
11	김수준	33	남	sk	18500
12	허정민	30	여	sk	18500
13	이신성	27	남	sk	18500
14	황나연	27	여	kt	17000
15	김준환	29	남	kt	17000
16	송종미	29	여	kt	17000
17	김소라	29	여	kt	17000
18	최세민	36	남	sk	18500
19	김우진	44	여	lg	18000
20	박재진	28	여	sk	18500
21	허상준	29	남	kt	17000
22	홍재현	28	여	kt	17000

061 여러 테이블의 데이터를 조인해서 출력하기 4(SELF JOIN)

self join : 자기 자신의 테이블과 조인하는 조인 문법

필요성 : 사원 테이블을 예를 들면 사원이름과 그 사원을 관리하는 관리자의 이름을 하나의 결과로 볼 수 있기 때문

ex1) 사원번호, 사원이름, 관리자 번호(mgr) 를 출력하세요!

select empno, ename, mgr

from emp;

	EMPNO	ENAME	MGR
1	7839	KING	(null)
2	7698	BLAKE	7839
3	7782	CLARK	7839
4	7566	JONES	7839
5	7654	MARTIN	7698
6	7499	ALLEN	7698
7	7844	TURNER	7698
8	7900	JAMES	7698
9	7521	WARD	7698
10	7902	FORD	7566
11	7369	SMITH	7902
12	7788	SCOTT	7566
13	7876	ADAMS	7788
14	7934	MILLER	7782

위 결과를 보면 mgr 의 data 는 해당 row 의 해당하는 관리자를 나타낸다(empno와 연계가능)
empno(ename)[mgr]의 형식으로 보면..

7369(SMITH)[7902] 의 관리자는 7902(FORD)[7566]

7902(FORD)[7566]의 관리자는 7566(JONES)[7839]

7566(JONES)[7839]의 관리자는 7839(KING)[null] >> MGR과 EMPNO가 서로 연계

ex2) 사원이름, 해당 사원의 관리자의 이름을 출력하시오!

select 사원.ename, 관리자.ename

from emp 사원, emp 관리자

where 사원.mgr = 관리자.empno;

	ENAME	ENAME_1
1	BLAKE	KING
2	CLARK	KING
3	JONES	KING
4	MARTIN	BLAKE
5	ALLEN	BLAKE
6	TURNER	BLAKE
7	JAMES	BLAKE
8	WARD	BLAKE
9	MILLER	CLARK
10	FORD	JONES
11	SCOTT	JONES
12	SMITH	FORD
13	ADAMS	SCOTT

self join 의 핵심은 where 절의 join 조건 설정인데 테이블 내에서 서로 연관시키고자 하는 항목, 위 예시에서는 mgr 과 empno 가 일치하는 경우에서의(mgr=empno) ename 이 각각 출력된다. 그리고 별칭은 나는 영어 1~2단어로 설정하는 게 좋다고 생각함(한/영 바꾸다 오타날 확률)

★

231) 위의 결과를 다시 출력하는데 컬럼명을 한글로 사원, 관리자라고 출력되게 하시오!

단 위의 결과 : 사원이름, 해당 사원의 관리자의 이름 출력

select 사원.ename 사원, 관리자.ename 관리자

from emp 사원, emp 관리자

where 사원.mgr = 관리자.empno;

232) 사원이름, 사원월급, 관리자이름, 관리자의 월급을 출력하시오!

select s.ename, s.sal, m.ename as "M ename", m.sal as "M sal"

from emp s, emp m

where s.mgr = m.empno;

233) 위의 결과를 다시 출력하는데 사원의 월급이 관리자의 월급보다 더 큰 사원들만 출력하시오

select s.ename, s.sal, m.ename as "M ename", m.sal as "M sal"

from emp s, emp m

where s.mgr = m.empno and s.sal > m.sal;

234) 관리자보다 먼저 입사한 직원들의 사원 이름과 사원 입사일, 관리자 이름과 관리자 입사일을 출력하시오!

select s.ename, s.hiredate, m.ename as "M ENAME", m.hiredate as "M HIREATE"

from emp s, emp m

where s.mgr = m.empno and s.hiredate < m.hiredate;

235) 관리자이름을 출력하고 그 옆에 해당 관리자에 속한 직원들의 이름을 가로로 출력하시오!

```
select m.ename as "M ENAME", listagg(s.ename, ',') within group (order by s.ename asc) as ename
  from emp s, emp m
  where s.mgr = m.mgr
  group by m.ename;
```

236) 직업, 이름, 월급, 순위를 출력하는데 순위가 월급이 높은 순서대로 순위를 부여하시오!

```
select job, ename, sal, dense_rank() over(order by sal desc) as rank
  from emp;
```

237) 위 결과를 다시 출력하는데 직업별로 각각 순위가 부여되게 하시오!

```
select job, ename, sal, dense_rank() over(partition by job
                                         order by sal desc) as rank
  from emp;
```

238) 부서위치, 이름, 월급, 순위를 출력하는데 부서위치별로 각각 월급이 높은 순서대로 순위를 부여하시오!

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

239) 부서위치, 부서위치별로 속한 직원들의 이름을 가로로 출력하시오!

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

240) 부서위치, 부서위치별 토달월급을 출력하는데 맨 아래에 전체 토달월급이 출력되게 하시오!

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

>>2 grouping sets 이용 가능

241) 직업, 직업별 최대월급, 직업별 최소월급, 직업별 평균월급을 출력하시오!

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

242) 부서위치, 부서위치별 최대월급, 부서위치별 최소월급, 부서위치별 인원수를 출력하시오!

```
select d.loc, max(e.sal), min(e.sal), count(e.empno)
  from emp e, dept d
  where e.deptno = d.deptno
  group by d.loc;
```

※count () 안에 * 넣는 것이 제일 확실함

062 여러 테이블의 데이터를 조인해서 출력하기 5(ON절)

join 문법

1) oracle join : equi join, non equi join, outer join, self join

2) 1999 ANSI join문법

- on
- using
- natural join
- left/right/full outer join
- cross join

ex1) oracle equi join

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

ex2) 1999ANSI join : on 절 사용

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

★

243) 이름과 월급과 부서위치를 출력하는데 월급이 2400 이상인 직원들만 출력하시오!

단, on절을 사용한 조인 문법으로 시행

```
select e.ename, e.sal, d.loc
      from emp e join dept d
     on (e.deptno = d.deptno)
     where e.sal >= 2400;
```

※1999ANSI join 문법(on)은 from 다음 on에서 조인조건을 기입하고 다음 where 절 기입

244) DALLAS 에서 근무하는 직원들의 이름과 월급과 부서위치와 직업을 출력하는데 ON절을 사용한 조인문법으로 수행하시오!

```
select e.ename, e.sal, d.loc, e.job
      from emp e join dept d
     on (e.deptno = d.deptno)
     where d.loc = 'DALLAS';
```

245) emp 테이블과 salgrade 테이블을 서로 조인해서 이름, 월급, 급여등급(grade) 를 출력하는데 2등급만 출력되게 하고 on절을 사용한 조인문법으로 수행하시오!

```
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 = 2;
```

※non-equi join 참고

063 여러 테이블의 데이터를 조인해서 출력하기 5(USING절)

using

ex1) **select e.ename, d.loc
 from emp e join dept d
 using (deptno) ;**

	ENAME	LOC
1	KING	NEW YORK
2	BLAKE	CHI CAGO
3	CLARK	NEW YORK
4	JONES	DALLAS
5	MARTIN	CHI CAGO
6	ALLEN	CHI CAGO
7	TURNER	CHI CAGO
8	JAMES	CHI CAGO
9	WARD	CHI CAGO
10	FORD	DALLAS
11	SMITH	DALLAS
12	SCOTT	DALLAS
13	ADAMS	DALLAS
14	MILLER	NEW YORK

연결고리가 되는 컬럼명을 테이블 별칭없이 사용해야 함. (3행의 deptno) 출력결과on과 동일

★

246) 직업이 SALESMAN 인 직원들의 이름과 월급과 직업과 부서위치를 출력하는데 using 절을 사용한 조인문법으로 수행하시오!

```
select e.ename, e.sal, e.job, d.loc
      from emp e join dept d
      using (deptno)
      where e.job = 'SALESMAN';
```

064 여러 테이블의 데이터를 조인해서 출력하기 6(NATURAL JOIN)

natural join

ex1) select e.ename, d.loc
 from emp e natural join dept d

	ENAME	LOC
1	KING	NEW YORK
2	BLAKE	CHI CAGO
3	CLARK	NEW YORK
4	JONES	DALLAS
5	MARTIN	CHI CAGO
6	ALLEN	CHI CAGO
7	TURNER	CHI CAGO
8	JAMES	CHI CAGO
9	WARD	CHI CAGO
10	FORD	DALLAS
11	SMITH	DALLAS
12	SCOTT	DALLAS
13	ADAMS	DALLAS
14	MILLER	NEW YORK

where 절 없이 간단하게 join 이 가능함

오라클이 알아서 두 테이블 사이에 공통된 컬럼이 있는지 찾아보고 조인함

※Join문법 중요도

1)oracle join : 4가지 전부 잘 사용

2)1999 ansi join : on절, outer join 두개 중요(자주 사용)

065 여러 테이블의 데이터를 조인해서 출력하기 7(LEFT/RIGHT OUTER JOIN)

outer join

ex1) oracle join(outer join)

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

ex2) 1999 ansi join

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


	ENAME	LOC
1	KING	NEW YORK
2	BLAKE	CHI CAGO
3	CLARK	NEW YORK
4	JONES	DALLAS
5	MARTIN	CHI CAGO
6	ALLEN	CHI CAGO
7	TURNER	CHI CAGO
8	JAMES	CHI CAGO
9	WARD	CHI CAGO
10	FORD	DALLAS
11	SMITH	DALLAS
12	SCOTT	DALLAS
13	ADAMS	DALLAS
14	MILLER	NEW YORK
15	(null)	BOSTON

ex1) 과 ex2) 의 결과값은 서로 동일하다

ex1)의 4행과 ex2)의 2행은 서로 동일한 의미이다

ex3) data 추가 >> emp table_jack_data 추가

```
insert into emp(empno, ename, sal, job, deptno)
values(1221, 'jack', 3500, 'SALESMAN', 70);

commit;
```

위 data 를 추가 후 dept 테이블에는 70번 부서가 없기 때문에 조인할 때 equi 조인을 하게 되면 결과가 안나오고 outer join 을 사용해야 결과값을 얻을 수 있다.

★

247) 아래 결과를 1999 ansi 문법으로 구현하시오!

	ENAME	LOC
1	KING	NEW YORK
2	CLARK	NEW YORK
3	MILLER	NEW YORK
4	JONES	DALLAS
5	FORD	DALLAS
6	SMITH	DALLAS
7	SCOTT	DALLAS
8	ADAMS	DALLAS
9	BLAKE	CHI CAGO
10	MARTIN	CHI CAGO
11	ALLEN	CHI CAGO
12	TURNER	CHI CAGO
13	JAMES	CHI CAGO
14	WARD	CHI CAGO
15	jack	(null)

```
select e.ename, d.loc
```

```
from emp e left outer join dept d
on ( e.deptno = d.deptno);
```

066 여러 테이블의 데이터를 조인해서 출력하기 8(FULL OUTER JOIN)

full outer join

ex1) select e.ename, d.loc
from emp e, dept d
where e.deptno (+) = d.deptno(+); >> 이 경우 err 발생

위 ex1)의 결과 출력을 가능하게 하는 join 문법이 1999 ansi 의 full outer join 이다

ex2) select e.ename, d.loc
from emp e full outer join dept d
on (e.deptno = d.deptno); >> 양쪽으로 출력이 가능하게 해주는 join 문법

	ENAME	LOC
1	jack	(null)
2	KING	NEW YORK
3	BLAKE	CHI CAGO
4	CLARK	NEW YORK
5	JONES	DALLAS
6	MARTIN	CHI CAGO
7	ALLEN	CHI CAGO
8	TURNER	CHI CAGO
9	JAMES	CHI CAGO
10	WARD	CHI CAGO
11	FORD	DALLAS
12	SMITH	DALLAS
13	SCOTT	DALLAS
14	ADAMS	DALLAS
15	MILLER	NEW YORK
16	(null)	BOSTON

CROSS Join

cross join : 1999 ansi 문법의 한 종류로, where 절 없이 조인하여 전체를 다 조인하는 문법

ex1) 오라클 조인 문법

```
select e.ename, d.loc
from emp e, dept d; >> jack 까지 60 개의 row 생성
```

ex2) 1999 ansi 문법

```
select e.ename, d.loc
```

from emp e cross join dept d; >> ex1) 과 동일한 결과 출력

★

248) 우리반 테이블과 telecom_price 테이블과 조인을 해서 이름이 김정민 학생의 이름과 나이와 통신사와 통신요금(price) 를 출력하시오!

>>1 oracle join 사용

```
select e.ename, e.age, e.telecom, t.price
  from emp12 e, telecom_price t
 where e.telecom = t.telecom and e.ename = '김정민';
```

>>2 1999 ansi join 사용

```
select e.ename, e.age, e.telecom, t.price
  from emp12 e join telecom_price t
 on( e.telecom = t.telecom )
 where e.ename = '김정민';
```

249) 나이가 28 이상인 학생들의 이름과 나이와 통신사와 통신요금(price) 를 출력하시오!

```
select e.ename, e.age, e.telecom, t.price
  from emp12 e, telecom_price t
 where e.telecom = t.telecom and e.age >= 28;
```

☆250) 부서위치, 부서위치별 토탈월급을 가로로 출력하시오!

NEW YORK	DALLAS	CHICAGO
8750	10875	9400

```
select sum(decode(d.loc, 'NEW YORK', e.sal ) ) as "NEW YORK",
       sum(decode(d.loc, 'DALLAS', e.sal ) ) as "DALLAS",
       sum(decode(d.loc, 'CHICAGO', e.sal ) ) as "CHICAGO"
  from emp e, dept d
 where e.deptno = d.deptno;
```

>>2 pivot 으로 수행하기

```
select *
  from ( select d.loc, e.sal
        from emp e, dept d
        where e.deptno=d.deptno)
 pivot( sum(sal) for loc in ('NEW YORK' as "NEW YORK",
                           'DALLAS' as "DALLAS",'CHICAGO' as"CHICAGO") );
```

※pivot문

위의 결과를 보기 위해 필요한 raw data 는 부서위치와 월급. from 절에 부서위치와 월급 컬럼만 가져오는 쿼리문을 작성해주면 >>1 과 동일한 결과를 얻을 수 있다.

from 절 괄호안의 조인쿼리문은 결과를 보기위한 raw data이다. from 절의 조인 쿼리문에서는 d.loc라고 작성했지만 실제로 출력되는 컬럼명은 loc 이므로 pivot 문에서 d.loc 가 아닌 loc 로 작성해주어야 한다.

##1104

2020년 11월 4일 수요일 오전 9:33

067 집합 연산자로 데이터를 위아래로 연결하기 1(UNION ALL)

데이터를 연결해서 출력하는 방법 2가지

1. join : 데이터를 양옆으로 연결해서 출력
2. 집합연산자 : 데이터를 **위아래로** 연결해서 출력

집합 연산자의 종류 4가지

- 1) union all
- 2) union
- 3) intersect
- 4) minus

ex1) 직업, 직업별 토탈월급을 세로로 출력하시오!

```
select job, sum(sal)
```

```
from emp
```

```
group by job;
```

ex2) 전체 토탈월급을 출력하시오!

```
select sum(sal)
```

```
from emp;
```

ex2-2)

```
select '전체토탈:' as job, sum(sal)
```

```
from emp
```

JOB	SUM(SAL)
1 전체토탈:	32525

ex3)

위(ex1, ex2-2)의 SQL을 하나로 합쳐서 데이터가 위아래로 출력되게 하시오!

```
select job, sum(sal)
```

```
from emp
```

```
group by job
```

```
union all
```

```
select '전체토탈:' as job, sum(sal)
```

```
from emp;
```

JOB	SUM(SAL)
1 SALESMAN	9100
2 CLERK	4150
3 ANALYST	6000
4 MANAGER	8275
5 PRESIDENT	5000
6 전체토탈:	32525

※**union all** 로 ex1)쿼리문의 결과와 ex2-2)쿼리문의 결과를 하나로 합쳐로 출력.

집합연산자 사용시 주의사항

- 1) 집합연산자 위 아래 쿼리문의 컬럼의 갯수가 동일해야 함
- 2) 집합연산자 위 아래 쿼리문의 컬럼의 데이터 타입(숫자형, 문자형, 날짜형)도 동일해야 함
- 3) 집합연산자 위 아래의 쿼리문의 컬럼의 컬럼명이 동일해야 함
- 4) **order by** 절은 맨 아래 쿼리문에만 작성할 수 있음

★

251) 부서번호, 부서번호별 토탈월급을 출력하는데 맨 아래에 전체토탈 월급이 출력되게 하시오!

>>1 union all

```

select to_char(deptno) as deptno, sum(sal)
  from emp
  group by deptno
union all
select '전체토탈:' as deptno, sum(sal)
  from emp;
>>2 rollup
select deptno, sum(sal)
  from emp
  group by rollup(deptno);

```

252) 입사한 년도(4자리), 입사한 년도별 토탈월급을 출력하시오!

```

select to_char(hiredate, 'RRRR'), sum(sal)
  from emp
  group by to_char(hiredate, 'RRRR');

```

253) 입사한 년도(4자리), 입사한 년도별 토탈월급을 출력하는데 맨 아래에 전체 토탈월급이 출력되게 하시오! 단, union all 을 사용하시오!

```

select to_char(hiredate, 'RRRR') as hire_year, sum(sal)
  from emp
  group by to_char(hiredate, 'RRRR')
union all
select '토탈월급' as hire_year, sum(sal)
  from emp;

```

※union all 의 주의사항을 항상 고려해서 SQL 을 작성해야 한다

254) 우리반 테이블에서 통신사, 통신사별 인원수를 출력하시오!

```

select telecom, count(*)
  from emp12
  group by telecom;

```

255) 아래와 같이 전체인원수가 맨 아래에 출력되게 하시오!

```

select telecom, count(*)
  from emp12
  group by telecom
union all
select '전체:' as telecom, count(*)
  from emp12;

```

256) 문제 253번의 결과를 정렬해서 출력하시오! 입사년도가 빠른 순으로 정렬하시오!

```

select to_char(hiredate, 'RRRR') as hire_year, sum(sal)
  from emp
  group by to_char(hiredate, 'RRRR')
union all
select '토탈월급' as hire_year, sum(sal)
  from emp

```

order by 1 asc;

※union all 활용시 order by 절은 항상 맨 아래의 쿼리문에만 사용가능함!

068 집합 연산자로 데이터를 위아래로 연결하기 2(UNION)

union : union all 과 같은 합집합 연산자이나 union 은 order by 절을 사용하지 않아도 정렬을 암시적으로 수행한다.
또한 중복된 데이터를 하나로 출력(distinct기능) 한다.

ex1) 256)에서 union all 을 union 을 사용하면 정렬된 값을 출력한다

	HIRE_YEAR	SUM(SAL)
1	1980	800
2	1981	22825
3	1982	4300
4	1983	1100
5	토탈월급	29025

★

257) 부서번호, 부서번호별 토탈월급을 출력하는데 맨 아래에 전체 토탈월급이 출력되게 하고 부서번호는 10, 20, 30번 순으로 정렬해서 출력되게 하시오!

```
select to_char(deptno) as deptno, sum(sal)
```

```
from emp
```

```
group by deptno
```

```
union
```

```
select '토탈월급' as deptno, sum(sal)
```

```
from emp;
```

※union 은 암시적으로 정렬 작업을 수행하기 때문에 굳이 정렬된 결과를 볼 필요가 없다면 union all 을 활용하는게 더 검색 성능에 유리하다

258) 부서위치, 부서위치별 토탈월급을 출력하시오! (단, 세로로 출력하시오)

```
select d.loc as loc, sum(e.sal)
```

```
from emp e, dept d
```

```
where e.deptno=d.deptno
```

```
group by d.loc;
```

259) 맨 아래에 전체 토탈월급도 출력되게 하시오!

```
select d.loc as loc, sum(e.sal)
```

```
from emp e, dept d
```

```
where e.deptno=d.deptno
```

```
group by d.loc
```

```
union all
```

```
select '토탈월급:' as loc, sum(e.sal)
```

```
from emp e;
```

260) 위의 결과를 다시 출력하는데 장소를 ABCD 순서대로 정렬해서 출력되게 하시오!

259) 5행의 union all 대신 union 을 사용

261) 직업, 직업별 최대월급, 직업별 최소월급, 직업별 평균월급, 직업별 인원수를 출력하시오!

```
select job, max(sal), min(sal), avg(sal), count(*)
```

```
from emp
```

```
group by job;
```

262) 사원 테이블에서 최대월급, 최소월급, 평균월급, 전체 인원수를 출력하시오!

```
select max(sal), min(sal), avg(sal), count(*)  
from emp;
```

263) 문제 261의 결과와 262의 결과를 위아래로 연결해서 출력하시오!

```
select job, max(sal), min(sal), avg(sal), count(*)  
from emp  
group by job  
union all  
select '전체' as job, max(sal), min(sal), avg(sal), count(*)  
from emp;
```

☆264) 위의 결과에서 숫자에 전부 천단위가 부여되게 하시오!

```
select job, to_char(max(sal), '999,999'), to_char(min(sal), '999,999'),  
        to_char(avg(sal), '999,999'), count(*)  
from emp  
group by job  
union all  
select '전체' as job, to_char(max(sal), '999,999'), to_char(min(sal), '999,999'),  
        to_char(avg(sal), '999,999'), count(*)  
from emp;
```

※avg 에서 '999,999' 에 '999,999.999' 로 전환하면 소수점 이하도 출력된다

069 집합 연산자로 데이터의 교집합을 출력하기(INTERSECT)

집합 연산자 종류

- 1) 합집합 연산자 : union all, union
- 2) 교집합 연산자 : intersect
- 3) 차집합 연산자 : minus

※여담

데이터분석, 딥러닝회사 : SQL 쪽지시험, 파이썬 코딩시험

제약회사 데이터 분석가 : 수학시험도 포함

intersect

ex1) 테이블 생성 : 우리반 테이블 백업 : emp12_backup

```
select *  
from emp12_backup >> 으로 백업자료를 확인할 수 있음
```

ex2) 테이블 생성 : 김씨 자료 : emp12_backup2

ex3) union all

```
select ename, age, telecom  
from emp12
```

union all

```
select ename, age, telecom  
from emp12_backup2; >>> 38개의 행 생성
```

ex4) union

```
select ename, age, telecom  
from emp12
```

union

select ename, age, telecom

from emp12_backup2; >>> 30개의 행 생성

※union 은 합집합 연산자이지만 중복된 데이터를 제거(distinct)하고 정렬작업도 수행함

ex5) intersect : 교집합 생성 : 아래 예시에서는 emp12_backup2 에 해당하는 부분이 교집합

select ename, age, telecom

from emp12

intersect

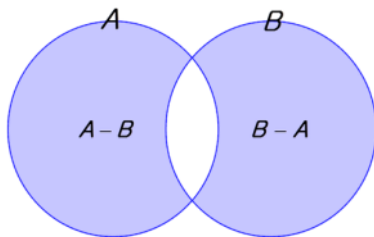
select ename, age, telecom

from emp12_backup2;

	ENAME	AGE	TELECOM
1	김미승	27	lg
2	김소라	29	kt
3	김승순	33	sk
4	김예린	27	kt
5	김정민	28	lg
6	김정원	28	kt
7	김주원	44	lg
8	김홍비	24	kt

070 집합 연산자로 데이터의 차이를 출력하기(MINUS)

minus : 차집합을 구하는 집합 연산자



ex1) minus

select ename, age, telecom

from emp12

minus

select ename, age, telecom

from emp12_backup2; >>> '김'씨 성 빼고 다 출력됨(차집합이 정상적으로 출력됨)

071 서브 쿼리 사용하기 1(단일행 서브쿼리)

ex1) 사원 테이블에서 최대월급을 받는 사원의 이름과 월급을 출력하시오!

서브 쿼리 : ex1) 의 SQL을 수행하기 위해 필요함

ex2) JONES 의 월급을 출력하시오!

select ename, sal

from emp

where ename = 'JONES';

	ENAME	SAL
1	JONES	2975

ex3) JONES 월급보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하시오!

select ename, sal

from emp

where sal > 2975; >>> JONES의 월급을 알고 있을 때 사용할 수 있는 SQL

ex4) **main query** / **sub query**

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

※쿼리를 ex2) 와 ex3) 을 나누어서 실행하지 않고 한번에 실행하는 것이 sub query 이다

blue = main query

red = sub query

★

265) SCOTT 과 같은 월급을 받는 직원들의 이름과 월급을 출력하시오! (SCOTT포함/포함X)

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

	ENAME	SAL
1	FORD	3000
2	SCOTT	3000

※위 경우, oracle 내부적으로 sub query 부터 수행하기 때문에 SCOTT 도 같이 출력된다.

