

제 3 장

검색 (SELECT)





SQL의 특징

- **SQL은 대화식 언어이다. 단편적인 질문 위주여서 명령문이 짧고 간결하다.**
 - 가장 많은 월급은 얼마인가? → 520만원이다.
 - 520만원을 받는 직원의 이름은 무엇인가? → 홍길동이다.
 - 홍길동은 어느 부서 소속인가? → 영업부이다.
- **단순한 명령을 조합하여 복잡한 명령을 처리한다. 가급적 한번에 처리하는 것이 좋다.**
 - 월급을 가장 많이 받는 직원이 속한 부서는?
- **제어문이 빈약해 C#, 자바 같은 고수준 언어와 함께 사용한다.**
- **둘 째로, SQL은 선언적인 언어이다. 구체적인 절차를 일일이 명령으로 기술하는 것이 아니라 원하는 것만 밝힌다.**
 - 부서별 월급 평균을 구하고 평균에 미달하는 월급을 받는 직원이 가장 많은 부서를 구하라.
- **절차적 언어는 앞쪽부터 순서대로 처리한다. SQL은 과정은 기술하지 않고 요구 사항만 전달한다.**
 - 절차적 언어 : 찬장의 흰 사발을 꺼내 우물가에 가서 물을 가득 담아 안방으로 냉큼 가져 와라.
 - 선언적 언어 : 돌쇠야, 물 떠와라.
- **지시가 복잡한 건 상관없지만 모호해서는 안되며 원하는게 뭔지 정확히 밝혀야 한다.**



SQL 테이블

- 관계형 데이터베이스는 정보를 표 형태로 정리한다. 예 : 주소록

이름	주소	전화번호	키	성별
김상형	경기도 오산시	111-2222	180	남자
김한슬	경기도 용인시	333-4444	178	여자
권성직	경기도 화성시	555-6666	175	남자
최상미	경기도 화성시	777-8888	166	여자
문종민	서울시 송파구	999-0000	179	남자

- 익숙하고 직관적이다. 정보를 표 형태로 정리해 놓은 것을 테이블(table)이라고 부른다.
- 엔터티(Entity) : 테이블이 표현하는 대상. 세상의 모든 것.
- 레코드(Record) : 테이블에 저장된 엔터티 하나. 도표의 가로줄에 해당한다.
- 필드(Field) : 레코드의 세부 속성. 테이블의 세로줄에 해당한다. 엔터티에 따라 필드의 목록은 달라진다.
 - 컴퓨터 : CPU, 메모리 용량, 그래픽 카드, 사운드 카드, 하드 디스크
 - 가계부 : 날짜, 금액, 사용처, 수입, 지출, 잔액
 - 재고 : 분류, 품목, 제품명, 입고날짜, 재고, 유통기간



SQL 테이블

- 필드 여러 개가 모여 레코드 하나가 되며 레코드 여러 개가 모여 테이블이 된다.

	이름	주소	전화번호	키	성별
레코드 →	김상형	경기도 오산시	111-2222	180	남자
레코드 →	김한솔	경기도 용인시	333-4444	178	여자
	권성직	경기도 화성시	555-6666	175	남자
	최상미	경기도 화성시	777-8888	166	여자
	문종민	서울시 송파구	999-0000	179	남자

테이블

- 레코드와 필드는 문맥에 따라 여러 가지 동의어가 정의되어 있다.
- 보는 관점에 따라 용어가 조금씩 다를 뿐이며 칭하는 대상은 같다.
- 실무에서는 별 구분없이 섞어서 사용한다.

정식 명칭	도표 관련 용어	한국말로	모델링 용어
레코드	로(Row)	행	튜플(Tuple)
필드	컬럼(Column)	열	어트리뷰트(Attribute)



SQL DB 오브젝트

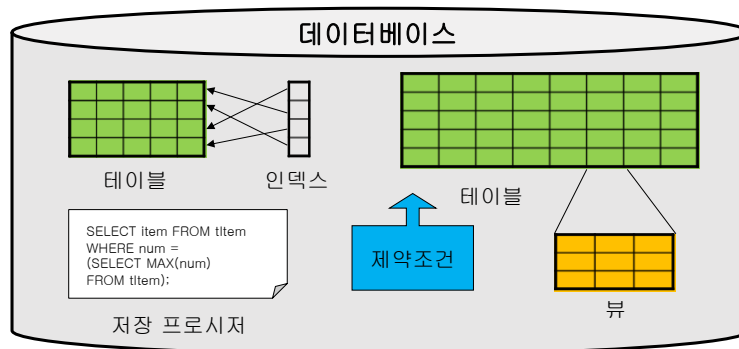
- 여러 개의 테이블이 모여야 정보로서의 가치를 발휘한다. 회원, 상품, 구매라는 세 가지 정보가 모여야 온전한 쇼핑몰을 표현할 수 있다.

번호	이름	주소
1	구홍녀	무거동
2	김상영	신북 로터리
3	김영주	운산면

번호	상품	가격
1	노트북	120
2	원피스	32
3	옥장판	16

날짜	고객	상품
2021-5-8	3	2
2021-5-12	2	1
2021-5-13	1	3

- 관련 있는 테이블과 이를 지원하는 장치가 모두 구비되어야 완전한 데이터베이스가 되어 현실 세계를 제대로 반영할 수 있다.
- 데이터베이스에 저장되는 모든 것을 통칭하여 DB 오브젝트라고 하며 간단하게 개체 (Object)라고 부른다.



목차

3.1 SELECT

3.2 조건 검색

3.3 검색 결과의 순서화 (정렬)

3.4 함수 (Function)

3.1 SELECT

- 데이터를 검색하기 위한 명령어가 **SELECT**이며, 형식은 다음과 같다.

```
SELECT [DISTINCT] column-commalist  
FROM table-names  
[WHERE predicate]  
[GROUP BY column-commalist [HAVING predicate]]  
[ORDER BY column-commalist]
```

3.1 SELECT

- 테이블의 구조를 알기 위해 DESCRIBE 명령어를 사용한다.

SQL> describe student;

NAME	NULLABLE	TYPE	DEFAULT	COMMENT
STU_NO	NOT NULL	NUMBER(9)		
STU_NAME		VARCHAR2(12)		
STU_DEPT		VARCHAR2(20)		
STU_GRADE		NUMBER(1)		
STU_CLASS		CHAR(1)		
STU_GENDER		CHAR(1)		
STU_HEIGHT		NUMBER(5,2)		
STU_WEIGHT		NUMBER(5,2)		

3.1 SELECT

- 테이블의 구조를 알기 위해 DESCRIBE 명령어를 사용한다.

