Database Part5

Cross JOIN
Inner JOIN
Outer JOIN
Sub QUERY

JOIN 문법을 보기 전에 우선 SELECT쿼리의 생략된 부분과 테이블에 별칭을 부여하는 문법을 알아야 한다.

아래는 EMP 테이블에 존재하는 직원 정보 중 사원번호, 사원명, 급여, 부서번호를 조회하는 쿼리문이다.

```
SELECT EMPNO, ENAME, SAL, DEPTNO
FROM EMP;
```

위 쿼리문에서 생략된 부분을 작성하면 다음과 같이 작성할 수 있다.

```
SELECT EMP.EMPNO, EMP.ENAME, EMP.SAL, EMP.DEPTNO
FROM EMP;
```

결국 SELECT 절의 컬럼명 앞에서 테이블. 이 생략되어 있는 것이다. 그렇기에 테이블명에 별칭을 부여하면 쿼리문은 다음과 같이 변경할 수 있다.

```
SELECT E.EMPNO, E.ENAME, E.SAL, E.DEPTNO
FROM EMP E;
```

테이블에 별칭을 부여했다고 해서 SELECT 절에 무조건 테이블별칭.컬럼명 으로 작성할 필요는 없다. 필요한 컬럼에만 사용하면 되는 것이다.

```
SELECT A.EMPNO, ENAME, SAL, A.DEPTNO
FROM EMP A;
```

위 쿼리에서 EMPNO컬럼과 DEPTNO컬럼에는 테이블에 부여한 별칭을 적용했지만, ENAME, SAL 컬럼은 컬럼명만 작성하였다.

테이블의 별칭은 자바의 변수처럼 단어를 보면 그 의미를 유추할 수 있게 작성하지 않고, 최대한 짧은 단어로 별칭을 부여한다.

JOIN(조인)

단일 테이블 조회가 아닌, 둘 이상의 테이블에서 데이터를 조회하는 것을 조인이라 한다.

조인의 종류로는 CROSS 조인, INNER 조인, OUTER 조인, SELF 조인 등이 있다.

JOIN 기본 문법

SELECT 절에는 두 테이블에서 조회하려는 모든 컬럼을 쉼표로 나열한다.

FROM 절에는 조회하려는 컬럼이 존재하는 두 테이블을 JOIN이라는 키워드로 나열한다.

SELECT 절에서 작성하려는 컬럼이 두 테이블에 모두 존재하는 컬럼이라면 반드시 해당 컬럼명 앞에 테이블. 혹은 테이블별칭. 을 붙여 구분해준다.

예를 들어 EMP테이블에서는 사번, 사원명, 부서번호를 조회하고, DEPT 테이블에서는 부선번호와 부서명을 조회하는 쿼리문은 다음과 같다.

```
SELECT EMPNO, ENAME, E.DEPTNO, D.DEPTNO, DNAME
FROM EMP E JOIN DEPT D;
```

두 테이블 중 하나의 테이블에만 존재하는 컬럼명 앞으로 구지 테이블명이나 테이블의 별칭을 붙일 필요가 없다.

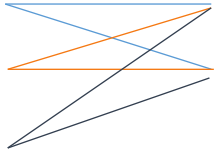
DEPTNO 컬럼은 EMP, DEPT 두 테이블에 모두 존재하는 컬럼이므로 반드시 컬럼명 앞에 테이블명 혹은 테이블의 별칭을 붙여 사용해야 한다.

이렇게 FROM절에는 조회하려는 모든 컬럼이 존재하는 테이블명을 나열하고, SELECT 절에는 조회하려는 모든 컬럼을 나열한 것을

Cross JOIN이라 한다.

Cross JOIN(크로스 조인) : 두 테이블에서 가능한 모든 조합을 조회하는 쿼리. 조회되는 데이터가 유의미하지 않아, 잘 사용하지 않음

사원번호	사원명	부서번호
101	홍길동	10
102	이순신	20
103	유관순	10



부서번호	부서명
10	개발부
20	인사부

```
SELECT EMPNO, ENAME, E.DEPTNO, D.DEPTNO, DNAME
FROM EMP E JOIN DEPT D;
```

* 두 테이블에 동일한 컬럼이 존재한다면 쿼리 작성 시 주의해야 한다.

홍길동 사원의 부서가 10번, 20번이라고 조회되지만 이 중 하나는 잘못된 데이터다.