그럴때는 **main query** 에 and 이후 조건을 추가해 주면 된다. 아래 SQL을 보자

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

266) SMITH 와 직업이 같은 직원들의 이름과 직업을 출력하는데 SMITH 는 제외하고 출력!

```
select ename, job
  from emp
 where job = (select job
              from emp
              where ename = 'SMITH')
       and ename != 'SMITH';
```

267) ALLEN 보다 늦게 입사한 직원들의 이름과 입사일을 출력하시오!

```
select ename, hiredate
  from emp
 where hiredate > (select hiredate
                  from emp
                  where ename = 'ALLEN');
```

268) 직업이 SALESMAN 인 사원의 최대월급보다 더 많은 월급을 받는 직원들의 이름과 월급을 출력하시오!

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

```

from emp
where job = 'SALESMAN');

```

269) 최대월급을 받는 사원의 이름과 월급을 출력하시오!

```

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

```

※suq query 에는 main query 의 where 검색조건에 맞는 데이터 형식을 따라주기만 하면 큰 err 없이 SQL의 결과를 출력할 수 있다. 그런데 **order by** 절은 **union** 과 마찬가지로 가장 마지막 순서로 코딩해야 하나?? **필요한곳에 하면됨**

270) 전공에 통계가 포함되어져 있는 학생들중에서의 최대나이인 학생의 이름과 나이와 전공을 출력하시오!

```

select ename, age, major
from emp12
where age = (select max(age)
from emp12
where major like '%통계%')
and major like '%통계%';

```

271) KING 에게 보고하는 사원들의 이름을 출력하시오!

단, KING 에게 보고하는 사원들은 KING 의 직속 부하 사원들이다 / mgr 활용)
mgr = 관리자번호, 즉 어떤 사원의 mgr 은 그 사원의 관리자의 empno 이다

```

select ename
from emp
where mgr = (select empno
from emp
where ename = 'KING');

```

072 서브 쿼리 사용하기 2(다중 행 서브쿼리)

서브쿼리의 종류

1) 단일행 서브쿼리 : 서브쿼리에서 메인쿼리로 하나의 값이 리턴되는 경우

연산자 : =, <, >, <=, >=, !=, <>, ^=

2) 다중행 서브쿼리 : 서브쿼리에서 메인쿼리로 여러개의 값이 리턴되는 경우

연산자 : in, not in, >all, <all, >any, <any

3) 다중컬럼 서브쿼리 : 서브쿼리에서 메인쿼리로 여러개의 컬럼값이 리턴되는 경우

ex1) 직업이 SALESMAN 인 사원들과 월급이 같은 사원들의 이름과 월급을 출력하시오!

```

select ename, sal
from emp
where sal = (select sal
from emp
where job = 'SALESMAN'); >>> 아래와 같은 err발생, = 대신 in 사용

```

ORA-01427: 단일 행 하위 질의에 2개 이상의 행이 리턴되었습니다.
01427. 00000 - "single-row subquery returns more than one row"

위와 같은 err 가 발생하는 이유는 서브쿼리의 값이 여러개의 행으로 리턴되기 때문.

따라서 main 과 sub 을 연결하는 연산자 = 대신 in 을 사용해야 값이 정상적으로 출력된다

★

272) 통신사가 SK 인 학생들과 나이가 같은 학생들의 이름과 나이와 통신사를 출력하시오!

```
select ename, age, telecom
  from emp12
 where age in (select age
               from emp12
               where telecom = 'sk');
```

273) 통신사가 sk 인 학생들과 나이가 같지 않은 학생들의 이름과 나이와 통신사를 출력하시오!

```
select ename, age, telecom
  from emp12
 where age not in (select age
                   from emp12
                   where telecom = 'sk');
```

073 서브 쿼리 사용하기 3(NOT IN)

ex1) 관리자인 직원들의 이름을 출력하시오! 단 자기 밑에 직속부하가 1명이라도 있는 직원.

```
select ename
  from emp
 where empno in (select mgr
                 from emp
                 where mgr is not null);
```

ENAME
1 KING
2 BLAKE
3 JONES
4 FORD
5 SCOTT
6 CLARK

★

274) 관리자가 아닌 직원들의 이름을 출력하시오! (관리자인 직원이 6명이 나왔으므로 관리자가 아닌 직원들이 8명이 나와야 함)

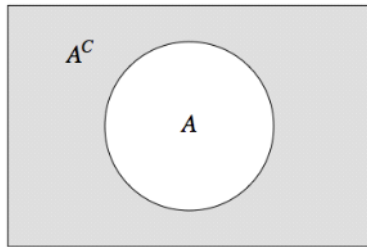
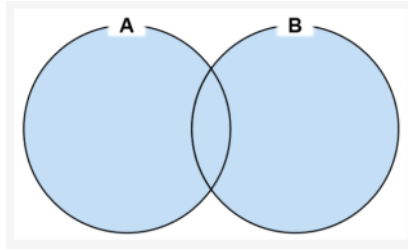
```
select ename
  from emp
 where empno not in (select mgr
                     from emp
                     where mgr is not null);
```

1 ADAMS
2 WARD
3 ALLEN
4 JAMES
5 SMITH
6 MILLER
7 MARTIN
8 TURNER

※ sub query 에서 not in 연산자 사용시 주의사항

서브쿼리에서 null 값이 하나라도 리턴되면 결과가 출력되지 않음. 한편, in 구문에서는 서브쿼리문에 null 이 있어도 값이 나오는데 그 이유는 **in 이 합집합의 개념**이므로 서브쿼리의 data 전체 중 메인쿼리의 조건에 해당하는 것을 출력하기 때문에 null 을 포함하여 출력하기 때문이다. 같은 논리로 **not in 은 여집합의 개념**이기 때문에 null 의 여집합은 '알

수 없는 것'의 여집합을 뜻하므로 전체가 아닌(not in)은 공집합, 즉 결과가 출력되지 않는 것이다.



※선생님 설명

where 절 조건 중 true가 포함되면 결과가 잘 출력된다 : in의 경우 : true or false(null) = true

where 절 조건 모두 true 이면 결과가 잘 출력된다 : not in 의 경우 : true and false(null) = false

※not in

서브쿼리문 사용시 not in 연사자를 사용하면 반드시 서브쿼리에서 메인쿼리로 null 값이 리턴되지 않도록 처리를 해주어야 함. ex) nvl, is not null 등

275) 274)를 값이 제대로 나오도록 하시오! (274에서 해결)

074 서브 쿼리 사용하기 4(EXISTS와 NOT EXISTS)

서브쿼리문에서 **exists** 와 **not exist** 를 사용하여 메인쿼리에 있는 데이터 중에 해당 데이터가 서브쿼리에 존재하는지 존재 유무를 파악할 때 사용하는 SQL 문법

ex1) emp 테이블에 권세원 데이터를 입력을 합니다. emp_권세원_data 파일

ex2) 우리반 테이블 학생들 중 사원 테이블에도 존재하는 학생이 있으면 이름을 출력하시오!

```
ENAME
1 권세원
```

```
select e12.ename
  from emp12 e12
 where exists (select *
               from emp e
               where e.ename = e12.ename);
```

위 결과를 보면 exists 문은 main query 컬럼이 서브쿼리 안으로 들어가면서 실행된다.

그러면서 메인쿼리의 데이터 중 서브쿼리에도 같은 데이터가 존재하는지 찾아본다.

★

276) 부서테이블에서 부서번호와 부서위치를 출력하는데 부서 테이블에 있는 부서번호중에 사원 테이블에 존재하는 부서번호에 대한것만 출력하시오!

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

※이때, 4행에서 emp e 옆에 dept d 를 추가하면 서브쿼리가 join 으로 바뀌게 되므로 exists 를 사용해서 서로 다른 컬럼에서 메인쿼리의 data가 서브쿼리에 존재하는지를 찾고자 할 때는 각각의 쿼리에 from 을 하나만 지정해주어야 원하는 값을 출력할 수 있다.

277) 이번에는 존재하지 않는 부서번호와 부서위치를 출력하시오!

276에서 3행의 exists 대신 **not exists** 기입

278) 어제 마지막 문제에 deptno 도 같이 출력하시오! (250번 참조)

```
select e.deptno,
       sum(decode(d.loc, 'NEW YORK', e.sal ) ) as "NEW YORK",
       sum(decode(d.loc, 'DALLAS', e.sal ) ) as "DALLAS",
       sum(decode(d.loc, 'CHICAGO', e.sal ) ) as "CHICAGO"
from emp e, dept d
where e.deptno = d.deptno
group by e.deptno;
```

☆279) 위 결과의 null 값 대신 0으로 출력되게 하시오!

```
select e.deptno,
       sum(decode(d.loc, 'NEW YORK', e.sal, 0 ) ) as "NEW YORK",
       sum(decode(d.loc, 'DALLAS', e.sal, 0 ) ) as "DALLAS",
       sum(decode(d.loc, 'CHICAGO', e.sal, 0 ) ) as "CHICAGO"
from emp e, dept d
where e.deptno = d.deptno
group by e.deptno;
```

※ **e.sal** 부분을 **nvl(e.sal, '0')** 으로 하게 되면 **null 값이 출력된다**. 얼핏 생각하면 279 와 같이 null 이 아닌 0 값이 출력되어야 하지만 코딩 순서 및 실행 순서가 sum >> decode >> e.sal >>>> 순으로 진행되고 있기 때문에 **e.sal을 출력하는 것은 decode 의 지배를 받고 있는 하위 개념**으로 이해되기 때문에 decode 의 끝 인자에 값이 없으므로 우선적으로 null 값이 출력되어버리는 것이다.

##1105

2020년 11월 5일 목요일 오전 8:31

075 서브 쿼리 사용하기 5(HAVING절의 서브 쿼리)

select 문장에서 서브쿼리를 쓸 수 있는 절

select 서브쿼리 사용가능 : scalar subquery(확장된 서브쿼리)
from 서브쿼리 사용가능 : in line view
where 서브쿼리 사용가능
group by X
having 서브쿼리 사용가능
order by 서브쿼리 사용가능 : scalar subquery

★

280) JAMES 보다 더 많은 월급을 받는 직원들의 이름과 월급을 출력하시오!

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

281) 직업, 직업별 토탈월급을 출력하시오!

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

282) 위 결과를 다시 출력하는데 직업이 SALESMAN 의 토탈월급보다 더 큰 직업들만 출력하시오!

```
select job, sum(sal)
from emp
where sum(sal) > (select sum(sal)
                  from emp
                  where job = 'SALESMAN')
group by job; >>> 이렇게 실행하면 err 발생 / 그룹함수는 where절에서 쿼리사용 불가
```

```
ORA-00934: 그룹 함수는 허가되지 않습니다
00934. 00000 - "group function is not allowed here"
*Cause:
*Action:
3행, 11열에서 오류 발생
```

>>SQL 수정

```
select job, sum(sal)
from emp
group by job
having sum(sal) > (select sum(sal)
                  from emp
                  where job = 'SALESMAN');
```

283) 부서번호와 부서번호별 인원수를 출력하는데 10번 부서번호의 인원수보다 더 큰것만 출력하시오!

```
select deptno, count(*)
  from emp
 group by deptno
 having count(*) > (select count(*)
                    from emp
                   where deptno = 10);
```

076 서브 쿼리 사용하기 6(FROM절의 서브 쿼리)

from : 해당 절에도 서브쿼리 사용가능 . from 절의 서브쿼리는 **in line viewer**.

in line view 는 임시로 table 을 가공하여 가공된 table을 검색조건을 활용하고자 할때 주로 사용
ex1) 이름, 월급, 순위를 출력하는데 순위가 월급이 높은 직원순으로 순위를 부여하시오!

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

ex2) 위에서 순위가 4등인 직원만 출력하시오!

```
select ename, sal, dense_rank() over(order by sal desc) 순위
  from emp
 where 순위 = 4;
```

```
ORA-00904: '*순위': 부적합한 식별자
00904. 00000 - "%s: invalid identifier"
*Cause:
*Action:
3행, 15열에서 오류 발생
```

err 발생이유 : 실행순서 : from>>where 수행하기때문에 emp 테이블에는 순위라는 컬럼존재X

err 해결위해서는 from 절의 서브쿼리를 사용해야 함

5 select *

1 from(3 select ename, sal, dense_rank() over(order by sal desc) 순위

2 from emp);

4 where 순위 = 4;

※ from 절의 서브쿼리문의 결과가 마치 하나의 테이블처럼 만들어져서 서브쿼리문의 결과 데이터가 메모리에 올라가게 된다. 위 예시의 2-3행이 해당하며, where 절에서 그것을 색인한다.

위 예시의 숫자는 실행 순서이다.

★

284) 직업, 이름, 월급, 순위를 출력하는데 순위가 직업별로 각각 월급이 높은 순서대로 순위를 부여하시오!

```
select job, ename, sal, dense_rank() over( partition by job
                                         order by sal desc) as rank
  from emp;
```

285) 위 결과에서 순위가 1등인 직원들만 출력하시오!

```
select *
  from(select job, ename, sal, dense_rank() over( partition by job
                                                  order by sal desc) as rank
        from emp
       )
 where rank = 1;
```

286) 우리반 테이블에서 통신사별로 나이가 가장 많은 학생의 이름과 나이와 통신사와 나이의 순위를 출력하시오!

```
select *
  from ( select ename, age, dense_rank() over( partition by telecom
                                              order by age desc) rank
        from emp12
        )
 where rank = 1;
```

※price table추가(웹 data 창 / 1번 price 자료)

주요 column 소개

M_NAME : 마트 이름 또는 시장이름

A_NAME : 식료품 이름

A_PRICE : 식료품 가격

287) price 테이블의 전체 건수가 어떻게 되는지 확인하시오!

```
select count(*)
  from price;
```

☆288) 서울시에서 가장 비싼 식료품의 이름과 가격과 파는곳(마트이름)을 출력하시오!

```
select *
  from ( select a_name, a_price, m_name, dense_rank() over(order by a_price desc) rank
        from price
        )
 where rank = 1;
```

077 서브 쿼리 사용하기 7(SELECT절의 서브 쿼리)

select 절에 서브쿼리를 사용 : **select 절의 서브쿼리를 scala subquery** 라고 한다

그룹함수를 컬럼으로 출력할때, 단일함수와 연관성이 없는 상태로 보고자 할 때 주로 사용함

ex1) 이름, 월급, 사원테이블의 평균월급을 출력하시오!

(자기의 월급과 평균월급을 비교해보기 위해서)

>>1

sets 를 이용하면..

```
select ename, sal, avg(sal)
  from emp
 group by grouping sets( (ename, sal) )
```

	ENAME	SAL	AVG(SAL)
1	ALLEN	1600	1600
2	KING	5000	5000
3	CLARK	2450	2450
4	MILLER	1300	1300
5	FORD	3000	3000
6	ADAMS	1100	1100
7	MARTIN	1250	1250
8	BLAKE	2850	2850
9	JAMES	950	950
10	JONES	2975	2975
11	TURNER	1500	1500
12	SMITH	800	800
13	WARD	1250	1250
14	SCOTT	3000	3000

이 경우 전체의 평균월급이 아닌 ename, sal 의 avg값, 즉 sal 과 동일한 값이 출력된다

>>2 select sub query 이용

select ename, sal, (select avg(sal)

from emp)

from emp;

★

289) 사원이름, 월급, 사원 테이블의 최대월급, 사원 테이블의 최소월급을 출력하시오!

```
select ename, sal, (select max(sal) from emp) as max,  
              (select min(sal) from emp) as min  
from emp;
```

※ **sacalar subquery**는 단 **한개의 값만 리턴할 수 있다**

290) 이름, 나이, 우리반 나이의 평균나이를 출력하시오!

```
select ename, age, (select avg(age) from emp12) as avage  
from emp12;
```

291) 위 결과를 소수점 이하가 나오지 않도록 반올림하시오!

```
select ename, age, round( (select avg(age) from emp12), 0) as avage  
from emp12;
```

☆☆292) 위 결과를 다시 출력하는데 우리반 평균나이보다 더 많은 학생들만 출력하시오!

>>1 in line view(from subquery)

select *

```
from (select ename, age, round( (select avg(age) from emp12), 0) as avage  
from emp12)
```

where age > avage;

>>2 where subquery

```
select ename, age, round( (select avg(age) from emp12), 0) as avage  
from emp12  
where age > (select avg(age)  
from emp12);
```

293) 사원 테이블에서 이름, 월급, 사원 테이블의 최대월급, 사원 테이블의 최소월급, 평균월급을 출력하시오!

>>튜닝 전 : emp 테이블을 4번 select함

```
select ename, sal, (select max(sal) from emp) as max,  
              (select min(sal) from emp) as min,  
              (select avg(sal) from emp) as avg  
from emp;
```

>> **튜닝 후** : emp 테이블을 1번 select함

```
select ename, sal, max(sal) over () max,  
              min(sal) over () min,  
              avg(sal) over () avg  
from emp;
```

294) 우리반 테이블에서 이름, 나이, 우리반 나이의 (최대나이, 최소나이, 평균나이) 와 우리반 학생들의 인원수를 출력하시오! (위 튜닝을 적용해서!)

```
select ename, age, max(age) over() max,  
              min(age) over() min,  
              avg(age) over() avg,
```

```
count(*) over() count
```

```
from emp12;
```

※select 절의 서브쿼리인 스칼라 서브쿼리가 성능이 느리므로 위와 같이 데이터 분석함수를 이용해서 튜닝을 하면 빠르게 대용량의 데이터를 검색 할 수 있다. **max(column) over()**

078 데이터 입력하기(INSERT)

insert : 테이블에 데이터를 입력하는 SQL 문장

```
ex1) insert into emp(empno, ename, sal)
      values( 1234, 'jack', 4500 );
```

문법 : **insert into table_name(column1, column2, column3)**
values(data1, data2, data3);

문법 SQL 의 1행의 column 순서대로 해당하는 data 값을 기술한다

★

295) 사원 테이블에 아래의 데이터를 입력하시오!

사원번호 : 2939

사원이름 : jane

월급 : 4700

입사일 : 오늘날짜

```
insert into emp(empno, ename, sal, hiredate)
      values (2939, 'jane', 4700, sysdate);
```

※ 날짜 data 는 sysdate 를 이용해도 되고 오늘날짜가 고정으로 유지되게 할 경우 to_date('20/11/05', 'RR/MM/DD') 를 이용해서 insert 해주면 된다

296) 오늘 입사한 사원의 이름과 입사일을 출력하시오!

```
>>1
```

```
select ename, hiredate
      from emp
```

```
      where hiredate = sysdate;
```

※ 위 데이터가 조회되지 않는 이유는 sysdate 가 시/분/초 까지 기록하기 때문에 295)의 insert 시의 시/분/초 와 296)의 시/분/초 가 다르기 때문에 서로 다른 sysdate 값이기 때문이다.

to_date('AA/BB/CC', 'RR/MM/DD') : AA년 BB월 CC일 00시 00분 00초로 기입된다

```
>>2 delete jane date
```

```
delete
```

```
      from emp where empno = 2939;
```

```
>>3 insert jane data with to_date ~
```

```
insert into emp(empno, ename, sal, hiredate)
```

```
      values (2939, 'jane', 4700, to_date('20/11/05', 'RR/MM/DD') );
```

```
>>4 resolve 296) question with changed data(jane)
```

```
select ename, hiredate
```

```
      from emp
```

```
      where hiredate = to_date('20/11/05', 'RR/MM/DD');
```

297) 2020년 11월 05일에 입사한 사원의 이름과 입사일을 출력하시오!

```
select ename, hiredate
```

```
      from emp
```

```
where hiredate = to_date('2020/11/05', 'RRRR/MM/DD');
```

※where 절의 column 에 함수를 사용하지 않고 SQL을 구성하는 것이 검색속도가 더 빠르다

위 문제의 경우 hiredate 를 to_char 로 가공하는것보다 2020/11/05 를 to_date로 가공하는것이 작업량이 더 적기 때문에 더 좋은 SQL 이라고 할 수 있다

298) 아래의 데이터를 사원 테이블에 입력하시오!

```
insert into emp(empno, ename, sal, deptno)
```

```
values( 4945, 'mike ', 3000, 20 ); >>> mike 옆에 공백 3칸 기입
```

299) 이름이 mike 인 사원의 이름과 월급을 출력하시오!

>>1 튜닝전

```
select ename, sal
```

```
from emp
```

```
where rtrim(ename) = 'mike';
```

>>2 튜닝후

```
select ename, sal
```

```
from emp
```

```
where ename like 'mike%'; >>> %를 뒤에 붙이는 것은 검색속도에 크게 영향주지않는다
```

※튜닝시 컬럼은 함수로 가공하지 않는것이 좋으며 가공한다면 가공량이 적은 것으로 한다

300) 아래의 데이터를 우리반 테이블에 입력하시오!

이름 : 김인호

성별 : 남

이메일 : abcd1234@gmail.com

주소 : 서울시 강남구 역삼동 삼원빌딩 4층

```
insert into emp12( ename, gender, email, address)
```

```
values( '김인호', '남', 'abcd1234@gmail.com', '서울시 강남구 역삼동 삼원빌딩 4층');
```

079 데이터 수정하기(UPDATE)

ex) update emp

```
set sal = 0
```

```
where ename = 'SCOTT';
```

위 SQL은 SCOTT 의 월급을 0으로 수정하는 update 문장

문법 : update **table_name**

```
set column = N
```

```
where condition
```

ex2) rollback; : commit 이후에 작업한 모든 변경사항을 취소하겠다

ex3) commit; : 지금까지 변경한 모든 작업을 다 database에 영구히 저장하겠다

★

301) KING 의 월급을 9000으로 변경하시오!

```
update emp
```

```
set sal = 9000
```

```
where ename = 'KING';
```

302) 직업이 SALESMAN 인 사원들의 커미션을 9500 으로 수정하시오!

```
update emp
  set comm = 9500
  where job = 'SALESMAN';
```

080 데이터 삭제하기(DELETE, TRUNCATE, DROP)

오라클에서 데이터를 삭제하는 방법 3가지

- 1) delete
- 2) truncate
- 3) drop

1. delete

ex1) delete from emp
 where empno = 7788;

문법 : delete from **table_name**
 where **column** = **N**

★

303) 직업이 SALESMAN 인 직원들을 삭제하시오!

```
delete from emp
  where job = 'SALESMAN';
```

※타임머신 기능(resque SQL)

1) delete from emp;

commit; >>> 이 경우 emp 테이블이 날아간 것이 commit 되어서 rollback 이 불가

2) resque SQL

1. emp 테이블을 flashback 이 가능한 상태로 변경한다(타임머신 기능 on 설정)

alter table emp enable row movement;

2. 현재시간의 5분 전으로 emp 테이블을 되돌린다

flashback table emp to timestamp
(systimestamp - interval '5' minute);

3. 골든타임은 15분 (default time = 15minute)

show parameter undo_retention

NAME	TYPE	VALUE
db_flashback_retention_target	integer	1440
undo_retention	integer	900
NAME	TYPE	VALUE
undo_retention	integer	900

2. truncate

- 1) 데이터 모두삭제
- 2) rollback 불가
- 3) flashback 불가
- 4) 테이블 구조는 잔존 / 이외 data는 다 delete

ex1) truncate table emp;

Table EMP이(가) 잘렸습니다.