SQL> describe enrol;

NAME	NULLABLE	TYPE	DEFAULT	COMMENT
SUB_NO	NOT NULL	CHAR(3)		
STU_NO	NOT NULL	NUMBER(9)		
ENR_GRADE		NUMBER(3)		

SQL> describe subject;

NAME	NULLABLE	TYPE	DEFAULT	COMMENT
SUB_NO	NOT NULL	CHAR(3)		
SUB_NAME		VARCHAR2(30)		
SUB_PROF		VARCHAR2(12)		
SUB_GRADE		NUMBER(1)		
SUB_DEPT		VARCHAR2(20)		

3.1 SELECT

●테이블에 있는 모든 데이터 검색

SQL> select * from student;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20153075	옥한빛	기계	1	C	M	177	80
20153088	이태연	기계	1	C	F	162	50
20143054	유가인	기계	2	C	F	154	47
20152088	조민우	전기전자	1	C	M	188	90
20142021	심수정	전기전자	2	A	F	168	45
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131001	김종헌	컴퓨터정보	3	C	M		72
20131025	옥성우	컴퓨터정보	3	A	F	172	63

3.1 SELECT

●특정 열의 내용 검색

SQL> select stu_no, stu_name from student;

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20152088	조민우
20142021	심수정
20132003	박희철
20151062	김인중
20141007	진현무
20131001	김종현
20131025	옥성우

3.1 SELECT

●중복 행 제거

```
SQL> select stu_dept  
2 from student;
```

STU_DEPT
기계
기계
기계
전기전자
전기전자
전기전자
컴퓨터정보
컴퓨터정보
컴퓨터정보
컴퓨터정보

```
SQL> select distinct stu_dept  
2 from student;
```

STU_DEPT
전기전자
기계
컴퓨터정보

3.1 SELECT

●중복 행 제거

SQL> select **distinct stu_grade, stu_class from student;**

STU_GRADE	STU_CLASS
1	B
1	C
3	A
2	C
2	A
3	B
3	C

3.1 SELECT

●수식을 포함한 검색

SQL> select stu_no, sub_no, enr_grade, enr_grade+10 from enrol;

STU_NO	SUB_NO	ENR_GRADE	ENR_GRADE+10
20131001	101	80	90
20131001	104	56	66
20132003	106	72	82
20152088	103	45	55
20131025	101	65	75
20131025	104	65	75
20151062	108	81	91
20143054	107	41	51
20153075	102	66	76
20153075	105	56	66
20153088	102	61	71
20153088	105	78	88

3.1 SELECT

- 결과열에 별칭(Alias) 부여하기

SQL> select stu_no as ID, stu_name as NAME from student;

ID	NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20152088	조민우
20142021	심수정
20132003	박희철
20151062	김인중
20141007	진현무
20131001	김종헌
20131025	옥성우

3.1 SELECT

●연결 연산자

SQL> select stu_dept || stu_name as 학과성명 from student;

학과성명
기계육한빛
기계이태연
기계유가인
전기전자조민우
전기전자심수정
전기전자박희철
컴퓨터정보김인중
컴퓨터정보진현무
컴퓨터정보김종헌
컴퓨터정보옥성우

3.1 SELECT

●연결 연산자

SQL> select stu_dept || ',' || stu_name || ' 입니다.' as "학과 와 성명"
from student;

	학과 와 성명
1	기계, 옥한빛 입니다.
2	기계, 이태연 입니다.
3	기계, 유가인 입니다.
4	전기전사, 소민우 입니다.
5	전기전사, 소성철 입니다.
6	전기전사, 박희정 입니다.
7	컴퓨터성정보, 김진수 입니다.
8	컴퓨터성정보, 김진현 입니다.
9	컴퓨터성정보, 김준호 입니다.
10	컴퓨터성정보, 김동우 입니다.
11	인공시등, 연내빈 입니다.
12	인공시등, 김수환 입니다.
13	빅데이터개발, 연새호 입니다.
14	빅데이터개발, 박새남 입니다.

3.2 조건 검색

●WHERE절 사용하기

- 일반적으로 비교연산자를 사용한다.
- 이때 사용되는 비교연산자는 =, <, >, <=, >=, <>의 6가지이다.
- 특히 '<>'는 '!=', '^='를 사용할 수도 있다.

```
SQL> select stu_name, stu_dept, stu_grade, stu_class  
2  from student  
3  where stu_dept = '컴퓨터정보';
```

STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS
김인중	컴퓨터정보	1	B
진현무	컴퓨터정보	2	A
김종현	컴퓨터정보	3	C
육성우	컴퓨터정보	3	A

3.2 조건 검색

●논리 연산자

➤ 논리 연산자 **NOT, AND, OR**를 사용하여 여러 개의 조건을 결합하여 표현할 수 있다.

```
SQL> select stu_name, stu_dept, stu_grade, stu_class  
2  from student  
3  where stu_dept = '컴퓨터정보' and stu_grade = 2;
```

STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS
진현무	컴퓨터정보	2	A

```
select stu_name, stu_dept, stu_grade, stu_class  
from student  
where stu_dept = '전기전자'  
and not stu_grade = 2;
```

3.2 조건 검색

●범위 조건

➤ WHERE절에 BETWEEN ~ AND을 사용하여 검색할 수 있다.

SQL> select *

2 from student

3 where stu_weight between 60 and 70;

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20132003	박희철	전기전자	3	B	M		63
20151062	김인중	컴퓨터정보	1	B	M	166	67
20141007	진현무	컴퓨터정보	2	A	M	174	64
20131025	육성우	컴퓨터정보	3	A	F	172	63

3.2 조건 검색

SQL> select *

2 from student

3 where stu_no between '20140001' and '20149999';

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20141007	진현무	컴퓨터정보	2	A	M	174	64
20142021	심수정	전기전자	2	A	F	168	45
20143054	유가인	기계	2	C	F	154	47

select *

from student

where stu_no between '20231001' and '20239999';

3.2 조건 검색

●LIKE를 이용한 검색

➢ 데이터의 일부를 알고 있을 경우, 와일드카드 문자와 함께 사용한다.

```
SQL> select stu_no, stu_name, stu_dept  
2   from student  
3   where stu_name like '김%';
```

STU_NO	STU_NAME	STU_DEPT
20151062	김인중	컴퓨터정보
20131001	김종현	컴퓨터정보

●오라클에서 LIKE와 같이 사용하는 와일드카드 문자는 다음과 같다.

기호	의 미
%	0개 이상의 문자
_	1개의 문자

- select stu_no, stu_name, stu_dept
- from student
- where stu_name like '김_';

3.2 조건 검색

```
SQL> select stu_no, stu_name, stu_dept  
2  from student  
3  where stu_name like '_수%';
```

STU_NO	STU_NAME	STU_DEPT
20142021	심수정	전기전자

```
SQL> select *  
2  from student  
3  where stu_no like '2014%';
```

STU_NO	STU_NAME	STU_DEPT	STU_GRADE	STU_CLASS	STU_GENDER	STU_HEIGHT	STU_WEIGHT
20143054	유가인	기계	2	C	F	154	47
20142021	심수정	전기전자	2	A	F	168	45
20141007	진현무	컴퓨터정보	2	A	M	174	64

3.2 조건 검색

● 널(NULL) 값 처리

➤ 널 값이 존재할 수 있으며, 이는 때때로 자료처리에 있어서 문제를 발생시킨다.

```
SQL> select stu_no, stu_name, stu_height  
2 from student;
```

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20153088	이태연	162
20143054	유가인	154
20152088	조민우	188
20142021	심수정	168
20132003	박희철	
20151062	김인중	166
20141007	진현무	174
20131001	김종헌	
20131025	옥성우	172

➤ 신장(stu_height)열에 널 값이 나타남.

