

SQLD 출제 예상문제 - 2회
과목 1. 데이터 모델링의 이해
1-1. 1번~5번

문제1. 다음 중 데이터 독립성에서는 크게 2가지 사상(MAPPING)이 도출된다. 그에 대한 설명으로 알맞은 것은?

- ① 외부적/개념적 사상은 외부적 뷰와 개념적 뷰 간의 상호 독립성을 정의한다.
- ② 개념적/내부적 사상은 개념적 뷰와 저장된 데이터베이스의 상호 독립성을 정의한다.
- ③ 외부적/개념적 사상은 논리적 사상이라고도 한다.
- ④ 개념적/내부적 사상은 논리적 사상이라고도 한다.

문제2. 다음 중 엔터티의 명명 규칙으로 가장 부적절한 것은 무엇인가?

- ① 가능하면 현업업무에서 사용하는 용어를 사용한다.
- ② 가급적 단수 명사를 사용하되 대량의 집합인 경우 복수 명사를 허용한다.
- ③ 가능하면 약어를 사용하지 않는다.
- ④ 엔터티 생성 의미대로 이름을 부여한다.

문제3. 다음 중 속성의 특성에 따른 분류에 대한 설명 중 가장 적절한 것은 무엇인가?

- ① 업무로부터 분석한 속성이라도 이미 업무상 코드로 정의한 경우 기본 속성이 아닌 설계 속성이라고 할 수 있다.
- ② 코드성 속성은 원래 속성을 업무상 필요에 의해 변형하여 만든 파생 속성이라고 할 수 있다.
- ③ 파생 속성은 다른 속성에 영향을 받아 발생하는 속성으로써 보통 코드성 데이터가 해당한다.
- ④ 파생 속성은 데이터 정합성을 유지하기 위해 주의해야할 점이 많지만 성능 측면을 고려하여 적극 사용하는 것이 좋다.

문제4. 다음 중 관계명으로 사용하기에 가장 부적절한 것은 무엇인가?

- ① 수강신청한다
- ② 강의한다
- ③ 주문을 한다
- ④ 입금예정이다

문제5. [아래]는 식별자의 속성의 수가 많아지지 않도록 하기 위해서 인조식별자를 사용한 모습이다. 인조식별자를 사용한 경우의 장점으로 가장 옳바르지 않는 것은 무엇인가?

접수

- ☐ # 관할부서
- ☐ # 접수일자
- ☐ # 접수순번
- ☐ * 접수방법코드
- ☐ * 신청인구분코드
- ☐ * 신청인주민번호
- ☐ * 신청인명

접수_인조식별자

- ☐ # 접수번호
- ☐ * 접수방법코드
- ☐ * 신청인구분코드
- ☐ * 신청인주민번호
- ☐ * 신청인명

관할부서+접수일자+접수순번=접수번호(인조식별자)

- ① 인조식별자를 사용함으로써 SQL문의 복잡성을 피할 수 있다.
- ② 인조식별자를 사용함으로써 애플리케이션 개발자가 SQL문을 개발 시 복잡한 소스 구성을 피할 수 있다.
- ③ 인조식별자를 사용함으로써 데이터 모델을 단순하게 하는데 기여한다.
- ④ 인조식별자를 사용함으로써 식별자의 유일성을 더욱 더 강화시킨다.

SQLD 출제 예상문제 - 2회
과목 1. 데이터 모델링의 이해
1-2. 6번~10번

문제6. [아래] '일재고' 엔터티는 1정규형 위반이며 이를 해소 하기 위하여 '일재고상세' 엔터티를 추가하여 제1정규형을 진행하고자 한다. '일재고상세' 엔터티의 PK구성으로 가장 적절한 것은 무엇인가?

<데이터 모델>

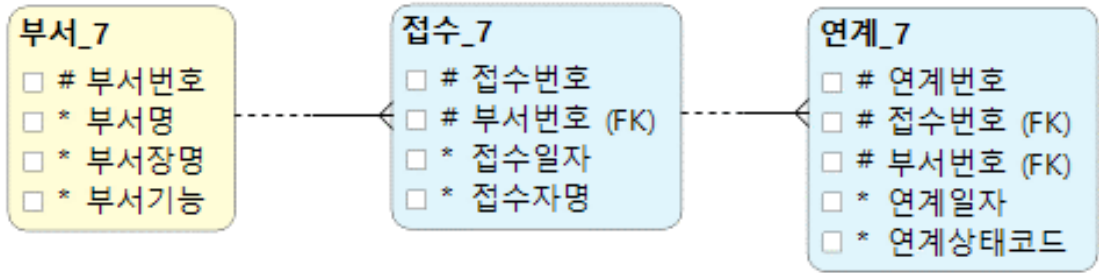
일재고

- ☐ # 물류센터코드
- ☐ # 재고일자
- ☐ o 월초재고수량
- ☐ o 장기재고1개월수량
- ☐ o 장기재고2개월수량
- ☐ o 장기재고3개월수량
- ☐ o 장기재고1개월주문수량
- ☐ o 장기재고2개월주문수량
- ☐ o 장기재고3개월주문수량
- ☐ o 장기재고1개월금액
- ☐ o 장기재고2개월금액
- ☐ o 장기재고3개월금액
- ☐ o 장기재고1개월주문금액
- ☐ o 장기재고2개월주문금액
- ☐ o 장기재고3개월주문금액

- ① 물류센터코드+재고일자
- ② 물류센터코드+재고기간
- ③ 물류센터코드+재고일시
- ④ 물류센터코드+재고일자+재고기간

문제7. [아래] 데이터 모델에서 '연계' 엔터티는 원격지 데이터베이스에 존재하는 엔터티이다. 제시된 <SQL문>은 이 시스템에서 빈번하게 수행되는 <SQL문>이다. 이 <SQL문>의 성능을 향상시키기 위한 방법으로 가장 알맞은 것은 무엇인가?

<데이터 모델>

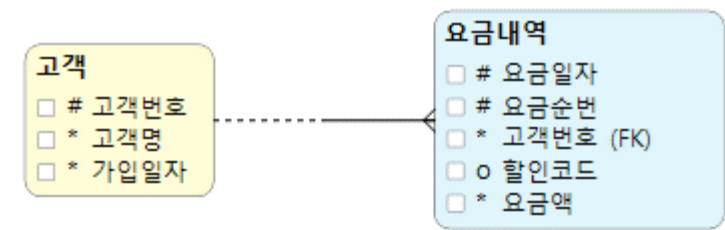


<SQL문>

```
SELECT A.부서명
      , C.연계일자
      , C.연계상태코드
FROM   부서 A
      , 접수 B
      , 연계@원격서버 C
WHERE  C.연계일자 BETWEEN '20200801' AND '20200831'
AND    C.부서번호 = B.부서번호
AND    B.부서번호 = A.부서번호
AND    C.접수번호 = B.접수번호
;
```

- ① 부서 테이블을 원격 서버로 이동시켜 원격지에서 부서와 연계 테이블을 조인한다.
- ② 부서 테이블에 연계일자, 연계상태코드 칼럼을 추가하여 연계 테이블과 조인을 제거한다.
- ③ 연계 테이블에 부서명 칼럼을 추가하여 연계 테이블 만을 조회한다.
- ④ 접수 테이블에 부서명 칼럼을 추가하여 접수와 연계 테이블을 조인 한다.

문제8. [아래] 데이터 모델은 고객별 요금에 대한 업무를 정의하고 있다. 업무 특성상 월 1천만건의 요금 데이터가 저장된다. 요금 엔터티가 대용량인 점을 감안하여 요금일자 칼럼을 기준으로 월별 범위 파티셔닝을 적용하려고 한다. 이에 따른 장점으로 가장 알맞은 것은 무엇인가?



- ① 요금 테이블은 논리적으로는 하나의 테이블 처럼 보이지만 물리적으로는 여러개로 쪼개져 있어 성능 상 불리하다.
- ② 범위 파티셔닝은 데이터 보관 주기에 따라 데이터를 물리적으로 제거할 수 있다.
- ③ 범위 파티셔닝은 데이터 보관 주기에 따라 데이터를 지우지면 실제 물리적으로 지워지진 않는다.
- ④ 요금순번을 기준으로 해시 파티셔닝으로 바꾼다면 해싱 알고리즘으로 데이터 보관 주기에 따라 매우 쉽게 데이터를 물리적으로 제거할 수 있다.

문제9. [아래] 데이터 모델은 학사기준과 수강신청 업무를 정의한 것이다. 수강신청 엔터티의 학사기준번호 칼럼은 논리 데이터 상으로 FK 관계이지만 물리 구축을 하면서 FK 지정을 하지 않았다. [아래]와 같은 <SQL문>이 빈번하게 호출되는 경우 성능을 향상시키는 방법 중 가장 타당한 것은 무엇인가?

<데이터 모델>

학사기준

☐ # 학사기준번호

☐ * 년도

☐ * 학기

☐ o 특이사항

수강신청

☐ # 강의번호

☐ # 학생번호

☐ * 학사기준번호 (FK)

☐ * 성명

☐ * 연락처

☐ * 등록년도

☐ o 감면코드

<SQL문>

```
SELECT COUNT(B.학생번호)
FROM 학사기준 A
      , 수강신청 B
WHERE A.학사기준번호 = B.학사기준번호
      AND A.년도 >= '2018'
      AND A.학기 = '1'
;
```

- ① 학사기준 테이블에 년도+학기로 이루어진 인덱스를 생성한다.

② 학사기준 테이블에 학기+년도로 이루어진 인덱스를 생성한다.

③ 수강신청 테이블에 학사기준번호로 이루어진 인덱스를 생성한다.

④ 수강신청 테이블의 PK순서를 학번+강의번호 순으로 변경 생성한다.