현업에서 일하다가 저 출력결과를 보면 빨리 팀장님께 말해서 복구해달라고 요청하자! ㅋㅋ

3. drop

- 1) 모든 data 삭제
- 2) 테이블 구조까지 다 삭제
- 3) rollback 불가

4) flashback 가능 : 휴지통 기능이 있어서 flashback 은 가능

ex1) drop table emp;

drop 이후 select * from emp 를 입력하면 아래 err발생

```
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다
00942. 00000 - "table or view does not exist"
*Cause:
*Action:
3행, 15열에서 오류 발생
```

이때 show recyclebin; 를 입력해서 휴지통 내 정보 확인가능

ORIGINAL NAME	RECYCLEBIN NAME	OBJECT TYPE	DROP TIME
EMP	BIN\$ji0wr71nSfSPc06uiZ3sSw==\$0	TABLE	2020-11-05:16:28:15

alter table emp enable row movement;

flashback table emp to timestamp

(systimestamp - interval '5' minute);

```
select *
from emp;
```

★

304) 부서번호, 부서번호별 토달월급을 출력하는데 가로로 출력하시오(sum+decode사용)

```
select sum( decode( deptno, 10, sal) ) as "10",
       sum( decode( deptno, 20, sal) ) as "20",
       sum( decode( deptno, 30, sal) ) as "30"
from emp;
```

☆305) 위 결과에서 집계결과를 아래와 같이 출력하시오! (rollup 이랑 nvl 쓰면 간단하긴 함)

JOB	10	20	30	토달값
1 SALESMAN	(null)	(null)	5600	5600
2 CLERK	1300	1900	950	4150
3 ANALYST	(null)	6000	(null)	6000
4 MANAGER	2450	2975	2850	8275
5 PRESIDENT	5000	(null)	(null)	5000
6 전체 토달:	8750	10875	9400	29025

```
select job, sum( decode( deptno, 10, sal) ) as "10",
       sum( decode( deptno, 20, sal) ) as "20",
       sum( decode( deptno, 30, sal) ) as "30",
       sum( sal) as 토달값
```

from emp

group by job

union all

```
select '전체 토달:' as job, sum( decode( deptno, 10, sal) ) as "10",
       sum( decode( deptno, 20, sal) ) as "20",
       sum( decode( deptno, 30, sal) ) as "30",
       sum(sal) as 토달값
```

from emp;

##1106

2020년 11월 5일 목요일 오후 5:37

DW서버 << SQL로 데이터 분석 (레포팅 데이터 분석 함수)

※회사별 db system

금융 : oracle

판교 게임회사 : MSSQL

쇼셜커머스 : MySQL

교육 업체(해커스 등) : MySQL, Maria

**하둡(hive) : NoSQL(애도 그냥 SQL이다)

081 데이터 저장 및 취소하기(COMMIT, ROLLBACK)

SQL의 종류 : 아래 1-5 는 ~~문 이라고 칭함

1) query : select 문의 6가지절, 조인, 서브쿼리

select / scalar sub.q
from / in-line view suq.q
where / single&multi sub.q, single funtion condition
group by / grouping sets(), coulumn
having / group-funtion condition
order by / scalar suq.q

2) DML : insert, update, delete, merge

3) DDL : create, alter, drop, truncate, rename

4) DCL : grant, revoke

5) TCL : commit, rollback, savepoint

commit : 지금까지 변경한 데이터 베이스 작업들(DML 문장) 을 데이터베이스에 영구히 반영하겠다는

TCL(Transaction Control Language) 문

rollback : commit 이후부터 지금까지 변경한 데이터 베이스 작업(DML 문)을 취소하는 명령어

ex1) select count(*) from emp;

delete from emp;

select * from emp;

★

306) rollback 이 마지막 commit 한 시점까지만 rollback이 된다는것을 테스트하기 위해 commit을 지금 수행하고 사원테이블의 월급을 모두 다 0으로 변경하시오!

update emp

set sal = 0;

※ 암시적 commit

1) 정상종료 : exit

2) DDL 문 수행

3) DCL 문 수행

※ emp 테이블을 생성하는 스크립트를 편하게 저장하고 수행하는 방법

1) cmd >> scott 로 접속 : **sqlplus scott/tiger** 입력하면 바로 접속된다

1-2) sqlplus 입력 후 scott 과 tiger 을 따로 쳐도 된다

- 2) **ed demo.sql**
- 3) 메모장에 저장할 script 를 붙여넣은 후 (혹은 기입한 후) 저장하고 메모장을 종료한다
- 4) \$dir : 디렉토리 확인 명령어
- 5) **@demo.sql** : demo.sql을 수행하는 명령어(script에 이전 잔존 table인 emp와 dept 를 delete하는 명령어있음)

※ cmd창에서 수행하기

SQL > delete from emp;

SQL > create table emp902

(empno number(10)
ename varchar2(10));

위 두 과정 이후 rollback; 하고 emp 테이블을 재 검색해도 암시적 commit 을 한 이후이기 때문에 결과출력X

오라클은 DML 문장을 수행하고서 정상종료 또는 DDL문, DCL문을 수행하지 않고 명시적으로 commit 을 수행하지 않은 경우에는 rollback 으로 DML 작업을 취소할 수 있다.

※ MSSQL 의 경우

delete from emp; 를 수행하면 자동으로 commit 된다. 이를 방지하기 위해 begin tran 을 사용

ex) begin tran

delete from fmp;

※ 백업을 생활화!

create table emp_backup8

as

select *

from emp;

※ 암시적 롤백

- 1) database 시스템이 비정상 종료되었을때 (정전)

082 데이터 입력, 수정, 삭제 한번에 하기(MERGE)

merge : 데이터 입력과 수정과 삭제를 한번에 수행하는 명령어

SQL튜닝을 위해서 자주 사용되는 SQL

ex1) 이름과 부서위치를 출력하시오!

select e.ename, d.loc

emp e, dept d

wherer e.deptno=d.deptno;

ex2) 사원 테이블에 부서 위치(loc)컬럼을 추가한다.

alter table emp

add loc varchar(10);

ex3) 사원 테이블에 추가한 부서위치(loc)컬럼에 데이터를 해당 사원의 부서위치로 값을 갱신하시오!

merge into emp e

using dept d

on (e.deptno = d.deptno)

when matched then

update set e.loc = d.loc

※ merge into 다음에는 변경할 타겟(target) 테이블명을 작성하고

using 다음에는 소스(source) 테이블명을 작성한다.

on 다음에는 타겟과 소스 테이블간의 연결고리를 작성한다.

when matched then 다음에는 변경할 update 문을 작성한다.

★

307) 우리반 테이블에 telecom_price 컬럼을 추가하고 해당 텔레콤의 요금으로 값을 갱신하시오!

>>1 컬럼 추가

```
alter table emp12
```

```
    add telecom_price number(10);
```

>>2 컬럼 병합

```
merge into emp12 e12
```

```
using telecom_price tp
```

```
on (e12.telecom = tp.telecom)
```

```
when matched then
```

```
update set e12.telecom_price = tp.price;
```

308) 부서위치, 부서위치별 토탈월급을 출력하시오!

```
select d.loc, sum(e.sal)
```

```
    from emp e, dept d
```

```
    where e.deptno=d.deptno
```

```
    group by d.loc;
```

309) 부서번호, 부서번호별 토탈월급을 출력하시오!

```
select deptno, sum(sal)
```

```
    from emp
```

```
    group by deptno;
```

310) 사원 테이블과 급여등급(sqlgrade) 테이블을 조인해서 이름과 월급과 등급(grade)을 출력하시오!

```
select e.ename, e.sal, s.grade
```

```
    from emp e, salgrade s
```

```
    where e.sal between s.losal and s.hisal;
```

311) 사원 테이블에 급여등급(grade) 컬럼을 추가하시오!

```
alter table emp
```

```
    add grade number(10);
```

312) 사원 테이블에 추가한 grade 컬럼에 해당 사원의 급여등급으로 값을 갱신하시오!

```
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;
```

☆313) 부서명(dname) 컬럼을 emp 테이블에 추가하고 해당 사원의 부서명으로 emp 테이블의 dname을 갱신하시오!

```
alter table emp
```

```
    add dname varchar2(10);
```

```
merge into emp e
```

```
using dept d
```

```
on (e.deptno = d.deptno)
```



```
when matched then
update set e.dname=d.dname;
```

083 락(LOCK) 이해하기

lock : 특정 창(session)에 접속한 유저가 KING 의 월급을 9000으로 변경하고 있는 상태에서 다른 창(session)에 접속한 유저(scott)가 KING의 월급을 변경할 수 없도록 막는 기능을 LOCK 이라고 통칭함

ex1) 실습 - commit;

cmd 창 2개를 둘 다 scott 로 접속

```
>>cmd1 prompt
update emp
set sal = 9000
where ename = 'SCOTT';
```

두commit;

```
>>cmd2 prompt
update emp
set sal = 0
where ename = 'SCOTT';
```

내가 변경한 데이터를 남이 보려면 내가 commit 을 해야 볼 수 있다

내가 commit 을 하지 않은 상태에서 다른 사용자는 내가 변경한 데이터를 볼 수 없다

update 를 수행하면 update 를 수행하고 있는 행(row) 에 락(lock)이 걸린다

위 cmd1 prompt 에서 먼저 SQL 을 작성하면 cmd2 prompt 에서 작성한 SQL은 적용되지 않는다(아래 ex2 참고)

ex2) row lock 예시

```
update emp
set sal = 8000
where ename = 'KING';
```

이렇게 update 를 진행하면 KING 의 전체 행(row)에 락(lock)이 걸린다. 이때 다른창(session)에 접속한 유저가 KING의 자료를 절대로 갱신할 수 없다. 아래는 cmd1 prompt 실행 후 cmd2 prompt 실행예시

cmd1 prompt (먼저실행)

```
SQL> update emp
2   set sal = 8700
3   where ename = 'KING';

1 행이 업데이트되었습니다.
```

cmd2 prompt (lock걸리는 현상)

```
SQL> update emp
2   set deptno = 30
3   where ename = 'KING';
_
```

이때 cmd1 prompt 에서 commit 을 해주면 lock 이 해제가 되고, cmd2 prompt 에서 수행한 SQL은 commit 하기 전까지 cmd1 에서는 확인할 수 없다.

ex3) 서로 다른 행 적용은 가능

cmd1 prompt

```
SQL> update emp
  2   set sal = 9700
  3   where ename = 'JAMES';

1 행이 업데이트되었습니다.
```

cmd2 prompt

```
SQL> update emp
  2   set sal = 8000
  3   where ename = 'ALLEN';

1 행이 업데이트되었습니다.
```

cmd1 과 cmd2 를 순차적으로 실행해도 위 결과들은 서로 lock 이 겹쳐져 있지 않으므로 둘 다 정상적으로 수행된다

위 ex1) ~ ex3) 은 다른 DB 프로그램에서도 적용된다 (oracle, MSSQL, MySQL, NoSQL, 등)

084 SELECT FOR UPDATE절 이해하기

보통 lock 은 update 문을 수행할 때 주로 걸리지만 select 를 수행할 때는 lock 이 걸리지 않는다.

select 수행할 때도 lock 을 걸고 싶을 땐 select for update 문을 이용하면 된다.

이 기능은 내가 데이터를 보고 있는 동안 data가 갱신되는 것을 원하지 않을 때 활용한다.

ex1) 코스트코는 pm 9 에 문을 닫는데 pm 9 에 매장에 진열된 상품의 갯수를 파악하여 모자란 상품을 주문하여 채워넣으려고 할 때, pm 9 를 기준으로 지금 매장의 상품 개수를 파악하고 싶다. 이때 항상 10개의 제품이 있어야 하는 커피 제품이 있다면 지금 3개가 남았다면 내일 7개를 주문하여야 한다. 그런데 상품의 개수가 계속 변경되면 새로 주문을 넣을 때 혼란이 발생할 수 있기 때문에 pm9 를 기준으로 그 어떤 데이터도 갱신하지 못하도록 막는다.

ex1-2) cmd1 cmd2 실습 (1, 2 순으로 실행)

cmd1 prompt

```
SQL> select ename, sal
  2   from emp
  3   where ename = 'BLAKE'
  4   for update;

ENAME          SAL
-----
BLAKE          2850
```

cmd2 prompt

```
SQL> update emp
  2   set sal = 0
  3   where ename = 'BLAKE';
```

역시 cmd2 prompt 상에서 lock 이 걸린 것을 볼 수 있다. 이때 cmd1 에서 commit 을 하면 cmd2에서 수행했던 SQL이 적용된다. 아래의 예시를 보자

after commit; in cmd1 // cmd2 prompt show

```
SQL> select ename, sal
  2   from emp
  3   where ename = 'BLAKE';

ENAME          SAL
-----
BLAKE          0
```

또한 cmd2 prompt 에서 수행한 작업을 cmd1 prompt 에서 보려면 cmd2 에서 commit; 을 적용해야 확인가능하다.

085 서브 쿼리를 사용하여 데이터 입력하기

우리가 지금까지 배운 insert 문장은 한번에 한건만 입력할 수 있었다

ex1) 한건씩 입력하기

```
insert into emp (empno, ename, sal)
values (1234, 'aa', 4000);
```

그런데 서브쿼리를 사용한 insert 문장을 이용하면 한번에 여러건의 데이터를 입력할 수 있다.

ex2) 실습전 emp12 백업

```
create table emp12_backup
as
```

```
select *
from emp12;
```

ex2-2) truncate table emp12;

ec2-3) 서브쿼리 사용 insert (from sub.q)

insert into emp12

select*

from emp12_backup3;

★

314) dept 테이블을 백업하시오!

```
create table dept_backup
```

```
as
```

```
select*
```

```
from dept;
```

315) dept 테이블을 truncate 하시오!

```
truncate table dept;
```

316) 복구하시오

```
insert into dept
```

```
select *
```

```
from dept_backup;
```

086 서브 쿼리를 사용하여 데이터 수정하기

```
update emp
```

```
set sal = 8900
```

```
where ename = 'SCOTT'
```

위 예시에서 1, 2, 3행 모두에서 sub query 사용 가능!

ex1) SCOTT보다 더 많은 월급을 받는 직원들의 직업을 SALESMAN 으로 변경하시오!

```
update emp
```

```
set job = 'SALESMAN'
```

```
where sal > ( select sal
from emp
```

```
where ename = 'SCOTT');
```

★

317) ALLEN 보다 더 늦게 입사한 직원들의 커미션을 9000 으로 수정하시오!

```
update emp
  set comm = 9000
  where hiredate > ( select hiredate
                    from emp
                    where ename = 'ALLEN');
```

318) SMITH 와 같은 직업을 갖는 직원들의 월급을 9800으로 변경하시오! 단, SMITH 는 제외하시오!

```
update emp
  set sal = 9800
  where job = ( select job
                from emp
                where ename = 'SMITH')
  and ename != 'SMITH';
```

319) ALLEN 의 월급을 KING 의 월급으로 변경하시오!

```
update emp
  set sal = (select sal
             from emp
             where ename='KING')
  where ename = 'ALLEN';
```

320) JONES 보다 월급이 많은 직원들의 직업을 MARTIN 의 직업으로 변경하시오!

```
update emp
  set job = ( select job
              from emp
              where ename = 'MARTIN')
  where sal > ( select sal
                from emp
                where ename = 'JONES');
```

087 서브 쿼리를 사용하여 데이터 삭제하기

ex1) SCOTT 보다 월급을 많이 받는 직원들을 삭제하시오!

```
delete from emp
  where sal > ( select sal
                from emp
                where ename = 'SCOTT');
```

★

321) ALLEN 보다 늦게 입사한 직원들을 삭제하시오!

```
delete from emp
  where hiredate > ( select hiredate
                    from emp
                    where ename = 'ALLEN');
```

322) JONES 와 같은 부서번호에서 일하는 직원들을 삭제하시오! 단, JONES는 제외하시오!

```
delete from emp
  where deptno = ( select deptno
                  from emp
                  where ename = 'JONES')
and ename != 'JONES';
```

088 서브 쿼리를 사용하여 데이터 합치기

merge 문에 서브쿼리를 사용하기

ex1) 부서번호, 부서번호별 토탈월급을 출력하시오! (세로로 출력)

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

ex2) 부서테이블에 sumsal 이라는 컬럼을 추가하시오!

```
alter table dept
  add sumsal number(10);
```

ex3) 위 dept 테이블의 추가한 sumsal 컬럼에 해당 부서번호의 토탈월급으로 값을 갱신하시오!

```
merge into dept d
using ( select deptno, sum(sal) as total
        from emp
        group by deptno) e
```

```
on ( e.deptno = d.deptno)
```

```
when matched then
```

```
update set d.sumsal = e.total;
```

위의 ex3) 에서 sub query 는 붉은색으로 표시되어있다.

★

323) 부서번호, 부서번호별 인원수를 출력하시오! (세로출력)

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

324) 부서테이블에 cnt 라는 컬럼을 추가하시오!

```
alter table dept
  add cnt number(10);
```

325) 지금 dept 테이블에 추가한 cnt 컬럼에 해당 부서번호에 인원수로 값을 갱신하시오!

```
merge into dept d
using ( select deptno, count(*) as count
        from emp
        group by deptno) e
on (e.deptno = d.deptno)
when matched then
```

```
update set d.cnt = e.count;
```

※ 325)의 using절에서 table 대신 sub.q 가 가상의 table 역할을 하는 것으로 이해하면 된다

089 계층형 질의문으로 서열을 주고 데이터 출력하기 1

ex1) 1## 1부터 10까지의 합을 구하시오!

```
select 1+2+3+4+5+6+7+8+9+10
      from dual;
```

ex2) 1부터 100까지 합을 구하시오!

이 경우, ex1 은 너무 소요가 많고 계층형 질의문을 활용하여 간단히 구할 수 있다

>>1

```
select level
      from dual
      connect by level <= 100;
```

LEVEL
1
2
3
4
5
6
7

1번부터 connect by 절의 level 다음에 오는 숫자만큼 숫자가 증가해서 출력된다

>>2

```
select sum(level)
      from dual
      connect by level <= 100;
```

SUM(LEVEL)
1 5050

★

2## 1부터 10사이의 숫자를 짝수만 출력하시오!

```
select level
      from dual
      where mod(level,2) = 0
      connect by level <= 10;
```

327) 밑수가 자연상수(e) 이고 진수가 10 인 로그값을 출력하시오!

```
select ln(10) from dual;
```

328) 자연상수(e) 의 10승은 얼마인가?

```
select exp(10) from dual;
```

329) 1부터 10까지의 곱을 SQL로 출력하시오!

```
select exp( sum( ln(level) ) )
      from dual
      connect by (level) <= 10;
```

3## 1부터 10까지의 곱을 출력하시오! (로그의 성질을 이용하면 풀 수 있다)

```
select exp( sum( ln(level) ) )
```

from dual

```
connect by (level) <= 10;
```

 EXP(SUM(LN(LEVEL)))

[illegible]

이때, 맨 끝에 11이 나오는 것은 컴퓨터 연산시 2진수 연산으로 나오는 것과 관련이 있고 정확한 이유는 부동소수점 연산에 의한 값에 의한 것이다.

level 에 대한 개략적인 개념을 잡고싶다면 아래 링크를 참조 // 계층형 질의

<http://www.gurubee.net/lecture/2379>

330) emp 테이블의 서열을 SQL 로 시각화 하시오!

```
select rpad( ' ', level*2) || ename as employee, level
      from emp
      start with ename = 'KING'
      connect by prior empno = mgr;
```

EMPLOYEE	LEVEL
1 KING	1
2 JONES	2
3 SCOTT	3
4 ADAMS	4
5 FORD	3
6 SMITH	4
7 BLAKE	2
8 ALLEN	3
9 WARD	3
10 MARTIN	3
11 TURNER	3
12 JAMES	3
13 CLARK	2
14 MILLER	3

rpad(' ', level*2) 은 공백(' ') 을 level*2 의 숫자만큼 채워넣겠다는 의미이다. level 이 클수록 공백이 많이 들어간다.

331) 위 결과에서 BLAKE 는 제외하고 출력하시오!

```
select rpad( ' ', level*2) || ename as employee, level
      from emp
      where ename != 'BLAKE'
      start with ename = 'KING'
      connect by prior empno = mgr;
```

332) 이때, BLAKE 만 제외하는 것이 아닌 BLAKE 와 연관된 하위서열까지 제외하려고 할 땐 connet by 절에 조건을 준다

```
select rpad( ' ', level*2) || ename as employee, level
      from emp
      start with ename = 'KING'
      connect by prior empno = mgr and ename != 'BLAKE'
```

이 경우 331) 원문제에서 BLAKE data 만 제외한 것과 별개로 BLAKE 에 해당하는 node를 제외한 것으로 이해가능

333) 다시 BLAKE 와 BLAKE 팀원들을 포함시킨 서열을 출력하는 SQL을 아래와 같이 실행하는데 월급이 높은 순서대로 출력하시오!

```
select rpad( ' ', level*2) || ename as employee, level, sal
      from emp
      start with ename = 'KING'
      connect by prior empno = mgr
      order by sal desc;
```

EMPLOYEE	LEVEL	SAL
1 KING	1	5000
2 SCOTT	3	3000
3 FORD	3	3000
4 JONES	2	2975
5 BLAKE	2	2850
6 CLARK	2	2450
7 ALLEN	3	1600
8 TURNER	3	1500
9 MILLER	3	1300
10 WARD	3	1250
11 MARTIN	3	1250
12 ADAMS	4	1100
13 JAMES	3	950
14 SMITH	4	800

위 결과는 sal desc 로 정렬이 되면서 서열로 정렬된 결과가 사라졌다.

334) 위 결과를 다시 서열로 정렬된 결과를 유지하면서 월급이 높은 순서대로 정렬되서 출력되게 하시오!

```
select rpad( ' ', level*2) || ename as employee, level, sal
from emp
start with ename = 'KING'
connect by prior empno = mgr
order siblings by sal desc;
```

EMPLOYEE	LEVEL	SAL
1 KING	1	5000
2 JONES	2	2975
3 SCOTT	3	3000
4 ADAMS	4	1100
5 FORD	3	3000
6 SMITH	4	800
7 BLAKE	2	2850
8 ALLEN	3	1600
9 TURNER	3	1500
10 WARD	3	1250
11 MARTIN	3	1250
12 JAMES	3	950
13 CLARK	2	2450
14 MILLER	3	1300

order by 절에 siblings 를 사용하면 334) 의 요구조건을 만족시키는 SQL 을 출력할 수 있다.

계층형 질의문을 사용할 때 order by 절을 이용하려면 siblings 라는 키워드를 짝공으로 사용해야 한다.