3.2 조건 검색

- NULL값을 가지는 연산

**SQL> select stu_name, stu_height/30.46
2 from student;**

STU_NAME	STU_HEIGHT/30.48
옥한빛	5.807086614
이태연	5.31496063
유가인	5.052493438
조민우	6.167979003
심수정	5.511811024
박희철	
김인중	5.446194226
진현무	5.708661417
김종현	
옥성우	5.643044619

3.2 조건 검색

- 데이터에 널 값의 존재 여부 질의문

```
SQL> select stu_no, stu_name, stu_height  
2  from student  
3  where stu_height is null;
```

STU_NO	STU_NAME	STU_HEIGHT
20132003	박희철	
20131001	김종현	

3.2 조건 검색

● 널 값이 아닌 행의 결과

**SQL>select stu_no, stu_name, stu_height, stu_height + 5
from student
where stu_height is not null;**

STU_NO	STU_NAME	STU_HEIGHT
20153075	옥한빛	177
20153088	이태연	162
20143054	유가인	154
20152088	조민우	188
20142021	심수정	168
20151062	김인중	166
20141007	진현무	174
20131025	옥성우	172

3.2 조건 검색

●IN 연산자

➤ 여러 개 조건 값 중 하나만 만족하는 행을 처리할 경우 사용한다.

```
SQL> select stu_no, stu_name  
2   from student  
3   where stu_dept in ('컴퓨터정보', '기계');
```

STU_NO	STU_NAME
20153075	옥한빛
20153088	이태연
20143054	유가인
20151062	김인중
20141007	진현무
20131001	김종현
20131025	옥성우

**stu_dept = '컴퓨터정보' or
stu_dept = '기계'**

```
select stu_no, stu_name, stu_dept  
from student  
where stu_dept = '컴퓨터정보'  
or stu_dept = '기계';
```

3.3 검색 결과의 순서화 (정렬)

●정렬(SORT) : 데이터를 어떤 기준에 의해 나열하는 것

➤ 기준(key)

- 기본키(primary key)
- 보조키(secondary keys)

➤ 나열방법

- 오름차순(ascending) : 생략가능
- 내림차순(descending)

===> **ORDER BY stu_no**

3.3 검색 결과의 순서화 (정렬)

```
SQL> select stu_no, stu_name  
2   from student  
3   order by stu_no;
```

STU_NO	STU_NAME
20131001	김종현
20131025	옥성우
20132003	박희철
20141007	진현무
20142021	심수정
20143054	유가인
20151062	김인중
20152088	조민우
20153075	옥한빛
20153088	이태연

3.3 검색 결과의 순서화 (정렬)

- 학생들의 신상을 학번의 내림차순으로 검색하는 질의문

```
SQL> select stu_no, stu_name  
2   from student  
3   order by stu_no desc;
```

STU_NO	STU_NAME
20153088	이태연
20153075	옥한빛
20152088	조민우
20151062	김인중
20143054	유가인
20142021	심수정
20141007	진현무
20132003	박희철
20131025	옥성우
20131001	김종현

3.3 검색 결과의 순서화 (정렬)

- 별칭이 붙어 있는 열을 기준으로 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target
2  from student
3  order by target;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20142021	심수정	전기전자	40
20143054	유가인	기계	42
20153088	이태연	기계	45
20132003	박희철	전기전자	58
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종헌	컴퓨터정보	67
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85

```
select stu_no, stu_name, stu_dept,
stu_weight-5 as target
from student
order by target desc;
```


3.3 검색 결과의 순서화 (정렬)

- 열의 순서번호를 이용하여 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by 4;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20142021	심수정	전기전자	40
20143054	유가인	기계	42
20153088	이태연	기계	45
20132003	박희철	전기전자	58
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종헌	컴퓨터정보	67
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85

3.3 검색 결과의 순서화 (정렬)

- 산술식의 열의 이름을 이용하여 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by stu_weight-5;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20142021	심수정	전기전자	40
20143054	유가인	기계	42
20153088	이태연	기계	45
20132003	박희철	전기전자	58
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종헌	컴퓨터정보	67
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85

3.3 검색 결과의 순서화 (정렬)

●여러 개의 열을 기준으로 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target
2  from student
3  order by stu_dept, target;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20143054	유가인	기계	42
20153088	이태연	기계	45
20153075	육한빛	기계	75
20142021	심수정	전기전자	40
20132003	박희철	전기전자	58
20152088	조민우	전기전자	85
20131025	육성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20151062	김인중	컴퓨터정보	62
20131001	김종헌	컴퓨터정보	67

```
select stu_no, stu_name, stu_dept,
stu_weight-5 as target
from student
order by stu_dept desc, target desc;
```

```
select stu_no, stu_name, stu_dept,
stu_weight-5 as target
from student
order by stu_dept, target desc;
```

3.3 검색 결과의 순서화 (정렬)

- 오름차순과 내림차순이 혼용된 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by stu_dept, stu_weight-5 desc;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20153075	옥한빛	기계	75
20153088	이태연	기계	45
20143054	유가인	기계	42
20152088	조민우	전기전자	85
20132003	박희철	전기전자	58
20142021	심수정	전기전자	40
20131001	김종헌	컴퓨터정보	67
20151062	김인중	컴퓨터정보	62
20141007	진현무	컴퓨터정보	59
20131025	옥성우	컴퓨터정보	58

3.3 검색 결과의 순서화 (정렬)

- SELECT절에 포함되지 않는 열을 이용하여 정렬하는 질의문

```
SQL> select stu_no, stu_name, stu_dept, stu_weight-5 as target  
2  from student  
3  order by stu_height;
```

STU_NO	STU_NAME	STU_DEPT	TARGET
20143054	유가인	기계	42
20153088	이태연	기계	45
20151062	김인중	컴퓨터정보	62
20142021	심수정	전기전자	40
20131025	옥성우	컴퓨터정보	58
20141007	진현무	컴퓨터정보	59
20153075	옥한빛	기계	75
20152088	조민우	전기전자	85
20132003	박희철	전기전자	58
20131001	김종현	컴퓨터정보	67

3.4 함수(Function)

● 함수란 ?