문제10. 다음 중 분산 데이터베이스의 적용 기법의 종류가 아닌 것은 무엇인가?

- ① 테이블 위치 분산
- ② 테이블 중복 분산
- ③ 테이블 요약 분산
- ④ 테이블 분할 분산

SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-1. 1번~5번

문제1. 다음 중 테이블에 관련된 용어에 대한 설명으로 가장 옳바르지 않는 것은 무엇인가?

- ① 테이블(Table) : 행과 칼럼의 2차원 구조를 가진 데이터 저장 장소이며, 데이터베이스의 가장 기본적인 개념이다.
- ② 칼럼(Column) : 2차원 구조를 가진 테이블에서 세로 방향으로 이루어진 하나하나의 특징 속성
- ③ 정규화 : 테이블을 분할하여 데이터의 정합성을 확보하고 필요한 중복을 허용하는 프로세스
- ④ 기본키 : 테이블에 존재하는 각 행을 한가지 의미로 특정할 수 있는 한 개 이상의 칼럼

문제2.

테이블에 데이터를 삭제 하는 방법에는 DELETE, TRUNCATE, DROP이 있다. 이에 대한 설명으로 가장 올바른 것은 무엇인가?

- ① DROP 명령으로 테이블을 삭제하면 테이블 내 데이터만 데이터베이스에서 영구적으로 삭제된다.
- ② TRUNCATE 명령으로 테이블을 삭제하면 테이블 스키마 및 데이터가 데이터베이스에서 영구적으로 삭제된다.
- ③ TRUNCATE 명령 수행 시 데이터 내 데이터가 삭제되지만 ROLLBACK명령으로 복구가 가능하다.
- ④ DELETE 명령으로 데이터를 삭제하면 테이블 스키마 구조의 변화없이 ROLLBACK명령으로 복구가 가능하다.

문제3.

아래의 SQL문은 TB_PLAYER_3 테이블에서 1980년대에 태어난 선수의 선수명을 출력하는 SQL문이다. 다음 중 SQL문 실행에 실패하는 것은 무엇인가?

①

```
SELECT A.PLAYER_NM AS 선수명
FROM TB_PLAYER_3 A
WHERE A.BIRTH_DE >= '19800101'
AND A.BIRTH_DE <= '19891231'
;
```

②

```
SELECT TB_PLAYER_3.PLAYER_NM AS 선수명
FROM TB_PLAYER_3 TB_PLAYER_3
WHERE TB_PLAYER_3.BIRTH_DE >= '19800101'
AND TB_PLAYER_3.BIRTH_DE <= '19891231'
;
```

③

```
SELECT TB_PLAYER_3.PLAYER_NM AS 선수명
FROM TB_PLAYER_3
WHERE TB_PLAYER_3.BIRTH_DE >= '19800101'
AND TB_PLAYER_3.BIRTH_DE <= '19891231'
;
```

④

```
SELECT A.PLAYER_NM AS 선수명
FROM TB_PLAYER_3 AS A
WHERE A.BIRTH_DE >= '19800101'
AND A.BIRTH_DE <= '19891231'
;
```


문제4.

DBMS는 COMMIT 혹은 ROLLBACK 명령을 통해 변경된 사항을 영구적으로 반영하던지 취소할 수 있다. 다음 중 COMMIT 혹은 ROLLBACK 명령 실행 이전의 데이터 상태에 대해서 올바르게 설명한 것을 2개 고르시오.

- ① 변경사항이 디스크에 쓰여진 채로 적용 혹은 취소만 하지 않은 상태이다.
- ② 데이터를 변경한 사용자는 다른 사용자를 통해 변경내역을 확인 할 수 있다.
- ③ 다른 사용자는 데이터를 변경한 사용자가 수행한 명령의 결과를 볼 수 없다.
- ④ 변경된 행은 잠금(Locking)이 설정되어서 다른 사용자가 변경 할 수 없다.

문제5.

다음 [아래]의 테이블 구성에서 전공(MAJOR)이 '데이터'로 시작하고 이름(EMP_NM)의 성이 '이'씨인 직원을 추출하는 SQL문으로 가장 부적절한 것은? (입력되어 있는 데이터를 기준으로 결과 집합이 만족하면 된다.)

SELECT * FROM TB_EMP_5;

EMP_NO	EMP_NM	MAJOR	DEPT_CD
100001	이경오	컴퓨터소프트웨어학	101
100002	이수지	데이터사이언스학	101
100003	이지수	컴퓨터공학	101
100004	김성민	컴퓨터공학	102
100005	김민선	데이터사이언스학	102
100006	김선미	컴퓨터소프트웨어학	102

①

```
SELECT A.*
FROM TB_EMP_5 A
WHERE A.MAJOR LIKE '데이터____' --언더바(_)의 개수는 5개임
AND A.EMP_NM LIKE '이%';
```

②

```
SELECT A.*
FROM TB_EMP_5 A
WHERE A.MAJOR LIKE '데이터%____' --언더바(_)의 개수는 5개임
AND A.EMP_NM LIKE '이__' --언더바(_)의 개수는 2개임;
```

③

```
SELECT A.*
FROM TB_EMP_5 A
WHERE A.MAJOR LIKE '데이터____%' --언더바(_)의 개수는 5개임
AND A.EMP_NM LIKE '이__%' --언더바(_)의 개수는 2개임;
```

④

```
SELECT A.*
FROM TB_EMP_5 A
WHERE A.MAJOR LIKE '데이터____' --언더바(_)의 개수는 5개임
AND A.EMP_NM LIKE '이____%' --언더바(_)의 개수는 5개임;
```

SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-2. 6번~10번

문제6. [아래]와 같이 TB_EMP_6 테이블을 생성하고 데이터를 입력하였다. <SQL문>의 실행 결과는 무엇인가?

```
DROP TABLE TB_EMP_6;

CREATE TABLE TB_EMP_6
(
  EMP_NO CHAR(6) NOT NULL
, EMP_NM VARCHAR2(50) NOT NULL
, JOIN_DT DATE NULL
, CONSTRAINT TB_EMP_6_PK PRIMARY KEY (EMP_NO)
)
;

INSERT INTO TB_EMP_6 VALUES ('100001', '이경오', TO_DATE('20200316090000', 'YYYYMMDDHH24MISS'));
INSERT INTO TB_EMP_6 VALUES ('100002', '이수지', TO_DATE('20200816090000', 'YYYYMMDDHH24MISS'));
INSERT INTO TB_EMP_6 VALUES ('100003', '김정훈', NULL);
INSERT INTO TB_EMP_6 VALUES ('100004', '박수진', TO_DATE('20190211090000', 'YYYYMMDDHH24MISS'));
INSERT INTO TB_EMP_6 VALUES ('100005', '최동진', TO_DATE('20190311090000', 'YYYYMMDDHH24MISS'));

COMMIT;
```

<SQL문>

```
SELECT COUNT(*) CNT
FROM TB_EMP_6 A
WHERE A.JOIN_DT NOT BETWEEN TO_DATE('20200101000000', 'YYYYMMDDHH24MISS')
AND TO_DATE('20201231235959', 'YYYYMMDDHH24MISS')
;
```

- ① 0
- ② 1
- ③ 2
- ④ 3

문제7. [아래]의 <SQL문>에서 ㉠에 들어갈 알맞은 키워드를 작성하고, 해당 SQL문의 결과값 ㉡을 작성하시오.

<SQL문>

```
SELECT CASE WHEN 1 = 1 THEN 1 ELSE 0 ㉠ AS RESULT  
FROM DUAL  
;
```

<결과값>

㉡

문제8. [아래] <SQL문>을 수행 후 출력된 결과값을 참고하여 ㉠에 들어갈 내용을 기재 하시오. (오라클 기준)

<SQL문>

```
SELECT TO_DATE('20200901' || '120000', 'YYYYMMDDHH24MISS') - ㉠ AS RESULT_VAL
FROM DUAL;
```

<결과값>

RESULT_VAL
2020/09/01 11:59:00

문제9. [아래]와 같이 TB_EMP_9 테이블을 생성하고 데이터를 입력하였다. [아래]의 <SQL>문의 결과로 올바른 것은?

```
DROP TABLE TB_EMP_9;

CREATE TABLE TB_EMP_9
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, DEPT_CD CHAR(3) NULL
, CONSTRAINT TB_EMP_9_PK PRIMARY KEY(EMP_NO)
);

INSERT INTO TB_EMP_9 VALUES ('100001', '이경오', '101');
INSERT INTO TB_EMP_9 VALUES ('100002', '김태호', '101');
INSERT INTO TB_EMP_9 VALUES ('100003', '박태훈', '102');
INSERT INTO TB_EMP_9 VALUES ('100004', '김수지', '102');
INSERT INTO TB_EMP_9 VALUES ('100005', '황정식', '103');
INSERT INTO TB_EMP_9 VALUES ('100006', '황태섭', '103');
INSERT INTO TB_EMP_9 VALUES ('100007', '김미선', '104');
INSERT INTO TB_EMP_9 VALUES ('100008', '박수경', '104');
INSERT INTO TB_EMP_9 VALUES ('100009', '최태경', NULL);
INSERT INTO TB_EMP_9 VALUES ('100010', '김승리', NULL);

COMMIT;
```

<SQL문>

```
SELECT DEPT_CD AS DEPT_CD
      , COUNT(DEPT_CD) AS DEPT_CD_CNT
      , COUNT(*) AS CNT
  FROM TB_EMP_9
 GROUP BY DEPT_CD
 ORDER BY DEPT_CD
;
```

①

DEPT_CD	DEPT_CD_CNT	CNT
101	2	2
102	2	2
103	2	2
104	2	2

②

DEPT_CD	DEPT_CD_CNT	CNT
101	2	2
102	2	2
103	2	2
104	2	2
NULL	0	2

③

DEPT_CD	DEPT_CD_CNT	CNT
101	2	8
102	2	8
103	2	8
104	2	8

④

DEPT_CD	DEPT_CD_CNT	CNT
101	2	10
102	2	10
103	2	10
104	2	10
NULL	2	10

문제10.