335) 이름과 입사일과 서열순위를 출력하는데 서열 순위의 정렬상태를 유지하면서 먼저 입사한 사원순으로 정렬이 되어 서 출력되게 하시오!

```
select rpad( ' ', level*2) || ename, hiredate, level
from emp
start with ename = 'KING'
connect by prior empno = mgr
order siblings by hiredate asc;
```

091 계층형 질의문으로 서열을 주고 데이터 출력하기 3

계층형 질의문과 짝공 함수인 **sys_connect_by_path** 를 이용하면 listagg 를 사용한 것처럼 가로로 결과를 출력할 수 있다.

ex1)

```
select ename, sys_connect_by_path(ename, '/') as path
from emp
start with ename = 'KING'
connect by prior empno = mgr;
```

ENAME	PATH
1 KING	/KING
2 JONES	/KING/JONES
3 SCOTT	/KING/JONES/SCOTT
4 ADAMS	/KING/JONES/SCOTT/ADAMS
5 FORD	/KING/JONES/FORD
6 SMITH	/KING/JONES/FORD/SMITH
7 BLAKE	/KING/BLAKE
8 ALLEN	/KING/BLAKE/ALLEN
9 WARD	/KING/BLAKE/WARD
10 MARTIN	/KING/BLAKE/MARTIN
11 TURNER	/KING/BLAKE/TURNER
12 JAMES	/KING/BLAKE/JAMES
13 CLARK	/KING/CLARK
14 MILLER	/KING/CLARK/MILLER

★

336) 위 ex1 에서 path column 의 / 를 잘라내시오!

```
select ename, ltrim( sys_connect_by_path(ename, '/'), '/' ) as path
from emp
start with ename = 'KING'
connect by prior empno = mgr;
```

※ 계층형 질의문 작성지 반드시 알아야하는 짝꿍 키워드 두 가지

- 1) order by 절의 siblings : order siblings by
- 2) 서열을 가로로 출력하는 sys_connect_by_path 함수

092 계층형 질의문으로 서열을 주고 데이터 출력하기 4

계층형 질의문에서도 조인문장을 같이 사용할 수 있다

ex1) 이름, 서열, 월급과 부서위치를 출력하시오!

```
select rpad(' ', level*2) || e.ename, level, e.sal, d.loc
  from emp e, dept d
 where e.deptno = d.deptno
 start with ename = 'KING'
 connect by prior empno = mgr;
```

★

337) 이름, 서열, 월급, 급여등급(grade)을 출력하시오! emp 와 salgrade 를 조인!

```
select rpad(' ', level*2) || e.ename as employee, level, e.sal, s.grade
  from emp e, salgrade s
 where e.sal between s.losal and s.hisal
 start with ename = 'KING'
 connect by prior empno = mgr;
```

338) 위 결과를 다시 출력하는데 서열의 정렬을 유지하면서 급여등급이 높은 사원부터 출력되게 하시오!

```
select rpad(' ', level*2) || e.ename as employee, level, e.sal, s.grade
  from emp e, salgrade s
 where e.sal between s.losal and s.hisal
 start with ename = 'KING'
 connect by prior empno = mgr
 order siblings by s.grade desc;
```

339) DALLAS 에서 근무하는 사원들의 이름, 서열, 부서위치를 출력하시오! (단 서열은 전체 사원을 기준으로 서열순위를 부여하시오!)

```
select rpad(' ', level*2) || e.ename as employee, level, d.loc
  from emp e, dept d
 where e.deptno = d.deptno and d.loc = 'DALLAS'
 start with ename = 'KING'
 connect by prior empno = mgr;
```

##은 알고리즘 practice number 임

340) ##4) 계층형 질의문을 이용해서 구구단 2단을 출력하시오!

```
select '2 X ' || level || ' = ' || level*2 as "2단"
  from dual
 connect by level <= 9;
```

2단		
1	2	X 1 = 2
2	2	X 2 = 4
3	2	X 3 = 6
4	2	X 4 = 8
5	2	X 5 = 10
6	2	X 6 = 12
7	2	X 7 = 14
8	2	X 8 = 16
9	2	X 9 = 18

341) ##5) 계층형 질의문을 이용해서 삼각형을 출력하시오!

```
select lpad( '★', level, '★' )
  from dual
 connect by level <= 10;
```

```

1 ★
2 ★★
3 ★★★
4 ★★★★
5 ★★★★★
6 ★★★★★★
7 ★★★★★★
8 ★★★★★★
9 ★★★★★★
10 ★★★★★★
```

- 1) lpad(column, 10, '*') >>> lpad 원형 : column 을 출력하는데 전체 10자리를 잡고, 나머지 빈자리에 *을 채워라
이때, lpad 면 * 을 왼쪽에, rpad 면 오른쪽에 출력한다.
- 2) lpad('★', level, '★') >>> ★을 출력하는데 level 수만큼 자리를 잡고 ★을 하나 출력하고 나머지 자리에 ★을 채워넣어라

☆342) ##6) 아래와 같이 결과를 출력하시오!

```

★
★★
★★★
★★★★
★★★★★
★★★★★
★★★★
★★★
★★
★
```

```
select rpad( '★', level, '★' )
  from dual
 connect by level <= 5
union all
select rpad( '★', decode(level, 1,4, 2,3, 3,2, 4,1), '★' )
  from dual
 connect by level <= 4
```

위 SQL 도 가능하긴 하지만 5행부터 **select lpad('★', 5-level, '★') from dual connect by level <= 4**
로 작성하는 것이 더 알고리즘적 SQL 이다 >> 단순 노가다성 SQL 은 지양하자 +case 구문으로 union all 생략가능

093 일반 테이블 생성하기(CREATE TABLE)

DDL 문 (Data Definition Language) : create, alter, drop, truncate, rename

활용 : 겨울왕국 대본을 오라클에 입력하고 elsa 가 많이 나오는지 anna 가 많이 나오는지 또는 긍정단어가 많은지
부정단어가 많은지 등을 SQL로 검색하려면 DDL 문을 이용해서 테이블을 생성할 수 있어야 함

ex1) 테이블 생성 script 예시

```
create table emp500          <<< 테이블명
( empno number(10),         <<< 컬럼명 데이터 유형(길이)
  ename varchar2(20),        ↓
  sal    number(10) );       문자형(varchar2, char), 숫자형( number(10) ), 날짜형(date)
```

이때 숫자형 number 옆의 숫자는 허용 숫자 자리수를 의미

varchar2(20) : 영어철자 20개를 허용

number(10) : 숫자 10자리를 사용

★

343) emp500 테이블에 아래의 데이터를 입력하시오!

1111 scott 3000

2222 smith 2900

>>commit; 은 하고싶으면 하기

insert into emp500 values(1111, 'scott', 3000);

insert into emp500 values(2222, 'smith', 2900);

344) 아래의 테이블을 생성하는 테이블을 emp501 로 해서 생성하시오

empno, ename, sal, hiredate, deptno

>>>SQL 실행하면 data 없이 column 만 들어있는 table 인 emp501 이 생성됨

create table emp501 <<< 테이블명은 문자로 시작해야 한다

(empno number(10),

ename varchar2(20),

sal number(10),

hiredate date, <<< 9999년 12월 31일까지의 날짜 데이터 입력

deptno number(20));

EMPNO	ENAME	SAL	HIREDATE	DEPTNO
-------	-------	-----	----------	--------

※ 오라클 메뉴얼사이트

https://docs.oracle.com/cd/E11882_01/index.htm

※ datatype 의 유형

데이터 유형

char : 고정길이 문자 데이터유형이고 최대길이가 2000

varchar2 : 가변길이 문자 데이터유형이고 최대길이가 4000

long : 가변길이 문자 데이터유형이고 최대 2GB 까지 데이터 허용

clob : 문자 데이터 유형이고 최대 4GB 까지 문자 데이터 허용

blob : 바이너리 데이터 유형이고 최대 4GB (사진, 동영상)

number : 숫자 데이터 유형이고 최대 38 까지 허용

date : 날짜 데이터 유형이고 기원전 4712 년 1월 1일부터 기원후 9999년 12월 31일까지 날짜를 허용

ex1) number(10, 2) : 숫자데이터 10자리를 허용하는데 그중에 소숫점 2자리를 허용하겠다.

number(10, 2) >>> data 는 3500.23 와 같이 기입할 수 있다

★

345) 아래의 emp501 테이블에 데이터를 2건 입력하시오

insert into emp501 values(7839, 'KING', 5000, to_date('81/11/17', 'RR/MM/DD'), 10);

insert into emp501 values(7698, 'BLAKE', 2850, to_date('81/05/01', 'RR/MM/DD'), 30);

각각의 data 를 입력해줄 때 ;(실행)은 각각 해 주어야 정상적으로 insert 된다(insert 문이 각각 들어가므로)

	EMPNO	ENAME	SAL	HIREDATE	DEPTNO
1	7839	KING	5000	81/11/17	10
2	7698	BLAKE	2850	81/05/01	30

ex2) long 데이터 타입을 사용하는 방법 테스트

long 데이터 타입은 아주 긴 텍스트 데이터를 입력할 때 사용하는 데이터 유형이다

create table profile2

```
( ename varchar2(20),
  self_intro long );
insert into profile2(ename, self_intro)
  values( '김인호', '어렸을때부터 우리집은 가난했었습니다. 그리고 어머니는 짜장면이 싫다고 하셨습니다.야히 야히
야' );
```

346) 겨울왕국 대본을 입력하기 위한 테이블을 winter_kingdom 이라는 이름으로 생성하시오!

```
create table winter_kingdom
( win_text long );
```

347) 영화 겨울왕국 대본에는 elsa 가 많이 나오는가? 혹은 anna 가 많이 나오는가?

정규식 함수인 regexp_count 를 이용하면 쉽게 구현할 수 있다

```
select regexp_count( lower( win_text), 'elsa' ) as cnt
  from winter_kingdom;
```

```
ORA-00932: 일관성 없는 데이터 유형: CHAR01(가) 필요하지만 LONG임
00932. 00000 - "inconsistent datatypes: expected %s got %s"
*Cause:
*Action:
7행, 29열에서 오류 발생
```

그런데 데이터 타입이 long 이어서 해당 오류가 발생하였다

348) 겨울왕국 테이블을 drop 하고 다시 생성하는데 데이터 타입을 long 이 아닌 varchar2(4000) 으로 하시오!

```
create table winter_kingdom
( win_text varchar2(4000) );
```

설정 후 winter script를 좌측 메뉴바에서 table/winter_kingdom/우클릭 후 데이터 임포트

348-2) 347)을 다시 실행하시오

```
select sum( regexp_count( lower( win_text), 'elsa' ) )as cnt
  from winter_kingdom;
select sum( regexp_count( lower( win_text), 'anna' ) )as cnt
  from winter_kingdom;
```

※ 위 SQL 은 win_text 를 다 소문자로 변경하고 win_text 컬럼의 데이터 중에 elsa 라는 단어가 몇번 나오는지 count 한다.

그리고 그 카운트된 숫자들을 sum 을 이용하여 다 더해준다.

위 SQL 이 유용한 경우

- 1) 뉴스기사를 분석해서 회사 주식에 영향을 주는지의 여부(미래에셋 로보어드바이저)
- 2) 외국 케냐의 사례 중 은행 고객 대출 한도를 정할 때(대출 신청자의 SNS 글을 분석하여 긍정/부정 단어 분석)

349) 긍정단어를 저장하기 위한 테이블을 positive 라는 이름으로 아래와 같이 생성하시오!

```
create table positive
( p_text varchar2(2000) );
```

350) 부정단어를 저장하기 위한 테이블을 negative 라는 이름으로 아래와 같이 생성하시오!

```
create table negative
( p_text varchar2(2000) );
```

351) 긍정단어는 positive 테이블에 입력하고 부정단어는 negative 테이블에 입력하시오!

```
select count(*)
  from negative;
```

```
select count(*)
  from positive;
```

각각 348) 을 참고하여 data 입력 후 위 SQL 을 이용하여 확인하여야 함

352) 겨울왕국 대본에는 긍정단어가 많은가 부정단어가 많은가?

>>>현 진도로는 조금 어려운 문제

1) from 절 서브쿼리

2) join

3) regexp_substr

```
select count(*) 긍정단어
  from (
        select regexp_substr( lower( win_text), '[^ ]+', 1, a ) word
          from winter_kingdom, ( select level a
                                from dual
                                connect by level <= 15 )
      )
 where word in ( select lower( p_text )
                  from positive );
```

353) 엑셀 데이터를 오라클의 테이블의 데이터로 입력하시오!

(단, 전국 대학교 등록금 현황 데이터를 저장할 테이블을 먼저 아래와 같이 생성하시오!)

[univ 입력문] text 참조 // 혹은 카페 data 32 번 게시물 참고

※ 공공 데이터 포털

354) 우리나라에서 대학등록금이 가장 비싼 학교는 어디인가?

tuition : 등록금 / admission_fee(입학금)

```
select university
  from univ
 where tuition = (select max( tuition)
                  from univ) ;
```

UNIVERSITY
1 명지대 학교 자연캠퍼스

355) data 게시판의 범죄 발생 지역 데이터를 내려받아 테이블을 생성하고 데이터를 입력하시오!

data 게시판 34번 참조 // table : crime_loc

CRIME_TYPE	C.LOC	CNT
1 절도	아파트	25389
2 절도	집	37787

crime_type : 범죄유형

c_loc : 범죄장소

cnt : 범죄건수

356) 범죄유형을 출력하는데 중복제거해서 출력하시오!

```
select distinct(crime_type)
  from crime_loc;
```

357) 살인이 가장 많이 일어나는 장소가 어디입니까?

>>한결씨 답

```

select *
  from crime_loc
 where crime_type = '살인'
 order by cnt desc
 fetch first 1 rows only;
>>sub query
select crime_type, c_loc, cnt
  from crime_loc
 where cnt = (
            select max(cnt)
              from crime_loc
             where crime_type = '살인'
            );

```

358) 절도가 가장 많이 일어나는 장소가 어디인지 1위부터 3위까지 출력하시오!

```

select *
  from crime_loc
 where crime_type = '절도'
 order by cnt desc
 fetch first 3 rows only;

```

359) 데이터 게시판에서 범치원인 데이터를 가져와서 오라클 테이블로 구성하고 데이터를 입력하시오! **crime_cause**

360) 살인을 일으키는 가장 큰 원인은 무엇인가?

데이터 >>> 컬럼 : pivot 문

컬럼 >>> 데이터 : unpivot 문

※ 위 360)질문에 대한 답을 구하려면 컬럼을 데이터로 넣어야 가능하다. unpivot 문 사용

아래의 SQL은 as 다음에 나오는 쿼리문의 결과를 crime_cause2 로 생성하겠더라는 뜻이다.

```

create table crime_cause2
as
select *
  from crime_cause
 unpivot ( cnt for term in (생계형,
                           유흥,
                           도박,
                           허영심,
                           복수,
                           해고,
                           징벌,
                           가정불화,
                           호기심,
                           유혹,
                           사고,
                           불만,
                           부주의,

```



```

기타));
>>>360) answer
select *
  from crime_cause2
 where crime_type = '살인'
 order by cnt desc
 fetch first 1 rows only;

```

361) 동전을 던져서 앞면이 나오는지 뒷면이 나오는지 확인하시오! (단 앞면은 숫자 0, 뒷면은 숫자1)

>>1

```

select dbms_random.value(0, 1)
  from dual;

```

위 설명 : 0~1 사이의 실수를 랜덤으로 출력하는데 가우시안 분포를 따르는 데이터로 출력이 되며, 시행시마다 값이 다르다

>>2

```

select round( dbms_random.value(0, 1) )
  from dual;

```

2번은 시행할때마다 동전을 던지는 것과 같다

362) 동전을 10번 던지시오(던질때마다 다른 값 출력)

```

select round( dbms_random.value(0, 1) )
  from dual
 connect by level <= 10;

```

위 SQL은 3행의 connect by level <= 10 으로 인해 10번 시행한 것과 같은 효과를 보인다

☆363) ##7) 동전을 10만번 던졌을 때 동전이 앞면이 나올 확률은 얼마인가?

>>1 from subquery 활용

```

select sum( regexp_count( p, 0) ) * 0.001 || '%'
  from (select round( dbms_random.value(0, 1) ) as p
        from dual
        connect by level <= 100000
       );

```

>>2 그외 답들

```

select sum(round(dbms_random.value(0,1)))/100000
  from dual
 connect by level <=100000;

```

>>3 그외 답들

```

select count(round( dbms_random.value( 0, 1 ) )
  from dual
 where round( dbms_random.value( 0, 1 ) ) =0
 connect by level <= 100000;

```


##1110

2020년 11월 10일 화요일 오전 9:41

복습

1. select 문의 6가지 절

select
from
where
group by
having
order by

2. 함수 :

단일행 함수 : 문자, 숫자, 날짜, 변환, 일반

복수행 함수 : max, min, avg, sum, count

3. 조인문장 :

oracle join : equi join, non equi join, outer join, self join

1999ansi join : on join, using join, natural join, left/right/full outer join, cross join

4. 서브쿼리

single row subquery
multiple row subquery
multiple column subquery >>> 아직 안배움

5. 집합 연산자

union all
union
intersect
minus

6. 계층형 질의문 : 서열을 보여주는 SQL 문

```
select level, ename
from dual
start with ename = 'KING'
connect by prior empno = mgr;
(부모 키) (자식 키)
```

7. DML 문장 : insert, update, delete, merge

8. DDL 문장 : create, alter, drop, truncate, rename

데이터 분석가 : SQL, Python, R, 하둡 //

R은 시각화, 통계에 강점

Python 은 시각화, 통계가 점점 발전 + 딥러닝 가능

하둡 은 SQL >> HIVE >> JAVA

개발자 루트

1) 딥러닝 개발자(연구원)

2) 데이터 분석가

공부방법

1) 위 1~8 을 머리속에 박아넣기

2) 알고리즘 문제 풀기(천천히)

시작

현업에서 계층형 질의문을 활용했을때의 장점

: 자바, C, 기타 프로그래밍 언어는 길게 작성해야 하는 코드를 SQL 하나로 끝낼 수 있다.

※ 치환변수 '&' 사용법

SQL 문장을 수행할 때 매번 검색해야 하는 데이터 값이 다른데 SQL 문장이 같을 때 검색을 용이하게 하는 오라클 SQL 문법

```
ex) select empno, ename, sal
      from emp
      where empno = &empno;
```

ex2) 어제 문제에서 10만번 동전 던지는 대신..

undefine 던진횟수

```
select count(*) / &&던진횟수 as "동전이 앞면이 나올 확률"
      from ( select round( dbms_random.value(0, 1) ) as 동전
            from dual
            connect by level <= &던진횟수)
      where 동전 = 0;
```

던진횟수를 SQL 내에서 두 번 입력해야 하는데 한번만 입력하게 하려면 앤퍼센트(&) 를 앞에 두개쓰고 뒤에 한개 쓰면 된다. 이때 SQL 내에서 치환변수시 사용하는 & 를 두개를 한번이라도 사용하게 되면 두 번째 실행할때는 안물어보고 바로 절에 입력된 값으로 실행을 해버린다. 그래서 치환변수 내의 내용을 비우려면 SQL 앞에 [undefine 치환변수이름]을 입력해주면 된다.

★

364 아래의 SQL에서 붉은 부분을 하드코딩하지 않고 인라인뷰에서 시행한 동전 던진 전체 횟수가 들어가게 하려면 어떻게 해야 할까?

```
select count(*) / 100000 as "동전이 앞면이 나올 확률"
      from ( select round( dbms_random.value(0, 1) ) as 동전
            from dual
            connect by level <= 100000)
      where 동전 = 0;
```

##12) 참조 (& 활용)

※ 구구단 1단부터 9단까지 출력하기 위해 알아야 할 SQL

ex1) 숫자 1과 2를 출력하는 SQL 문을 작성하시오

```
select level
      from dual
      connect by level <= 2;
```

ex2) 위 결과를 from 절의 서브쿼리로 만들고 emp 테이블과 조인하시오!

```
select empno, ename, sal
      from emp e, (select level
            from dual
            connect by level <= 2);
```

14개 * 2 = 28개

emp 테이블의 모든 데이터가 숫자 1과 조인해서 14개 출력하고

emp 테이블의 모든 데이터가 숫자 2와 조인해서 14개 출력되어 총 28개가 출력되었다.

EMPNO	ENAME	SAL	LEVEL
1	7839 KING	5000	1
2	7698 BLAKE	2850	1
3	7782 CLARK	2450	1
4	7566 JONES	2975	2
5	7654 MARTIN	1250	2
6	7499 ALLEN	1600	
7	7844 TURNER	1500	
8	7900 JAMES	950	
9	7521 WARD	1250	
10	7902 FORD	3000	
11	7369 SMITH	800	
12	7788 SCOTT	3000	
13	7876 ADAMS	1100	
14	7934 MILLER	1300	

이 두 결과가 서로 조인하였기 때문에 $14 \times 2 = 28$ 이 도출된다

ex3) 숫자를 1부터 9까지 출력하는 SQL 문장을 작성하시오!

```
select level
  from dual
 connect by level <= 9;
```

ex4) 위의 숫자를 1부터 9까지 출력하는 SQL 문을 아래와 같이 from 절의 서브쿼리로 2개로 만드시오!

```
select a.num1, b.num2
  from (
    select level as num1
      from dual
    connect by level <= 9) a,
  (
    select level as num2
      from dual
    connect by level <= 9) b;
```

★

365) 8##) 구구단 전체를 출력하시오!

```
select a.num1+1 || ' x ' || b.num2 || ' = ' || a.num1*b.num2 as 구구단
  from ( select level as num1
        from dual
        connect by level <= 8) a,
  ( select level as num2
    from dual
    connect by level <= 9) b;
```

1	2	3	4	5	6	7	8	9
1	x	2	=	2				
2	x	3	=	3				
3	x	4	=	4				
4	x	5	=	5				
5	x	6	=	6				
6	x	7	=	7				
7	x	8	=	8				
8	x	9	=	9				
9	x	1	=	2				
10	x	2	=	4				
11	x	3	=	6				
12	x	4	=	8				
13	x	5	=	5				

이 문제 같은 경우는 a 테이블과 b 테이블 모두가 조인해야 결과값이 출력되므로 조인조건이 필요가 없다. 하지만 그렇지 않은 경우에는 서로 조인되지 않도록 inline view를 구성해야 한다

094 임시 테이블 생성하기(CREATE TEMPORAY TABLE)

데이터를 영구히 database 에 저장하는게 아니라 임시로 저장하는 테이블

데이터 중 영구히 저장할 필요는 없고 지금 잠깐 테스트를 위해 볼 데이터나, 지금 현재만 필요하고 나중에는 필요하지 않은 데이터가 있는데 그 데이터를 잠깐 저장할 때 사용하는 테이블

임시 테이블의 종류

- 1) **on commit delete rows** 옵션 : 데이터를 commit 할때까지만 보관
- 2) **on commit preserve rows** 옵션 : 데이터를 접속한 유저가 로그아웃할때까지만 보관

```
ex1) on commit delete rows
create global temporary table emp700
( empno number(10),
  ename varchar2(10),
  sal    number(10) )
on commit delete rows;
```

```
insert into emp700
select empno, ename, sal
  from emp;
```

>>> 이렇게 작성 후 commit; 하면 없어진다. 그러나 테이블은 남아있다

```
ex2) on commit preserve rows
create global temporary table emp800
( empno number(10),
  ename varchar2(10),
  sal    number(10) )
on commit preserve rows;
```

```
insert into emp800
select empno, ename, sal
  from emp;
```

>>> 이렇게 작성 후 commit 해도 사라지지 않으며 exit 혹은 developer 종료 시 사라진다

★

366) 구구단 문제를 풀기위한 from 절의 서브쿼리문의 결과를 임시테이블로 각각 생성하시오!

>> num1_9 table 생성

```
create global temporary table num1_9
( num1 number(10) )
on commit preserve rows;
```

```
insert into num1_9
select level
  from dual
 connect by level <= 9;
```

>> num2_9 table 생성

```
create global temporary table num2_9
( num2 number(10) )
on commit preserve rows;
```

```
insert into num2_9
select level
  from dual
 connect by level <= 9;
```

>>3 테이블 조인

```
select a.num1, b.num2
```

```
from num1_9 a, num2_9 b;
```

367) ##9) 주사위를 10만 번 던져서 주사위의 눈이 6이 나올 확률은 어떻게 되는가?

undefine n

select count(*) /&&n

```
from (select round( dbms_random.value(0.5, 6.5) ) as p
      from dual
      connect by level <= &n
      )
```

where p = 6;

여기서 random value 가 바뀌는 이유는 1과 6의sel 확률을 지키기 위해서이다

☆368) ##10) 두 개의 주사위를 던져서 두개의 주사위의 눈의 합이 10이 되는 확률을 구하시오!

>>1 from 절 sub query 2개 사용

select count(*)/1000000

```
from ( select round( dbms_random.value(0.5, 6.5) ) as p1
      from dual
      connect by level <= 1000) a,
      ( select round( dbms_random.value(0.5, 6.5) ) as p2
      from dual
      connect by level <= 1000) b
```

where a.p1+b.p2 = 10;

>>2 from 절 sub query 하나로 끝내기(>>1에 비해 속도가 월등히 빠름)

undefine flip

select count(*) / &&flip

```
from ( select round( dbms_random.value(0.5,6.5) ) as dice_1,
      round( dbms_random.value(0.5,6.5) ) as dice_2, &flip
      from dual
      connect by level <= &flip )
where dice_1+dice_2 = 10;
```

095 복잡한 쿼리를 단순하게 하기 1(VIEW)

view : 테이블은 아니고 데이터를 바라보는 쿼리문을 테이블처럼 하나의 object(객체)로 생성

ex1)

create table emp708

as

```
select empno, enmae, sal, deptno
      from emp;
```

as 다음에 나오는 쿼리문의 결과대로 emp708 테이블이 생성된다.

emp708 은 emp 테이블과는 다른 별개의 또 다른 테이블이다.

※ emp 로 시작하는 내가 만든 테이블이 뭐가 있는지 확인하는 방법

select table_name

from user_tables

where table_name like 'EMP%'; <<< 강조부분은 반드시 대문자로 해야 검색이 됨

★

369) DALLAS 에서 근무하는 직원들의 이름과 부서위치를 출력하시오!