- 하나 이상의 인수를 전달받아 처리한 결과 값을 함수의 이름을 변수처럼 활용하여 반환해 주는 프로그램 모듈
- **단일 행 함수**는 함수가 정의된 SQL문장이 실행 될 때 각각의 행에 대해 수행되며 각 행별로 하나의 결과 값을 반환함
- **그룹함수**는 데이터를 그룹화하고 그룹 각각에 대한 결과를 반환하며, **GROUP BY 절**을 사용함

3.4 함수(Function)

● 단일 행 함수

➤ 인수의 데이터 타입에 따라

- ✓ 숫자함수
- ✓ 문자함수
- ✓ 날짜함수
- ✓ 형변환함수
- ✓ 일반함수

➤ 함수의 활용에서 함수는 **인수의 수** 및 각 **인수의 역할**이 중요함으로
인수를 바꾸어가며 충분한 실습이 필요함

3.4 함수(Function)

●숫자 함수

➤ 숫자 인수를 사용하는 함수

함 수	설 명
ROUND(인수1,인수2)	인수1의 값을 인수2의 자리로 반올림하여 반환
TRUNC(인수1,인수2)	인수1의 값을 인수2 자리까지 유지하고, 나머지는 절삭하여 반환
MOD(인수1, 인수2)	인수1 값을 인수2 값으로 나눈 나머지를 반환
ABS(인수)	인수의 절대값을 반환
FLOOR(인수)	소숫점 이하 자리를 절삭하여 반환

3.4 함수(Function)

●ROUND함수

```
SQL> select round(345.678), round(345.678, 0),  
2 round(345.678, 1), round(345.678, -1)  
3 from dual;
```

ROUND(345.678)	ROUND(345.678,0)	ROUND(345.678,1)	ROUND(345.678,-1)
346	346	345.7	350

```
select round(345.678), round(345.678, 0),  
round(345.678, 1), round(345.678, 2), round(345.678, 3),  
round(655.678, 1), round(655.678, 2), round(655.678, 3),  
round(345.678, -1), round(345.678, -2), round(345.678, -3),  
round(655.678, -1), round(655.678, -2), round(655.678, -3)  
from dual;
```

```
select trunc(345.678), trunc(345.678, 0),  
trunc(345.678, 1), trunc(355.678, 2), trunc(355.678, 3), trunc(355.67890, 3), trunc(355.67890, 4)  
from dual;
```

```
select mod(100, 3) from dual;
```

```
select floor(123.4567) from dual;--123  
select trunc(345.678), trunc(345.678, 0) from dual;-- 345
```

3.4 함수(Function)

● 문자 함수

➤ 인수를 문자로 하는 함수

함 수	설 명
LOWER(인수)	인수를 모두 소문자로 변환하여 반환
UPPER(인수)	인수를 모두 대문자로 변환하여 반환
INITCAP(인수)	인수 단어의 첫 번째 문자를 대문자로 변환하여 반환
CONCAT(인수1,인수2)	두 개의 문자 인수를 연결하여 반환
SUBSTR(인수1,인수2,인수3,인수4)	문자열인수1의 일부분을 추출하여 반환
LENGTH(인수)	문자인수의 길이를 반환
INSTR(인수1,인수2,인수3,인수4)	문자인수 중 특정 문자의 절대위치를 반환
LPAD(인수1,인수2,인수3)	자릿수를 지정하고 빈 공간을 특정 문자로 왼쪽부터 채워서 문자열을 반환
RPAD(인수1,인수2,인수3)	자릿수를 지정하고 빈 공간을 특정 문자로 오른쪽부터 채워서 문자열을 반환

HR 계정에서 employees 테이블을 대상으로 편집

```
SQL> conn system as sysdba;
```

```
Enter password: 1234
```

```
Connected.
```

```
SQL>
```

```
SQL> alter user hr identified by hr account unlock; -- hr 계정 lock 해제
```

```
User altered.
```

Developer에서 HRManager 만들고, hr/hr계정으로 로그인합니다.

- 안에는 countries, departmenys, employees, job, locations등의 테이블 파일들이 존재합니다.

```
create table employees(
```

EMPLOYEE_ID	NUMBER(6,0) primary key,
EMPLOYEE_NAME	VARCHAR2(20) not null,
EMPLOYEE_EMAIL	VARCHAR2(25) not null,
PHONE_NUMBER	VARCHAR2(20) not null,
HIRE_DATE	DATE not null,
JOB_ID	VARCHAR2(10) not null,
SALARY	NUMBER(8,2) not null,
COMMISSION	NUMBER(6,2) not null,
MANAGER_ID	NUMBER(6,0) not null,
DEPARTMENT_ID	NUMBER(4,0) not null

```
);
```

```
insert into employees values(100,'StevenKing','StevenKing@StevenKing.com','010.123.4567','03/06/17','ITProg',24000,90,109, 10);
```

```
insert into employees values(101,'NeenaKochhar','Neena@NeenaKochhar.com','010.123.4568','05/09/21','Account',17000,100,90,20);
```

```
insert into employees values(102,'Lex','LDEHAAN@ldhan.com','010.123.4569','01/01/13','ITDataBase',17000,100,90,30);
```

```
insert into employees values(103,'Alexander','Hunold@AHUNOLD.com','010.423.4567','06/01/03','IT_PROG',9000,102,60,40);
```

```
insert into employees values(104,'Bruce','Ernst@BERNST.com','010.423.4568','07/05/21','IT_PROG',6000,103,60,50);
```

```
insert into employees values(105,'David','Austin@DAUSTIN.com','010.423.4569','05/06/25','IT_PROG',4800,103,60,60);
```

```
insert into employees values(106,'Valli','Pataballa@VPATABAL.com','010.423.4560','06/02/05','IT_PROG',4800,103,60,70);
```

```
insert into employees values(107,'Diana','Lorentz@DLORENTZ.com','010.423.5567','07/02/07','IT_PROG',4200,103,60,80);
```

```
insert into employees values(108,'Nancy','Greenberg@NGREENBE.com','010.124.4569','02/08/17','FI_MGR',12008,101,100,90);
```

```
insert into employees values(109,'Daniel','Faviet@DFAVIET.com','010.124.4169','02/08/16','FI_ACCOUNT',9000,108,100,100);
```

```
select * from employees;
```

```
commit;
```

3.4 함수(Function)

●UPPER함수

```
SQL> select upper('korea')  
2  from dual;
```

UPPER('KOREA')
korea

```
select lower(stu_gender) from student;
```

```
select lower(EMPLOYEE_NAME) from employees;
```

```
select upper(EMPLOYEE_NAME) from employees;
```

```
select initcap(EMPLOYEE_NAME) from employees;
```

```
select concat(EMPLOYEE_NAME, job_id) from employees;
```

```
select EMPLOYEE_NAME, substr(EMPLOYEE_NAME, 3,7) from employees;
```

```
select EMPLOYEE_NAME, length(EMPLOYEE_NAME) from employees;
```

```
select EMPLOYEE_NAME, instr(EMPLOYEE_NAME, 'ex') from employees;
```

```
select lpad(EMPLOYEE_NAME, 15, '%') from employees;
```

```
select rpad(EMPLOYEE_NAME, 15, '&') from employees;
```