아래와 같이 TB_PLAYER_10 테이블을 생성하고 데이터를 입력하였다. 포지션(POS_NM) 칼럼을 기준으로 GROUP BY한 후 포지션 기준으로 정렬한 결과를 출력하고 있다. 다음 보기의 SQL문 올바르지 않은 것을 고르시오.

```
DROP TABLE TB_PLAYER_10;
CREATE TABLE TB_PLAYER_10
(
    PLAYER_ID CHAR(6)
,   PLAYER_NM VARCHAR2(50) NOT NULL
,   POS_NM VARCHAR2(5)
,   BIRTH_DE CHAR(8) NOT NULL
,   HEIGHT NUMBER(10, 2) NOT NULL
,   WEIGHT NUMBER(10, 2) NOT NULL
,   CONSTRAINT TB_PLAYER_10_PK PRIMARY KEY (PLAYER_ID)
)
;

INSERT INTO TB_PLAYER_10 VALUES ('100001', '황선홍', 'FW', '19680714', '183.4', '81.2');
INSERT INTO TB_PLAYER_10 VALUES ('100002', '안정환', 'FW', '19760127', '178.1', '68.3');
INSERT INTO TB_PLAYER_10 VALUES ('100003', '박지성', 'MF', '19810330', '175.2', '71.2');
INSERT INTO TB_PLAYER_10 VALUES ('100004', '유상철', 'MF', '19711018', '184.1', '77.8');
INSERT INTO TB_PLAYER_10 VALUES ('100005', '이천수', 'MF', '19810709', '172.1', '63.1');
INSERT INTO TB_PLAYER_10 VALUES ('100006', '설기현', 'MF', '19790204', '187.1', '77.2');
INSERT INTO TB_PLAYER_10 VALUES ('100007', '차두리', 'DF', '19800725', '181.1', '75.8');
INSERT INTO TB_PLAYER_10 VALUES ('100008', '이영표', 'DF', '19770423', '177.4', '70.2');
INSERT INTO TB_PLAYER_10 VALUES ('100009', '홍명보', 'DF', '19690212', '181.1', '72.4');
INSERT INTO TB_PLAYER_10 VALUES ('100010', '최진철', 'DF', '19710326', '187.8', '86.1');
INSERT INTO TB_PLAYER_10 VALUES ('100011', '이운재', 'GK', '19730426', '182.1', '88.1');

COMMIT;
```

①

```
SELECT A.POS_NM AS 포지션
FROM TB_PLAYER_10 A
GROUP BY A.POS_NM
ORDER BY A.POS_NM;
```

②

```
SELECT A.POS_NM AS 포지션
FROM TB_PLAYER_10 A
GROUP BY A.POS_NM
ORDER BY 포지션;
```

③

```
SELECT 포지션
FROM
(
    SELECT A.POS_NM AS 포지션
    FROM TB_PLAYER_10 A
    GROUP BY A.POS_NM
) A
ORDER BY 포지션;
```

④

```
SELECT A.포지션
FROM
(
    SELECT A.POS_NM AS 포지션
    FROM TB_PLAYER_10 A
    GROUP BY A.POS_NM
) A
ORDER BY A.POS_NM;
```


SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-3. 11번~15번

문제11.

아래의 <SQL문>은 TB_PLAYER_11 테이블에서 포지션(POS_NM)별 선수 명수를 구하는 <SQL문>이다. 결과 집합은 각 칼럼 별로 각각의 포지션의 선수 명수를 나타내고 있다. 이러한 <SQL문>의 결과 집합과 동일한 <새로운SQL문>을 작성하려고 한다. <새로운SQL문>에서 ㉠에 들어갈 알맞은 함수를 기입 하시오.

```
DROP TABLE TB_PLAYER_11;

CREATE TABLE TB_PLAYER_11
(
  PLAYER_ID CHAR(6)
, PLAYER_NM VARCHAR2(50) NOT NULL
, POS_NM VARCHAR2(5)
, BIRTH_DE CHAR(8) NOT NULL
, HEIGHT NUMBER(10, 2) NOT NULL
, WEIGHT NUMBER(10, 2) NOT NULL
, CONSTRAINT TB_PLAYER_11_PK PRIMARY KEY (PLAYER_ID)
);

INSERT INTO TB_PLAYER_11 VALUES ('100001', '황선홍', 'FW', '19680714', '183.4', '81.2');
INSERT INTO TB_PLAYER_11 VALUES ('100002', '안정환', 'FW', '19760127', '178.1', '68.3');
INSERT INTO TB_PLAYER_11 VALUES ('100003', '박지성', 'MF', '19810330', '175.2', '71.2');
INSERT INTO TB_PLAYER_11 VALUES ('100004', '유상철', 'MF', '19711018', '184.1', '77.8');
INSERT INTO TB_PLAYER_11 VALUES ('100005', '이천수', 'MF', '19810709', '172.1', '63.1');
INSERT INTO TB_PLAYER_11 VALUES ('100006', '설기현', 'MF', '19790204', '187.1', '77.2');
INSERT INTO TB_PLAYER_11 VALUES ('100007', '차두리', 'DF', '19800725', '181.1', '75.8');
INSERT INTO TB_PLAYER_11 VALUES ('100008', '이영표', 'DF', '19770423', '177.4', '70.2');
INSERT INTO TB_PLAYER_11 VALUES ('100009', '홍명보', 'DF', '19690212', '181.1', '72.4');
INSERT INTO TB_PLAYER_11 VALUES ('100010', '최진철', 'DF', '19710326', '187.8', '86.1');
INSERT INTO TB_PLAYER_11 VALUES ('100011', '이운재', 'GK', '19730426', '182.1', '88.1');

COMMIT;
```

<SQL문>

```
SELECT
  SUM(CASE WHEN A.POS_NM = 'FW' THEN 1 ELSE 0 END) AS FW_CNT
, SUM(CASE WHEN A.POS_NM = 'MF' THEN 1 ELSE 0 END) AS MF_CNT
, SUM(CASE WHEN A.POS_NM = 'DF' THEN 1 ELSE 0 END) AS DF_CNT
, SUM(CASE WHEN A.POS_NM = 'GK' THEN 1 ELSE 0 END) AS GK_CNT
FROM TB_PLAYER_11 A
;
```

<결과>

FW_CNT	MF_CNT	DF_CNT	GK_CNT
2	4	4	1

<새로운SQL문>

```
SELECT
  SUM(㉠(A.POS_NM, 'FW', 1, 0)) AS FW_CNT
, SUM(㉠(A.POS_NM, 'MF', 1, 0)) AS MF_CNT
, SUM(㉠(A.POS_NM, 'DF', 1, 0)) AS DF_CNT
, SUM(㉠(A.POS_NM, 'GK', 1, 0)) AS GK_CNT
FROM TB_PLAYER_11 A
;
```

문제12.

TB_EMP_SAL_12 테이블을 생성하고 각 사원의 연도별 연봉정보 데이터를 입력하였다. 2019년도 기준 가장 높은 액수의 연봉액을 구하고 있다. 다음 보기의 SQL문 중 가장 높은 액수의 연봉을 구하는 SQL문이 아닌 것은? (오라클 기준)

```
DROP TABLE TB_EMP_SAL_12;
CREATE TABLE TB_EMP_SAL_12
(
  EMP_NO CHAR(6)
, STD_YYYY CHAR(4)
, SAL NUMBER(15) NULL
, CONSTRAINT TB_EMP_SAL_12_PK PRIMARY KEY(EMP_NO, STD_YYYY)
);

INSERT INTO TB_EMP_SAL_12 VALUES ('100001', '2020', '70000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100001', '2019', '65000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100001', '2018', '60000000');

INSERT INTO TB_EMP_SAL_12 VALUES ('100002', '2020', '60000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100002', '2019', '55000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100002', '2018', '50000000');

INSERT INTO TB_EMP_SAL_12 VALUES ('100003', '2020', '50000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100003', '2019', '45000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100003', '2018', '40000000');

INSERT INTO TB_EMP_SAL_12 VALUES ('100004', '2020', '40000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100004', '2019', '35000000');
INSERT INTO TB_EMP_SAL_12 VALUES ('100004', '2018', '30000000');

INSERT INTO TB_EMP_SAL_12 VALUES ('100005', '2020', NULL);
INSERT INTO TB_EMP_SAL_12 VALUES ('100005', '2019', NULL);
INSERT INTO TB_EMP_SAL_12 VALUES ('100005', '2018', NULL);
COMMIT;
```

①

```
SELECT A.SAL AS SAL
FROM
(
  SELECT A.SAL
  FROM TB_EMP_SAL_12 A
  WHERE A.STD_YYYY = '2019'
  ORDER BY NVL(A.SAL, 0) DESC
) A
WHERE ROWNUM <= 1;
```

②

```
SELECT MAX(A.SAL) AS SAL
FROM TB_EMP_SAL_12 A
WHERE A.STD_YYYY = '2019';
```

③

```
SELECT A.SAL AS SAL
FROM
(
  SELECT A.SAL
  FROM TB_EMP_SAL_12 A
  WHERE A.STD_YYYY = '2019'
  ORDER BY CASE WHEN A.SAL IS NULL THEN 999999999999999 ELSE 0 END DESC
) A
WHERE ROWNUM <= 1;
```

④

```
SELECT A.SAL AS SAL
FROM
(
  SELECT A.SAL
  FROM TB_EMP_SAL_12 A
  WHERE A.STD_YYYY = '2019'
  ORDER BY DECODE(A.SAL, NULL, 0, A.SAL) DESC
) A
WHERE ROWNUM <= 1;
```

문제13.