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

370) 위 결과의 데이터를 담은 테이블 emp710 을 생성하시오

```
create table emp710
as
select e.ename, d.loc
      from emp e, dept d
     where e.deptno = d.deptno and d.loc = 'DALLAS';
```

371) 숫자 1번부터 10번까지의 숫자를 담은 테이블을 emp705 로 생성하시오!

```
create table emp705
as
select level as num1 >> level 별칭을 지정하지 않으면 err 발생
      from dual
     connect by level <= 10;
```

372) emp705 의 숫자 데이터 중 임의로 아무거나 하나를 지우시오

```
delete from emp705
      where num1 = 4;
```

※카카오에 입사하는 방법

1. 신입 : 알고리즘 문제(매우 어려움)

2. 경력 : 포트폴리오 제출(딥러닝, 머신러닝 등) >>> 알고리즘 문제 1문제 >>> 면접

373) ##11) 1부터 10 사이의 숫자 중 하나가 없다. 그 수는?

emp705 를 쿼리해서 한번에 알아내시오! (372번이랑 연계)

>>내 답

```
select level as num2
      from dual
     connect by level <= 10
```

minus

```
select num1
      from emp705;
```

>>2 inline view + where 조건

```
select num2
      from ( select level as num2
            from dual
           connect by level <=10 )
     where num2 not in ( select num1
                       from emp705 );
```

>>3

```
select max(b.원본) - sum(a.num1)
      from emp705 a,(select sum(level) 원본
                    from dual
                   connect by level <= 10) b;
```


view설명

```
create view emp801
as
```

```
select empno, ename, sal, deptno
from emp;
```

view는 테이블과는 다르게 별도로 데이터를 저장하고 있지 않고 그냥 그 테이블을 바로보는 쿼리문이다.

```
update emp801
```

```
set sal = 0
```

```
where ename = 'SCOTT';
```

이후 emp801 을 select 하면 view emp801 은 변경된다

또한, emp 테이블도 변경된다 >>> view 를 업데이트하면 view 를 create 할때 바라보았던 원형 테이블도 수정된다

활용 : 회사의 데이터중에 공개하면 안되는 데이터가 있다. 민감한 데이터 외에는 공개를 원함

ex) 라이나생명 데이터 분석을 하러 갔는데 라이나 생명에서 일하는 직원들의 데이터가 있는데 직원 데이터 중 월급은 라이나 생명에 공개하지 않고 나머지 데이터만 공개를 하고 얼마든지 조회하고 업데이트 가능하게 하고 싶을 때 월급을 제외하고 나머지 데이터를 view로 생성하면 가능하다

```
ex1-2) create view emp809
```

```
as
```

```
select empno, ename, job, hiredate, mgr, deptno
from emp; <<< sal 은 제외되었다
```

※ 예약어 level

```
create table emp705
```

```
as
```

```
select level as num1
```

```
from dual
```

```
connect by level <= 10;
```

위의 level 은 SQL 문과 같은 예약어여서 예약어를 테이블의 컬럼명으로 생성할 수 없다

예약어 : select, from, where,level 등등

★

374) 직업, 직업별 토달월급을 출력하시오! (단, 세로로 출력)

```
select job, sum(sal)
```

```
from emp
```

```
group by job;
```

375) 위 결과를 출력하는 view 를 생성하시오. (단, view 이름은 emp403이다)

```
create view emp403
```

```
as
```

```
select job, sum(sal) as A
```

```
from emp
```

```
group by job;
```

view 를 생성할때 그룹함수를 사용하게 되면 컬럼별칭을 주어야 한다

376) 부서번호, 부서번호별 평균월급을 출력하는 view 를 deptno_avg 라는 이름으로 생성하시오!

```
create view deptno_avg
```

```
as
```

```
select deptno, avg(sal) as 부서평균
```

```
from emp
```

group by deptno;

377) emp 와 지금 만든 deptno_avg view 와 조인해서 이름, 월급, 부서번호, 부서평균을 출력하시오!

```
select e.ename, e.sal, e.deptno, d.부서평균  
from emp e, deptno_avg d  
where e.deptno=d.deptno;
```

view 가 있어서 위 문제에서 from 절의 서브쿼리를 사용하지 않고 해결할 수 있었음

만약 viewc 테이블을 생성하고 inline view 를 이용하면 아래와 같다

```
select e.ename, e.sal, e.deptno, d.부서평균  
from emp e, (  
    select avg(sal) as 부서평균, deptno  
    from emp  
    group by deptno  
) d  
where e.deptno=d.deptno and e.sal > d.부서평균;
```

※ view 의 장점

1. 민감한 컬럼을 감춰서 데이터를 제공할 수 있다.
2. 복잡한 쿼리문을 단순하게 만들 수 있다. >> 검색속도의 향상

378) 이름, 월급, 부서번호, 부서평균을 출력하는데 월급이 부서평균보다 더 큰 사원들만 출력하시오!

```
select e.ename, e.sal, e.deptno, d.부서평균  
from emp e, deptno_avg d  
where e.deptno=d.deptno and e.sal > d.부서평균;
```

379) emp 테이블에서 퇴사할 것 같은 사원들을 예측하기 위해 자기의 월급이 자기가 속한 직업의 평균월급보다 더 작은 사원들의 이름과 월급과 직업과 직업평균을 출력하시오!

```
create table job_avg  
as  
select deptno, avg(sal) as 직업평균  
from emp  
group by sal;
```

```
select e.ename, e.sal, e.job, v.직업평균  
from emp e, job_avg v  
where e.job = v.job and e.sal < v.직업평균;
```

380) 직업, 이름, 월급, 순위를 출력하는데 순위가 직업별로 각각 월급이 높은 순서대로 순위를 부여하시오!

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

381) 위 쿼리의 결과를 view로 만들고 view를 쿼리해서 순위가 1등인 사원들만 출력하시오!

※ view의 종류

	테이블 개수	그룹함수	DML여부
1) 단순 view :	1개	포함X	가능
2) 복합 view :	2개	포함	불가능할 수도 있음

382) 이름과 부서위치를 출력하는 view 를 `ename_loc` 라는 이름으로 생성하시오!

```
create view ename_loc
as
select e.ename, d.loc
  from emp e, dept d
 where e.deptno = d.deptno;
```

383) `ename_loc` 의 데이터 중에서 SCOTT 의 부서위치를 WASHINGTON 으로 바꾸시오!

```
update ename_loc
  set loc='WASHINGTON'
 where ename = 'SCOTT';
```

SQL 오류: ORA-01779: 키-보존된것이 아닌 테이블로 대응한 열을 수정할 수 없습니다.
01779. 00000 - "cannot modify a column which maps to a non key-preserved table"
*Cause: An attempt was made to insert or update columns of a join view which
map to a non-key-preserved table.
*Action: Modify the underlying base tables directly.

err 발생 : 변경불가. SCOTT 의 부서위치가 DALLAS 에서 WS로 변경이 된다면 실제로 DEPT 테이블의 부서위치가 DALLS 에서 WS 로 변경되는거라서 SCOTT이 아닌 다른 사원들도 DALLAS에서 WS으로 변경되어지므로 변경이 불가하도록 oracle에서 설정한 것이다. ws : wasington

위 문제 중 375)에서 만든 emp403 view 를 아래와 같이 쿼리하고 결과를 보시오!

```
update emp403
set a = 2000
where job = 'SALESMAN';
```

SQL 오류: ORA-01732: 뷰에 대한 데이터 조작이 부적합합니다
1732. 00000 - "data manipulation operation not legal on this view"
*Cause: Windows 정중
*Action: [설정]으로 이동하여

위와 같이 복합 뷰는 데이터를 갱신할 수 없다. 만약 토탈월급(a) 이 갱신된다면 실제값도 갱신해줘야 하기 때문에 갱신할 수도 없고 해서도 안된다.

384) 내가 그동안 만든 view 들이 무엇이 있는지 확인하시오!

```
select view_name
  from user_views;
```

이때 1행에 select view_name, text 이나 select * 로 수행하면

VIEW_NAME	TEXT
EMP801	select empno, ename, sal, deptno from emp
EMP403	select job, sum(sal) as a from emp group by job
DEPTNO_AVG	select deptno, avg(sal) as 부서평균 from emp group by deptno
JOB_AVG	select job, avg(sal) as 직업평균 from emp group by job
RANKING	select dense_rank() over(partition by job order
ENAME_LOC	select e.ename, d.loc from emp e, dept d where e.deptno = d.deptno

위와같이 view 를 만들때 사용한 테이블명도 볼 수 있다. 외부에서 온 데이터 분석가들은 위의 쿼리를 조회할 수 있는 권한을 제외하는 경우가 많다.

※ delete view

```
drop view emp801
```

097 데이터 검색 속도를 높이기(INDEX)

오라클 데이터베이스의 객체(object)의 종류 5가지

- 1) table : 데이터를 저장하는 기본 저장소
- 2) view : 데이터를 바로보는 쿼리문
- 3) index : 검색 속도를 높이기 위한 db object
- 4) sequence : 순서대로 번호를 생성하는 db object
- 5) synonym : 테이블명에 대한 또 다른 이름

db object : SQL 튜닝을 위해 반드시 알고 있어야 하는 검색속도를 높여주는 기술

index : 책이 테이블이면, index 는 목차

ex1) 이름이 SCOTT 인 사원의 이름과 월급을 출력하시오!

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

위 SQL은 ename에 index 가 없기 때문에 SCOTT 을 emp 테이블에서 찾을 때 처음부터 끝까지 full table scan 했을 것이다

ex2) emp 테이블에 ename 에 index 를 생성하시오!

```
create index emp_ename << index 이름
on emp(ename);
테이블명(컬럼명)
```

위와같이 ename 에 목차(index) 를 만들면 ename 을 이용해서 만든 인덱스의 구조는 컬럼명+rowid 로 구성되어 있고 컬럼명은 abc 순으로 정렬되어 index를 구성함.

rowid : 그 행을 대표하는 주소

ex3) rowid 활용

```
select rowid, ename, sal
from emp;
```

ROWID	ENAME	SAL
1 AAAR/GAAHAAAAIeAAA	KING	5000
2 AAAR/GAAHAAAAIeAAB	BLAKE	2850
3 AAAR/GAAHAAAAIeAAC	CLARK	2450
4 AAAR/GAAHAAAAIeAAD	JONES	2975
5 AAAR/GAAHAAAAIeAAE	MARTIN	1250
6 AAAR/GAAHAAAAIeAAF	ALLEN	1600
7 AAAR/GAAHAAAAIeAAG	TURNER	1500
8 AAAR/GAAHAAAAIeAAH	JAMES	950
9 AAAR/GAAHAAAAIeAAI	WARD	1250
10 AAAR/GAAHAAAAIeAAJ	FORD	3000
11 AAAR/GAAHAAAAIeAAK	SMITH	800
12 AAAR/GAAHAAAAIeAAL	SCOTT	3000
13 AAAR/GAAHAAAAIeAAM	ADAMS	1100
14 AAAR/GAAHAAAAIeAAN	MILLER	1300

위 rowid 를 이용해서 검색도 가능하다

```
select rowid, ename, sal
from emp
where rowid = 'AAAR/GAAHAAAAIeAAA';
```

ROWID	ENAME	SAL
1 AAAR/GAAHAAAAIeAAA	KING	5000

한편, emp_ename의 index 구조를 보려면..

```
select ename, rowid
from emp
where ename > ' '; <<< 공백을 준 것
```

ENAME	ROWID
1 ADAMS	AAAR/GAAHAAAAIeAAM
2 ALLEN	AAAR/GAAHAAAAIeAAF
3 BLAKE	AAAR/GAAHAAAAIeAAB
4 CLARK	AAAR/GAAHAAAAIeAAC
5 FORD	AAAR/GAAHAAAAIeAAJ
6 JAMES	AAAR/GAAHAAAAIeAAH
7 JONES	AAAR/GAAHAAAAIeAAD
8 KING	AAAR/GAAHAAAAIeAAA
9 MARTIN	AAAR/GAAHAAAAIeAAE
10 MILLER	AAAR/GAAHAAAAIeAAN
11 SCOTT	AAAR/GAAHAAAAIeAAL
12 SMITH	AAAR/GAAHAAAAIeAAK
13 TURNER	AAAR/GAAHAAAAIeAAG
14 WARD	AAAR/GAAHAAAAIeAAI

위 3행의 where절에서 [ename > ' '] 조건을 주었기 때문에 index 에서 ename 을 가져와서 order by 절 없이도 정렬된 ename 결과를 볼 수 있다.

★

385) 사원 테이블의 월급에 인덱스를 거시오!

```
create index emp_sal
on emp(sal);
```

386) emp 테이블의 sal 의 인덱스인 emp_sal 의 구조를 확인하시오!

```
select sal, rowid
from emp
where sal >= 0;
```

앞의 예시와 마찬가지로 index 에서 읽어 온 것이므로 (3행의 조건에 의해) sal 이 desc 정렬됨

☆387) ##12) 두 개의 주사위를 동시에 던져서 주사위의 눈의 합이 짝수가 되는 확률은?

>>1 in line view 사용

```
select count(*)/100000
from ( select round( dbms_random.value(0.5, 6.5)) as d1,
round( dbms_random.value(0.5,6.5)) as d2
from dual
connect by level <=100000)
where mod(d1+d2, 2) = 0;
```

>>2 한결씨 정답 : 근데 10번 알고리즘처럼 검색속도의 향상은 없었음(둘 다 비슷함)

하지만 검색속도를 입력하고 초기화할 수 있다는 점에서 더 정교한 SQL임

undefine 던진횟수;

```
select (count(*)/ &던진횟수) * 100 || '%' as "두 눈의 합이 짝수"
```

```
from
(select round(dbms_random.value(0.5,6.5)) num1,
round(dbms_random.value(0.5,6.5)) num2, &던진횟수
from dual
connect by level <= &던진횟수)
```

where mod(num1 + num2,2) = 0;

>>3 다시 정리 -- 이게 현재 내가 쓸수있는 SQL 중 가장 깔끔함

undefine flip;

select (count(*)/ &flip) * 100 || '%' as mod2

```
from
(select round(dbms_random.value(0.5,6.5)) num1,
round(dbms_random.value(0.5,6.5)) num2, &flip
```

```
from dual
  connect by level <= &flip)
where mod(num1 + num2,2) = 0;
```

##1111

2020년 11월 11일 수요일 오전 9:42

복습

1. view 생성하는 방법과 view 를 사용해야 하는 이유
2. 인덱스가 무엇인지, 인덱스를 생성하는 방법, 인덱스의 구조
3. 알고리즘 문제 → 확률의 고전적 정의 : $P(A) = \text{사건 A에 속하는 원소수} / \text{표본공간의 전체 원소수}$
 - 1) **dbms_random.value(n1, n2) : n1 과 n2 사이의 실수를 랜덤으로 출력하는 함수(단, 가우시안 분포를 따른다)**
 - 2) 위 함수와 round 를 이용해서 동전을 던지는 경우(2가지 경우의 수), 주사위(6가지) 등의 환경조성이 가능

시작

388) ##13) 주사위 하나와 동전 한개를 동시에 던져서 주사위의 눈은 5가 나오고 동전은 앞면이 나올 확률은 어떻게 되는가? 단 앞면이 0, 뒷면은 1이다

undefine flip

```
select count(*)/&flip
from ( select round(dbms_random.value(0,1)) as coin,
              round(dbms_random.value(0.5,6.5)) as dice, &flip
        from dual
        connect by level <= &flip
      )
where coin = 0 and dice = 5;
```

※ index

- 1) index 를 이용해야 데이터를 빠르게 검색가능
- 2) table 이 책이면 index 는 목차
- 3) 20분 >> 0.1초 (쉬운 튜닝) , 0.1초>>0.01초 (어려운 튜닝)
- 4) 튜닝을 위해서는 인덱스의 구조를 이해하고 활용을 잘 할 수 있어야 함
- 5) SQL튜너 중 중급과 고급의 차이는 알고리즘 문제 해결 능력(스스로 해결했는지)

index 의 구조 : 컬럼명 + rowid (row의 물리적 주소)

컬럼명이 오름차순으로 정렬되어 있음

ex1) emp table >> ename index making

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

emp 테이블의 ename 에 인덱스가 있으므로 ename의 인덱스인 emp_ename 을 통해 테이블의 데이터를 검색.

만약 index 가 없다면 full table scan 을 해야 한다. 대용량 데이터가 있는 환경에서 SQL 수행시 너무 느리다면 반드시 **실행계획**을 확인하여 full table scan 을 했는지 index scan 을 했는지 확인을 해야 한다.

explain plan for

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

select *

from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT							
1	Plan hash value: 3956160932						
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	10	3 (0)	00:00:01
7	* 1	TABLE ACCESS FULL	EMP	1	10	3 (0)	00:00:01
8							
9							
10	Predicate Information (identified by operation id):						
11							
12							
13	1	- filter('ENAME'='SCOTT')					

실행계획을 만드는 옵티마이저라는 프로세서가 있는데 이 프로세서가 점점 인공지능화 되어가고있다.

위 음영부분처럼 full table scan 으로 나왔다면 SCOTT 를 검색하기 위해 emp 테이블을 풀스캔했다는 의미이다.

만약 위 경우에서 index 가 존재한다면 (기존에 만든 emp_ename) 아래와 같다

PLAN_TABLE_OUTPUT							
1	Plan hash value: 2855689319						
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	10	2 (0)	00:00:01
7	1	TABLE ACCESS BY INDEX ROWID BATCHED	EMP	1	10	2 (0)	00:00:01
8	* 2	INDEX RANGE SCAN	EMP_ENAME	1		1 (0)	00:00:01
9							
10							
11	Predicate Information (identified by operation id):						
12							
13							
14	2	- access('ENAME'='SCOTT')					

그런데 만약 위 index 를 생성하고 explain 을 시도했을 때 full scan 으로 뜬다면(data 가 너무 작아서 그럴수 있음)

힌트를 이용해서 [emp 테이블 emp_ename 에 index 를 통해서 검색해라] 라고 옵티마이저에게 직접 지정해줄수있다
explain plan for

```
select /*+ index(emp emp_ename) */ ename, sal <<< /*+ 는 다 붙여써야 됨. + 사이에 공백이 있으면 주석
from emp
where ename = 'SCOTT';
```

select *

from table(dbms_xplan.display);

★

389) 사원 테이블에 월급이 인덱스를 생성하고 사원 테이블을 검색하는데 월급이 3000 인 사원들의 이름과 월급을 검색하시오!

create index emp_sal

on emp(sal);

select ename,sal

from emp

where sal = 3000;

이후 plan 확인을 하여 full scan 인지 index range scan 인지 확인해볼 수 있다.

390) 위 SQL의 실행계획을 확인하시오!

explain plan for

select ename, sal

from emp

where sal = 3000;

select *


```
from table(dbms_xplan.display);
```

>> 2행-4행 드래그 후 F10 눌러서 볼 수도 있다(양식은 다름)

391) 위 SQL이 emp_sal 의 인덱스를 통해서 검색할 수 있도록 힌트를 주시오!

```
select /*+ (ename emp_sal) */ ename, sal
from emp
where sal = 3000;
```

happy babaro day

오전 11:11
2020-11-11

※ index 테이블 조회구조

이름이 BLAKE 인 사원의 이름과 월급을 출력하시오!

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

index scan	full table scan																																																																											
select ename, rowid from emp where ename > ' ' ;	select rowid, ename, sal from emp;																																																																											
<table><tr><th>ENAME</th><th>ROWID</th></tr><tr><td>1 ADAMS</td><td>AAASCoAAHAAAAJmAAM</td></tr><tr><td>2 ALLEN</td><td>AAASCoAAHAAAAJmAAF</td></tr><tr><td>3 BLAKE</td><td>AAASCoAAHAAAAJmAAB</td></tr><tr><td>4 CLARK</td><td>AAASCoAAHAAAAJmAAC</td></tr><tr><td>5 FORD</td><td>AAASCoAAHAAAAJmAAJ</td></tr><tr><td>6 JAMES</td><td>AAASCoAAHAAAAJmAAH</td></tr><tr><td>7 JONES</td><td>AAASCoAAHAAAAJmAAD</td></tr><tr><td>8 KING</td><td>AAASCoAAHAAAAJmAAA</td></tr><tr><td>9 MARTIN</td><td>AAASCoAAHAAAAJmAAE</td></tr><tr><td>10 MILLER</td><td>AAASCoAAHAAAAJmAAN</td></tr><tr><td>11 SCOTT</td><td>AAASCoAAHAAAAJmAAL</td></tr><tr><td>12 SMITH</td><td>AAASCoAAHAAAAJmAAK</td></tr><tr><td>13 TURNER</td><td>AAASCoAAHAAAAJmAAG</td></tr><tr><td>14 WARD</td><td>AAASCoAAHAAAAJmAAI</td></tr></table>	ENAME	ROWID	1 ADAMS	AAASCoAAHAAAAJmAAM	2 ALLEN	AAASCoAAHAAAAJmAAF	3 BLAKE	AAASCoAAHAAAAJmAAB	4 CLARK	AAASCoAAHAAAAJmAAC	5 FORD	AAASCoAAHAAAAJmAAJ	6 JAMES	AAASCoAAHAAAAJmAAH	7 JONES	AAASCoAAHAAAAJmAAD	8 KING	AAASCoAAHAAAAJmAAA	9 MARTIN	AAASCoAAHAAAAJmAAE	10 MILLER	AAASCoAAHAAAAJmAAN	11 SCOTT	AAASCoAAHAAAAJmAAL	12 SMITH	AAASCoAAHAAAAJmAAK	13 TURNER	AAASCoAAHAAAAJmAAG	14 WARD	AAASCoAAHAAAAJmAAI	<table><tr><th>ROWID</th><th>ENAME</th><th>SAL</th></tr><tr><td>1 AAASCoAAHAAAAJmAAA</td><td>KING</td><td>5000</td></tr><tr><td>2 AAASCoAAHAAAAJmAAB</td><td>BLAKE</td><td>2850</td></tr><tr><td>3 AAASCoAAHAAAAJmAAC</td><td>CLARK</td><td>2450</td></tr><tr><td>4 AAASCoAAHAAAAJmAAD</td><td>JONES</td><td>2975</td></tr><tr><td>5 AAASCoAAHAAAAJmAAE</td><td>MARTIN</td><td>1250</td></tr><tr><td>6 AAASCoAAHAAAAJmAAF</td><td>ALLEN</td><td>1600</td></tr><tr><td>7 AAASCoAAHAAAAJmAAG</td><td>TURNER</td><td>1500</td></tr><tr><td>8 AAASCoAAHAAAAJmAAH</td><td>JAMES</td><td>950</td></tr><tr><td>9 AAASCoAAHAAAAJmAAI</td><td>WARD</td><td>1250</td></tr><tr><td>10 AAASCoAAHAAAAJmAAJ</td><td>FORD</td><td>3000</td></tr><tr><td>11 AAASCoAAHAAAAJmAAK</td><td>SMITH</td><td>800</td></tr><tr><td>12 AAASCoAAHAAAAJmAAL</td><td>SCOTT</td><td>3000</td></tr><tr><td>13 AAASCoAAHAAAAJmAAM</td><td>ADAMS</td><td>1100</td></tr><tr><td>14 AAASCoAAHAAAAJmAAN</td><td>MILLER</td><td>1300</td></tr></table>	ROWID	ENAME	SAL	1 AAASCoAAHAAAAJmAAA	KING	5000	2 AAASCoAAHAAAAJmAAB	BLAKE	2850	3 AAASCoAAHAAAAJmAAC	CLARK	2450	4 AAASCoAAHAAAAJmAAD	JONES	2975	5 AAASCoAAHAAAAJmAAE	MARTIN	1250	6 AAASCoAAHAAAAJmAAF	ALLEN	1600	7 AAASCoAAHAAAAJmAAG	TURNER	1500	8 AAASCoAAHAAAAJmAAH	JAMES	950	9 AAASCoAAHAAAAJmAAI	WARD	1250	10 AAASCoAAHAAAAJmAAJ	FORD	3000	11 AAASCoAAHAAAAJmAAK	SMITH	800	12 AAASCoAAHAAAAJmAAL	SCOTT	3000	13 AAASCoAAHAAAAJmAAM	ADAMS	1100	14 AAASCoAAHAAAAJmAAN	MILLER	1300
ENAME	ROWID																																																																											
1 ADAMS	AAASCoAAHAAAAJmAAM																																																																											
2 ALLEN	AAASCoAAHAAAAJmAAF																																																																											
3 BLAKE	AAASCoAAHAAAAJmAAB																																																																											
4 CLARK	AAASCoAAHAAAAJmAAC																																																																											
5 FORD	AAASCoAAHAAAAJmAAJ																																																																											
6 JAMES	AAASCoAAHAAAAJmAAH																																																																											
7 JONES	AAASCoAAHAAAAJmAAD																																																																											
8 KING	AAASCoAAHAAAAJmAAA																																																																											
9 MARTIN	AAASCoAAHAAAAJmAAE																																																																											
10 MILLER	AAASCoAAHAAAAJmAAN																																																																											
11 SCOTT	AAASCoAAHAAAAJmAAL																																																																											
12 SMITH	AAASCoAAHAAAAJmAAK																																																																											
13 TURNER	AAASCoAAHAAAAJmAAG																																																																											
14 WARD	AAASCoAAHAAAAJmAAI																																																																											
ROWID	ENAME	SAL																																																																										
1 AAASCoAAHAAAAJmAAA	KING	5000																																																																										
2 AAASCoAAHAAAAJmAAB	BLAKE	2850																																																																										
3 AAASCoAAHAAAAJmAAC	CLARK	2450																																																																										
4 AAASCoAAHAAAAJmAAD	JONES	2975																																																																										
5 AAASCoAAHAAAAJmAAE	MARTIN	1250																																																																										
6 AAASCoAAHAAAAJmAAF	ALLEN	1600																																																																										
7 AAASCoAAHAAAAJmAAG	TURNER	1500																																																																										
8 AAASCoAAHAAAAJmAAH	JAMES	950																																																																										
9 AAASCoAAHAAAAJmAAI	WARD	1250																																																																										
10 AAASCoAAHAAAAJmAAJ	FORD	3000																																																																										
11 AAASCoAAHAAAAJmAAK	SMITH	800																																																																										
12 AAASCoAAHAAAAJmAAL	SCOTT	3000																																																																										
13 AAASCoAAHAAAAJmAAM	ADAMS	1100																																																																										
14 AAASCoAAHAAAAJmAAN	MILLER	1300																																																																										
ename, sal 의 index 존재	index no exist (오래걸린다)																																																																											

- index 도 저장공간을 차지
- index 이름이 겹치면 덮어쓰기
- index 항목이 많아지면 insert 시 속도가 느려짐 <<< insert 시 index 고려하여 삽입하기 때문

392) 입사일에 인덱스를 생성하고 81년 11월 17일에 입사한 사원의 이름과 입사일을 조회하시오!

```
create index emp_hiredate
on emp(hiredate);
select ename, hiredate
from emp
where hiredate = to_date('81/11/17', 'RR/MM/DD');
```

393) 위 SQL의 실행계획을 확인하고 혹시 full table scan 을 했으면 **index scan** 을 할 수 있도록 튜닝하시오!

explain plan for

```
select /*+ emp emp_hiredate */ ename, hiredate
  from emp
  where hiredate = to_date('81/11/17', 'RR/MM/DD');
```

select *

```
  from table(dbms_xplan.display);
```

※ 인덱스의 구조를 전부 읽어오는 방법(아래 조건이 where 절에 있어야 index 전체를 읽어 올 수 있다)

1) 문자 : ' '

2) 숫자 : >= 0

3) 날짜 : <= to_date('9999/12/31', 'RRRR/MM/DD')

****이때, where 절에 조건을 추가해야만 index 를 통해 data를 출력할 수 있다. (없으면 full scan함)**

※ index 의 구조를 알면 SQL 튜닝이 가능하다. 그 방법중 하나

1) order by 절을 사용하지 않고 정렬된 결과를 볼 수 있다.

order by 절을 남발하면 검색성능이 느려진다.

