다음 내장 함수의 결과를 적으시오

ABS(-15) =15
CEIL(15.7) =16
COS(3.14159) =-1
FLOOR(15.7) =15
LOG(10,100) =2
MOD(11,4) =3
POWER(3,2) =9
ROUND(15.7) =16
SIGN(-15) =-1
TRUNCATE(15.7) =15
CHAR(67 USING uft8) =C
CONCAT('HAPPY', Birhday')
LOWER('Birthday') =birthday
LPAD('Page 1', 15, '*.') ='*.*.*.*.*Page 1'
REPLACE('JACK', 'J', 'BL') ='BLACK'
RPAD('Page 1', 15, '*') ='Page 1*********'

SUBSTAR('ABCDEFG', 3, 4) ='CDEF
TRIM(LEADING 0 FROM '00AA00') ='AA00'
UPPER('BIRTHDAY') ='BIRTHDAY'
ASCII('A') =65
LENGTH('Birthday') =8
ADDDATE('2024-02-14', INTERVAL 10 DAY) ='2024'-02-24'
LAST_DAY(STYSDATE()) ='2025-06-30'
NOW() '2025-06-06-13:45:00'
DATE_FORMAT(SYSDATE(), '%Y') ='2025'
CONCAT(123) =123
STR_TO_DATE('12 05 2024', %d %m %Y) ='2024-05-12'
CAST('12.3' AS DECIMAL(3, 1)) = 12.3
IF(1=1, 'aa', 'bb') ='aa'
IFNULL(123, 345) = 123
IFNULL(NULL, 123) =123

MyBook 테이블을 생성하고 NULL에 관한 다음 SQL 문에 답하시오 또한 질의의 결과를 보면서 NULL에 대한 개념도 정리해 보시오

```
mysql> SELECT * FROM MyBook;
+--------+-------+
| bookid | price |
+--------+-------+
|      1 | 10000 |
|      2 | 20000 |
|      3 |  NULL |
+--------+-------+
3 rows in set (0.001 sec)
```

```
mysql> SELECT bookid, IFNULL(price,0) FROM MyBook;
+--------+-----------------+
| bookid | IFNULL(price,0) |
+--------+-----------------+
|      1 |           10000 |
|      2 |           20000 |
|      3 |               0 |
+--------+-----------------+
3 rows in set (0.001 sec)
```

```
mysql> SELECT * FROM MyBook WHERE price IS NULL;
+--------+-------+
| bookid | price |
+--------+-------+
|      3 |  NULL |
+--------+-------+
1 row in set (0.001 sec)
```

```
mysql> SELECT SUM(price), AVG(price), COUNT(*) FROM MyBook WHERE bookid>=4;
+------------+------------+----------+
| SUM(price) | AVG(price) | COUNT(*) |
+------------+------------+----------+
|       NULL |       NULL |        0 |
+------------+------------+----------+
1 row in set (0.001 sec)
```

```
mysql> SELECT COUNT(*), COUNT(price) FROM MyBook;
+----------+--------------+
| COUNT(*) | COUNT(price) |
+----------+--------------+
|        3 |            2 |
+----------+--------------+
1 row in set (0.001 sec)
```

```
mysql> SELECT SUM(price), AVG(price) FROM MyBook;
+------------+------------+
| SUM(price) | AVG(price) |
+------------+------------+
|      30000 | 15000.0000 |
+------------+------------+
1 row in set (0.001 sec)
```

다음 SQL 문을 비어 있는 릴레이션 R(A; int, B: int)에 실행했을 때 결과 값을 적으시오

(SQL)

```
INSERT INTO R VALUES(NULL, 10);
INSERT INTO R VALUES(12, NULL);
INSERT INTO R VALUES(NULL, NULL);
INSERT INTO R VALUES(10, 12);
```

1) SELECT COUNT(A) FROM R; = 2

2) SELECT * FROM R WHERE A IN(12, 10, NULL); =12|NULL, 10|12

3) SELECT A, COUNT(*) FROM R GROUP BY A; =10|1, 12|1

부속질의에 관한 다음 SQL 문을 수행해보고, 어떤 질의에 대한 답인지 설명하시오

1) SELECT

```
custid,
(SELECT address FROM Customer cs WHERE cs.custid = od.custid) AS address,
SUM(saleprice) AS total
```

```
FROM
    Orders od
GROUP BY
    od.custid;
```

고객별로 주문 금액의 합을 구하고 고객의 주소도 보이게 함

```
2) SELECT cs.name s
FROM (
    SELECT custid, AVG(saleprice) s
    FROM Orders
    GROUP BY custid
) od, Customer cs
WHERE cs.custid = od.custid;
```

주문한 적이 있는 고객의 이름을 보이게 함

```
3) SELECT SUM(saleprice) AS total
FROM Orders od
WHERE EXISTS (
  SELECT * FROM Customer cs
  WHERE cs.custid <= 3 AND cs.custid = od.custid
);
```

고객번호가 3 이하인 고객이 주문한 총 주문 금액을 보이게 함

릴레이션 EMP, Dept가 다음과 같이 정의되어 있다. Emp의 deptno는 Dept의 deptno를 참조하는 외래키다. 사원이 1명도 없는 부서(deptno)를 검색하는 질의를 다양한 방법으로 작성했을 때 가장 거리가 먼 질의는 어느 것인가?