아래와 같이 TB_EMP_SAL_13 테이블을 생성 후 데이터를 입력하였다. 다음 중 아래 <SQL문> 및 <결과>에 대한 설명으로 가장 적절한 것은 무엇인가?

```
DROP TABLE TB_EMP_SAL_13;

CREATE TABLE TB_EMP_SAL_13
(
  EMP_NO CHAR(6)
, STD_YYYY CHAR(4)
, SAL NUMBER(15) NULL
, CONSTRAINT TB_EMP_SAL_13_PK PRIMARY KEY(EMP_NO, STD_YYYY)
)
;

INSERT INTO TB_EMP_SAL_13 VALUES ('100001', '2020', '700000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100001', '2019', '650000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100001', '2018', '600000000');

INSERT INTO TB_EMP_SAL_13 VALUES ('100002', '2020', '600000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100002', '2019', '550000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100002', '2018', '500000000');

INSERT INTO TB_EMP_SAL_13 VALUES ('100003', '2020', '500000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100003', '2019', '450000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100003', '2018', '400000000');

INSERT INTO TB_EMP_SAL_13 VALUES ('100004', '2020', '400000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100004', '2019', '350000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100004', '2018', '300000000');

INSERT INTO TB_EMP_SAL_13 VALUES ('100005', '2020', NULL );
INSERT INTO TB_EMP_SAL_13 VALUES ('100005', '2019', NULL );
INSERT INTO TB_EMP_SAL_13 VALUES ('100005', '2018', NULL );

COMMIT;
```

<SQL문>

SELECT *
FROM TB_EMP_SAL_13 A
WHERE STD_YYYY = '2019'
AND ROWNUM <= 1
ORDER BY A.SAL DESC
;

<결과>

EMP_NO	STD_YYYY	SAL
100001	2019	650000000

①

2019년도 기준으로 연봉(SAL)이 가장 높은 단 한 건이 출력되었다.

②

STD_YYYY조건을 '2020'으로 바꾼다면 2020년도 기준 연봉이 가장 높은 단 한 건이 출력된다.

③

만약 ORDER BY를 'A.SAL ASC'로 바꾼다면 2019년도 기준 연봉이 가장 낮은 단 한 건이 출력된다.

④

STD_YYYY조건이 '2019'인 행 중에서 무작위로 단 한 건의 데이터가 나와서 ORDER BY는 의미가 없어진다.

문제14. 아래와 같이 TB_EMP_14, TB_EMP_SAL_14 테이블을 생성하고 데이터를 입력하였다. 아래 SQL문의 결과는 무엇인가?

```
DROP TABLE TB_EMP_SAL_14;
DROP TABLE TB_EMP_14;

CREATE TABLE TB_EMP_14
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, CONSTRAINT TB_EMP_14_PK PRIMARY KEY (EMP_NO)
);

INSERT INTO TB_EMP_14 VALUES ('100001', '이경오');
INSERT INTO TB_EMP_14 VALUES ('100002', '이수지');
COMMIT;

CREATE TABLE TB_EMP_SAL_14
(
  EMP_NO CHAR(6)
, SAL_STD_YYYY CHAR(8)
, SAL NUMBER
, CONSTRAINT TB_EMP_SAL_14_PK PRIMARY KEY (EMP_NO, SAL_STD_YYYY)
);

ALTER TABLE TB_EMP_SAL_14
ADD CONSTRAINTS TB_EMP_SAL_14_FK FOREIGN KEY (EMP_NO)
REFERENCES TB_EMP_14(EMP_NO);

INSERT INTO TB_EMP_SAL_14 VALUES ('100001', '2020', 75000000);
INSERT INTO TB_EMP_SAL_14 VALUES ('100001', '2019', 65000000);
INSERT INTO TB_EMP_SAL_14 VALUES ('100001', '2018', 55000000);
INSERT INTO TB_EMP_SAL_14 VALUES ('100002', '2020', 55000000);
INSERT INTO TB_EMP_SAL_14 VALUES ('100002', '2019', 35000000);
INSERT INTO TB_EMP_SAL_14 VALUES ('100002', '2018', 25000000);
COMMIT;
```

<SQL문>

```
SELECT COUNT(DISTINCT A.EMP_NO) +
       COUNT(DISTINCT B.SAL_STD_YYYY) +
       COUNT(DISTINCT B.SAL) AS CNT_SUM
FROM TB_EMP_14 A
     , TB_EMP_SAL_14 B
WHERE A.EMP_NO = B.EMP_NO
;
```

- ① 18
- ② 3
- ③ 6
- ④ 10

문제15. 조인과 SQL성능에 대한 설명으로 가장 올바른 것은 무엇인가?

- ① 정규화를 통해 테이블을 분할 하여 중복이 제거된 상태로 데이터가 저장되기 때문에 SQL성능이 무조건 좋아진다.
- ② 정규화를 하지 않고 데이터를 중복적으로 관리하면 데이터의 양을 커지나 조인이 필요 없어서 SQL성능이 비약적으로 빨라진다.
- ③ 논리적인 연관관계 및 PK, FK, 인덱스 등을 고려해서 조인 연산을 수행하면 조인으로 인한 성능 저하를 최소화 시킬 수 있다.
- ④ 조인 조건이 잘못 기술되게 되면 DBMS시스템에서 자동으로 해당 연산을 중지해서 DBMS의 장애 발생을 예방한다.

SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-4. 16번~20번

문제16.

순수 관계연산자는 관계형 데이터베이스를 구현하기 위해 새롭게 만들어진 연산자이다. 다음 중 순수 관계 연산자를 현재 SQL 문과 비교한 것 중 올바른 것은 무엇인가?

- ① SELECT 연산 -> SELECT절로
- ② PROJECT 연산 -> SELEC절로
- ③ (NATURAL) JOIN 연산 -> NATURAL JOIN 기능으로
- ④ DIVIDE 연산 -> EXCEPT 기능으로

문제17.

아래와 같이 TB_EMP_17, TB_DEPT_17 테이블을 생성하고 데이터를 입력하였다. 아래의 <SQL문> 은 TB_DEPT_17과 TB_EMP_17 테이블을 LEFT OUTER JOIN하고 있는 SQL문이다. 해당 <SQL문>의 결과 집합은 무엇인가?

```
DROP TABLE TB_EMP_17;
DROP TABLE TB_DEPT_17;

CREATE TABLE TB_DEPT_17
(
  DEPT_NO CHAR(6)
, DEPT_NM VARCHAR2(150) NOT NULL
, CONSTRAINT TB_DEPT_17_PK PRIMARY KEY (DEPT_NO)
)
;

INSERT INTO TB_DEPT_17 VALUES ('D00001', 'Data시각화팀');
INSERT INTO TB_DEPT_17 VALUES ('D00002', 'Data플랫폼팀');
INSERT INTO TB_DEPT_17 VALUES ('D00003', 'Data분석팀' );

COMMIT;

CREATE TABLE TB_EMP_17
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, DEPT_NO CHAR(6)
, CONSTRAINT TB_EMP_17_PK PRIMARY KEY (EMP_NO)
)
;

INSERT INTO TB_EMP_17 VALUES ('E00001', '이경오', 'D00001');
INSERT INTO TB_EMP_17 VALUES ('E00002', '이수지', 'D00001');
INSERT INTO TB_EMP_17 VALUES ('E00003', '김호선', 'D00002');
INSERT INTO TB_EMP_17 VALUES ('E00004', '박상진', 'D00003');

COMMIT;

ALTER TABLE TB_EMP_17
ADD CONSTRAINTS TB_EMP_17_FK FOREIGN KEY (DEPT_NO)
REFERENCES TB_DEPT_17(DEPT_NO)
;
```

<SQL문>

```
SELECT A.DEPT_NO, A.DEPT_NM, B.EMP_NO, B.EMP_NM
FROM TB_DEPT_17 A LEFT OUTER JOIN TB_EMP_17 B
ON (A.DEPT_NO = B.DEPT_NO AND A.DEPT_NM = 'Data시각화팀')
WHERE A.DEPT_NO IS NOT NULL
;
```

①

DEPT_NO	DEPT_NM	EMP_NO	EMP_NM
D00001	Data시각화팀	E00001	이경오
D00001	Data시각화팀	E00002	이수지
NULL	NULL	E00003	김호선
NULL	NULL	E00004	박상진

②

DEPT_NO	DEPT_NM	EMP_NO	EMP_NM
D00001	Data시각화팀	E00001	이경오
D00001	Data시각화팀	E00002	이수지
D00003	Data분석팀	NULL	NULL
D00002	Data플랫폼팀	NULL	NULL

③

DEPT_NO	DEPT_NM	EMP_NO	EMP_NM
D00001	Data시각화팀	E00001	이경오
D00001	Data시각화팀	E00002	이수지

④

DEPT_NO	DEPT_NM	EMP_NO	EMP_NM
D00001	Data시각화팀	E00001	이경오

문제18. 다음 중 집합연산자의 UNION과 UNION ALL에 대한 설명으로 가장 부적절한 것은 무엇인가?

- ① UNION은 여러 개의 SQL문 결과에 대한 합집합이다.
- ② UNION은 중복된 행은 하나의 행으로 만든다.
- ③ UNION ALL 일반적으로 여러 개의 질의 결과가 중복이 없을 때(상호 배타적 일 때) 사용한다.
- ④ UNION ALL 중복된 행을 그대로 보여주며 중복된 행을 제외하면 UNION과 결과 집합 및 그 순서가 동일하다.