2) order by 절을 사용하지 않고 index 를 통해서 정렬하면 SQL 튜닝이 된다.

ex) order by 절 사용

```
select ename, sal
  from emp
  order by sal asc;
```

ex2) index 사용

```
select /*+ index_desc(emp emp_sal) */ ename, sal
  from emp
  where sal >= 0;
```

index 를 통해서 정렬된 데이터를 보려면 where 절에 반드시 해당 컬럼을 검색하는 조건이 있어야 한다.
이 때, 원하는 대로 정렬이 안된다면 힌트(위 갈색)를 주어 원하는 대로 결과를 출력할 수 있다.

※ 힌트(반드시 where 절과 함께 사용)

1) /*+ index_asc(table index_name) */ <<< index_name(인덱스)에 대해 asc 정렬하는 힌트

2) /*+ index_desc(table index_name) */

3) /*+ index(table index_name) */ <<< index를 하지 않을때 명령하기

394) 이름과 월급을 출력하는데 월급이 높은 사원부터 출력하시오! (단 order by 절을 사용하지 말고 index를 사용)

```
select /*+ index_desc(emp emp_sal) */ ename, sal
  from emp
  where sal >= 0;
```

index 를 이용해 정렬하고자 할때 반드시 where 절에 조건을 주어야 원하는 결과 출력 가능

☆395) 아래의 SQL을 튜닝하시오!

튜닝 전

```
select ename, hiredate
  from emp
  order by hiredate desc;
```

튜닝후

```
select /*+ index_desc(emp emp_hiredate) */ ename, hiredate
from emp
where hiredate <= to_date('9999/12/31', 'RRRR/MM/DD');
```

인덱스 여부 확인

explain plan for

```
select /*+ index_desc(emp emp_hiredate) */ ename, hiredate
from emp
where hiredate <= to_date('9999/12/31', 'RRRR/MM/DD');
```

```
select *
from table(dbms_xplan.display);
```

※ 인덱스를 사용한 SQL 튜닝하는 방법(가공된 column에 대해)

ex1) 튜닝전

```
select /*+ index(emp emp_sal) */ ename, sal
from emp
where sal*12 = 36000;
```

```
select * from table(dbms_xplan.display);
```

6	0	SELECT STATEMENT		1	10	3	(0)	00:00:01	
7	1	TABLE ACCESS FULL	EMP	1	10	3	(0)	00:00:01	

emp 테이블에 emp_sal 인덱스가 있고, 힌트를 주어도 full scan 으로 실행된다.

그 이유는 where 절에 sal 이 sal*12 로 가공이 되어있기 때문이다.

ex2) 튜닝 후

```
select /*+ index(emp emp_sal) */ ename, sal
from emp
where sal = 36000/12;
```

ENAME	SAL
1 FORD	3000
2 SCOTT	3000

display :

1	TABLE ACCESS BY INDEX ROWID BATCHED	EMP
2	INDEX RANGE SCAN	EMP_SAL

396) 사원 테이블에 직업에 인덱스를 생성하시오!

```
create index emp_job
on emp(job);
```

397) 아래의 SQL을 튜닝하시오!

튜닝전

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

튜닝후

```
select /*+ index(emp emp_job) */ ename, job
from emp
where job like 'SALES%';
```

튜닝후 SQL에서 본 것처럼 where 절의 좌변의 컬럼을 가공하면 full table scan 하게 되므로 가공X

추가로, 위 hint 는 줘도 되고 안줘도 된다(index가 안될때만 주자), hint는 asc/desc 정렬을 정하고 싶을 때 활용가능

398) 아래의 SQL을 튜닝하시오!

튜닝전

```
select ename, hiredate
  from emp
 where to_char(hiredate, 'RRRR') = '1981';
```

튜닝후

```
select ename, hiredate
  from emp
 where hiredate between to_date('81/01/01','RR/MM/DD') and to_date('81/12/31','RR/MM/DD');
```

확인

explain plan for ~튜닝후~ select * from table(dbms_xplan.display);

	0		SELECT STATEMENT	
*	1		FILTER	
	2		TABLE ACCESS BY INDEX ROWID BATCHED	
*	3		INDEX RANGE SCAN	

399) 직업이 SALESMAN 인 사원들의 이름과 월급과 직업을 출력하는데 월급이 높은 사원부터 출력하시오!

```
select /*+ index_desc(emp emp_sal) */ ename, sal, job
  from emp
 where sal >= 0 and job = 'SALESMAN';
```

튜닝시 order by 절 사용 최소화

※ 내가 생성한 index 목록 확인하기

```
select index_name
```

```
  from user_indexes;
```

3행에 조건을 주어서 특정 조건을 만족하는 index 만 확인할 수 있다.

```
select index_name
```

```
  from user_indexes
```

```
 where index_name like 'EMP%'; << EMP 는 대문자로 검색해야 검색이 가능함
```

※ delete index

```
drop index emp_ename;
```

400) 나머지 index 도 전부 drop 하시오!

```
drop index emp_ename;
```

```
drop index emp_sal;
```

```
drop index emp_hiredate;
```

```
drop index emp_job;
```

098 절대로 중복되지 않는 번호 만들기(SEQUENCE)

sequene : 번호를 중복되지 않게 순서대로 생성하는 번호 생성기

ex1) 숫자 생성하기

```
create sequence seq1
```

```
  start with 1          <<<<< 시작숫자
```

```
  maxvalue 100;        <<<<< 최대숫자
```

ex2) 실행하기

select seq1.nextval

from dual; <<< 실행할 때마다 nextval(column) 의 값이 1씩 증가한다(100까지)

이때, ex2)가 실행시 100 이 넘어가면 아래 err 발생

```
ORA-08004: 시퀀스 SEQ1.NEXTVAL exceeds MAXVALUE로 사례로 될 수 없습니다
ORA-08004. 00000 - "sequence %s.NEXTVAL %s %sVALUE and cannot be instantiated"
*Cause:   instantiating NEXTVAL would violate one of MAX/MINVALUE
*Action:  alter the sequence so that a new value can be requested
```

sequence 을 이용하면 어떤점이 좋은가?

- 1) 번호를 중복되지 않게 일관되게 테이블에 입력할 수 있다
- 2) 번호가 중복되면 안되는 컬럼들 : 주식 테이블 매매번호, 사원 테이블의 사원번호 등

ex3) 시퀀스를 seq2 라고 해서 만드시오! (1번부터 1000번까지)

create sequence seq2

start with 1

maxvalue 1000;

ex4) 아래의 컬럼을 담는 테이블을 emp534 라는 이름으로 생성하시오! (empno, ename, sal 컬럼)

create table emp534

(empno number(10),

ename varchar2(20),

sal number(10));

ex5) ex2)에서 만든 seq2 를 이용해서 emp534 가 empno 에 일관된 번호가 입력되게 하시오!

insert into emp534

values(seq2.nextval, 'scott', 3000);

select * from emp534;

	EMPNO	ENAME	SAL
1	222	scott	3000
2	223	scott	3000
3	224	scott	3000

위 결과값과 같이 ex5) 의 첫 SQL을 반복적으로 수행하면 empno가 1개씩 증가되어 data 가 입력된다.

※시퀀스

현재값 확인	내가 생성한 시퀀스 값 확인	시퀀스 삭제
select seq2.currval from dual;	select sequence_name from user_sequences;	drop sequence seq2;

103 실수로 지운 데이터 복구하기 5(FLASHBACK TRANSACTION QUERY)

오라클은 10g 버전부터 타임머신 기능이 있다. 이 기능으로 테이블을 과거로 돌릴 수 있다.

flashback query 는 과거의 테이블을 확인하는 기능이다.

ex1) delete table data

delete from emp;

commit;

ex2) 과거데이터 확인

select *

```
from emp as of timestamp (systimestamp - interval '1' minute);
```

2행의 안의 숫자부분은 삭제되고 난 후 경과된 시간내로 입력해야 정보가 출력된다

401) 백업받은 emp_backup_20201111 의 데이터를 emp 테이블에 입력하시오!

```
insert into emp
select *
from emp_backup_20201111;
```

402) 아래와 같이 salgrade 테이블을 전부 delete 하고 commit 한 후에 복구하시오!

```
delete from salgrade;
commit;
>>1 자료위치찾기
select *
from salgrade as of timestamp ( systimestamp - interval '3' minute);
>>2 백업파일생성
create table salgrade_backup
as
select *
from salgrade as of timestamp ( systimestamp - interval '3' minute);
>>3 백업파일로 salgrade data 주입
insert into salgrade
select *
from salgrade_backup;
>>4 데이터 확인
select *
from salgrade;
```

103 실수로 지운 데이터 복구하기 2(FLASHBACK TRANSACTION QUERY)

flashback : 특정 테이블을 과거로 완전히 되돌려버리는 기능

ex1) delete
delete from dept;
commit;

ex2) 복구하기

1) dept 테이블이 flashback 될 수 있도록 설정한다

alter table dept enable row movement;

2) dept 테이블을 과거로 되돌린다.

**flashback table dept to timestamp
(systimestamp - interval '5' minute);**

3) 확인

select * from dept;

4) flashback 성공 후 commit; 으로 고정

※ 과거로 되돌리지 못하는 경우

- 1) sys 유저에서 작업했을 때
- 2) delete 하고 commit 한 이후에 DDL 명령어를 수행한 경우

DML 문장 : insert, update, delete, merge

DDL 문장 : create, alter, drop, truncate, rename

※ 과거로 되돌릴 수 있는 골든타임확인

```
show parameter undo_retention
NAME                TYPE        VALUE
-----
undo_retention      integer     900
```

101 실수로 지운 데이터 복구하기 3(FLASHBACK DROP)

테이블을 drop 한 경우에도 복구가능한데 10g 버전 이후부터는 drop 한 테이블이 휴지통에 입력된다.
이 휴지통에 입력된 data 를 끄집어 내면 복구 가능하다

ex1) 휴지통에 넣었다 복구하기

drop table dept; <<< 휴지통에 넣기

show recyclebin; <<< 휴지통에 있는 정보 보기

select *

from user_recyclebin; <<< 휴지통에 있는 정보 보기2

flashback table dept to before drop; <<< flashback 하기

위 경우, 휴지통만 비우지 않으면 다 복구할 수있다.

※ 휴지통 비우기

purge recyclebin; << 비우기

show recyclebin; << 휴지통 확인하면 아래와 같이 뜬다

SP2-0564: ** 객체가 부적합합니다. 설명되지 않을 수 있습니다.

※ 테이블 삭제할 때 휴지통에 넣지 않고 삭제하는 방법

drop table emp500 purge;

102 실수로 지운 데이터 복구하기 4(FLASHBACK VERSION QUERY)

version query : 특정 테이블이 과거로부터 현재까지 어떻게 변경이 되어왔는지 그 이력정보를 확인할때 사용

ex) 실습

- 1) 현재 시간을 확인 후 text 로 따내기

select systimestamp from dual;

SYSTIMESTAMP
1 20/11/11 15:50:16.571000000 +09:00

20/11/11 15:50:16.571000000 +09:00

- 2) KING의 이름과 월급과 부서번호를 조회

ENAME	SAL	DEPTNO
1 KING	5000	10

- 3) KING의 월급을 8000으로 변경하고 commit; 한다

update emp

set sal = 8000

```

where ename = 'KING';
commit;
4) KING의 부서번호를 20번으로 변경하고 commit; 한다
update emp
set deptno = 20

```

```

where ename = 'KING';
commit;

```

5) 그동안 KING의 데이터가 어떻게 변경되어 왔는지 그 이력정보를 확인하시오!

```

select ename, sal, deptno, versions_starttime, versions_endtime,
       versions_operation
from emp
versions between timestamp
       to_timestamp('20/11/11 15:50:16', 'RR/MM/DD HH24:MI:SS')
       and maxvalue
where ename = 'KING'
order by versions_starttime;

```

ENAME	SAL	DEPTNO	VERSIONS_STARTTIME	VERSIONS_ENDTIME	VERSIONS_OPERATION
1 KING	8000	10	20/11/11 15:53:08.000000000	20/11/11 15:54:06.000000000	U
2 KING	8000	20	20/11/11 15:54:06.000000000	(null)	U
3 KING	5000	10	(null)	20/11/11 15:53:08.000000000	(null)

위 파란색 시간은 1)의 시간이다

6) 위 시간중에 하나를 확인해서 그 시간대에 emp 테이블의 상태를 확인하시오!
(5번째 컬럼인 versions_endtime 으로 상태확인)

```

select *
from emp as of timestamp
to_timestamp('20/11/11 15:54:06', 'RR/MM/DD HH24:MI:SS');

```

7) 20/11/11 15:54:06 이 시간으로 emp 테이블을 복구하시오!

```

alter table emp enable row movement;

```

```

flashback table emp to timestamp

```

```

to_timestamp('20/11/11 15:54:06', 'RR/MM/DD HH24:MI:SS');

```

1행에서 table 의 성질을 바꾸고 2-3행에서 flashback 을 하면 언급된 시간대로 flashback 할 수 있다.

103 실수로 지운 데이터 복구하기 5(FLASHBACK TRANSACTION QUERY)

위 과정은 실습이 잘 안되고 dba 영역에 가까워 따로 실습X

flashback transaction query 는 그동안 작업했던 DML 문장을 확인해 볼 수 있다.

예를 들면 어떤 update, delete, insert 문을 수행했는지 그 문장들을 반대로 수행하는 DML 문장이 출력된다.

104 데이터의 품질 높이기 1(PRIMARY KEY)

데이터 분석을 하다 보면 가장 많은 시간을 할애하는 작업이 **데이터 전처리** 이다. 처음부터 품질이 높은 데이터를 입력 받게끔 강제화하면 데이터 전처리에 소요되는 시간이 줄어든다. 그래서 데이터 품질을 높이기 위한 방법으로 **제약**을 사용한다. 즉, 처음부터 테이블에 데이터를 입력받을 때부터 엄격한 기준으로 제약을 걸어 활용한다.

※제약의 종류

- 1) **primary key** : 중복된 데이터와 null 값을 허용하지 않게 하는 제약
- 2) **unique** : 중복된 데이터를 허용하지 않게 하는 제약
- 3) **not null** : null 값을 허용X
- 4) **check** : 특정 데이터 외에 다른 데이터는 입력불가
- 5) **foreign key** : 참조하는 컬럼에 거는 제약

※ primary key 제약을 생성해서 테이블 생성하기

```
create table emp307
( empno number(10) primary key,
  ename varchar2(20) );
```

위와 같이 테이블을 생성하면 empno 에는 앞으로 중복된 데이터와 null 값이 입력되지 않는다.

```
insert into emp307 values( 1111, 'scott');
insert into emp307 values( 2222, 'smith');
insert into emp307 values( 1111, 'allen'); <<< 중복된 데이터 입력불가 (primary key)
```

명령의 7 행에서 시작하는 중 오류 발생 -

```
insert into emp307 values( 1111, 'allen')
```

오류 보고 -

ORA-00001: 무결성 제약 조건 (SCOTT.SYS_C007548)에 위배됩니다

위 insert SQL 중 3번째 SQL을 실행하면 위와 같은 err 발생(primary key 에 걸림)

```
insert into emp307 values( null, 'jones'); <<< null 값 입력 불가 (primary key)
```

2류 보고 -

ORA-01400: NULL을 ('SCOTT'. 'EMP307'. 'EMPNO') 안에 삽입할 수 없습니다

위 SQL도 마찬가지로 null 값 때문에 해당 data 가 emp307 에 insert 되지 않는 것을 볼 수 있다.

☆403) ##14) 스마일게이트, 엑셈 면접 알고리즘문제

factorial 을 SQL로 구현하시오!

undefine ft;

```
select &ft || ' 팩토리얼은 ' || round(exp( sum( ln(level) ) ) ) || ' 입니다 ' as factorial
```

```
from dual
```

```
connect by (level) <= &ft;
```

FACTORIAL	
1	12 팩토리얼은 479001600 입니다

&ft 로 실행시마다 값을 변경할 수 있다. 단, 변경된 값을 적용하려면 undefine ft; 까지 같이 실행해주어야 한다

※ 위 문제를 풀기 위한 이전 알고리즘 : 1부터 10까지의 곱

```
select exp( sum( ln(level) ) )
```

```
from dual
```

```
connect by (level) <= 10;
```

사실 ##14)보다는 log 의 성질을 이용해서 1부터 N까지의 곱을 ln 과 exp 함수로 유도할 수 있는것이 중요하다
만약 팩토리얼이 아닌 N 행까지 존재하는 짝수의 곱을 물어본다면 어떻게 풀 것인가?

undefine ft;

```
select &&ft as try, round(exp( sum( ln(level) ) ) ) as factorial
from dual
where mod(level, 2) = 0
connect by (level) <= &ft;
```

where 절에 column을 level 로 주어서 조건절을 주면 구할 수 있다. 이때 select 절의 별칭인 factorial 을 주지 않는 이유는 실행순서 때문인데, from -- **where** -- group by -- having -- order by -- **select** -- 순이기 때문이다.

##1112

2020년 11월 12일 목요일 오전 9:33

1. SQL 쪽지시험, SQL 인터넷 시험
2. 파이썬 알고리즘

시작

※제약의 종류

- 1) **primary key** : 중복된 데이터와 null 값을 허용하지 않게 하는 제약
- 2) **unique** : 중복된 데이터를 허용하지 않게 하는 제약
- 3) **not null** : null 값을 허용X
- 4) **check** : 특정 데이터 외에 다른 데이터는 입력불가
- 5) **foreign key** : 참조하는 컬럼에 거는 제약

105 데이터의 품질 높이기 2(UNIQUE)

unique : 중복된 데이터를 허용하지 않게 하는 제약

데이터 품질을 높이기 위해서 중복된 데이터가 테이블에 입력되지 못하도록 역할수행

ex1) unique 제약 걸기

```
create table emp507
( empno number(10),
  enmae varchar2(10) unique );
insert into emp507 values(1111, 'scott');
insert into emp507 values(2222, 'scott');
위 2번째 insert 는 unique 제약조건으로 insert 불가
```

※ 제약삭제

- 1) 삭제할 제약 이름 확인

```
select table_name, constraint_name <<<<< 제약조건검색
from user_constraints
where table_name='EMP507'; <<<<< emp507은 대문자로 입력해야 검색된다
```

TABLE_NAME	CONSTRAINT_NAME
EMP507	SYS_C007471

- 2) emp507 의 unique 제약을 삭제

```
alter table emp507
drop constraint SYS_C007471;
```

- 3) 중복된 데이터가 입력이 잘 되는지 확인

```
insert into emp507 values(2222, 'scott');
```

위와 같이 제약의 이름이 SYS_C00741 이면 이름만 보서는 이 제약이 어떤 제약인지 확인이 어렵기 때문에 처음 제약 생성시 이름을 의미있게 부여하는 것이 관리에 편함

- 4) 제약이름을 주고 테이블을 생성하는 방법

```
create table emp508
( empno number(10),
  ename varchar2(10) constraint emp508_ename_un unique);
```

[테이블명_컬럼명_제약이름축약] 으로 이름설정

5) emp508 enmae 에 걸린 unique 제약 확인

```
select table_name, constraint_name
from user_constraints
where table_name = 'EMP508';
```

TABLE_NAME	CONSTRAINT_NAME
1 EMP508	EMP508_ENAME_UN

404) emp508 테이블의 ename 에 걸린 unique 제약을 삭제하시오!

>>1 제약조건 검색

```
select table_name, constraint_name
from user_constraints
where table_name = 'EMP508';
```

>>2 제약조건 삭제(table을 alter해주어야함)

```
alter table emp508
```

```
drop constraint emp508_ename_un;
```

※ 제약을 생성하는 방법

1) 테이블 생성할때 제약 생성

2) 이미 만들어 놓은 테이블에 제약을 추가

ex) 실습

1. emp 테이블에 ename 에 unique 제약을 거시오!

```
alter table emp
```

```
add constraint emp_ename_un unique(ename);
```

추가 제약이름 제약의종류(column)

기존에 테이블에 중복된 데이터가 있는 경우 위 SQL은 실행불가

Q : DATABASE 에서 제약을 사용하는 이유가 무엇입니까?

A : 데이터의 품질을 높이기 위해서

405) 우리반 테이블에 이름에 unique 제약을 거시오!

```
alter table emp12
```

```
add constraint emp12_ename_un unique(ename);
```

TABLE_NAME	CONSTRAINT_NAME
1 EMP12	EMP12_ENAME_UN

이후 제약이 걸렸는지 검색하여 확인한다

106 데이터의 품질 높이기 3(NOT NULL)

not null : null 값을 입력하지 못하도록 막는 제약

제약생성

1) 테이블 생성시 제약

```
create table emp707
```

```
( empno    number(10) constraint emp707_empno_nn not null,
  ename    varchar2(20) );
```

```
insert into emp707 values(1111, 'smith');
```

```
insert into emp707 values(null, 'scott');
```

두번째 insert 불가

2) 만들어진 테이블에 제약

```
alter table emp
```

```
    modify empno constraint emp_empno_nn not null;
```

unique 제약과 조금 형태가 다름(아래는 unique 제약 조건)

```
alter table emp
```

```
    add constraint emp_ename_un unique(ename);
```

406) 사원 테이블에 월급에 not null 제약을 거시오!

```
alter table emp
```

```
    modify sal constraint emp_sal_nn not null;
```

407) 사원 테이블의 월급에 걸린 not null 제약을 삭제하시오!

```
alter table emp
```

```
drop constraint EMP_sal_nn;
```

107 데이터의 품질 높이기 4(CHECK)

check : 지정된 데이터만 입력되고 다른 데이터는 입력되지 못하게 막는 제약

예를 들면 성별(남/여) 이 있다

제약걸기

1) 테이블 생성시 제약

```
create table emp745
```

```
( ename  varchar2(10),
```

```
   loc    varchar2(20) constraint emp745_loc_ck
```

```
    check(loc in ('DALLAS', 'CHICAGO', 'BOSTON')) );
```

제약을 check 로 주면 체크 내 조건에 해당하는 데이터만 입력된다. 대소문자는 구분됨

만약 조건이 하나이면 연결연산자 = 을 사용해도 된다

2) 만들어진 테이블에 제약

```
alter table dept
```

```
    add constraint dept_loc_ck
```

```
    check(loc in ('NEW YORK', 'DALLAS', 'CHICAGO', 'BOSTON')) );
```

위 조건의 데이터만 입력가능하도록 check 하겠다.