(릴레이션)

Emp(empno, ename, job, mgr, hiredate, sal, comm, deptno)

Dept(deptno, dname, loc)

```
1) SELECT deptno FROM Dept WHERE deptno NOT IN (SELECT deptno FROM Emp);
2) SELECT deptno FROM Dept A WHERE NOT EXISTS (SELECT * FROM Emp B WHERE
```

A.deptno=B.deptno);

3) SELECT B.deptno FROM EMP A RIGHT OUTER JOIN Dept B ON A.deptno = B.deptno
WHERE empno IS Null;

4) SELECT deptno FROM Dept WHERE deptno !=ANY(SELECT deptno FROM EMP);

다음 세 개의 질이는 각각 부속질의, EXISTS, 조인을 사용하며 '대한민국에 거주하면서 도서를 주문한 적이 없는 고객의 이름'을 찾는 질의이다. 세 질의의 결과가 같은지 확인하시오. 이 결과를 참조하여 '대한민국에 거주하면서 도서를 주문한 적이 없는 고객의 이름'을 세 가지 방법으로 작성하시오

1) 부속질의 사용

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%'
   AND name IN (
      SELECT name FROM Customer
      WHERE custid IN (SELECT custid FROM Orders)
);
```

2) EXISTS 사용

```
SELECT name
FROM Customer c1
WHERE address LIKE '대한민국%'
   AND EXISTS (
      SELECT name FROM Customer
      WHERE custid IN (SELECT custid FROM Orders)
         AND c1.name LIKE name
);
```

3) 조인 사용

```
SELECT c1.name
FROM Customer c1, Customer c2
WHERE c1.name = c2.name
```

```
  AND c1.address LIKE '대한민국%'
  AND c2.name IN (
    SELECT name FROM Customer
    WHERE custid IN (SELECT custid FROM Orders)
);
```

세 질의 결과는 똑같다.

대한민국에 거주하면서 도서를 주문한 적이 없는 고객의 이름

1) 부속질의 사용

```
SELECT name
FROM Customer
WHERE address LIKE '대한민국%'
  AND custid NOT IN (SELECT custid FROM Orders);
```

2) EXISTS 사용

```
SELECT name
FROM Customer c
WHERE address LIKE '대한민국%'
  AND NOT EXISTS (
    SELECT * FROM Orders o WHERE o.custid = c.custid
  );
```

3) 조인 사용

```
SELECT c.name
FROM Customer c
LEFT JOIN Orders o ON c.custid = o.custid
WHERE c.address LIKE '대한민국%'
  AND o.custid IS NULL;
```

테이블 R, S에 대하여 다음 SQL 문을 수행한 결과를 적으시오

회원번호	등급
1	1
2	2

다음 SQL 문의 실행 순서를 번호순으로 적으시오

1) SELECT deptno, COUNT(empno)
2) FROM Emp
3) WHERE sal >= 500
4) GROUP BY deptno
5) HAVING COUNT(empno) > 2
6) ORDER BY deptno;

2 -> 3 -> 4-> 5-> 1-> 6

뷰의 장점 세 가지를 설명하시오

보안 강화 – 민감한 데이터에 직접 접근하지 못하도록 제한 할 수 있다.
복잡한 쿼리 단순화 – 자주 쓰는 복잡한 조인이나 조건을 뷰로 만들어 놓으면, 이후에는 간단한 SELECT 문으로 결과를 얻을 수 있다.
논리적 독립성 제공 – 기본 테이블 구조가 바뀌어도 뷰를 유지하면 사용자 쿼리를 그대로 사용할 수 있어, 프로그램 변경을 줄일 수 있다.

릴레이션 R(A, B)에 대한 뷰(view)가 다음과 같이 정의될 때 SQL 문의 실행 결과를 적으시오

5000

마당서점 데이터베이스를 이용하여 다음에 해당하는 뷰를 작성하시오

1) 판매가격이 20,000원 이상인 도서의 도서번호, 도서이름, 고객이름, 출판사, 판매가격을 보여주는 highorders 뷰를 생성하시오

```
CREATE VIEW highorders AS
SELECT
    b.bookid,
    b.bookname,
    c.name AS customer_name,
    b.publisher,
    o.saleprice
FROM Orders o
JOIN Book b ON o.bookid = b.bookid
JOIN Customer c ON o.custid = c.custid
```

```
WHERE o.saleprice >= 20000;
```

2) 생성한 뷰를 이용하여 판매된 도서의 이름과 고객의 이름을 출력하는 SQL 문을 작성하시오

```
SELECT bookname, customer_name FROM highorders;
```

3) highorders 뷰를 변경하고자 한다. 판매가격 속성을 삭제하는 명령을 수행하시오

```
DROP VIEW IF EXISTS highorders;

CREATE VIEW highorders AS
SELECT
    b.bookid,
    b.bookname,
    c.name AS customer_name,
    b.publisher
FROM Orders o
JOIN Book b ON o.bookid = b.bookid
JOIN Customer c ON o.custid = c.custid
WHERE o.saleprice >= 20000;
```