문제19. 아래와 같은 테이블 및 데이터 구성에서 <SQL문>의 결과집합으로 올바른 것은 무엇인가?

```
DROP TABLE TB_EMP_19;
DROP TABLE TB_DEPT_19;

CREATE TABLE TB_DEPT_19
(
  DEPT_NO CHAR(6)
, DEPT_NM VARCHAR2(150) NOT NULL
, CONSTRAINT TB_DEPT_19_PK PRIMARY KEY (DEPT_NO)
);

INSERT INTO TB_DEPT_19 VALUES ('D00001', 'Data시각화팀');
INSERT INTO TB_DEPT_19 VALUES ('D00002', 'Data플랫폼팀');
INSERT INTO TB_DEPT_19 VALUES ('D00003', 'Data분석팀');

COMMIT;

CREATE TABLE TB_EMP_19
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, DEPT_NO CHAR(6)
, CONSTRAINT TB_EMP_19_PK PRIMARY KEY (EMP_NO)
);

INSERT INTO TB_EMP_19 VALUES ('E00001', '이경오', 'D00001');
INSERT INTO TB_EMP_19 VALUES ('E00002', '이수지', 'D00001');
INSERT INTO TB_EMP_19 VALUES ('E00003', '김호선', 'D00002');
INSERT INTO TB_EMP_19 VALUES ('E00004', '박상진', 'D00003');

COMMIT;

ALTER TABLE TB_EMP_19
ADD CONSTRAINTS TB_EMP_19_FK FOREIGN KEY (DEPT_NO)
REFERENCES TB_DEPT_19(DEPT_NO)
;
```

<SQL문>

```
SELECT DEPT_NO
FROM
(
  SELECT A.DEPT_NO
  FROM TB_EMP_19 A
  WHERE A.DEPT_NO = 'D00001'
MINUS
  SELECT A.DEPT_NO
  FROM TB_EMP_19 A
  WHERE A.DEPT_NO = 'D00002'
  ORDER BY DEPT_NO
)
GROUP BY DEPT_NO
;
```

①

DEPT_NO

D00001

②

DEPT_NO

D00001
D00002

③

DEPT_NO

④

SQL문법 에러

문제20. 다음과 같이 TB_DEPT_20 테이블을 생성하고 데이터를 입력하였다. 아래의 계층형 쿼리의 결과집합으로 올바른 것은 무엇인가?

```
DROP TABLE TB_DEPT_20;

CREATE TABLE TB_DEPT_20
(
  DEPT_NO CHAR(6)
, DEPT_NM VARCHAR2(150) NOT NULL
, UPPER_DEPT_NO CHAR(6) NULL
, CONSTRAINT TB_DEPT_20_PK PRIMARY KEY (DEPT_NO)
)
;

INSERT INTO TB_DEPT_20 VALUES ('D00001', '회장실', NULL);

INSERT INTO TB_DEPT_20 VALUES ('D00002', '영업본부', 'D00001');
INSERT INTO TB_DEPT_20 VALUES ('D00003', '기술본부', 'D00001');

INSERT INTO TB_DEPT_20 VALUES ('D00004', '국내영업부', 'D00002');
INSERT INTO TB_DEPT_20 VALUES ('D00005', '해외영업부', 'D00002');

INSERT INTO TB_DEPT_20 VALUES ('D00006', '개발사업부', 'D00003');
INSERT INTO TB_DEPT_20 VALUES ('D00007', '데이터사업부', 'D00003');

INSERT INTO TB_DEPT_20 VALUES ('D00008', '기업영업팀', 'D00004');
INSERT INTO TB_DEPT_20 VALUES ('D00009', '공공영업팀', 'D00004');

INSERT INTO TB_DEPT_20 VALUES ('D00010', '북미영업팀', 'D00005');
INSERT INTO TB_DEPT_20 VALUES ('D00011', '남미영업팀', 'D00005');

INSERT INTO TB_DEPT_20 VALUES ('D00012', '서버개발팀', 'D00006');
INSERT INTO TB_DEPT_20 VALUES ('D00013', '화면개발팀', 'D00006');

INSERT INTO TB_DEPT_20 VALUES ('D00014', '오라클기술팀', 'D00007');
INSERT INTO TB_DEPT_20 VALUES ('D00015', '오픈소스기술팀', 'D00007');

COMMIT;
```

<SQL문>

```
SELECT DEPT_NO, LPAD(' ', 4 * (LEVEL - 1)) || DEPT_NM AS DEPT_NM
      , UPPER_DEPT_NO
      , LEVEL AS LVL
FROM TB_DEPT_20
START WITH UPPER_DEPT_NO IS NULL
CONNECT BY PRIOR DEPT_NO = UPPER_DEPT_NO
           AND DEPT_NM LIKE '영업%';
```

①

DEPT_NO	DEPT_NM	UPPER_DEPT_NO	LVL
D00002	영업본부	D00001	2
D00001	회장실		1

②

DEPT_NO	DEPT_NM	UPPER_DEPT_NO	LVL
D00001	회장실		1
D00002	영업본부	D00001	2
D00004	국내영업부	D00002	3
D00005	해외영업부	D00002	3

③

DEPT_NO	DEPT_NM	UPPER_DEPT_NO	LVL
D00001	회장실		1
D00002	영업본부	D00001	2

④

DEPT_NO	DEPT_NM	UPPER_DEPT_NO	LVL
D00001	회장실		1
D00002	영업본부	D00001	2
D00004	국내영업부	D00002	3
D00008	기업영업팀	D00004	4
D00009	공공영업팀	D00004	4
D00005	해외영업부	D00002	3
D00010	북미영업팀	D00005	4
D00011	남미영업팀	D00005	4

SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-5. 21번~25번

문제21.

다음과 같이 TB_DEPT_21 테이블을 생성하고 데이터를 입력하였다. 아래의 <SQL문> 과 실행 결과를 참고하여 <SQL문>에 ㉠에 들어갈 내용을 작성하시오.

```
DROP TABLE TB_DEPT_21;

CREATE TABLE TB_DEPT_21
(
    DEPT_NO CHAR(6)
    , DEPT_NM VARCHAR2(150) NOT NULL
    , UPPER_DEPT_NO CHAR(6) NULL
    , CONSTRAINT TB_DEPT_21_PK PRIMARY KEY (DEPT_NO)
)
;

INSERT INTO TB_DEPT_21 VALUES ('D00001', '회장실'      , NULL );
INSERT INTO TB_DEPT_21 VALUES ('D00002', '영업본부'    , 'D00001');
INSERT INTO TB_DEPT_21 VALUES ('D00003', '기술본부'    , 'D00001');

INSERT INTO TB_DEPT_21 VALUES ('D00004', '국내영업부'   , 'D00002');
INSERT INTO TB_DEPT_21 VALUES ('D00005', '해외영업부'   , 'D00002');

INSERT INTO TB_DEPT_21 VALUES ('D00006', '개발사업부'   , 'D00003');
INSERT INTO TB_DEPT_21 VALUES ('D00007', '데이터사업부' , 'D00003');

INSERT INTO TB_DEPT_21 VALUES ('D00008', '기업영업팀'   , 'D00004');
INSERT INTO TB_DEPT_21 VALUES ('D00009', '공공영업팀'   , 'D00004');

INSERT INTO TB_DEPT_21 VALUES ('D00010', '북미영업팀'   , 'D00005');
INSERT INTO TB_DEPT_21 VALUES ('D00011', '남미영업팀'   , 'D00005');

INSERT INTO TB_DEPT_21 VALUES ('D00012', '서버개발팀'   , 'D00006');
INSERT INTO TB_DEPT_21 VALUES ('D00013', '화면개발팀'   , 'D00006');

INSERT INTO TB_DEPT_21 VALUES ('D00014', '오라클기술팀' , 'D00007');
INSERT INTO TB_DEPT_21 VALUES ('D00015', '오픈소스기술팀', 'D00007');

COMMIT;
```

<SQL문>

```
SELECT B.DEPT_NO, B.DEPT_NM, B.UPPER_DEPT_NO, A.UPPER_DEPT_NO
FROM TB_DEPT_21 A ㉠ OUTER JOIN TB_DEPT_21 B
ON (B.UPPER_DEPT_NO = A.DEPT_NO)
WHERE A.UPPER_DEPT_NO IS NULL
ORDER BY B.DEPT_NO;
```

<실행결과>

DEPT_NO	DEPT_NM	UPPER_DEPT_NO	UPPER_DEPT_NO
D00001	회장실	NULL	NULL
D00002	영업본부	D00001	NULL
D00003	기술본부	D00001	NULL

문제22. 다음 중 반환 되는 데이터 형태에 따른 서브 쿼리의 분류 중 가장 적절하지 않은 것은 무엇인가?

- ① 단일 행 서브쿼리
- ② 복합 행 서브쿼리
- ③ 다중 행 서브쿼리
- ④ 다중 칼럼 서브쿼리

문제23.