사원번호	사원명	부서번호1	부서번호2	부서명
101	홍길동	10	10	개발부
101	홍길동	10	20	인사부
102	이순신	20	10	개발부
102	이순신	20	20	인사부
103	유관순	10	10	개발부
103	유관순	10	20	인사부

Cross JOIN(크로스 조인) 실행 결과

크로스 조인은 개념상으로 조인의 한 종류로 보지만, 크로스 조인의 결과는 두 테이블에서 가능한 조합의 모든 경우를 조회하는 것이 그치기 때문에 사실상 사용할 일이 없다. 하지만 가장 많이 사용하는 Inner JOIN은 크로스 조인 문법에서 추가적으로 한 줄만 더 작성하면 되기 때문에 문법적으로 알아본 것이다.

Cross JOIN에서 Inner JOIN으로의 변경

이전 슬라이드에서 사용한 Cross JOIN을 다시 살펴보자.

```
SELECT EMPNO, ENAME, E.DEPTNO, D.DEPTNO, DNAME
FROM EMP E JOIN DEPT D;
```

사원번호	사원명	부서번호1	부서번호2	부서명
101	홍길동	10	10	개발부
101	홍길동	10	20	인사부
102	이순신	20	10	개발부
102	이순신	20	20	인사부
103	유관순	10	10	개발부
103	유관순	10	20	인사부

쿼리문 실행 결과 조회된 데이터를 보면 똑같은 사원명이 여러번 조회된 것을 확인할 수 있다. EMP 테이블에서 확인할 수 있듯, 사원명이 중복인 직원은 없기 때문에 여러 행 조회된 사원명 중 한 행만이 정확한 데이터이며, 나머지 데이터는 잘못된 데이터임을 판단할 수 있다.

그럼 사원명이 중복으로 조회된 여러 데이터 중 어떤 행의 데이터가 맞는 데이터일까? EMP 테이블과 DEPT 테이블에서 각각 조회한 DEPTNO 컬럼의 데이터를 보면 확인할 수 있다. 두 테이블에서 조회한 DEPTNO가 애초에 서로 다를 수는 없다.

예를 들어, 사원명이 홍길동인 사원의 부서번호가 EMP 테이블에서는 10번으로 조회되고, DEPT 테이블에서는 20번으로 조회됐다면, 두 테이블에서 조회된 부서번호 중 하나는 잘못된 것이라 판단할 수 있다는 것이다.

결국 위 Cross JOIN의 쿼리 결과로 조회된 모든 데이터 중 EMP테이블에서의 DEPTNO컬럼과 DEPT테이블의 DEPTNO컬럼이 같은 값을 가진 행 정보가 맞는 데이터가 되는 것이다.

앞선 슬라이드의 내용을 Cross JOIN에 적용하면 다음과 같은 쿼리문이 나온다.

```
SELECT EMPNO, ENAME, E.DEPTNO, D.DEPTNO, DNAME
FROM EMP E JOIN DEPT D
WHERE E.DEPTNO = D.DEPTNO;
```

위 쿼리는 Cross JOIN 쿼리문에서 EMP테이블의 DEPTNO와 DEPT테이블의 DEPTNO 컬럼의 값이 같은 것만 조회하겠다는 조건을 추가한 것이다.

Inner JOIN의 최종 형태

위 쿼리문을 보면 Cross JOIN 문법에 WHERE 절을 작성하여 올바른 데이터만을 조회하는 조건을 추가한 것이라는 걸 알 수 있다.

이렇듯 Cross JOIN 결과에서 올바른 데이터만을 조회하기 위해 작성한 조건절을 JOIN 조건절이라 부른다. 그리고 이러한 JOIN 조건절은 다른 조건문과 쉽게 구별하기 위해서 JOIN 조건절만을 위한 별도의 문법을 추가하였다. JOIN 조건절은 WHERE절에 작성하기 않고, 다음과 같이 ON절에 작성한다.

```
SELECT EMPNO, ENAME, E.DEPTNO, D.DEPTNO, DNAME
FROM EMP E  [INNER] JOIN DEPT D                    # INNER 키워드는 작성하거나, 생략 가능
ON E.DEPTNO = D.DEPTNO;
```

JOIN 문법을 처음 학습하면 JOIN 조건절에 어떤 내용을 작성해야 할지 프로그래밍처럼 감이 잘 오지 않을 수 있다. 프로그래밍과 마찬가지로 JOIN 쿼리문을 계속 작성하며 경험치를 쌓고, 이해력을 키워야 한다. 익숙하지 않을 때, JOIN조건절은 다음과 같이 작성하면 90%정도 맞을 것이다.