3.4 함수(Function)

● 날짜 함수

➤ 기본 날짜 형식은 'DD-MON-RR'형식(RR형식 : 21세기와 20세기 데이터의 처리가 가능)

함 수	설 명
SYSDATE	시스템의 오늘 날짜를 반환
날짜 연산	날짜에 +, - 연산을 함
MONTHS_BETWEEN(인수1, 인수2)	인수1, 2의 날수 차이를 반환
NEXT_DAY(인수1,인수2)	인수1에서 가장 가까운 인수2의 요일을 반환
ADD_MONTHS(인수1, 인수2)	인수1에 인수2의 달을 더하여 반환
LAST_DAY(인수1)	인수1이 속한 달의 마지막 날을 반환
ROUND(인수1,인수2)	인수1의 값을 인수2를 기준으로 반올림하여 반환
TRUNC(인수1)	인수1의 값을 인수2를 기준으로 절사하여 반환

날짜 함수

-- select sysdate from dual;

select sysdate + 1 from dual;

select months_between(sysdate, hire_date) from employees; --467개월, 81-11-17, 2020-11-11
select * from employees;

--기준: 달력 일요일:1 ~ 토요일:7

select hire_date, next_day(hire_date, 5) from employees;

select hire_date, add_months(hire_date, 13) from employees;--개월수 증가

select last_day(sysdate) from dual;

select hire_date, last_day(hire_date) from employees;

--기준: year(6개월미만이면 round [X]: 2003.06.30까지)

select hire_date, round(hire_date, 'year') from employees;

--기준: month에서 16일 부터 round합니다.

select hire_date, round(hire_date, 'month') from employees;

--기준: 달력 일요일:1 ~ 토요일:7

-- 일, 월, 화는 그대로 날짜를 유지하고, 수요일부터 +1일 입니다.

select hire_date, round(hire_date, 'day') from employees;

3.4 함수(Function)

SQL> select sysdate from dual;

SYSDATE
2016-07-26 15:49

➤ SYSDATE 함수는 시스템의 현재 날짜를 반환한다.

**SQL> select next_day(sysdate, 4)
from dual;
select next_day(sysdate, '수요일')
from dual;
select next_day(sysdate, '금요일')
from dual;**

- 결과에서처럼 현재의 날짜를 기준으로 다음에 오늘 수요일을 구하게 된다.
- 이때 요일에 해당하는 숫자를 이용할 수 있다.
- 일-1, 월-2, 화-3, 수-4, 목-5, 금-6, 토-7

3.4 함수(Function)

SQL> select round(sysdate, 'MON')

2 from dual;

ROUND(SYSDATE, 'MON')
2016-08-01

- 날짜도 숫자와 같이 반올림을 할 수 있다.
- 예에서 보듯이 두 번째 인수를 MON으로 하였을 경우 달을 기준으로 반올림하여 결과를 보여준다.
- 두 번째 인수를 조절하면 년, 월, 일 등을 기준으로 반올림할 수 있다.
- **select round(sysdate, 'MON') -- 2021.4.21 수요일**
- **from dual; -- 21/05/01 토요일**
- **--'MON'을 month로 인식하여 달을 일자가 21일 이므로 반을 넘겼으므로 반올림 합니다.**

3.4 함수(Function)

●변환 함수

➤ 데이터의 형을 변환함

함 수	내 용
TO_NUMBER	문자 데이터를 숫자 데이터로 변환
TO_DATE	문자 데이터를 날짜 데이터로 변환
TO_CHAR	숫자, 날짜 데이터를 문자 데이터로 변환

3.4 함수(Function)

●TO_CHAR 함수

- TO_CHAR 함수는 주로 출력에 형식을 지정하기 위해 사용되며, 날짜형, 숫자형 모든 데이터에 사용한다.

```
SQL> select empno, ename,  
           to_char(hiredate, 'yyyy-mm-dd') as 입사년월  
2  from emp;
```

```
●select employee_id, employee_name,  
   to_char(hire_date, 'yyyy-mm-dd') as 입사년월  
from employees;
```

SQL 인출된 모든 행: 14(0.003초)			
	EMPNO	ENAME	입사년월
1	7839	KING	1981-11-17
2	7566	JONES	1981-04-02
3	7698	BLAKE	1981-05-01
4	7782	CLARK	1981-06-09
5	7788	SCOTT	1987-07-13
6	7902	FORD	1981-12-03
7	7499	ALLEN	1981-02-20
8	7521	WARD	1981-02-22
9	7654	MARTIN	1981-09-28
10	7844	TURNER	1981-09-08
11	7900	JAMES	1981-12-03
12	7934	MILLER	1982-01-23
13	7369	SMITH	1980-12-17
14	7876	ADAMS	1987-07-13

3.4 함수(Function)

●TO_NUMBER 함수

➤ TO_NUMBER 함수는 숫자 형태의 문자를 숫자로 변환할 때 사용한다.

```
SQL> select to_char(to_number(1234.5678), '9999.999')  
2 from dual;
```

TO_CHAR(TO_NUMBER(1234.5678), '9999.999')
1234.568

```
SQL> select to_char(to_number(1234.5678), '999.999')  
2 from dual;
```

TO_CHAR(TO_NUMBER(1234.5678), '999.999')
#####

3.4 함수(Function)

●TO_DATE 함수

➤ TO_DATE 함수는 날짜 형태의 문자를 날짜로 변환할 때 사용한다.

```
select EMPLOYEE_ID, EMPLOYEE_NAME  
from employees  
where hire_date = to_date('02/08/16','yy-mm-dd');
```

	EMPLOYEE_ID	EMPLOYEE_NAME
1	109	Daniel

3.4 함수(Function)

- 일반 함수

- NVL 함수

➢ 인수 1이 null이면, 인수2를 아니면 인수를 반환함

```
select nvl(stu_height, 0), stu_name  
from student;
```

	NVL(STU_HEIGHT,0)	STU_NAME
1	177	목한빛
2	162	이태연
3	154	유가인
4	188	소빈우
5	168	심수성
6	0	박희철
7	166	김인중
8	174	신현우
9	0	김승연
10	172	목성우
11	174.88	현내빈
12	192.88	김수환
13	172.77	현새호
14	169.23	곽새님

3.4 함수(Function)

● NVL2 함수

➤ 인수 1이 널이 아니면 인수 2를 널이면 인수 3을 반환함

```
SQL> select ename, sal, comm, nvl2(comm, sal+comm, sal) from emp;  
select employee_name, salary, commission, nvl2(commission, salary +  
commission, salary)  
from employees;
```

