5장 Select 문

강사 금영석

CONTENT

001 SELECT 문의 기본 문법

❖SELECT 문의 기본 문법

- SELECT 문은 항상 SELECT 절과 FROM 절을 같이 작성한다.
- 다른 조건이 추가로 필요하면 WHERE 절에 기술하고 데이터가 출력되는 순서를 정하려면 ORDER BY 절을 기술한다.

001 SELECT 문의 기본 문법

SELECT [DISTINCT] [열 이름] [or 별칭(alias)]
FROM [테이블 이름]
[WHERE 조건식]
[ORDER BY 열 이름 [ASC or DESC];

- 열 이름 : 필수로 입력해야 하는 항목
- SELECT : 열을 선택하기 위한 명령어
- 대괄호([]) 안에 들어 있는 항목은 선택 사항으로 생략이 가능
- 문장을 모두 작성했다면 문장이 끝났다는 의미로 세미콜론(;)을 입력해야 한다. SQL 문장이 하나뿐이면 세미콜론을 입력하지 않아도 SQL문이 실행된다.

001 SELECT 문의 기본 문법

- ❖필수 입력 항목만 입력하고 SQL문을 실행하면 결과를 확인 할 수 있다.
- ❖대괄호 안에 들어 있는 선택 사항을 추가로 작성하면 더 자세한 결과를 얻을 수 있음
- ❖SELECT 절은 열을 선택하는 역할을 하고, FROM 절은 데이터를 가져올 테이블을 지정하는 역할을 한다.

001 SELECT 문의 기본 문법

❖SQL 문 작성 규칙

- SQL 문은 대문자와 소문자를 구별하지 않는다
- SQL 문은 한 줄 또는 여러 줄로 작성할 수 있다.
  - ➢가독성과 편집의 용이성을 위해 내용이 달라지면 줄을 내릴 수 있다.
- 코드 수준에 따른 들여쓰기는 SQL 문장을 좀 더 읽기 쉽게 한다.
- 명령어를 대문자로 작성하고 나머지를 소문자로 작성하면 가독성이 좋아진다.
- 표준 관리와 가독성 면에서는 가능하면 규칙을 준수하는 것이 좋다.

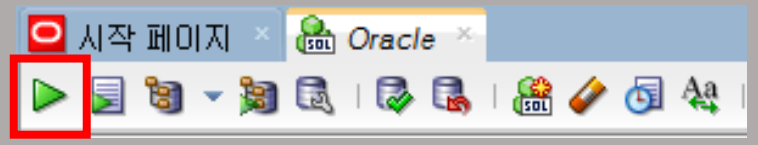
001 SELECT 문의 기본 문법

❖전체 데이터 조회하기

- 데이터 값을 실제로 조작(manipulation)하여 반영하는 명령어는 뒤에 자세히 다루는 DML 이다.
- 테이블의 전체 데이터를 조회 하는 것 부터 살펴 보자

SELECT * FROM [테이블 이름]

- * 은 모든 열을 의미한다.
- FROM을 테이블이름의 테이블로 부터 데이터를 가져오라는 의미다.
- 테이블 이름은 조회하고자하는 테이블이름을 넣으면 된다.
- SQL Developer 에서 실행은 Ctrl + Enter 를 하면 된다.



또는 실행 버튼을 누르면 된다.

001 SELECT 문의 기본 문법

❖원하는 열만 조회하고 정렬하기

- SQL 문을 사용해 특정 열만 조회 할 수 있다.

SELECT [열 이름] FROM [테이블 이름]

- 열 이름 : 조회하고자 하는 열이름을 넣으면 된다.
- 테이블 이름 : 데이터를 조회할 테이블 이름을 넣으면 된다.

```
SELECT dept_name FROM department;
```

	DEPT_NAME
1	컴퓨터 과학부
2	전기전자공학부
3	데이터 싸이언스

001 SELECT 문의 기본 문법

❖원하는 열만 조회하고 정렬하기

- SELECT 명령문 뒤에 나열한 열 이름 순선대로 결과가 조회 된다.
- 열 이름은 쉼표(,) 를 붙여 계속해서 나열할 수 있으며 결과는 나열한 순서대로 출력 된다.
- 모든 열을 조회하는 * 대신 열 이름을 모두 나열 해도 같은 결과가 출력 된다.