JOIN 조건절 : JOIN하려는 두 테이블이 공통으로 지닌 컬럼의 값이 같은 것만 조회하는 조건을 작성한다.

다음은 EMP 테이블과 DEPT 테이블의 데이터를 동시에 조회하는 몇 가지 Inner join 예시이다.

직급이 '사원'이 아닌 직원의 사번, 사원명, 직급, 부서번호, 부서명 조회

```
SELECT EMPNO, ENAME, JOB, EMP.DEPTNO, DNAME          #EMP.DEPTN와 DEPT.DEPTNO 둘 다 가능.
FROM EMP INNER JOIN DEPT
ON EMP.DEPTNO = EMP.DEPTNO
WHERE JOB <> '사원';
```

부서번호가 10, 20번인 직원의 사번, 사원명, 부서명, 부서지역 조회

```
SELECT EMPNO, ENAME, DNAME, LOC
FROM EMP INNER JOIN DEPT
ON EMP.DEPTNO = EMP.DEPTNO
WHERE EMP.DEPTNO IN (10, 20);          #WHERE 조건절에서도 두 테이블이 공통으로 지닌 컬럼명은 앞에 테이블. 붙임!
```

다음과 같은 두 테이블이 존재할 때, 이너 조인 쿼리문을 살펴보자

회원ID (M_ID)	회원명 (M_NAME)	성별 (GENDER)
aaa	신사임당	여
bbb	이순신	남
ccc	유관순	여

MEMBER(회원 테이블)

주문번호 (ORDER_NUM)	상품명 (I_NAME)	수량 (O_CNT)	주문일자 (O_DATE)	회원ID (M_ID)
1	체크셔츠	2	2025.01.10	aaa
2	청바지	1	2025.01.05	bbb
3	맨투맨	1	2015.01.02	aaa

ORDER_INFO(주문 정보 테이블)

```
SELECT M.M_ID, M_NAME,  ORDER_NUM, I_NAME, O_CNT, O.M_ID
FROM MEMBER M INNER JOIN ORDER_INFO O
ON M.M_ID = O.M_ID;
```

위 쿼리문은 회원 테이블에서는 회원ID, 회원명을 조회하고, 주문정보 테이블에서는 주문번호, 주문상품명, 주문수량, 회원ID를 조회한 쿼리이다.

두 테이블 모두에 M_ID 컬럼이 존재하므로, SELECT와 WHERE 절에 사용 시 컬럼명 앞에 반드시 테이블명. 혹은 테이블별칭. 을 붙여야 한다.

두 테이블이 공통으로 지닌 컬럼이 M_ID 컬럼이므로 해당 컬럼이 같은 값만 조회하는 조건을 조인조건에 작성하였다.

앞선 슬라이드의 조회 결과는 다음과 같다.

```
SELECT M.M_ID, M_NAME,  ORDER_NUM, I_NAME, O_CNT, O.M_ID
FROM MEMBER M INNER JOIN ORDER_INFO O
ON M.M_ID = O.M_ID;
```

M_ID 1	M_NAME	ORDER_NUM	I_NAME	O_CNT	M_ID 2
aaa	신사임당	1	체크셔츠	2	aaa
aaa	신사임당	3	맨투맨	1	aaa
bbb	이순신	2	청바지	1	bbb

쿼리 결과를 보면 알겠지만 회원ID가 aaa인 회원은 주문을 두 번했기 때문에 데이터가 2건 조회되었다.

반면, 회원ID가 ccc인 회원이 MEMBER 테이블에는 존재하지만, 주문한 데이터가 없기 때문에 조인 쿼리의 결과로는 나오지 않은 것을 확인 할 수 있다.

JOIN조건에서 두 테이블의 M_ID가 같은 값만 조회한다는 조건을 넣었으며, 회원 테이블에는 회원ID가 ccc인 회원이 존재하지만, 주문 정보 테이블에서는 ccc라는 회원ID가 존재하지 않기 때문에 이러한 조회 결과는 당연한 것이다.

이런 상태에서 만약 요구사항이 다음과 같다면 어떨지 생각해보자.