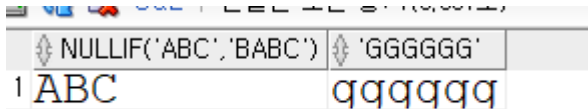
	ENAME	SAL	COMM	NVL2(COMM,SAL+COMM,SAL)
1	KING	5000	(null)	5000
2	JONES	2975	(null)	2975
3	BLAKE	2850	(null)	2850
4	CLARK	2450	(null)	2450
5	SCOTT	3000	(null)	3000
6	FORD	3000	(null)	3000
7	ALLEN	1600	300	1900
8	WARD	1250	500	1750
9	MARTIN	1250	1400	2650
10	TURNER	1500	0	1500
11	JAMES	950	(null)	950
12	MILLER	1300	(null)	1300
13	SMITH	800	(null)	800
14	ADAMS	1100	(null)	1100

3.4 함수(Function)

●NULLIF 함수

- 인수 1과 인수 2의 값을 비교하여 그 값이 같으면 NULL을 아니면 인수1의 값을 반환함

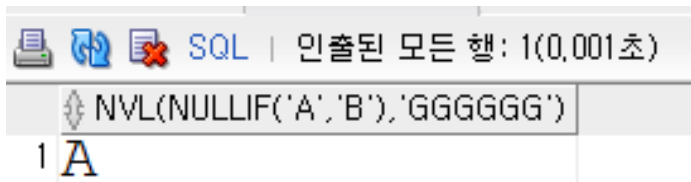
```
select nullif('ABC', 'BABC'), 'ggggggg'  
from dual;
```



A screenshot of a SQL query result window. The title bar shows icons for a document, a refresh button, and a close button, followed by the text 'SQL | 인출된 모든 행: 1(0,001초)'. The query text 'NULLIF('ABC','BABC')' and 'GGGGGG' is visible in the top section. The result table has two columns. The first column contains the value 'ABC' and the second column contains the value 'ggggggg'.

NULLIF('ABC','BABC')	GGGGGG
ABC	ggggggg

```
select nvl(nullif('A', 'B'), 'ggggggg')  
from dual;
```



A screenshot of a SQL query result window. The title bar shows icons for a document, a refresh button, and a close button, followed by the text 'SQL | 인출된 모든 행: 1(0,001초)'. The query text 'NVL(NULLIF('A','B'),'GGGGGG')' is visible in the top section. The result table has one column containing the value 'A'.

NVL(NULLIF('A','B'),'GGGGGG')
A

3.4 함수(Function)

●COALESCE 함수

- 인수 1의 값이 널값이 아니면 인수1의 값을, 널값이면 인수 2의 값을 검사하여 널 값이 아니면 인수 2의 값을 반환하고, 아니면 인수 3의 값을 검사하여 값을 반환함

```
SQL> select coalesce(null, null, 10, 100, null)
2  from dual;
```

COALESCE(NULL,NULL,10,100,NULL)
10

```
select coalesce(null, 45678, 10, 100, null)
from dual;--45678
```

```
select coalesce(null, null, null,100, null)
from dual;--100
```

```
select coalesce(null, null, null, null, 12345)
from dual;--12345
```

--숫자끼리, 문자끼리 따로 체크합니다.

```
select coalesce(null, null, '자장면', null)
from dual;
```

3.4 함수(Function)

- CASE 함수 조건에 따른 처리

```
SELECT column-names
  CASE WHEN condition-1 THEN statement-1,
        WHEN condition-2 THEN statement-2,
        .....
        WHEN condition-n THEN statement-n,
        ELSE statement
  END
FROM table-name;
```

3.4 함수(Function)

●CASE 함수

```
select * from employees;  
select EMPLOYEE_ID, EMPLOYEE_NAME, JOB_ID, SALARY,  
       case JOB_ID when 'FI_MGR' then SALARY * 1.1  
              when 'Account' then SALARY * 1.15  
              when 'IT_PROG' then SALARY * 1.2  
              when 'ITDataBase' then SALARY * 1.3  
       else SALARY  
       end as 급여인상  
from employees;
```

SQL | 인출된 모든 행: 10(0.003초)

EMPLOY...	EMPLOYEE_NAME	JOB_ID	SALARY	급여인상
1	101 NeenaKochhar	Account	17000	19550
2	102 Lex	ITDataBase	17000	22100
3	103 Alexander	IT_PROG	9000	10800
4	104 Bruce	IT_PROG	6000	7200
5	105 David	IT_PROG	4800	5760
6	106 Valli	IT_PROG	4800	5760
7	107 Diana	IT_PROG	4200	5040
8	108 Nancy	FI_MGR	12008	13208.8
9	109 Daniel	FI_ACCOUNT	9000	9000
10	100 StevenKing	IT_PROG	24000	28800

3.4 함수(Function)

●DECODE 함수

```
DECODE ( column-name, condition-1, statement-1,  
                                condition-2, statement-2,  
                                .....  
                                condition-n, statement-n,  
                                statement)
```

3.4 함수(Function)

●DECODE 함수

```
select EMPLOYEE_ID, EMPLOYEE_NAME, JOB_ID, SALARY,  
       decode(JOB_ID, 'FI_MGR', SALARY * 1.1,  
              'Account', SALARY * 1.15,  
              'IT_PROG', SALARY * 1.2,  
              'ITDataBase', SALARY * 1.3,  
              SALARY) as 인상된급여  
from employees;
```

	EMPLOYEE_ID	EMPLOYEE_NAME	JOB_ID	SALARY	인상된급여
1	101	NeenaKochhar	Account	17000	19550
2	102	Lex	ITDataBase	17000	22100
3	103	Alexander	IT_PROG	9000	10800
4	104	Bruce	IT_PROG	6000	7200
5	105	David	IT_PROG	4800	5760
6	106	Valli	IT_PROG	4800	5760
7	107	Diana	IT_PROG	4200	5040
8	108	Nancy	FI_MGR	12008	13208.8
9	109	Daniel	FI_ACCOUNT	9000	9000
10	100	StevenKing	IT_PROG	24000	28800

3.4 함수(Function)

●그룹 함수

- 여러 행에 대한 연산 즉 평균, 개수 등의 결과값을 반환하는 함수
- SELECT문에서 **GROUP BY**절을 사용함

●그룹 함수의 종류

함 수	기 능
COUNT()	조건을 만족하는 열의 데이터 값들의 개수를 반환
COUNT(*)	모든 행의 개수를 반환
SUM()	조건을 만족하는 열의 데이터 값들의 합을 반환
AVG()	조건을 만족하는 열의 데이터 값들의 평균을 반환
MAX()	조건을 만족하는 열의 데이터 값들 중 최댓값을 반환
MIN()	조건을 만족하는 열의 데이터 값들 중 최솟값을 반환
STDDEV()	조건을 만족하는 열의 데이터 값들의 표준편차를 반환
VARIANCE()	조건을 만족하는 열의 데이터 값들의 분산 값을 반환

3.4 함수(Function)