SELECT * FROM [테이블이름] ORDER BY [열 이름] [ASC 또는 DESC]

- ORDER BY 명령문은 정렬 순서를 지정할 수 있다.
- ASC 는 오름차순, DESC 는 내림차순으로 정렬 된다.
- 열이름은 정렬할 열이름을 넣으면 된다.(쉼표를 넣어서 여러 개 넣을 수 있다)

001 SELECT 문의 기본 문법

```
SELECT dept_name FROM department ORDER BY dept_name ASC;
```

	DEPT_NAME
1	데이터 싸이언스
2	전기전자공학부
3	컴퓨터 과학부

```
SELECT dept_name FROM department ORDER BY dept_name DESC;
```

	DEPT_NAME
1	컴퓨터 과학부
2	전기전자공학부
3	데이터 싸이언스

001 SELECT 문의 기본 문법

❖중복된 출력 값 제거 하기

- SQL 연산이나 보고서를 작성할 때 데이터 값의 행이 중복되었다면 중복된 데이터를 제거하고 출력해야 한다.
- 이럴 때 사용하는 것이 DISTINCT 명령어이다.
- DISTINCT 명령어 뒤에 열 이름을 계속 나열 하면 나열한 순서대로 DISTINCT 가 모두 적용되므로 유의해야 한다.

SELECT DISTINCT [열 이름] FROM [테이블 이름]

- 현재는 중복된 데이터가 없기 때문에 추가적으로 데이터를 입력한다.

001 SELECT 문의 기본 문법

```
SELECT year FROM student;
```

	YEAR
1	3
2	3
3	4

```
SELECT DISTINCT year FROM student;
```

	YEAR
1	4
2	3

001 SELECT 문의 기본 문법

❖ 별칭 사용하기

- SELECT 문의 결과를 출력 할 때 일반적으로 열 이름은 테이블을 정의 할 때 명명한 열의 제목을 출력한다.
- 원래 명명된 열 이름 외에 열 이름으로 임의로 바꿔 쓰고자 할 때 사용하는 것이 별칭(Alias) 이다.

- 열 이름을 변경하려면 AS 접속사를 사용한다.
- 단, SELECT 문에 기술 할 때는 AS 접속사를 생략하고 바로 별칭을 기술할 수도 있다.
- 가독성이 필요한 경우에는 AS 접속사를 사용하기를 권장한다.

001 SELECT 문의 기본 문법

SELECT [열 이름] AS [별칭], [열 이름] AS [별칭]
FROM [테이블 이름]

- ‘,’ 를 이용해서 여러 개의 별칭을 지정할 수 있다.
- 별칭은 열 이름을 임시로 변경하는데 사용한다.
- 별칭에 공백, 특수문자, 대소문자 등을 사용하려면 “(큰 따옴표)로 묶어서 사용한다.

001 SELECT 문의 기본 문법

```
SELECT dept_id AS 순번,
dept_name AS 학과,
office AS 강의실
FROM department;
```

	순번	학과	강의실
1	1	컴퓨터 과학부	302호
2	2	전기전자공학부	403호
3	3	데이터 싸이언스	303호

001 SELECT 문의 기본 문법

❖데이터 값 연결하기

- 각 열에 따로 담겨 있는 데이터 값을 하나로 붙이거나 추가 수식을 붙여 출력하는 경우가 있다.
- 이럴 때 사용하는 연결 연산자는 || 이다.
- 연결 연산자 || 를 사용하면 각 열의 결과를 연결해 하나의 열로 결과를 표현 할 수 있고, 문자열을 추가해 새로운 데이터를 표현하는 열을 만들 수도 있다.
-

SELECT [열 이름] || [열 이름], [문자] || [열 이름], [열 이름] || [문자]
FROM [테이블 이름]

001 SELECT 문의 기본 문법

```
SELECT dept_name || '의 강의장은 '
|| office || '입니다' FROM department;
```

	DEPT_NAME||'의강의장은'||OFFICE||'입니다'
1	컴퓨터 과학부의 강의장은 302호입니다
2	전기전자공학부의 강의장은 403호입니다
3	데이터 싸이언스의 강의장은 303호입니다

001 SELECT 문의 기본 문법

❖데이터 값 계산하기

- 데이터를 사전에 가공하거나 리포트를 작성할 목적으로 데이터 값끼리 계산하려면 산술 연산자를 사용해야 한다.
- 산술 연산자를 이용하여 데이터의 값을 계산할 수 있다.
- 연산의 우선 순위는 (), *, /, +, - 순이다.
- () (괄호)를 이용하면 연산의 우선순위를 지정할 수 있다.
- 새로운 열을 만들거나 데이터베이스에 추가되는 것이 아니다.
- 산술 연산자는 실무에서 많이 쓰인다. 급여, 매출, 날짜 계산에 사용된다.