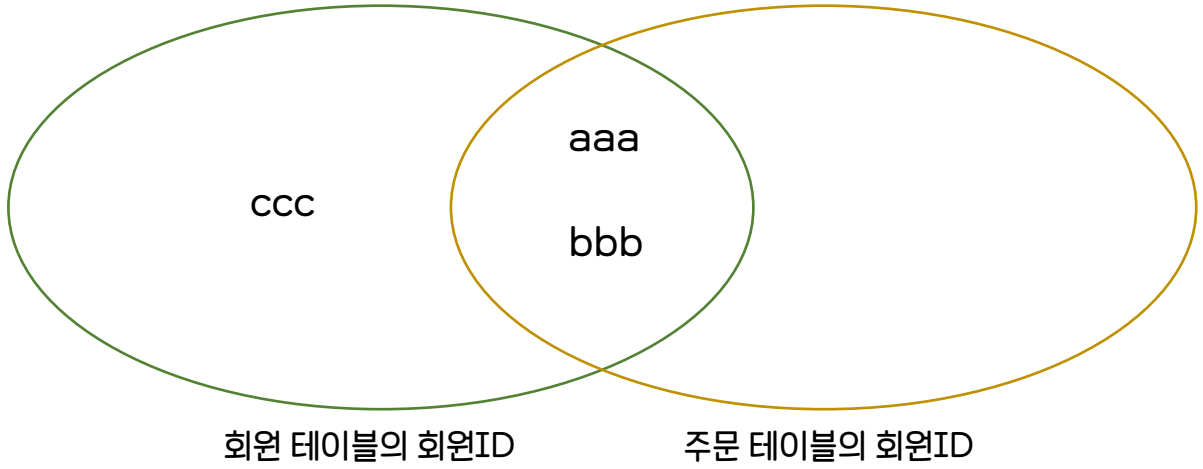
모든 회원들의 주문정보를 조회하고 싶다. 주문한 이력이 없는 회원들도 모두 조회되어야한다.

이러한 요구사항은 생각보다 빈번하게 일어난다. 이 요구사항을 충족하기 위해 사용하는 쿼리문이 Outer JOIN이다.

이너 조인과 아우터 조인은 집합관계로 생각하면 차이점을 알 수 있다. 다시 회원과 주문정보 테이블의 이너 조인 쿼리문을 보자.

```
SELECT M.M_ID, M_NAME,  ORDER_NUM, I_NAME, O_CNT, O.M_ID
FROM MEMBER M INNER JOIN ORDER_INFO O
ON M.M_ID = O.M_ID;
```

위 쿼리는 JOIN조건에서 두 테이블의 M_ID가 같은 값만 조회한다는 조건을 넣었으며, 이러한 이너 조인은 집합관계로는 교집합에 해당한다.

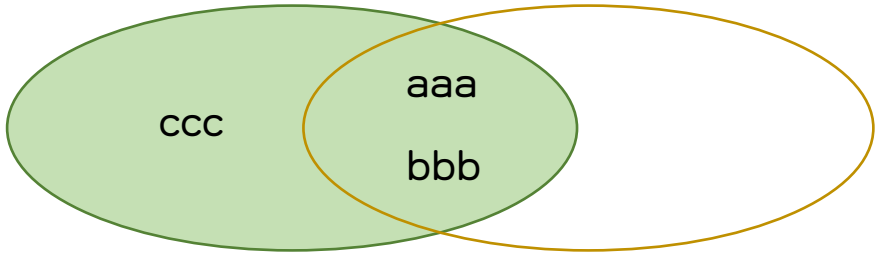


위 집합관계를 보면 회원 테이블과 주문 테이블에서 공통으로 지니는 M_ID는 aaa와 bbb 이기 때문에 이너 조인 결과 두 아이디에 대한 정보만 조회된다.

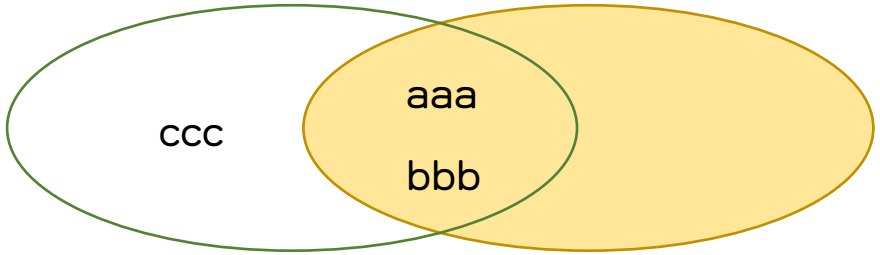
그렇기 때문에 모든 회원에 대한 주문정보를 이너 조인으로 조회하면 ccc회원은 교집합에 포함되지 않기 때문에 조회되지 않는 것이다.