●MAX와 MIN함수

```
SQL> select max(enr_grade), min(enr_grade)
2  from enrol;
```

MAX(ENR_GRADE)	MIN(ENR_GRADE)
81	41

```
SQL> select min(stu_weight), max(stu_weight)
2  from student
3  where stu_dept = '기계';

select min(stu_weight), max(stu_weight)
      from student
      where stu_dept = '전기전자';

select min(stu_weight), max(stu_weight)
      from student
      where stu_dept = '컴퓨터정보';
```

3.4 함수(Function)

●COUNT 함수

```
SQL> select count(*), count(stu_height)
2  from student;
```

COUNT(*)	COUNT(STU_HEIGHT)
10	8

```
SQL> select count(stu_dept), count(distinct stu_dept)
2  from student;
```

COUNT(STU_DEPT)	COUNT(DISTINCTSTU_DEPT)
10	3

3.4 함수(Function)

●SUM과 AVG함수

```
SQL> select sum(stu_weight), to_char(avg(stu_weight), '9999.99')  
2  from student  
3  where stu_dept = '컴퓨터정보';
```

SUM(STU_WEIGHT)	TO_CHAR(AVG(STU_WEIGHT), '9999.99')
266	66.50

```
select count(*) as 학생, sum(stu_height) as 신장합,  
count(stu_height)as 해당학생수, o_char(avg(stu_height), '999.999') as 평균신장  
from student;
```

학생	신장합	해당학생수	평균신장
10	1361	8	170.125

```
select count(*) as 직원수, sum(sal) as 연봉합계,  
       count(mgr)as 직장상사수, avg(comm)as 보너스평균  
from emp;
```

	직원수	연봉합계	직장상사수	보너스평균
1	14	29025	13	550

```
select count(*) as 직원수, sum(salary) as 연봉합계,  
       count(manager_id)as 직장상사수, avg(commission)as 보너스평균  
from employees;
```

	직원수	연봉합계	직장상사수	보너스평균
1	10	107808	10	101.3

3.4 함수(Function)

●단일행을 이용한 GROUP BY절

```
SQL> select stu_dept, avg(stu_weight)
2  from student
3  group by stu_dept;
```

STU_DEPT	AVG(STU_WEIGHT)
전기전자	66
기계	59
컴퓨터정보	66.5

```
select stu_dept, sum(stu_weight)
from student
group by stu_dept;
```

	STU_DEPT	SUM(STU_WEIGHT)
1	기계	177
2	전기전자	198
3	컴퓨터정보	266

3.4 함수(Function)

●단일행을 이용한 GROUP BY절

```
SQL> select stu_dept, count(*)  
2  from student  
3  where stu_weight >= 50  
4  group by stu_dept;
```

STU_DEPT	COUNT(*)
전기전자	2
기계	2
컴퓨터정보	4

```
select stu_dept, count(*)  
from student  
where stu_weight < 50  
group by stu_dept;
```

	STU_DEPT	COUNT(*)
1	기계	1
2	전기전자	1

3.4 함수(Function)

●다중열 GROUP BY절

```
SQL> select stu_dept, stu_grade, count(*)  
2  from student  
3  group by stu_dept, stu_grade;
```

STU_DEPT	STU_GRADE	COUNT(*)
기계	2	1
기계	1	2
전기전자	1	1
컴퓨터정보	1	1
컴퓨터정보	3	2
컴퓨터정보	2	1
전기전자	2	1
전기전자	3	1

```
select stu_dept, stu_grade, count(*)  
from student  
group by stu_dept, stu_grade  
order by stu_dept, stu_grade;
```

3.4 함수(Function)

●HAVING절 사용

➤ 그룹함수를 적용한 결과에 다시 조건을 부여할 때는 HAVING 절을 사용한다.

```
SQL> select stu_grade, avg(stu_height)
2   from student
3   where stu_dept = '기계'
4   group by stu_grade
   having avg(stu_height) >= 160;
```

STU_GRADE	AVG(STU_HEIGHT)
1	169.5

```
select stu_grade, avg(stu_height)
from student
where stu_dept = '기계'
group by stu_grade;
```

	STU_GRADE	AVG(STU_HEIGHT)
1	1	169.5
2	2	154

3.4 함수(Function)

- HAVING절 사용

```
SQL> select stu_dept, max(stu_height)
2  from student
3  group by stu_dept
   having max(stu_height) >= 175;
```

STU_DEPT	MAX(STU_HEIGHT)
전기전자	188
기계	177

```
SQL> select to_char(max(avg(stu_height)), '999.99')
2  from student
3  group by stu_dept;
```

TO_CHAR(MAX(AVG(STU_HEIGHT)), '999.99')
178.00

```
select stu_dept, to_char(avg(stu_height), '999.99')
from student
group by stu_dept; -- 178.00, 전기전자
```

<Quiz>

```
create table emp(  
  empno number(4)  -- 직원번호  
  constraint pk_emp primary key, --PK 선언  
  ename varchar2(10), --직원이름  
  job varchar2(9),  -- 직무  
  mgr number(4),    -- 매니저(상급자)  
  hiredate date,    -- 입사일자  
  sal number(7,2),  -- 연봉  
  comm number(7,2), -- 보너스(상여금)  
  deptno number(2)  -- 부서번호  
);
```

```
INSERT INTO EMP VALUES (7839,'KING','PRESIDENT',NULL,to_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);  
INSERT INTO EMP VALUES (7566,'JONES','MANAGER',7839,to_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);  
INSERT INTO EMP VALUES (7698,'BLAKE','MANAGER',7839,to_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);  
INSERT INTO EMP VALUES (7782,'CLARK','MANAGER',7839,to_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);  
INSERT INTO EMP VALUES (7788,'SCOTT','ANALYST',7566,to_date('13-07-1987','dd-mm-yyyy'),3000,NULL,20);  
INSERT INTO EMP VALUES (7902,'FORD','ANALYST',7566,to_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);  
INSERT INTO EMP VALUES (7499,'ALLEN','SALESMAN',7698,to_date('20-2-1981','dd-mm-yyyy'),1600,300,30);  
INSERT INTO EMP VALUES (7521,'WARD','SALESMAN',7698,to_date('22-2-1981','dd-mm-yyyy'),1250,500,30);  
INSERT INTO EMP VALUES (7654,'MARTIN','SALESMAN',7698,to_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);  
INSERT INTO EMP VALUES (7844,'TURNER','SALESMAN',7698,to_date('8-9-1981','dd-mm-yyyy'),1500,0,30);  
INSERT INTO EMP VALUES (7900,'JAMES','CLERK',7698,to_date('3-12-1981','dd-mm-yyyy'), 950,NULL,30);  
INSERT INTO EMP VALUES (7934,'MILLER','CLERK',7782,to_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);  
INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7902,to_date('17-12-1980','dd-mm-yyyy'), 800,NULL,20);  
INSERT INTO EMP VALUES (7876,'ADAMS','CLERK',7788,to_date('13-07-1987','dd-mm-yyyy'),1100,NULL,20);
```

```
select * from emp;
```