아래와 같이 TB_EMP_23 및 TB_DEPT_23 테이블을 생성 하고 데이터를 입력하였다. 부서테이블에 '차은영' 사원은 아직 부서를 배정받지 못한 인턴 사원이다. 다음 중 아래 결과와 같이 부서를 배정받지 못한 '차은영'사원만을 출력하는 SQL문을 무엇인가?

```
DROP TABLE TB_EMP_23;
DROP TABLE TB_DEPT_23;
CREATE TABLE TB_DEPT_23
(
    DEPT_CD CHAR(4)
, DEPT_NM VARCHAR2(150) NOT NULL
);
ALTER TABLE TB_DEPT_23
ADD CONSTRAINT TB_DEPT_23_PK PRIMARY KEY (DEPT_CD);
INSERT INTO TB_DEPT_23 (DEPT_CD, DEPT_NM) VALUES ('D001', 'Data시각화팀');
INSERT INTO TB_DEPT_23 (DEPT_CD, DEPT_NM) VALUES ('D002', 'Data플랫폼팀');
INSERT INTO TB_DEPT_23 (DEPT_CD, DEPT_NM) VALUES ('D003', 'Data분석팀');
COMMIT;

CREATE TABLE TB_EMP_23
(
    EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, SEX_CD CHAR(1)
, BIRTH_DE CHAR(8) NOT NULL
, DEPT_CD CHAR(4)
);
ALTER TABLE TB_EMP_23
ADD CONSTRAINT TB_EMP_23_PK PRIMARY KEY (EMP_NO);
ALTER TABLE TB_EMP_23
ADD CONSTRAINT TB_EMP_23_FK FOREIGN KEY (DEPT_CD) REFERENCES TB_DEPT_23(DEPT_CD);

INSERT INTO TB_EMP_23 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00001', '이경오', '1', '19840718', 'D001');
INSERT INTO TB_EMP_23 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00002', '이수지', '2', '19940502', 'D001');
INSERT INTO TB_EMP_23 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00003', '박경민', '1', '19830414', 'D002');
INSERT INTO TB_EMP_23 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00004', '최주연', '2', '19920508', 'D002');
INSERT INTO TB_EMP_23 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00005', '최철순', '1', '19860112', 'D003');
INSERT INTO TB_EMP_23 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00006', '이지연', '2', '19960218', 'D003');
INSERT INTO TB_EMP_23 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00007', '차은영', '2', '19980218', NULL);
COMMIT;
```


문제23.

아래와 같이 TB_EMP_23 및 TB_DEPT_23 테이블을 생성 하고 데이터를 입력하였다. 부서테이블에 '차은영' 사원은 아직 부서를 배정받지 못한 인턴 사원이다. 다음 중 아래 결과와 같이 부서를 배정받지 못한 '차은영'사원 만을 출력하는 SQL문을 무엇인가?

<실행결과>

EMP_NO	EMP_NM	SEX_CD	BIRTH_DE	DEPT_CD
E00007	차은영	2	19980218	NULL

①

```
SELECT *
FROM TB_EMP_23 A
WHERE NOT EXISTS (SELECT 1
FROM TB_DEPT_23 B
WHERE B.DEPT_CD = A.DEPT_CD
)
;
```

②

```
SELECT *
FROM TB_EMP_23 A
WHERE EXISTS (SELECT 1
FROM TB_DEPT_23 B
WHERE B.DEPT_CD <> A.DEPT_CD
)
;
```

③

```
SELECT *
FROM TB_EMP_23 A
WHERE A.DEPT_CD NOT IN (SELECT B.DEPT_CD
FROM TB_DEPT_23 B
)
;
```

④

```
SELECT *
FROM TB_EMP_23 A
WHERE NOT A.DEPT_CD IN (SELECT B.DEPT_CD
FROM TB_DEPT_23 B
)
;
```

문제24.

[아래]는 TB_EMP_24 테이블을 생성하고 데이터를 입력하고 V_TB_EMP_24라는 뷰를 생성하였다. 해당 뷰를 호출하는 <SQL문>이 출력 하는 결과를 기재 하시오. (만약 결과 집합이 공집합이라면 '공집합'이라고 기재 하시오.)

```
DROP TABLE TB_EMP_24;
CREATE TABLE TB_EMP_24
(
    EMP_NO CHAR(6)
    , EMP_NM VARCHAR2(50) NOT NULL
    , SEX_CD CHAR(1)
    , BIRTH_DE CHAR(8) NOT NULL
    , DEPT_CD CHAR(4)
)
;

ALTER TABLE TB_EMP_24
ADD CONSTRAINT TB_EMP_24_PK PRIMARY KEY (EMP_NO);

INSERT INTO TB_EMP_24 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00001', '이경오', '1', '19840718', 'D001');
INSERT INTO TB_EMP_24 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00002', '이수지', '2', '19940502', 'D001');
INSERT INTO TB_EMP_24 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00003', '박경민', '1', '19830414', 'D002');
INSERT INTO TB_EMP_24 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00004', '최주연', '2', '19920508', 'D002');
INSERT INTO TB_EMP_24 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00005', '최철순', '1', '19860112', 'D003');
INSERT INTO TB_EMP_24 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00006', '이지연', '2', '19960218', 'D003');
INSERT INTO TB_EMP_24 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00007', '차은영', '2', '19980218', NULL );
COMMIT;

DROP VIEW V_TB_EMP_24;
CREATE OR REPLACE VIEW V_TB_EMP_24 AS
SELECT A.EMP_NO, A.EMP_NM, A.SEX_CD, A.BIRTH_DE, A.DEPT_CD
FROM TB_EMP_24 A
WHERE A.DEPT_CD IS NULL
OR A.SEX_CD = '1'
;
```

<SQL문>

```
SELECT COUNT(*) CNT
FROM V_TB_EMP_24 A
WHERE A.SEX_CD = '2'
HAVING COUNT(*) >= 0
;
```

문제25.

아래와 같이 TB_EMP_25, TB_DEPT_25 테이블을 생성하고 데이터를 입력하였다. 아래 <SQL문> 과 같이 ROLLUP 함수를 사용하였을 때 결과 집합의 건수는 몇 건인가?

```
DROP TABLE TB_EMP_25;
DROP TABLE TB_DEPT_25;
CREATE TABLE TB_DEPT_25
(
  DEPT_CD CHAR(4)
, DEPT_NM VARCHAR2(150) NOT NULL
, CONSTRAINT TB_DEPT_25_PK PRIMARY KEY(DEPT_CD)
);
CREATE TABLE TB_EMP_25
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, JOB_NM VARCHAR2(150) NULL
, CUR_SAL NUMBER
, DEPT_CD CHAR(4)
, CONSTRAINT TB_EMP_25_PK PRIMARY KEY(EMP_NO)
);
ALTER TABLE TB_EMP_25
ADD CONSTRAINT TB_EMP_25_FK FOREIGN KEY (DEPT_CD)
REFERENCES TB_DEPT_25(DEPT_CD);
```

```
INSERT INTO TB_DEPT_25 VALUES ('D101', '데이터개발팀' );
INSERT INTO TB_DEPT_25 VALUES ('D102', '데이터플랫폼팀' );
INSERT INTO TB_DEPT_25 VALUES ('D103', '데이터사이언스팀' );
INSERT INTO TB_DEPT_25 VALUES ('D104', '데이터성능팀' );
INSERT INTO TB_DEPT_25 VALUES ('D105', '데이터마이그레이션팀' );
COMMIT;

INSERT INTO TB_EMP_25 VALUES ('100001', '이경오', 'SQL개발자', 45000000, 'D101');
INSERT INTO TB_EMP_25 VALUES ('100002', '이동민', 'SQL개발자', 40000000, 'D101');
INSERT INTO TB_EMP_25 VALUES ('100003', '김철수', 'SQL개발자', 40000000, 'D102');
INSERT INTO TB_EMP_25 VALUES ('100004', '박상진', 'SQL개발자', 35000000, 'D102');
INSERT INTO TB_EMP_25 VALUES ('100005', '박은정', 'SQL개발자', 50000000, 'D103');
INSERT INTO TB_EMP_25 VALUES ('100006', '김다연', 'SQL개발자', 45000000, 'D103');
INSERT INTO TB_EMP_25 VALUES ('100007', '박수진', 'SQL개발자', 65000000, 'D104');
INSERT INTO TB_EMP_25 VALUES ('100008', '김성수', 'SQL개발자', 60000000, 'D104');
INSERT INTO TB_EMP_25 VALUES ('100009', '추상미', 'SQL개발자', 35000000, 'D105');
INSERT INTO TB_EMP_25 VALUES ('100010', '박나래', 'SQL개발자', 30000000, 'D105');
COMMIT;
```

<SQL문>

```
SELECT JOB_NM
      , B.DEPT_NM
      , AVG(A.CUR_SAL) AS CUR_SAL
FROM TB_EMP_25 A
      , TB_DEPT_25 B
WHERE A.DEPT_CD = B.DEPT_CD
GROUP BY ROLLUP (A.JOB_NM, B.DEPT_NM)
;
```

- ① 5
- ② 6
- ③ 7
- ④ 11

SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-6. 26번~30번

문제26.