SELECT [열 이름]+[숫자], [열 이름]-[숫자], [열 이름]*[숫자],

[열 이름]/[숫자] FROM [테이블 이름]

001 SELECT 문의 기본 문법

```
SELECT dept_id + 10 AS 더하기,
dept_id - 10 AS 빼기,
dept_id * 10 AS 곱하기,
dept_id / 10 AS 나누기
FROM department;
```

	더하기	빼기	곱하기	나누기
1	11	-9	10	0.1
2	12	-8	20	0.2
3	13	-7	30	0.3

002 WHERE 절을 활용한 데이터 검색

❖WHERE 을 활용한 데이터 검색

- 특정 데이터 값을 조회하거나 비교하여 연산 처리 하는 방법이다.
- 사용자가 원하는 데이터를 조회할 때 사용하는 것이 WHERE 절이다.
- WHERE 절은 조건을 지정해 데이터 값을 어디에서 어떻게 가져올 지를 정할 수 있다.
- WHERE 절은 FROM 절 다음에 기술하여 수행될 조건식을 포함한다.
- 수행할 조건 절에는 비교 연산자, SQL 연산자, 논리 연산자, 열 이름, 표현식, 숫자, 문자 등을 포함시킬 수 있다.

002 WHERE 절을 활용한 데이터 검색

❖WHERE 절 형식

```
SELECT [열 이름]
FROM [테이블 이름]
WHERE [조건식];
```

002 WHERE 절을 활용한 데이터 검색

- WHERE 절에는 연산자를 같이 쓸 수 있는데, 연산자는 의미 그대로 데이터 값을 조작하는데 사용된다.
- 복잡한 조건을 만족하는 SQL 문을 작성하려면 다양한 연산자를 사용한다.
- 연산자는 크게 비교 연산자, SQL 연산자, 논리 연산자로 구분할 수 있다.
- 연산자의 우선 순위는 괄호 > 부정 연산 > 비교 연산 > SQL 연산순이고 논리 연산자는 NOT, AND, OR 순으로 처리 된다.

연산자 종류	설명	예시
비교 연산자	조건을 비교	=, <, > 등
SQL 연산자	조건 비교를 확장	BETWEEN, IN 등
논리 연산자	조건 논리를 연결	AND, OR 등

002 WHERE 절을 활용한 데이터 검색

❖비교 연산자

구분	연산자	의미
비교 연산자	=	같다
	<>(!=)	같지 않다.
	>	보다 크다.
	>=	보다 크거나 같다.
	<	보다 작다.
	<=	보다 작거나 같다.

002 WHERE 절을 활용한 데이터 검색

- = 연산자
  - 특정 데이터 값을 선택할 때 주로 사용하는 연산자로 '같다'는 의미를 가진 연산자이다.
  - 가장 많이 사용할 연산자 중 하나이다.
  - 열에 들어가 있는 데이터 중 문자는 대소문자를 구분한다.
  - id에 들어 있는 'hong' 과 'HONG' 또는 'Hong' 은 다른 데이터로 인식 한다.

SELECT [열 이름] FROM [테이블 이름] WHRE [열 이름] = [조건]

002 WHERE 절을 활용한 데이터 검색

```
SELECT * FROM student WHERE stu_id = 1292001;
```

	STU_ID	RESIDENT_ID	YEAR	DEPT_ID
1	1292001	900424	3	1

002 WHERE 절을 활용한 데이터 검색

- >= 연산자
  - >= 연산자는 '크거나 같을 경우', ~ 이상 이라는 의미를 가진다.

SELECT [열 이름] FROM [테이블 이름] WHRE [열 이름] >= [조건]

```
SELECT * FROM student WHERE stu_id >= 1292002;
```

	STU_ID	RESIDENT_ID	YEAR	DEPT_ID
1	1292002	900305	3	2
2	1292003	991021	4	3

002 WHERE 절을 활용한 데이터 검색

❖SQL 연산자

- SQL 연산자는 비교 연산자보다 조금 더 확장된 연산자로 자주 쓰는 연산자 이다.

구분	연산자	의미
SQL 연산자	BETWEEN a AND b	A와 b 사이에 값이 있다. (a, b 값 포함)
	In(list)	List 중 어느 값이라도 일치한다.
	LIKE '비교 문자 '	비교 문자와 형태가 일치한다. (%, _ 사용)
	IS NULL	Null 값을 갖는다.