모든 회원들의 주문정보를 조회하고 싶다. 주문한 이력이 없는 회원들도 모두 조회되어야한다.

결국 위 요구사항에 충족되는 쿼리문을 집합관계로 생각해보면 다음과 같을 것이다.



위 집합의 데이터를 조회하는 것을 Outer 조인이라 하며, 정확히 말하면 left outer join이라 한다.



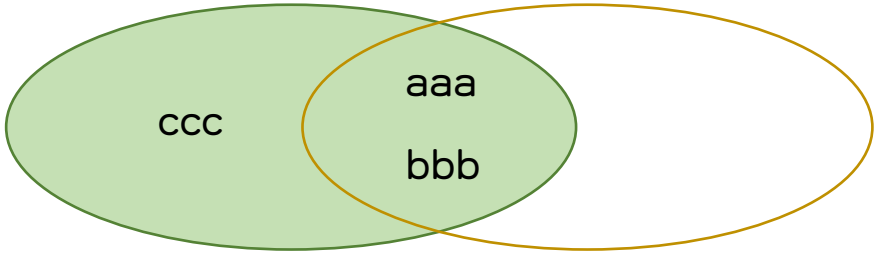
반면 위와 같은 집합관계를 표한하는 쿼리문은 right outer join이라 한다.

left와 right의 차이점은 말 그대로 왼쪽, 오른쪽을 의미한다.

조인 쿼리에서는 두 테이블 중 먼저 작성된 테이블이 left가 되고, 나중에 작성한 테이블이 right가 된다.

결론적으로, 우측의 집합관계를 요구하는 쿼리문의 작성은 다음과 같다.

```
SELECT M.M_ID, M_NAME,  ORDER_NUM, I_NAME, O_CNT, O.M_ID
FROM MEMBER M LEFT OUTER JOIN ORDER_INFO O
ON M.M_ID = O.M_ID;
```



위 쿼리의 조회 결과 데이터는 다음과 같다.

M_ID 1	M_NAME	ORDER_NUM	I_NAME	O_CNT	M_ID 2
aaa	신사임당	1	체크셔츠	2	aaa
aaa	신사임당	3	맨투맨	1	aaa
bbb	이순신	2	청바지	1	bbb
ccc	유관순	null	null	null	null

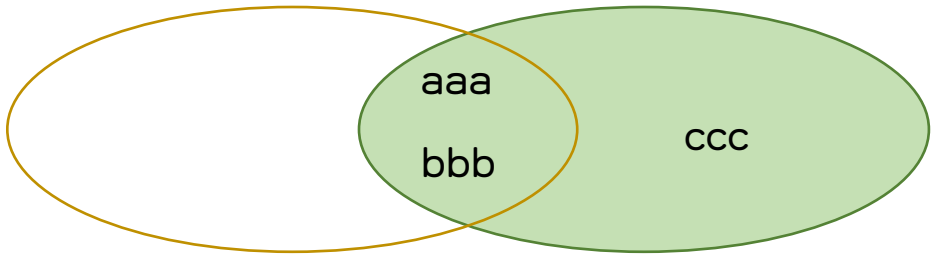
이렇게 아우터 조인은 이너 조인의 결과에서 한쪽 테이블에만 존재하는 데이터를 추가적으로 조회하는 기능을 한다.

한쪽 테이블에만 데이터가 존재하기 때문에, 데이터가 존재하지 않는 테이블의 데이터는 모두 null로 조회된다.

위 쿼리문에서 MEMBER 테이블을 우선 작성했기 때문에 MEMBER 테이블이 LEFT가 되고, ORDER_INFO 테이블을 나중에 작성했기 때문에 ORDER_INFO 테이블이 RIGHT 테이블이 된다.

앞선 아우터 쿼리문은 다음과 같이 작성해도 결국 동일한 데이터가 조회된다.

```
SELECT M.M_ID, M_NAME,  ORDER_NUM, I_NAME, O_CNT, O.M_ID
FROM ORDER_INFO O RIGHT OUTER JOIN MEMBER M
ON O.M_ID = M.M_ID;
```



Sub Query(서브 쿼리)란 하나의 쿼리문 안에 존재하는 또 다른 쿼리문을 말한다. 다음과 같은 요구사항은 어떻게 해결할 수 있을지 생각해보자.