아래와 같이 TB_EMP_26, TB_DEPT_26 테이블을 생성하고 데이터를 입력하였다. 다음 보기의 SQL문 중 아래와 같은 <결과>를 리턴 하는 SQL문으로 올바른 것을 2개 고르시오.

```
DROP TABLE TB_EMP_26;
DROP TABLE TB_DEPT_26;
CREATE TABLE TB_DEPT_26
(
    DEPT_CD CHAR(4)
    , DEPT_NM VARCHAR2(150) NOT NULL
    , CONSTRAINT TB_DEPT_26_PK PRIMARY KEY(DEPT_CD)
);
CREATE TABLE TB_EMP_26
(
    EMP_NO CHAR(6)
    , EMP_NM VARCHAR2(50) NOT NULL
    , JOB_NM VARCHAR2(150) NULL
    , CUR_SAL NUMBER
    , DEPT_CD CHAR(4)
    , CONSTRAINT TB_EMP_26_PK PRIMARY KEY(EMP_NO)
);
ALTER TABLE TB_EMP_26
ADD CONSTRAINT TB_EMP_26_FK FOREIGN KEY (DEPT_CD)
REFERENCES TB_DEPT_26(DEPT_CD);
```

```
INSERT INTO TB_DEPT_26 VALUES ('D101', '데이터개발팀');
INSERT INTO TB_DEPT_26 VALUES ('D102', '데이터플랫폼팀');
INSERT INTO TB_DEPT_26 VALUES ('D103', '데이터사이언스팀');
INSERT INTO TB_DEPT_26 VALUES ('D104', '데이터성능팀');
INSERT INTO TB_DEPT_26 VALUES ('D105', '데이터마이그레이션팀');
COMMIT;
INSERT INTO TB_EMP_26 VALUES ('100001', '이경오', 'SQL개발자', 45000000, 'D101');
INSERT INTO TB_EMP_26 VALUES ('100002', '이동민', 'SQL개발자', 40000000, 'D101');
INSERT INTO TB_EMP_26 VALUES ('100003', '김철수', 'SQL개발자', 40000000, 'D102');
INSERT INTO TB_EMP_26 VALUES ('100004', '박상진', 'SQL개발자', 35000000, 'D102');
INSERT INTO TB_EMP_26 VALUES ('100005', '박은정', 'SQL개발자', 50000000, 'D103');
INSERT INTO TB_EMP_26 VALUES ('100006', '김다연', 'SQL개발자', 45000000, 'D103');
INSERT INTO TB_EMP_26 VALUES ('100007', '박수진', 'SQL개발자', 65000000, 'D104');
INSERT INTO TB_EMP_26 VALUES ('100008', '김성수', 'SQL개발자', 60000000, 'D104');
INSERT INTO TB_EMP_26 VALUES ('100009', '추상미', 'SQL개발자', 35000000, 'D105');
INSERT INTO TB_EMP_26 VALUES ('100010', '박나래', 'SQL개발자', 30000000, 'D105');
COMMIT;
```

<결과>

CNT	MAX_CUR_SAL	MIN_CUR_SAL
10	65000000	30000000

①

```
SELECT
    COUNT(*) AS CNT
    , MAX(CUR_SAL) AS MAX_CUR_SAL
    , MIN(CUR_SAL) AS MIN_CUR_SAL
FROM TB_EMP_26 A
    , TB_DEPT_26 B
WHERE A.DEPT_CD = B.DEPT_CD
GROUP BY GROUPING SETS(
);
```

②

```
SELECT
    COUNT(*) AS CNT
    , MAX(CUR_SAL) AS MAX_CUR_SAL
    , MIN(CUR_SAL) AS MIN_CUR_SAL
FROM TB_EMP_26 A
    , TB_DEPT_26 B
WHERE A.DEPT_CD = B.DEPT_CD
;
```

③

```
SELECT
    COUNT(*) AS CNT
    , MAX(CUR_SAL) AS MAX_CUR_SAL
    , MIN(CUR_SAL) AS MIN_CUR_SAL
FROM TB_EMP_26 A
    , TB_DEPT_26 B
;
```

④

```
SELECT
    COUNT(*) AS CNT
    , MAX(CUR_SAL) AS MAX_CUR_SAL
    , MIN(CUR_SAL) AS MIN_CUR_SAL
FROM TB_EMP_26 A
    , TB_DEPT_26 B
GROUP BY ROLLUP(A.CUR_SAL)
;
```

문제27. 다음 아래의 설명 중에서 ㉠에 들어갈 알맞은 말을 기재 하시오.

- ㉠ 은 하나의 SQL로 테이블을 한번만 읽어서 빠르게 원하는 리포트를 작성하는데 도움을 준다.
- ㉠ 의 종류로는 소그룹 간의 소계를 계산하는 ROLLUP, 다차원적인 소계를 계산하는 CUBE, 특정 항목에 대한 소계를 계산하는 GROUPING SETS가 있다.

문제28.

아래와 같이 TB_EMP_28 테이블을 생성하고 데이터를 입력하였다. 이 상황에서 아래의 <SQL>을 실행하였을 경우 출력되는 결과 집합은 무엇인가?

```
DROP TABLE TB_EMP_28 ;

CREATE TABLE TB_EMP_28
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, JOB_NM VARCHAR2(150) NULL
, CUR_SAL NUMBER
, DEPT_CD CHAR(4)
, CONSTRAINT TB_EMP_28_PK PRIMARY KEY(EMP_NO)
)
;

INSERT INTO TB_EMP_28 VALUES ('100001', '이경오', 'SQL개발자', 45000000, 'D101');
INSERT INTO TB_EMP_28 VALUES ('100002', '이동민', 'SQL개발자', 40000000, 'D101');

INSERT INTO TB_EMP_28 VALUES ('100003', '김철수', 'SQL개발자', 40000000, 'D102');
INSERT INTO TB_EMP_28 VALUES ('100004', '박상진', 'SQL개발자', 35000000, 'D102');

INSERT INTO TB_EMP_28 VALUES ('100005', '박은정', 'SQL개발자', 50000000, 'D103');
INSERT INTO TB_EMP_28 VALUES ('100006', '김다연', 'SQL개발자', 45000000, 'D103');

INSERT INTO TB_EMP_28 VALUES ('100007', '박수진', 'SQL개발자', 65000000, 'D104');
INSERT INTO TB_EMP_28 VALUES ('100008', '김성수', 'SQL개발자', 60000000, 'D104');

INSERT INTO TB_EMP_28 VALUES ('100009', '추상미', 'SQL개발자', 35000000, 'D105');
INSERT INTO TB_EMP_28 VALUES ('100010', '박나래', 'SQL개발자', 30000000, 'D105');

COMMIT;
```

<SQL문>

```
SELECT DEPT_CD
      , MAX_CUR_SAL
FROM
(
  SELECT DEPT_CD
        , MAX(CUR_SAL) OVER (PARTITION BY DEPT_CD) AS MAX_CUR_SAL
    FROM TB_EMP_28
  ) A
GROUP BY DEPT_CD, A.MAX_CUR_SAL
ORDER BY 1, 2
;
```

①

DEPT_CD	MAX_CUR_SAL
D101	45000000
D102	40000000
D103	50000000
D104	65000000
D105	35000000

④

DEPT_CD	MAX_CUR_SAL
D101	40000000
D102	35000000
D103	45000000
D104	60000000
D105	30000000

②

DEPT_CD	MAX_CUR_SAL
D101	45000000
D101	45000000
D102	40000000
D102	40000000
D103	50000000
D103	50000000
D104	65000000
D104	65000000
D105	35000000
D105	35000000

③

DEPT_CD	MAX_CUR_SAL
D101	40000000
D101	45000000
D102	35000000
D102	40000000
D103	45000000
D103	50000000
D104	60000000
D104	65000000
D105	30000000
D105	35000000

문제29.

아래와 같이 TB_EMP_29 테이블을 생성하고 데이터를 입력하였다. 다음 중 아래의 <결과>를 출력하는 SQL문으로 올바른 것은 무엇인가?

```
DROP TABLE TB_EMP_29;

CREATE TABLE TB_EMP_29
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, SEX_CD CHAR(1)
, BIRTH_DE CHAR(8) NOT NULL
, DEPT_CD CHAR(4)
)
;

ALTER TABLE TB_EMP_29
ADD CONSTRAINT TB_EMP_29_PK PRIMARY KEY (EMP_NO);

INSERT INTO TB_EMP_29 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00001', '이경오', '1', '19840718', 'D001');
INSERT INTO TB_EMP_29 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00002', '이수지', '2', '19940502', 'D001');
INSERT INTO TB_EMP_29 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00003', '박경민', '1', '19830414', 'D002');
INSERT INTO TB_EMP_29 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00004', '최주연', '2', '19920508', 'D002');
INSERT INTO TB_EMP_29 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00005', '최철순', '1', '19860112', 'D003');
INSERT INTO TB_EMP_29 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00006', '이지연', '2', '19960218', 'D003');
INSERT INTO TB_EMP_29 (EMP_NO, EMP_NM, SEX_CD, BIRTH_DE, DEPT_CD) VALUES ('E00007', '차은영', '2', '19980218', NULL);

COMMIT;
```


문제29. 아래와 같이 TB_EMP_29 테이블을 생성하고 데이터를 입력하였다. 아래의 <SQL문>의 결과로 올바른 것은 무엇인가?

<결과집합>

EMP_NO	EMP_NM	DEPT_CD
E00001	이경오	D001
E00002	이수지	D001
E00003	박경민	D002
E00004	최주연	D002
E00005	최철순	D003
E00006	이지연	D003
E00007	차은영	NULL

①

```
SELECT EMP_NO, EMP_NM, DEPT_CD AS DEPT_CD
FROM TB_EMP_29
ORDER BY EMP_NO;
```

②

```
SELECT EMP_NO, EMP_NM, LAG(DEPT_CD) OVER(ORDER BY DEPT_CD) AS DEPT_CD
FROM TB_EMP_29
ORDER BY EMP_NO;
```

③

```
SELECT EMP_NO, EMP_NM, LEAD(DEPT_CD) OVER(ORDER BY DEPT_CD) AS DEPT_CD
FROM TB_EMP_29
ORDER BY EMP_NO;
```

④

```
SELECT EMP_NO, EMP_NM, LAG(DEPT_CD) OVER(ORDER BY DEPT_CD) AS DEPT_CD
FROM TB_EMP_29
ORDER BY DEPT_CD;
```

문제30.