002 WHERE 절을 활용한 데이터 검색

❖BETWEEN 연산자

- BETWEEN 연산자는 두 값의 범위에 해당하는 행을 출력할 때 사용한다.
- a 이상 b 이하의 값을 조회하라는 의미로 >= 와 <= 연산자를 함께 사용한 것과 같은 의미를 가진다.
- a에 작은 값을 기술하고 b에 큰 값을 기술한다.

```
SELECT [열 이름] FROM [테이블 이름]
WHERE [열 이름] BETWEEN [작은 값] AND [큰 값];
```

002 WHERE 절을 활용한 데이터 검색

```
SELECT * FROM department
WHERE dept_id  BETWEEN 2 AND 3;
```

	DEPT_ID	DEPT_NAME	OFFICE
1	2	전기전자공학부	403호
2	3	데이터 싸이언스	303호

002 WHERE 절을 활용한 데이터 검색

❖IN 연산자

- 조회하고자 하는 데이터 값이 여러 개일 때 사용한다.
- = 연산자와 같은 의미이지만 = 연산자는 조회 조건으로 데이터 값을 하나만 지정할 수 있지만 IN 연산자는 데이터 값을 여러 개를 지정할 수 있다.
- 이러한 연산자를 다중 행 연산자라고도 부른다.

```
SELECT [열 이름] FROM [테이블 이름]
WHERE [열 이름] IN (조건 값, 조건 값, 조건 값);
```

002 WHERE 절을 활용한 데이터 검색

```
SELECT * FROM department
WHERE dept_id = 2 OR dept_id = 3;
```

	DEPT_ID	DEPT_NAME	OFFICE
1	2	전기전자공학부	403호
2	3	데이터 싸이언스	303호

002 WHERE 절을 활용한 데이터 검색

```
SELECT * FROM department
WHERE dept_id IN (2,3);
```

	DEPT_ID	DEPT_NAME	OFFICE
1	2	전기전자공학부	403호
2	3	데이터 싸이언스	303호

002 WHERE 절을 활용한 데이터 검색

❖LIKE 연산자

- LIKE 연산자는 조회 조건 값이 명확하지 않을 때 사용한다.
- LIKE 연산자는 '~와 같다' 라는 의미를 가진다.
- LIKE 연산자는 %와 _ 같은 기호 연산자를 함께 사용한다.
- 조건에는 문자나 숫자를 포함할 수 있다.
- %는 '모든 문자' 라는 의미고, _ 는 '한 글자' 를 의미한다.

SELECT [열 이름] FROM [테이블 이름]

WHERE [열 이름] LIKE [조건]

002 WHERE 절을 활용한 데이터 검색

```
SELECT * FROM department WHERE dept_name like '전%';
SELECT * FROM department WHERE dept_name like '%전%';
SELECT * FROM department WHERE dept_name like '%전';
SELECT * FROM department WHERE dept_name like '전_____';
```

002 WHERE 절을 활용한 데이터 검색

❖IS NULL 연산자

- IS NULL 연산자는 데이터 값이 null 인 경우를 조회하고자 할 때 사용한다.
- null 은 값이 지정되지 않았기 때문에 값이 없어 알 수 없는 값을 말한다.
- null은 0이나 공백과는 엄연히 다르다.
- 0은 숫자 값이고 공백은 문자 값이므로 다른 유형의 데이터 값이다.

SELECT [열 이름] FROM [테이블 이름]

WHERE [열 이름] IS NULL;

```
SELECT * FROM dual WHERE abc IS NULL;
```

002 WHERE 절을 활용한 데이터 검색

❖논리 연산자

- 논리 연산자는 여러 조건을 논리적으로 연결할 때 사용하는 연산자이다.
- SQL 문의 조건을 계속 추가해야 할 때 필수 연산자 이다.

구분	연산자	의미
논리 연산자	AND	기술 순서로 봤을 때 앞의 조건과 뒤의 조건이 동시에 참이어야 참이다. 즉, 앞뒤 조건을 동시에 만족해야 한다.
	OR	앞의 조건이 참이거나 뒤의 조건이 참인 경우, 즉 한쪽이라도 참이면 참이다.
	NOT	뒤의 조건에 대해 반대 결과를 반환한다.

002 WHERE 절을 활용한 데이터 검색

❖AND 연산자

- 여러 개의 조건이 모두 참인 데이터를 조회하고자 할 때 사용한다.

SELECT [열 이름] FROM [테이블 이름]
WHERE [열 이름] [연산자] [조건 값]
AND [열 이름] [연산자] [조건 값]

```
SELECT * FROM department
WHERE dept_id = 2 AND office = '403호';
```

	DEPT_ID	DEPT_NAME	OFFICE
1	2	전기전자공학부	403호

002 WHERE 절을 활용한 데이터 검색

❖OR 연산자

- OR 연산자는 조회하고자 하는 데이터를 연결 할 때 사용한다.
- 대용량 데이터베이스에서 조회할 때 성능에 영향을 많이 주기 때문에 사용에 주의해야 한다.