408) 사원 테이블의 월급에 월급이 0~9500 사이의 데이터만 입력되도록 check 제약을 거시오!

```
alter table emp
```

```
    add constraint emp_sal_ck
```

```
    check( sal between 0 and 9500);
```

409) 우리반 테이블에 성별 컬럼의 남 아니면 여 만 입력되도록 check 제약을 거시오!

```
alter table emp12
```

```
    add constraint emp12_gender_ck
```

```
    check( gender in ('남', '여')) ;
```

108 데이터의 품질 높이기 5(FOREIGN KEY)

foreign key : 참조하는 컬럼에 거는 제약

EMPNO	ENAME	SAL	DEPTNO	DEPTNO	DNAME
1	KING	5000	10	1	10 ACCOUNTING
2	BLAKE	2850	30	2	20 RESEARCH
3	CLARK	2450	10	3	30 SALES
4	JONES	2975	20	4	40 OPERATIONS
5	MARTIN	1250	30		
6	ALLEN	1600	30		
7	TURNER	1500	30		
8	JAMES	950	30		
9	WARD	1250	30		
10	FORD	3000	20		
11	SMITH	800	20		
12	SCOTT	3000	20		
13	ADAMS	1100	20		
14	MILLER	1300	10		

emp table	dept table
deptno에 foreign key	deptno에 primary key
자식 키	부모 키

dept 테이블에 primary key 제약을 걸고 emp 테이블에 foreign key 제약을 걸면서

emp deptno 가 dept deptno 를 참조하겠다고 하면 그때부터 emp deptno 에 deptno 를 insert 할 때 dept 테이블에 있는 deptno 인 10, 20, 30, 40 번 데이터만 입력할 수 있다. 또한 dept 테이블의 deptno 의 데이터를 지우려고 하면 emp 테이블의 deptno 가 참조하고 있으므로 삭제가 되지 않는다.

위와 같이 복잡한 과정을 거쳐서 foreign key 설정을 하는 이유

- 1) 데이터 품질을 높이기 위해서
- 2) 조인할 때 outer join 을 남발하지 않기 위해서 (outer join 은 무거운 SQL)

※ 실습 (수행 전 @demo.sql 수행)

- 1) dept 테이블의 deptno 에 primary key 제약을 건다

alter table dept

add constraint dept_deptno_pk primary key(deptno);

- 2) emp 테이블의 deptno 에 foreign key 제약을 걸면서 dept 테이블의 deptno를 참조

alter table emp

add constraint emp_deptno_fk foreign key(deptno)

references dept(deptno);

- 3) err 확인

insert into emp(empno, ename, sal, deptno)

values (1234, 'jack', 4500, 70);

insert into emp(empno, ename, sal, deptno)

values (1234, 'jack', 4500, 70)

오류 보고 -

ORA-02291: 무결성 제약조건(SCOTT.EMP_DEPTNO_FK)이 위배되었습니다- 부모 키가 없습니다

위 insert 의 deptno가 70 으로 1),2) 에서 건 pk, fk 에 위배되는 것을 볼 수 있다

109 WITH절 사용하기 1(WITH ~ AS)

with~as 절

- 1) 복잡한 쿼리내에 동일한 쿼리가 두번 이상 발생한 경우에 좋은 성능을 보이는 SQL
- 2) 테스트 테이블과 같은 임시로 사용되는 데이터를 갖고 테스트 할 때 유용한 SQL

ex1) 1부터 10까지의 숫자를 출력하는 SQL을 작성하시오!

```
select level as num1
```

```
from dual
```

```
connect by level <=10;
```

위 결과를 테이블로 만들고 싶을 때 dba가 승인을 안해준다면 with절을 이용하여 내가 사용하는 SQL 내에서만 임시로 테이블을 만들수 있다

ex2) 위 SQL 을 with 절로 변경하시오!

```
with test_table as ( select level as num1
                        from dual
                        connect by level <=10 )
```

```
select num1
```

```
from test_table;
```

as 다음에 나오는 괄호안의 쿼리문의 결과 데이터로 임시 테이블을 생성하는데 그 테이블 이름이 test_table 이 된다. 그리고 그 아래의 쿼리에서 test table 을 활용하여 쿼리문장을 만든다

이 **임시테이블은 with절이 끝나면 없어진다.**

410) 위 결과에서 짝수만 출력하시오!

```
with test_table as
```

```
( select level as num1
```

```
from dual
```

```
connect by level <=10)
```

```
select num1
```

```
from test_table
```

```
where mod(num1, 2) = 0;
```

☆411) with 절로 구구단 2단을 출력하세요!

```
with test_1 as
```

```
(select level as num1
```

```
from dual
```

```
connect by level <=9)
```

```
select 2 || ' X ' || num1 || ' = ' || 2*num1 as "2단"
```

```
from test_1;
```

>>>치환변수를 이용한 구구단 (단수별 새로고침가능)

```
undefine flip
```

```
with test_1 as
```

```
(select level as num1
```

```
from dual
```

```
connect by level <=9)
```

```
select &&flip || ' X ' || num1 || ' = ' || &flip*num1 as "&flip"
```

```
from test_1;
```

오라클은 크게 메모리가 디스크 두 가지로 나뉜다.

memory(cash)	disk(HDD)
sub.q, with..as~	emp, dept, emp12

with tem.table as **sub query** 에서 query 의 결과가 disk에 저장이 되고 그 결과는 테이블로 tem.table 의 이름으로 임시 저장된다(memory)

412) with 절로 구구단 전체를 출력하시오! 이전 알고리즘 문제의 SQL 을 with절로 변경

```
with temp_t1 as ( select level as num1
                  from dual
                  connect by level <= 9),
temp_t2 as ( select level as num2
              from dual
              connect by level <= 9)
select t1.num1 || ' X ' || t2.num2 || ' = ' || (t1.num1)*t2.num2
      from temp_t1 t1, temp_t2 t2
      where t1.num1 between 2 and 9;
in line view 에 비해 속도가 훨씬 빠르다.
```

413) 아래의 인라인뷰 SQL 을 with 절로 변경하시오!

```
select e.ename, e.sal, e.deptno, v.부서평균
      from emp e, (select deptno, avg(sal) 부서평균
                  from emp
                  group by deptno) v
      where e.deptno = v.deptno and e.sal > v.부서평균;
>>with 절 사용 (from절의 서브쿼리를 with절로 변경)
with t_1 as (select deptno, avg(sal) 부서평균
              from emp
              group by deptno)
select e.ename, e.sal, e.deptno, v.부서평균
      from emp e, t_1 v
      where e.deptno = v.deptno and e.sal > v.부서평균;
```

※ with절의 의미

- 1) as다음에 나오는 쿼리문의 결과가 아주 많을때
- 2) inline-view에서 여러번 작성될 경우

414) ##15) with절을 이용해서 직각 삼각형을 출력하시오!

```
☆
☆☆
☆☆☆
☆☆☆☆
☆☆☆☆☆
...
with temp_table as (select level as num1
                    from dual
                    connect by level <= 9 )
select lpad('☆', num1, '☆')
```



```

from temp_table;
>>2 with 절 없이도 가능
select lpad( '★',level, '★')
  from dual
 connect by level <= 10;

```

415) 이번에는 숫자를 물어보게 하고 숫자를 입력하면 해당 숫자만큼 직각삼각형이 만들어지게 하시오!

```

with temp_table as (select level as num1
  from dual
 connect by level <= &f )

```

```

select lpad('☆', num1, '☆')

```

```

  from temp_table;

```

확장연산자 &는 하나일때는 계속 초기값을 물어보고, && 있으면 값이 저장되어
undefine f 를 추가해 주어야 초기화된다.

416) 이번에는 삼각형이 출력되게 하시오! (숫자입력값도 매번 줄 수 있게)

```

undefine p

```

```

with temp_table as (select level as num1
  from dual
 connect by level <= &&p )

```

```

select lpad(' ',&p-num1,' ') || lpad('☆', num1, '☆') as A
  from temp_table;

```

>>>2 이것도 with 절 없이 가능하다

```

undefine p

```

```

select lpad(' ',&&p-level,' ') || lpad( '★',level, '★') as triangle
  from dual
 connect by level <= &p;

```

with 절의 장점 : 똑같은 SQL을 하나의 SQL 내에서 여러개 사용될때 유용하다.

417) 직업, 직업별 토탈월급을 출력하시오!

JOB	SUM(SAL)
1 SALESMAN	5600
2 CLERK	4150
3 ANALYST	6000
4 MANAGER	8275
5 PRESIDENT	5000

418) 위 직업별 토탈월급들 중에서 평균값을 출력하시오!

```

select avg(sum(sal))
  from emp
 group by job;

```

419) 417쿼리문과 418쿼리문을 둘다 이용해서 직업, 직업별 토탈월급을 출력하는데 직업별 토탈월급이 직업별 토탈월급들의 평균값보다 더 큰 것만 출력하시오!

```

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

```

```

from emp
group by job);

```

위 문제와 같은 상황에서 with 절이 유용하다 << group 함수 조건시(좌변에) having절 사용

420) 위 SQL을 with절로 바꾸시오!

```

with job_sumsal as (select job, sum(sal) as 토탈
from emp
group by job)

```

```

select job, 토탈
from job_sumsal
where 토탈 > (select avg(토탈)
from job_sumsal);

```

복잡한 쿼리내에 동일 쿼리블럭이 두 번 이상 발생하는 경우에 사용하면 좋은 성능을 보이는 SQL이 with절이다. 420 SQL 의 경우 with 절을 한번 수행하면 memory 에 저장되어 활용가능

421) 위 SQL의 실행계획을 확인해서 HDD 에 TEMP 테이블로 만들어지는지 확인하시오!

explain plan for ~~ select * from table(dbms_xplan.display);

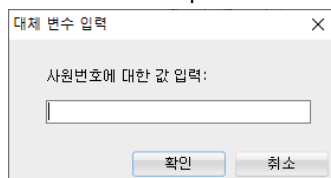
PLAN_TABLE_OUTPUT		
1 Plan hash value: 1646589203		
2		
3		
Id	Operation	
0	SELECT STATEMENT	
1	TEMP TABLE TRANSFORMATION	
2	LOAD AS SELECT (CURSOR DURATION MEMORY)	
3	HASH GROUP BY	
4	TABLE ACCESS FULL	
* 5	VIEW	
6	TABLE ACCESS FULL	
7	SORT AGGREGATE	
8	VIEW	
9	TABLE ACCESS FULL	

※ accept 절 사용 방법

```

select empno, ename, sal
from emp
where empno = &사원번호;

```



이 창의 [사원번호에 대한 값 입력:] 대신 다른 메시지를 나오게 하고 싶을 때 accept 활용

accept p_num1 prompt '사원번호를 입력하세요~'

```

select empno, ename, sal
from emp
where empno = &p_num1;

```

422) 415를 다시 푸는데, 아래와 같이 입력메세지 창에서 나오는 메세지가 숫자를~ 로 나오게하시오!

accept f prompt '숫자를 입력하세요~'

```

with temp_table as (select level as num1
from dual
connect by level <= &f )
select lpad('☆', num1, '☆')

```

```
from temp_table;
```

423) ##17) 아래와 같이 숫자를 물어보게 하고 숫자를 입력하면 마름모가 출력되게 하시오!

```
undefine p
accept p prompt '숫자를 입력하세요~'
with temp_table as (select level as num1
                    from dual
                    connect by level <= &p )
select lpad(' ',&p-num1,' ') || lpad('☆', num1, '☆') as A
from temp_table
union all
select lpad(' ',num1,' ') || lpad('☆', &p-num1, '☆') as A
from temp_table;
```

★@□ 문자들에 따라 출력모양이 달라진다

110 WITH절 사용하기 2(SUBQUERY FACTORING)

with 절을 사용할 때 with 절 내에서 같은 SQL이 아래와 같이 두 번 이상 사용하게 되면

TEMP table 에 자동으로 저장되고 한번만 사용하면 with 절의 as 다음에 나오는 쿼리문의 결과가 메모리에 저장된다.

with절을 사용할 때 쓰는 hint

- 1) /*+ materialize */ : temp table을 HDD 에 생성해라
- 2) /*+ inline */ : temp table 을 만들지 말고 in line view 를 사용해라

>>예시

explain plan for

```
with job_sumsal as (select /*+ materialize */ job, sum(sal) as 토탈
                    from emp
                    group by job)
```

select job, 토탈

from job_sumsal

where 토탈 > (select avg(토탈)

from job_sumsal);

select * from table(dbms_xplan.display);

3	0	SELECT STATEMENT	
7	1	TEMP TABLE TRANSFORMATION	
3	2	LOAD AS SELECT (CURSOR DURATION MEMORY)	SY:
3	3	HASH GROUP BY	
3	4	TABLE ACCESS FULL	EMI
1	* 5	VIEW	
2	6	TABLE ACCESS FULL	SY:
3	7	SORT AGGREGATE	
4	8	VIEW	
5	9	TABLE ACCESS FULL	SY:

114 SQL로 알고리즘 문제 풀기 4(사각형 출력)

ex1) ★을 아래와 같이 5개를 가로로 출력하시오!

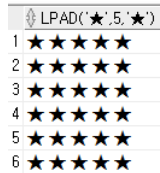
```
select lpad('★', 5, '★')
```

```
from dual;
```

```
LPAD('★',5,'★')
1 ★★★★★
```

ex2) 위의 별 다섯개가 아래와 같이 6개가 나오게 하시오!

```
select lpad('★', 5, '★')
      from dual
      connect by level <= 6;
```



```
LPAD('★',5,'★')
1 ★★★★★
2 ★★★★★
3 ★★★★★
4 ★★★★★
5 ★★★★★
6 ★★★★★
```

424) ##18) 아래와 같이 두 숫자를 각각 입력받아 사각형을 출력하시오!

가로의 숫자를 입력하세요~ : 5

세로의 숫자를 입력하세요~ : 4

★★★★★

★★★★★

★★★★★

★★★★★

accept p1 prompt '가로의 숫자를 입력하세요~'

accept p2 prompt '세로의 숫자를 입력하세요~'

```
select lpad('★', &p1, '★')
      from dual
      connect by level <= &p2;
```

121 SQL로 알고리즘 문제 풀기 11(최대 공약수)

최대공약수가 필요한 이유 : 약분을 빨리 하기 위해서

ex1) 16명의 인부들에게 24개의 빵을 공평하게 나눠주려면 1명당 몇개씩 주면 되는가?

최대공약수 8 16 24

(2) (3) >>>> 2명에게 3개의 빵을 나누어 주는것과 같다

즉 1명당 1.5개의 빵을 나누어 주는것이 공평하다

☆425) ##19) SQL로 최대공약수를 구하시오!

첫번째 숫자를 입력하세요~ 16

두번째 숫자를 입력하세요~ 24

최대공약수는 8입니다. 이렇게 결과가 나오도록 하시오!

accept p1 prompt '첫번째 숫자를 입력하세요~'

accept p2 prompt '두번째 숫자를 입력하세요~'

```
with c_1 as (select level as c
              from dual
              connect by level <= &p1+&p2)
select '최대 공약수는 ' || max(c) || ' 입니다.' as 출력값
      from c_1
      where mod(&p1, c) = 0 and mod(&p2, c) = 0;
```

>> 튜닝가능(with절이 필요없음)

accept p1 prompt '큰 숫자를 입력하세요~'

```
accept p2 prompt '두번째로 큰 숫자를 입력하세요~'  
select '최대 공약수는 ' || max(level) || ' 입니다.' as "p1>p2"  
  from dual  
 where mod(&p1, level) = 0 and mod(&p2, level) = 0  
 connect by level <= &p1;
```

##1113

2020년 11월 13일 금요일 오전 9:42

복습

1. 기본 SQL 문장
2. 함수
3. 조인문
 - 1) oracle join : equi, non equi, outer, self
 - 2) 1999ansi join : on, using, natural, left/right/full outer, cross
4. 집합연산자
5. 서브쿼리문
6. DML문
7. DDL 문
8. TCL 문
9. 계층형 질의문
10. with 절
11. 제약

■ 3개 이상의 테이블 조인

- 1) 2개의 테이블 조인

emp-----dept
(연결고리)

- 2) 3개의 테이블 조인

dept-----emp-----salgrade
(연결고리) (연결고리)
where절 where절
e.deptno=d.deptno and e.sal between s.losal and s.hisal

426) 이름과 부서위치와 월급과 부서번호를 출력하시오! (emp join dept)

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

427) emp, dept, salgrade 테이블을 조인해서 이름, 월급, 부서위치, 부여등급(grade) 을 출력하시오!

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

테이블이 3개이면 2개의 연결고리를 where 절에 기술해줘야 한다.

428) 아래와 같이 bonus 라는 테이블을 생성하시오!

```
create table bonus
as
```

```
select empno, sal*1.5 as comm2
from emp;
```

429) 사원이름, 월급, 부서위치, comm2 를 출력하시오!

(dept---emp---bonus)

```
select e.ename, e.sal, d.loc, b.comm2
from emp e, dept d, bonus b
where e.deptno=d.deptno and e.empno=b.empno;
```

430) 위 결과에서 월급이 2000 이상인 사원들만 출력하시오!

```
select e.ename, e.sal, d.loc, b.comm2
from emp e, dept d, bonus b
where e.deptno=d.deptno and e.empno=b.empno and e.sal>=2000;
```

123 SQL로 알고리즘 문제 풀기 13(피타고라스의 정리)

피타고라스의 정리 : $(a^2+b^2)=c^2$ ($a \leq b < c$)

##

밑변을 입력하세요~3

높이를 입력하세요~4

빗변을 입력하세요~5

직각삼각형이 맞습니다. 틀리면 직각삼각형이 아닙니다 라고 나와야 함

```
undefin p1 undefin p2 undefin p3
```

```
accept p1 prompt '밑변을 입력하세요~'
```

```
accept p2 prompt '높이를 입력하세요~'
```

```
accept p3 prompt '빗변을 입력하세요~'
```

```
select decode( (&p1*&p1)+(&p2*&p2), (&p3*&p3), '직각삼각형이 맞습니다', '직각삼각형이 아닙니다') as
pitagoras
```

```
from dual;
```

>>2 power함수 활용

```
accept p1 prompt '밑변을 입력하세요~'
```

```
accept p2 prompt '높이를 입력하세요~'
```

```
accept p3 prompt '빗변을 입력하세요~'
```

```
select decode( power(&p1,2)+power(&p2,2), power(&p3,2), '직각삼각형이 맞습니다', '직각삼각형이 아닙니다') as
pitagoras
```

```
from dual;
```

&가 하나씩 들어가서 undefine 해제 가능

※ 함수

1) 단일행 함수 : 문자, 숫자, 날짜, 변환, 일반

숫자함수 : round, trunc, mod, **power**

```
power : select power(2,3)
from dual;
```

2) 복수행 함수

■외부 테이블(external table)

오라클의 테이블의 종류

- 1) 일반 테이블(heap table)
- 2) 임시 테이블(temp table)
- 3) 외부 테이블(external table)
- 4) 파티션 테이블(partition table)

※면접질문 : Q : 병렬처리로 프로그래밍 해본 적이 있나요?? >>> partition table 과 연관이 높다
A : SQL로는 파티션 테이블 만들어 병렬 쿼리를 이용해 병렬처리를 해보았습니다.

외부테이블 : database 외부에 있는 엑셀 파일이나 csv 파일, text 파일을 insert 문으로 만들어
database 에 입력하지 않고 링크만 걸어서 해당 파일의 데이터를 select 할 수 있는 테이블

ex) 카페에서 emp.text 파일을 받아서 실습하시오

- 1) emp.txt를 c:/data/emp.txt 하시오
- 2) c 드라이브에 data 폴더를 오라클 디렉토리로 생성하시오

create directory emp_dir as 'c:\Wdata';

3) 외부 테이블을 생성하시오! (cafe file 참조)

```
create table ext_emp
( EMPNO   NUMBER(10),
  ENAME   VARCHAR2(20),
  JOB     VARCHAR2(20),
  MGR     NUMBER(10),
  HIREDATE DATE,
  SAL     NUMBER(10),
  COMM    NUMBER(10),
  DEPTNO  NUMBER(10))
```

설정에 대한 설명(위SQL과 이어지는 부분임)

organization external : **external table** 을 생성하겠다

(type oracle_loader : 데이터를 로드하는 엔진을 sql*loader를 사용

default directory emp_dir : **emp_dir** 를 기본 디렉토리로 하겠다

access parameters : **text** 파일의 내용을 테이블에 입력하기 위한 문법을 수행하겠다

(records delimited by newline : 행과 행 사이는 엔터로 구분하겠다

fields terminated by "," : 데이터와 데이터는 콤마로 구분하겠다

empno char, : 외부테이블의 컬럼들의 실제 데이터는 문자

ename char, : 인데 오라클에는 숫자로 입력한다

job char,

mgr char,

hiredate date "yyyy/mm/dd",

sal char

comm char,

deptno char))

location ('emp.txt')); : 링크를 걸 텍스트 파일 이름작성(여기에서는 emp.txt 가 된다)

대용량 엑셀 파일이나 text 파일을 db에 넣으려면 시간이 많이 필요하다

외부 테이블의 장점

- 1) database 에 저장공간이 필요없다
- 2) 대용량 데이터인 경우 테이블에 입력하는 시간을 줄일 수 있다

432) exp_emp11 테이블을 조회하여 직업이 SALESMAN 인 직원들의 이름과 직업과 월급을 출력하시오!

```
select ename, job, sal
from exp_emp11
where job='SALESMAN';
```

433) exp_emp11 과 dept 를 조인해서 이름과 부서위치를 출력하시오!

```
select e.ename, d.loc
from ext_emp11 e, dept d
where e.deptno=d.deptno;
```

조인도 가능하다.

※ external table 의 단점

- 1) DML 작업이 안되고 오직 select 만 가능
- 2) index 생성 불가

434) ext_emp11 의 scott 의 월급을 6000 으로 갱신하시오!

```
update ext_emp11
set sal = 6000
where ename = 'SCOTT';
```

불가능하다 아래 err를 보자

```
SQL 오류: ORA-30657: 외부 구성 테이블에 지원되지 않는 작업
30657.0000 - "operation not supported on external organized table"
*Cause:      User attempted on operation on an external table which is
              not supported.
*Action:     Don't do that!
```

☆435) dept.txt 를 외부테이블로 생성해서 아래와 같이 select 할 수 있도록 하시오!

1. dept.txt 를 카페에서 내려받고 c:\w\data\dept.txt 를 한다
2. ext_emp11 테이블 스크립트를 가져와서 exp_dept 테이블을 만들 수 있도록 수정하시오!

```
create table ext_dept
( DEPTNO  NUMBER(10),
  DNAME   VARCHAR2(20),
  LOC     VARCHAR2(20) )
organization external
(type oracle_loader
 default directory emp2_dir
 access parameters
 (records delimited by newline
  fields terminated by ",")
 ( DEPTNO  char,
  DNAME   char,
  LOC     char) )
location ('dept.txt');
```

실행 시 emp_dir 이 알 수 없는 오류로 인해 emp2_dir 로 생성하였다

```
create directory emp2_dir as 'c:\w\data';
```

3. select 되는지 확인하시오!

잘 된다.

4. 외부 테이블 확인

```
select table_name, external
      from user_tables
      where external = 'Yes';
```

5. drop

```
drop table ext_emp;
```

■ 정규식 함수 5가지

정규식 함수 : 데이터 검색을 좀 더 상세하게 할때 기존의 함수로는 표현할 수 없는 데이터 검색을 가능하게 하는 함수

regular expreesion(정규 표현식) :

- 정규 표현식 코드는 오라클 뿐 아니라 다른 언어에서도 공통적으로 사용하는 표현식 코드이다
- 오라클 10.1 버전부터 정규 표현식 지원
- 오라클 데이터베이스는 정규표현식의 posix 연산자를 지원
- posix : Portable Operation System Interface 의 약자로, 시스템간 호환성을 위해 미리 정의된 인터페이스를 의미

※ **posix 연산자** : 기본연산자, 앵커, 수량사, 서브표현식, 역참조, 문자리스트, posix 문자 클래스 등이 있다

1) 기본 연산자

연산자	영문	설명
.	dot	모든 문자와 일치
	or	대체 문자를 구분
\w	backslach	다음 문자를 일반 문자로 처리

ex) dot(.) 활용하기

```
select regexp_substr('aab', 'a.b') as c1,
       regexp_substr('abb', 'a.b') as c2,
       regexp_substr('acb', 'a.b') as c3,
       regexp_substr('adc', 'a.b') as c4
      from dual;
```

	C1	C2	C3	C4
1	aab	abb	acb	(null)

위 예제에서 .(dot) 은 그 자리에 무엇이 와도 상관없다는 의미이다

원래 substr 은 select substr('abc', 1, 2) from dual; 을 사용해서 ab를 출력하는데 위 ex)의 1행의 의미는 aab 중 a.b를 잘라내라는 것인데 이 때 .(dot) 부분은 어떤 문자가 와도 상관없다는 의미이기 때문에 아래 결과가 출력됨 한편 c4 는 adc 에서 a.b 를 잘라내려고 했기 때문에 null 값이 출력된다.