김사랑 사원과 같은 부서에 소속된 직원들의 사번, 사원명, 직급을 조화하시오.

위 요구사항을 충족시키는 쿼리문 작성을 위해 우선 김사랑 사원이 속한 부서를 조회해보자.

```
SELECT DEPTNO FROM EMP WHERE ENAME = '김사랑';
```

위 쿼리의 실행결과 김사랑 사원은 부서번호가 20번인 부서에 소속된 직원임을 알 수 있다. 그럼 김사랑 사원과 같은 부서에 소속된 직원들의 사번, 사원명, 직급을 조회하는 쿼리문은 다음과 같을 것이다.

```
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE DEPTNO = 20;
```

우리는 요구사항을 만족하는 쿼리문을 작성하기 위해 2개의 쿼리문을 작성하였다.

첫째가 김사랑 사원의 부서번호를 조회하는 것이였고, 두번째는 첫번째 쿼리로 알게된 김사랑 사원의 부서번호를 활용한 사원들의 정보를 조회한 쿼리다.

결국, 우리는 하나의 요구사항을 해결하기 위해 2개의 쿼리문을 따로 작성하였다. 하지만, 매번 이렇게 두번씩 조회를 하는 것은 번거로운 것이며,

만약 김사랑 사원의 부서가 변경된다면 두번째 쿼리문의 WHERE절 조건도 매번 달라져야 할 것이다.

위 두 개의 쿼리문을 하나의 쿼리문으로 만들면 이러한 번거로움 및 에러 사항을 해결할 수 있다.

앞선 슬라이드의 두 쿼리를 하나의 쿼리문으로 만든 결과는 다음과 같다.

김사랑 사원과 같은 부서에 소속된 직원들의 사번, 사원명, 직급을 조화하시오.

김사랑 사원의 부서번호 조회 쿼리

```
SELECT DEPTNO
FROM EMP
WHERE ENAME = '김사랑';
```

김사랑 사원과 동일한 부서번호를 같은 사원들의 정보 조회 쿼리

```
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE DEPTNO = 20;
```

왼쪽 쿼리의 조회 결과값을 우측 쿼리에 작성한다.

우측의 쿼리가 요구사항을 충족시키기 위한 쿼리문이다.

우측의 쿼리를 보면 하나의 쿼리문 안에 다른 쿼리가 포함된 것을 알 수 있다.

이렇게 하나의 쿼리 안에 또 다른 쿼리가 들어간 것을 Sub Query(서브쿼리)라 부른다.

서브쿼리는 예제처럼 WHERE절에도 들어갈 수 있지만 SELECT절, FROM절에도 들어갈 수 있다.

서브쿼리는 안쪽의 쿼리부터 해석해야 전체 쿼리문을 해석하기 편하다.

```
SELECT EMPNO, ENAME, JOB
FROM EMP
WHERE DEPTNO = (SELECT DEPTNO
                FROM EMP
                WHERE ENAME = '김사랑');
```

* 서브쿼리는 반드시 () 안에 작성해야 한다.

* 서브쿼리는 조인쿼리로, 조인쿼리는 서브쿼리로 대부분 변경 가능하기 때문에 쿼리를 변경해가며 작성하면 빨리 익숙해질 것이다.

다음의 문제를 풀며 서브쿼리 및 조인 쿼리를 연습해보자.

1. 김사랑 사원의 사번, 사원명, 직급, 급여, 부서번호, 부서명을 조회하시오. (조인 & 서브쿼리)
2. 강혜정 사원보다 급여를 더 많이 받는 사원들의 사번, 사원명, 급여를 조회하시오. (서브쿼리)
3. 영업부에 속한 직원 중, 커미션이 NULL이 아닌 직원들의 사원명, 직급, 부서번호, 커미션을 조회하시오. (조인 & 서브쿼리)
4. 인사부에 소속된 직원들의 급여 평균보다 더 높은 급여를 받는 사원들의 사번, 사원명, 급여, 부서명을 조회하시오.(서브쿼리)
5. 직급이 부장, 차장인 사원들의 사원명, 직급 , 부서번호, 부서명을 조회하시오(조인 & 서브쿼리)