SELECT [열 이름] FROM [테이블 이름]

WHERE [열 이름] [연산자] [조건 값]

OR [열 이름] [연산자] [조건 값]

```
SELECT * FROM department
WHERE dept_id = 2 OR office = '303호';
```

	DEPT_ID	DEPT_NAME	OFFICE
1	2	전기전자공학부	403호
2	3	데이터 싸이언스	303호

002 WHERE 절을 활용한 데이터 검색

❖NOT 연산자

- NOT 연산자는 조건을 부정으로 만드는 역할을 한다.

구분	연산자	의미
부정 비교	!=	같지 않다.
	<>	같지 않다.(ISO 표준)
	NOT [열 이름] =	~와 같지 않다.
	NOT [열 이름] >	~보다 크지 않다.
부정 SQL	NOT BETWEEN a AND b	a와 b 사이에 값이 없다.
	NOT IN (list)	list 값과 일치하지 않는다.
	IS NOT NULL	null 값을 갖지 않는다.

002 WHERE 절을 활용한 데이터 검색

```
SELECT * FROM department
WHERE NOT dept_id  BETWEEN 2 AND 3;
```

	DEPT_ID	DEPT_NAME	OFFICE
1	1	컴퓨터 과학부	302호

003 실습

❖모든 레코드 조회

- * 은 모든 레코드를 의미한다.

```
SELECT * FROM department;
```

	DEPT_ID	DEPT_NAME	OFFICE
1	1	컴퓨터 과학부	302호
2	2	전기전자공학부	403호
3	3	데이터 싸이언스	303호

003 실습

❖조건을 이용한 레코드 조회

- WHERE 조건절을 이용하여 특정 레코드를 출력할 수 있다.

```
SELECT * FROM department
WHERE dept_id = 3;
```

	DEPT_ID	DEPT_NAME	OFFICE
1	3	데이터 싸이언스	303호

003 실습

❖비교연산을 이용한 레코드 조회

```
SELECT * FROM department
WHERE dept_id < 3;
```

	DEPT_ID	DEPT_NAME	OFFICE
1	1	컴퓨터 과학부	302호
2	2	전기전자공학부	403호

003 실습

❖student 테이블 레코드 추가

```
INSERT INTO student VALUES('1292001', '900424', 3, 1);
INSERT INTO student VALUES('1292002', '900305', 3, 2);
INSERT INTO student VALUES('1292003', '991021', 1, 3);
INSERT INTO student VALUES('1292004', '930504', 4, 1);
INSERT INTO student VALUES('1292005', '970105', 2, 2);
INSERT INTO student VALUES('1292006', '961101', 2, 3);
```

003 실습

```
INSERT INTO student VALUES('1292007', '920214', 3, 1);
INSERT INTO student VALUES('1292008', '960305', 2, 2);
INSERT INTO student VALUES('1292009', '931224', 4, 3);
INSERT INTO student VALUES('1292010', '980824', 1, 1);
INSERT INTO student VALUES('1292011', '970625', 1, 2);
INSERT INTO student VALUES('1292012', '940721', 2, 3);
```

003 실습

❖2학년 이상인 학생들의 학년 출력

```
SELECT year FROM student
WHERE year > 2;
```

	YEAR
1	3
2	3
3	4
4	4
5	3
6	4

003 실습

❖2학년 이상인 학년 출력
- 중복 데이터 제거

```
SELECT DISTINCT year
FROM student WHERE year > 2;
```

	YEAR
1	4
2	3

003 실습

❖4학년 미만인 학생들의 학년을 1 더한 값 출력

```
SELECT stu_id, year+1
FROM student
WHERE year < 4;
```

	STU_ID	YEAR+1
1	1292001	4
2	1292002	4
3	1292005	3
4	1292006	3
5	1292007	4
6	1292008	3
7	1292010	2
8	1292011	2
9	1292012	3

004 QUIZ

❖아래와 같은 결과값 출력

	STU_ID	RESIDENT_ID	YEAR	DEPT_ID
1	1292002	900305	3	2
2	1292005	970105	2	2
3	1292008	960305	2	2
4	1292011	970625	1	2

004 QUIZ

❖아래와 같은 결과값 출력

	STU_ID	RESIDENT_ID	YEAR	DEPT_ID
1	1292003	991021	4	3
2	1292005	970105	2	2
3	1292006	961101	2	3
4	1292008	960305	2	2
5	1292010	980824	1	1
6	1292011	970625	1	2


수고하셨습니다.