436) 이름에 EN 또는 IN 을 포함하고 있는 사원들의 이름과 월급을 출력하시오!

```
select ename, sal
      from emp
      where ename like '%EN%' or ename like '%IN%';
```

※ **정규 표현식 함수**

1. **regexp_substr**

- 2. regexp_like
- 3. regexp_count
- 4. regexp_instr
- 5. regexp_replace

437) 위 SQL을 정규 표현식 함수인 regexp_like 를 이용해서 출력하시오!

```
select ename, sal
  from emp
 where regexp_like( ename, 'EN|IN' ); <<<< 붉게 칠해진 부분이 |(or) 이다
```

438) 우리반 테이블에서 전공이 통계, 수학, 컴퓨터, 전자가 포함된 학생들의 이름과 전공을 출력하시오!

>>1 like

```
select ename, major
  from emp12
 where major like '%통계%' or
        major like '%수학%' or
        major like '%컴퓨터%' or
        major like '%전자%';
```

>>2 regexp_like

```
select ename, major
  from emp12
 where regexp_like( major, '통계|수학|컴퓨터|전자');
```

439) employees.csv 파일을 오라클의 데이터로 생성하시오!

>>1 table 생성

```
create table employees
( EMPLOYEE_ID      number(10),
  FIRST_NAME       varchar2(20),
  LAST_NAME        varchar2(20),
  EMAIL            varchar2(50),
  PHONE_NUMBER     varchar2(30),
  HIRE_DATE        date,
  JOB_ID           varchar2(20),
  SALARY           number(10,2),
  COMMISSION_PCT   number(10),
  MANAGER_ID       number(10),
  DEPARTMENT_ID    number(10) );
```

테이블 생성시 데이터 양식을 잘 맞춰주어야 import 가능하다

>>2 employees.csv 파일을 sqldeveloper 를 이용해서 employees 테이블에 로드한다.

좌측 메뉴바에서 employees 테이블 우클릭 후 data import 실행 후 employees.csv 파일을 경로지정해주고 다음~완료

440. 이름의 첫글자가 St로 시작하면서 끝글자가 en으로 끝나는 직원들의 first_name을 출력하시오!

>>1

```
select first_name
  from employees
 where substr(first_name, 1, 2)= 'St' and first_name like '%en';
```

>>2 regexp_like 사용

```
where regexp_like(first_name, '^St(.)+en$');
```

regex_like 는 자동으로 %%를 양쪽 문자열에 둘러주기 때문에 **^ 와 \$ 로 그걸 막는 개념**이라고 생각하면 편하다
즉, 시작이 St 로 시작하면서 끝이 en 으로 끝나는 first_name 을 찾는다

양괄호() 는 . 과 + 를 연결하기 위해 사용해 준 것이다. 이것은 St 와 en 사이에 여러개의 철자가 가능한 것을 의미한다

```
where regexp_like( ename, '^ (김|허)(.)* 민$');
```

- 서브표현식은 표현식을 소괄호로 묶은 표현식
- 서브표현식은 하나의 단위로 처리됨

```
select regexp_substr( 'ababc', '(ab)+c' ) as c1,
       regexp_substr( 'ababc', 'ab+c' ) as c2,
       regexp_substr( 'abd', 'a(b|c)d' ) as c3,
       regexp_substr( 'abd', 'ab|cd' ) as c4
from dual;
```

괄호가 없을 때 + 는 바로 전 한개의 문자를 하나의 packet 으로 생각한다.

겨울왕국에서 elsa 가 몇 번 나오나 등을 알 수 있다

SQL 페이지 164

FROM dual;

위 SQL 에서 gtc 라는 단어가 몇 번 나오는지 찾는 함수이다.

형식은 **regexp_count(script , 'gtc')** 의 형식이다.

442) 겨울왕국(winer_kingdom) 에는 elsa 가 몇 번 나오는지 출력하시오!

```
select sum(elsa_count)
from( select win_text, regexp_count(lower(win_text), 'elsa' ) as elsa_count
      from winter_kingdom);
```

1109 page에서 더 inline view 사용X하고 출력가능하다 (참고!)

```
select sum(regexp_count(lower(win_text), 'elsa' ) ) as elsa_count
      from winter_kingdom;
```

어차피 sum 해줄거 굳이 inline view 사용할필요가 없음

■ regexp_replace

replae >>>> regexp_replace

ex) 이름과 월급을 출력하는데 월급을 출력할 때 replace 함수를 이용해서 숫자 0을 *로 출력하시오!

```
select ename, replace( sal, 0, '*')
      from emp;
```

ex2) 이름과 월급을 출력하는데 월급 숫자 0~3 까지를 * 로 출력하시오!

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

위 SQL 에서 [0-3] 은 0에서 3까지 라는 의미이다

443) 우리반 테이블에서 이름과 나이를 출력하는데 나이를 출력할 때, 아래와 같이 앞의 숫자를 *로 출력되게 하시오!

```
select ename, replace( substr(age,1,1), substr(age,1,1), '*') || substr(age,2,1) as s_age
      from emp12;
```

■ 다중 insert 문

단일 insert

```
insert into emp(empno, enmae, sal) values( 134, 'Scott', 30090);
```

다중 insert 를 사용하면 여러개의 테이블에 동시에 같은 데이터를 여러개 입력 가능하다

다중 insert 문의 종류

- 1) 무조건 all insert 문
- 2) 조건부 all insert 문
- 3) 조건부 first insert 문
- 4) pivoting insert 문

※ 무조건 all insert 문

all insert : 여러개의 테이블에 조건 없이 한번에 데이터를 입력하는 것

emp	→	target_a
	→	target_b

	→	target_c
--	---	----------

ex1) target_n 테이블 생성

create table target_a as select * from emp where 1 =2;	create table target_b as select * from emp where 1 =2;	create table target_c as select * from emp where 1 =2;
--	--	--

where 절에 1!=2 이므로 emp 테이블을 가져와서 target_c 테이블을 생성시 data는 못가져오고 구조만 가져와서 만든다

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
-------	-------	-----	-----	----------	-----	------	--------

이때 where 절을 주지 않거나 참이면 emp table 의 data 를 다 가져온다

ex1-2) data 동시에 입력하기

```
insert all into target_a
           into target_b
           into target_c
```

```
select * from emp;
```

444) 우리반 테이블(emp12) 를 emp12_a, emp12_b, emp12_c 로 구조만 가져와서 만들고 무조건 all insert 문으로 emp12 테이블의 전체 data를 세 개의 테이블에 동시에 입력하시오!

```
create table emp12_a as select * from emp12 where 1 = 2;
create table emp12_b as select * from emp12 where 1 = 2;
create table emp12_c as select * from emp12 where 1 = 2;
```

```
insert all into emp12_a
           into emp12_b
           into emp12_c
select * from emp12;
```

3	into emp12_c
4	select *
5	from emp12;

※ 조건부 all insert 문

조건에 맞는 데이터만 입력되게 조건을 주는 입력문

ex) target_a 테이블에는 comm 을 받는 직원들만 입력하고 target_b 테이블에는 comm 을 받지 않는 직원들만 입력!

```
insert all
  when comm is not null then
    into target_a(empno, ename, sal, comm)
  when comm is null then
    into target_b( empno, ename, sal, comm)
select empno, ename, sal, comm
from emp;
```

445) emp12 테이블의 데이터를 아래의 3개의 테이블에 입력하는데 통신사가 sk 면 emp12_sk 에 입력하고 lg면 emp12_lg에 입력하고 kt 면 emp12_kt 에 입력하시오! table 을 생성하고 조건부 all insert 로 data를 입력하시오!

```
create table emp12_sk as select * from emp12 where 1=2;
create table emp12_lg as select * from emp12 where 1=2;
create table emp12_kt as select * from emp12 where 1=2;
insert all
```

```

when telecom = 'sk' then into emp12_sk
when telecom = 'lg' then into emp12_lg
when telecom = 'kt' then into emp12_kt
select * from emp12;

```

※ 조건부 first insert 문

조건에 맞는 데이터가 첫번째 테이블에 입력되고 남은 나머지 데이터를 갖고 새로운 조건에 맞춰서 두번째 또는 세번째에 입력하는 insert 문

ex) 부서번호가 20번인 직원들은 target_a 에 입력하고 남은 나머지 부서번호인 직원들의 월급중에서 월급이 1200인 이상은 target_b 에 입력하고 1200보다 작은 직원은 target_c 에 입력하시오! (시작 전 truncate로 data 비우고 하기)

```

insert first
  when deptno = 20 then into target_a
  when sal >= 1200 then into target_b
  when sal < 1200 then into target_c
select * from emp;

```

124 SQL로 알고리즘 문제 풀기 14(몬테카를로 알고리즘)

몬테카를로 알고리즘은 수많은 노가다를 통해서 정답을 알아내는것을 뜻한다

카페 SQL 게시판 게시물 1128 몬테카를로 알고리즘으로 원주율을 알아내세요~

ex) 삼성생명 데이터 분석// 금융사고가 생겼을 때를 대비해서 최대 얼마만큼의 자금을 보유하고 있어야 하는가?

☆446)##21) 몬테카를로 알고리즘으로 원주율(pie) 을 구하시오

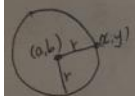
```

undefine p
accept p prompt '실험횟수를 입력하세요~'
with quater as ( select dbms_random.value(0,1) as x1,
                      dbms_random.value(0,1) as y1
                  from dual
                  connect by level <= &&p)
select count(*)*4/&p as pie
  from quater
 where power(x1,2)+power(y1,2) <= 1;

```

위 답을 위해서는 알고리즘 짜는것도 중요하지만 원의 성질과 수학적 정리가 더 중요하다

테가르조 algosism -



$$\sqrt{(x-a)^2 + (y-b)^2} = r$$

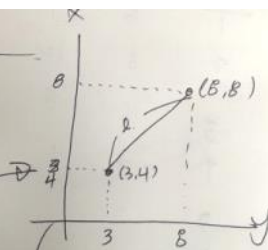
$$\Rightarrow (x-a)^2 + (y-b)^2 = r^2$$



원래의 모습이다

$$q_{\text{water}} = r^2 \pi / 4$$

$$\square = r^2$$



↓ 파타리 382

$$L = \sqrt{(8-3)^2 + (8-4)^2}$$

$$= \sqrt{25 + 16}$$

$$= \sqrt{41}$$

$$r^2 : r_c^2/4 = 1 : \frac{\pi}{4} \quad \begin{matrix} 57 \\ 26 = 1\text{의} \\ 6\% \end{matrix}$$

7.1의
이론

$$\frac{\pi}{4} : 1 = \text{정현 안쪽 개수} : \text{정현 바깥 개수}$$

$\pi = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2}$
 ↓
 16가지 P

x 4
 ↓
 4의 Random Value
 ↓
 3가지

SQL 페이지 168

##1116

2020년 11월 16일 월요일 오전 9:47

다중 insert 문의 종류

- 1) 무조건 all insert 문
- 2) 조건부 all insert 문
- 3) 조건부 first insert 문
- 4) **pivoting insert 문**

※ pivot inset 문

어렵게 pivot 문으로 작성하지 않고 pivoting insert 문으로 데이터를 입력한 후에 pivot 문을 사용하지 않은 간단한 SQL로 결과를 볼 때 필요한 insert 문

ex)

```
create table week_sum_sal
```

```
( empno number(10),  
  ename varchar2(10),  
  mon number(10),  
  tues number(10),  
  wed number(10),  
  thur number(10),  
  fri number(10) );
```

```
insert into week_sum_sal
```

```
select empno, ename, sal*0.1, sal*0.2, sal*0.3, sal*0.4, sal*0.5  
from emp;
```

446) week_sum_sal 에서 이름, 일당을 다 합친 주급을 출력하시오!

```
select ename, mon+tues+wed+thur+fri  
from week_sum_sal;
```

위와같이 sum 그룹함수를 이용하지 못하고 컬럼의 데이터를 다 일일이 더해서 SQL 을 작성하면 번거롭다

pivoting insert 문으로 위 문제를 해결

```
1)  
create table sales_info  
( ename varchar2(10),  
  sal number(10));
```

```
insert all into sales_info values(ename, mon)  
into sales_info values(ename, tues)  
into sales_info values(ename, wed)  
into sales_info values(ename, thur)  
into sales_info values(ename, fri)
```

```
select ename, mon, tues, wed, thur, fri  
from week_sum_sal;
```

```
ENAME SAL
```

ENAME	SAL
1 KING	500
2 BLAKE	285
3 CLARK	245
4 JONES	298
5 MARTIN	125
6 ALLEN	160
7 TURNER	150
8 JAMES	95
9 WARD	125
10 FORD	300
11 SMITH	80
12 SCOTT	300
13 ADAMS	110
14 MILLER	130
15 KING	1000
16 BLAKE	570
17 CLARK	490
18 JONES	595
19 MARTIN	250
20 ALLEN	320

2) 446번

```
select ename, sum(sal)
  from sales_info
 group by ename;
```

SQL복습

1. 기본 select 문장
2. 함수 : 단일행 함수, 복수행 함수, 데이터 분석함수
3. 조인
4. 집합 연산자
5. 서브쿼리문
6. DML 문장
7. DDL 문장
8. TCL 문장
9. 계층형 질의문
10. with절
11. 제약
12. database object 5가지
13. flashback 5가지
14. 다중 insert 문 4가지

■금요일 시험문제유형

#문제유형-유형별연습문제)

447 #1) 직업, 직업별 토달월급을 출력하는데 직업이 SALESMAN 은 제외하고 출력하고 직업별 토달월급이 4000 이상인것만 출력하고 직업별 토달월급이 높은것부터 출력하시오!

```
select job, sum(sal)
  from emp
 where job != 'SALESMAN'
 group by job
 having sum(sal) >= 4000
 order by 2 desc;
```

448 #2) 이름, 직업, 월급을 출력하는데 직업은 ABCD 순서대로 출력하고 직업을 abcd 순서대로 출력한 것을 기준으로 월급을 높은것부터 출력하시오!

```
select ename, job, sal
  from emp
 order by job asc, sal desc;
```

449 #2-2) 이름, 부서번호, 입사일을 출력하는데 부서번호를 asc 로 출력하고 부서번호를 ascending 하게 출력된 것을

기준으로 입사일을 descending 하게 출력하시오!

```
select ename, deptno, hiredate
  from emp
 order by deptno asc, hiredate desc;
```

450 #3) 직업, 직업별 토달월급을 출력하는데 맨 아래에 전체 토달월급이 출력되게 하시오!

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

451 #3-2) 위 결과를 grouping sets 를 이용하여 출력하시오! 198번 문제를 참고하면 grouping sets 에 대한 이해 가능

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

452 #3-3) 아래의 SQL의 결과를 union all 로 변경하시오! (단 rollup, cube, grouping sets 와 같은 레포팅 함수 사용X)

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

DEPTNO	JOB	SUM(SAL)
1	30 (null)	9400
2	10 (null)	8750
3	20 (null)	10875
4	(null) SALESMAN	5600
5	(null) CLERK	4150
6	(null) ANALYST	6000
7	(null) MANAGER	8275
8	(null) PRESIDENT	5000

```
select to_number(null) as deptno, job, sum(sal)
  from emp
  group by job
```

union all

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

union all 위아래의 쿼리의 **컬럼의 개수, 데이터 타입, 컬럼의 이름**이 동일해야 한다.
이 조건을 만족해야 order by 절을 사용하여 정렬할 때 문제가 생기지 않는다.

453 #3-4) 아래의 SQL 을 union all 로 변경하시오!

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

>> 답안

```
select deptno, null as job, sum(sal)
  from emp
  group by deptno
union all
select to_number(null) as deptno, job, sum(sal)
  from emp
  group by job
union all
select to_number(null) as deptno, to_char(null) as job, sum(sal)
  from emp
  order by deptno asc, job asc;
```

454 #4) 부서위치, 부서위치별 토탈월급을 출력하는데 부서위치별 토탈월급을 출력할 때 천단위 콤마를 부여해서 하시오!

```
select d.loc, to_char(sum(e.sal), '999,999') as sal
  from emp e, dept d
 where e.deptno=d.deptno
  group by d.loc;
```

455 #5) 부서위치, 부서위치별 토탈월급을 출력하는데 가로로 출력하시오!

```
select sum(decode( d.loc, 'NEW YORK', e.sal)) as "NEW YORK",
       sum(decode( d.loc, 'DALLAS', e.sal)) as DALLAS,
```

```

        sum(decode( d.loc, 'CHICAGO', e.sal)) as CHICAGO
    from emp e, dept d
    where e.deptno=d.deptno;
컬럼별칭에 '(싱글 쿼테이션 마크) 를 사용하면 안된다. 차라리 " 을 쓰자

```

456 #6) ALLEN 보다 더 늦게 입사한 직원들의 이름과 입사일을 출력하는데 최근에 입사한 직원부터 출력하시오!

```

select ename, hiredate
    from emp
    where hiredate > (select hiredate
                      from emp
                      where ename='ALLEN')
    order by 2 desc;

```

457 #7) 관리자가 아닌 직원들의 이름을 출력하시오! (자기 밑에 직속부하가 한명도 없는 직원들)

```

select ename
    from emp
    where empno not in ( select mgr
                        from emp
                        where mgr is not null);

```

458 #8) 직업, 이름, 월급, 순위를 출력하는데 순위가 월급이 직업별로 각각 월급이 높은 순서대로 순위를 부여하시오!

```

select job, ename, sal, dense_rank() over( partition by job order by sal desc) as rank
    from emp;

```

☆459 #8-2) 위 결과를 다시 출력하는데 순위가 1등인 직원들만 출력하시오!

```

select *
    from (select job, ename, sal, dense_rank() over( partition by job order by sal desc) as rank
          from emp)
    where rank = 1;

```

>> inline view, with, create view, order by ~ fetch 등 다양한 방법으로 구할 수 있다.

460 #9) 아래의 테이블을 생성하고 empno 에 primary 제약을 거시오!

테이블명 : emp460

컬럼명 : empno, enmae, sal, hiredate

>>1 생성 후 alter 로 추가

```

create table emp460

```

```

( empno  number(10),

```

```

  ename  varchar2(10),

```

```

  sal    number(10),

```

```

  hiredate date );

```

```

alter table emp460

```

```

    add constraint emp460_empno_pk primary key(empno);

```

>>2 생성시 추가

```
create table emp460
( empno number(10) constraint emp460_empno_pk primary key,
  ename varchar2(10),
  sal number(10),
  hiredate date );
```

461 #9-2) 아래의 데이터를 담은 테이블을 emp461 로 생성하고 아래의 데이터를 입력하고 ename 에 unique 제약을 거시오!

	EMPNO	ENAME	SAL	HIREDATE
1	3333	allen	5000	83/05/02
2	2222	smith	4000	82/12/21
3	1111	scott	3000	81/11/17

```
create table emp461
( empno number(10),
  ename varchar2(10),
  sal number(10),
  hiredate date );
alter table emp461
  add constraint emp461_ename_uk unique(ename);
insert into emp461 values(1111, 'scott', 3000, to_date('81/11/17', 'RR/MM/DD'));
insert into emp461 values(2222, 'smith', 4000, to_date('82/12/21', 'RR/MM/DD'));
insert into emp461 values(3333, 'allen', 5000, to_date('83/05/02', 'RR/MM/DD'));
```

462 #10) 이름, 월급, 부서번호, 자기가 속한 부서번호의 평균월급을 출력하시오!

```
select e.ename, e.sal, e.deptno, d.sd
  from emp e, ( select deptno, avg(sal) as sd
                from emp
                group by deptno) d
 where e.deptno=d.deptno;
```

463 #10-2) 위의 SQL을 with 절로 구현하시오!

```
with d as (select deptno, avg(sal) as sd
           from emp
           group by deptno)
select e.ename, e.sal, e.deptno, d.sd
  from emp e, d d
 where e.deptno=d.deptno;
```

464 #11) 1부터 10까지의 숫자중에 홀수만 출력하시오! // 11번 유형은 알고리즘 문제중에 하나 출제예정!

```
select level
  from dual
 where mod(level,2)!=0
 connect by level<=10;
```

>>with 절 활용

```

with num_table as (select level as num1
                    from dual
                    connect by level <=10)
select num1
   from num_table
  where mod(num1,2)=1;

```

SQL 문장의 종류 : TCL

1. 쿼리문
2. DML : insert, update, delete, merge
3. DDL : create, alter, drop, truncate, rename
4. **DCL : grant, revoke**
4. TCL : commit, rollback, savepoint

■ 1. grant : user 생성

```

create user king          <<<<< user name
   identified by tiger;    <<<<< password

```

위 상태로는 접속이 불가하기 때문에 **grant** 를 사용하여 권한을 부여한다

grant connect to king;

이후 show user 을 이용하여 사용자를 확인한다

465) allen 이라는 유저를 패스워드 tiger1234 로 생성하고 allen 으로 접속할 수 있도록 권한을 부여하고 접속하십시오!

```
create user allen
```

```
   identified by tiger1234;
```

```
grant connect to allen;
```

```
>>open cmd prompt
```

```
sqlplus allen/tiger1234
```

```
show user
```

```

C:\Users\STU-29>sqlplus allen/tiger1234

SQL*Plus: Release 19.0.0.0.0 - Production on 월 11월 16 14:24:17 2020
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

마지막 성공한 로그인 시간: 월 11월 16 2020 14:23:09 +09:00

다음에 접속됨:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> show user
USER은 "ALLEN"입니다
SQL>

```

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

```
grant dba to scott;
```

```
connect scott/tiger
```

위 SQL은 emp12 table 생성 script 인데 보면 scott 계정이 생성되고 dba 권한(가장 높은 권한)을 부여한 것이다

466) allen 유저에서 아래의 테이블을 생성하시오!

테이블명 : emp466

컬럼명 : empno, ename, sal

```
USER은 "ALLEN"입니다
SQL> create table emp466
  2  ( empno number(10),
  3    ename varchar2(10),
  4    sal   number(10));
create table emp466
*
1행에 오류:
ORA-01031: 권한이 불충분합니다
```

위와 같이 SQL 을 작성하면 권한이 없다고 err 발생. create table 권한이 있어야 테이블을 생성할 수 있다.

create table 권한을 allen 유저에게 부여하기 (아래 SQL 참고)

1) sqlplus "/as sysdba" 접속 후 show user로 확인

```
C:\Users\STU-29>sqlplus "/as sysdba"
SQL> show user
USER은 "SYS"입니다
```

2) grant create table to allen;

```
SQL> grant create table to allen;
권한이 부여되었습니다.
```

3) allen 으로 재접속해서 table 생성하기

```
SQL> create table emp466
  2  ( empno number(10),
  3    ename varchar2(10),
  4    sal   number(10));
테이블이 생성되었습니다.
```

※ 오라클에 존재하는 권한 리스트를 확인하는 방법

1) 내가 가지고있는 시스템 권한 확인하는 방법

```
select * from session_privs;
```

```
SQL> select * from session_privs;
PRIVILEGE
-----
SET CONTAINER
CREATE TABLE
CREATE SESSION
```

2) database에 있는 모든 테이블들을 다 볼 수 있는 권한

select any table <<< 강력한 권한

■2. revoke : 권한 취소

권한을 취소하는 sql : 권한을 부여한 계정에서 작성하면 기존 부여했던 권한을 취소할 수 있다.

ex) revoke create table from allen;

위 SQL 은 sysdba 계정에서 실시

```
SQL> revoke create table from allen;
```

권한이 취소되었습니다.

■3. user drop

sys유저에서 아래와 같이 작업하면 지워진다

drop user allen cascade;

```
SQL> drop user allen cascade;
```

사용자가 삭제되었습니다.

467) king 유저를 삭제하시오!

```
SQL> drop user king cascade;
```

사용자가 삭제되었습니다.

■4. db 에 있는 유저목록 확인

```
select username
```

```
from dba_users;
```