<퀴즈 문제>

```
select * from emp; //employees
```

--1. 10번 부서 월급의 평균, 최고, 최저, 인원수를 구하여 출력하라.

```
select to_char(avg(sal), '9999.999'), max(sal), min(sal), count(*)  
  from emp  
 where deptno = 10;
```

```
select avg(salary), min(salary), max(salary), count(*)  
  from employees  
 where department_id = 10;
```

--2. 각 부서별 급여의 평균, 최고, 최저, 인원수를 구하여 출력하라.

```
select deptno, avg(sal), min(sal), max(sal), count(*)  
  from emp  
 group by deptno;
```

```
select department_id, avg(salary), min(salary), max(salary), count(*)  
  from employees  
 group by department_id;
```

--3. 입사일을 "2022년 5월 14일" 의 형식으로 이름, 입사일을 출력하라.

```
select ename, to_char(hiredate, 'YYYY"년"MM"월"DD"일")  
from emp;
```

```
select EMPLOYEE_NAME, to_char(HIRE_DATE, 'YYYY"년"MM"월"DD"일"  
")as 입사년월일  
from employees;
```

--4. 입사일 부터 지금까지의 날짜수를 출력하라.

---출력양식은 부서번호, 이름, 입사일, 현재일, 근무일수(소수점 이하 절삭), 근무 년수, 근무 월수, 근무 주수를 출력하라.

```
select deptno, ename, hiredate, sysdate,  
       floor(sysdate - hiredate) + 1 as 근무일수,  
       floor(((sysdate - hiredate) + 1) / 365) as 근무년수,  
       floor(months_between(sysdate, hiredate)) as 근무월수,  
       floor((sysdate - hiredate) / 7) as 근무주수  
from emp;
```

```
select department_id, employee_name, hire_date, sysdate,  
       floor(sysdate - hire_date) + 1 as 근무일수,  
       floor(((sysdate - hire_date) + 1) / 365) as 근무년수,  
       floor(months_between(sysdate, hire_date)) as 근무월수,  
       floor((sysdate - hire_date) / 7) as 근무주수  
from employees;
```

--5. 커미션이 NULL이 아닌 사원의 정보를 출력하라.

```
select *  
from emp  
where comm is not null;
```

```
select * from employees  
where commission is not null;
```

--6. 모든 사원의 실수령액을 계산하여 출력하라.
--- 단 급여가 많은 순으로 이름, 급여, 실수령액을 출력하라.
--- (실수령액은 급여에 대해 10%의 세금을 뺀 금액)

```
select ename, sal, sal*0.9 "실수령액"  
from emp  
order by sal desc;
```

```
select employee_name, salary, salary*0.9 "실수령액"  
from employees  
order by salary desc;
```

--7. 입사일로부터 6개월이 지난 후의 입사일, 6개월 후의 날짜, 급여를 출력하라.

```
select hiredate, add_months(hiredate, 6), add_months(sysdate, 6), sal  
from emp;
```

--8. 이름의 글자수가 6자 이상인 사원의 이름을 앞에서 3자리만 구하여 소문자로 이름만을 출력하라.

```
select lower(substr(ename,1,3))  
from emp  
where length(ename) >=6 ;
```

```
select lower(substr(employee_name,1,3))  
from employees  
where length(employee_name) >=6 ;
```

--9. 같은 업무를 하는 사원의 수가 4명 이상인 업무와 인원수를 출력하라.

```
select job, count(*)  
from emp  
group by job  
having count(*) >= 4;
```

```
select job_id, count(*)  
from employees  
group by job_id  
having count(*) >= 4;
```

--10. 각 부서별 평균 월급, 전체 월급, 최고 월급, 최저 월급 을 구하여 평균 월급이 많은 순으로 출력하라.

```
select deptno, round(avg(sal)), sum(sal), max(sal), min(sal)
from emp
group by deptno
order by avg(sal) desc;
```

```
select department_id, round(avg(salary)), sum(salary), max(salary), min(salary)
from employees
group by department_id
order by avg(salary) desc;
```

--11. 이름이 ALLEN인 사원의 부서명과 부서위치를 출력하라.

```
--natural join
select ename, dname, loc
from emp natural join dept
where ename = 'ALLEN';
```

```
select employee_name, department_id, loc
from employees natural join dept
where employee_name = 'Nancy';
```



CREATE TABLE DEPT

**(DEPTNO NUMBER(2) CONSTRAINT PK_DEPT PRIMARY KEY, --부서번호
DNAME VARCHAR2(14), --부서이름
LOC VARCHAR2(13)); --부서위치**

INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');

INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');

INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');

INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

--12. 회사내의 최소급여와 최대급여의 차이를 구하라

```
select max(sal) - min(sal) "최대급여-최소급여"  
from emp;
```

```
select max(salary) - min(salary) "최대급여-최소급여"  
from employees;
```

--13. JOB과 그 JOB에 속한 사원수를 출력하라.

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

```
select job_id, count(*)  
from employees  
group by job_id;
```


--14. 이름에 "M"자가 들어간 사원들의 이름,부서명,급여를 구하라

```
select ename, dname, sal  
from emp e, dept d  
where e.deptno = d.deptno  
and ename like 'M%';
```

```
select ename, dname, sal  
from emp natural join dept  
where ename like '%M%';
```

```
select ename, dname, sal  
from emp join dept using(deptno)  
where ename like 'M%';
```

```
select employee_name, dname, salary  
from employees e, dept d  
where e.department_id = d.deptno  
and employee_name like 'A%';
```

--15. SCOTT의 급여에서 1000 을 뺀 급여보다 적게 받는 사원의 이름, 급여를 출력하라.

```
select ename, sal
from emp
where sal <= ( select sal-1000
               from emp
               where ename like 'SCOTT');
```

```
select employee_name, salary
from employees
where salary <= ( select salary - 1000
                  from employees
                  where employee_name like 'Nancy');
```

--16. JOB이 MANAGER인 사원들 중 최소급여를 받는 사원보다 급여가 적은 사원의 이름, 급여를 출력하라.

```
select ename, sal
from emp
where sal < (select min(sal)
             from emp
             where job = 'MANAGER');
```

```
select employee_name, salary
from employees
where salary < (select min(salary)
                from employees
                where job_id = 'ITProg');
```

--17. WARD가 소속된 부서 직원들의 평균 급여보다 급여가 높은 직원의 이름 ,
급여를 출력하라.

```
select ename, sal
from emp
where sal >= (select avg(sal)
              from emp
              where ename= 'WARD'
              group by deptno);
```

```
select employee_name, salary
from employees
where salary >= (select avg(salary)
                 from employees
                 where employee_name = 'Nancy'
                 group by department_id);
```

3장을 마치며.....

Q & A