아래와 같이 TB_EMP_30 테이블을 생성하고 데이터를 입력하였다. 이 상태에서 아래의 <SQL문>을 호출한 경우에 출력되는 결과를 기재 하시오.

```
DROP TABLE TB_EMP_30;

CREATE TABLE TB_EMP_30
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, JOB_NM VARCHAR2(150) NULL
, CUR_SAL NUMBER
, DEPT_CD CHAR(4)
, CONSTRAINT TB_EMP_30_PK PRIMARY KEY(EMP_NO)
)
;

INSERT INTO TB_EMP_30 VALUES ('100001', '이경오', 'SQL개발자', 80000000, 'D101');
INSERT INTO TB_EMP_30 VALUES ('100002', '이동민', '프로시저개발자', 60000000, 'D101');

INSERT INTO TB_EMP_30 VALUES ('100003', '김철수', '리눅스엔지니어', 40000000, 'D102');
INSERT INTO TB_EMP_30 VALUES ('100004', '박상진', '윈도우엔지니어', 20000000, 'D102');

COMMIT;
```

<SQL문>

```
SELECT SUM(CUR_SAL)/10000
FROM
(
  SELECT A.EMP_NO
        , A.EMP_NM
        , LAG(CUR_SAL) OVER(ORDER BY CUR_SAL) AS LAG_CUR_SAL
        , LEAD(CUR_SAL) OVER(ORDER BY CUR_SAL) AS LEAD_CUR_SAL
        , CUR_SAL
  FROM TB_EMP_30 A
  ORDER BY A.EMP_NO
) A
WHERE LAG_CUR_SAL IS NULL
      OR LEAD_CUR_SAL IS NULL
;
```

SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-7. 31번~35번

문제31. 다음 아래와 같은 스크립트를 실행한 경우 최종적으로 어떠한 동작을 하게 되는지에 대하여 가장 올바르게 설명한 것은 무엇인가?

```
CONN SYSTEM/1234 --a

CREATE USER DCL IDENTIFIED BY 1234; --b
GRANT CONNECT, RESOURCE, DBA TO DCL; --c

CONN DCL/1234 --d

CREATE TABLE DCL_TABLE_31 --e
(
    DCL_COL1 NUMBER
)
;

INSERT INTO DCL_TABLE_31 VALUES (1); --f

COMMIT; --g

DROP TABLE DCL_TABLE_31 PURGE; --h

DROP USER DCL; --i
```

- ① ⑥ 단계에서 비밀번호 지정 시 쌍 따옴표("1234") 감싸지 않았으므로 계정 생성이 실패한다.
- ② ③ 단계에서 테이블 스페이스 지정없이 RESOURCE 권한을 주었으므로 권한 부여에 실패한다.
- ③ ⑧ 단계에서 PURGE 권한없이 테이블을 PURGE 옵션으로 제거하려고 했으므로 테이블 제거에 실패한다.
- ④ ① 단계에서 DCL 계정 자신이 자기 자신을 제거하려고 했으므로 유저 제거에 실패한다.

문제32.

아래 SQL문은 DCL 사용자에게 CREATE TABLE 권한을 주고 다시 CREATE TABLE의 권한을 취소(회수)하는 SQL문이다. SQL문에서 ㉠, ㉡, ㉢, ㉣에 들어갈 알맞은 키워드를 기재 하시오.

<SQL문>

--SYSTEM 계정으로 접속

㉠ CREATE TABLE ㉡ DCL; --DCL 사용자에게 CREATE TABLE 권한 부여

㉢ CREATE TABLE ㉣ DCL; --DCL 사용자에게 CREATE TABLE 권한 취소(회수)

문제33.

아래와 같이 TB_DEPT_33, TB_EMP_33 테이블을 생성하고 데이터를 입력하였다. 그 후 FN_EMP_CNT_BY_DEPT_CD_33이라는 사용자 정의 함수를 생성 하였으며 아래의 <SQL문>을 실행하였을 경우 해당 SQL문의 결과집합으로 가장 적절한 것은 무엇인가?

```
DROP TABLE TB_EMP_33;
DROP TABLE TB_DEPT_33;

CREATE TABLE TB_DEPT_33
(
    DEPT_CD CHAR(4)
, DEPT_NM VARCHAR2(150)
, CONSTRAINT TB_DEPT_33_PK PRIMARY KEY(DEPT_CD)
);

INSERT INTO TB_DEPT_33 VALUES ('D101', '데이터시각화팀');
INSERT INTO TB_DEPT_33 VALUES ('D102', '데이터엔지니어링팀');
INSERT INTO TB_DEPT_33 VALUES ('D103', '데이터분석팀');

COMMIT;

CREATE TABLE TB_EMP_33
(
    EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50)
, DEPT_CD CHAR(4)
, CONSTRAINT TB_EMP_33_PK PRIMARY KEY(EMP_NO)
);

INSERT INTO TB_EMP_33 VALUES ('100001', '이경오', 'D101' );
INSERT INTO TB_EMP_33 VALUES ('100002', '이수진', 'D101' );
INSERT INTO TB_EMP_33 VALUES ('100003', '권수철', 'D102' );
INSERT INTO TB_EMP_33 VALUES ('100004', '이지은', 'D102' );
INSERT INTO TB_EMP_33 VALUES ('100005', '정수라', 'D103' );
INSERT INTO TB_EMP_33 VALUES ('100006', '김연정', NULL );

COMMIT;
```

```
ALTER TABLE TB_EMP_33
ADD CONSTRAINTS TB_EMP_33_FK FOREIGN KEY (DEPT_CD)
REFERENCES TB_DEPT_33(DEPT_CD);
```

```
DROP FUNCTION FN_EMP_CNT_BY_DEPT_CD_33;

CREATE OR REPLACE FUNCTION FN_EMP_CNT_BY_DEPT_CD_33(IN_DEPT_CD IN
TB_DEPT_33.DEPT_CD%TYPE)
RETURN NUMBER IS V_EMP_CNT NUMBER;

BEGIN
SELECT COUNT(*) CNT
    INTO V_EMP_CNT
    FROM TB_EMP_33
    WHERE DEPT_CD = IN_DEPT_CD
        OR NVL(DEPT_CD, '0000') = NVL(IN_DEPT_CD, '0000')
;

RETURN V_EMP_CNT;
END;
/
;
```

문제33.

아래와 같이 TB_DEPT_33, TB_EMP_33 테이블을 생성하고 데이터를 입력하였다. 그 후 FN_EMP_CNT_BY_DEPT_CD_33이라는 사용자 정의 함수를 생성 하였으며 아래의 <SQL문>을 실행하였을 경우 해당 SQL문의 결과집합으로 가장 적절한 것은 무엇인가?

<SQL문>

```
SELECT A.EMP_NO
      , A.DEPT_CD
      , B.DEPT_NM
      , FN_EMP_CNT_BY_DEPT_CD_33(A.DEPT_CD) EMP_CNT
FROM   TB_EMP_33 A
      , TB_DEPT_33 B
WHERE  A.DEPT_CD = B.DEPT_CD(+)
ORDER BY A.EMP_NO
;
```

①

EMP_NO	DEPT_CD	DEPT_NM	EMP_CNT
100001	D101	데이터시각화팀	2
100002	D101	데이터시각화팀	2
100003	D102	데이터엔지니어링팀	2
100004	D102	데이터엔지니어링팀	2
100005	D103	데이터분석팀	1

②

EMP_NO	DEPT_CD	DEPT_NM	EMP_CNT
100001	D101	데이터시각화팀	2
100002	D101	데이터시각화팀	2
100003	D102	데이터엔지니어링팀	2
100004	D102	데이터엔지니어링팀	2
100005	D103	데이터분석팀	1
100006	NULL	NULL	1

③

EMP_NO	DEPT_CD	DEPT_NM	EMP_CNT
100001	D101	데이터시각화팀	2
100002	D101	데이터시각화팀	2
100003	D102	데이터엔지니어링팀	2
100004	D102	데이터엔지니어링팀	2
100005	D103	데이터분석팀	1
100006	NULL	NULL	0

④

EMP_NO	DEPT_CD	DEPT_NM	EMP_CNT
100001	D101	데이터시각화팀	3
100002	D101	데이터시각화팀	3
100003	D102	데이터엔지니어링팀	3
100004	D102	데이터엔지니어링팀	3
100005	D103	데이터분석팀	2
100006	NULL	NULL	2

문제34.

아래는 프로시저와 트리거의 차이점에 대한 설명이다. 아래의 설명에서 ㉠, ㉡을 채우시오. (단, 정답은 영문으로 기재 하시오.)

- 1) 프로시저는 CREATE PROCEDURE 명령으로 생성하고 트리거는 CREATE TRIGGER 명령으로 생성한다.
- 2) 프로시저는 EXECUTE 명령어로 실행하고 트리거는 생성 후 자동으로 실행된다.
- 3) 프로시저는 프로시저내에서 ㉠과 ㉡이 실행가능하고 트리거는 트리거내에서 ㉠과 ㉡이 실행 불가능하다.

문제35. 다음 중 비용기반 옵티마이저에 대한 설명 중 가장 부적절한 것을 2개 고르시오.

- ① WHERE절에 조건에서 'BETWEEN'보다 '='조건이 적은 일의 양을 차지한다고 판단한다.
- ② 비용기반 옵티마이저는 SQL문을 처리하는데 필요한 비용이 가장 적은 실행계획을 선택한다.
- ③ 비용기반 옵티마이저는 비용을 예측하기 위해서 테이블, 인덱스, 칼럼 등의 다양한 객체 통계 정보와 시스템 통계정보 등을 이용한다.
- ④ 통계 정보가 없는 경우 비용기반 옵티마이저는 정확한 비용 예측이 불가능해서 실행계획을 생성할 수 없다.

SQLD 출제 예상문제 - 2회
과목 2. SQL 기본 및 활용
2-8. 36번~40번

문제36. 아래의 SQL문에 대한 실행 계획에 대한 설명으로 가장 적절하지 않은 것은 무엇인가?

<SQL문>

```
SELECT ENAME
FROM EMP
WHERE JOB = 'SALESMAN'
AND SAL BETWEEN 3000 AND 6000;
```

<실행 계획>

```
*****[Explain Plan Time: 2020/09/07 11:26:24]*****
Execution Plan
-----
0      SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2 Card=2 Bytes=36)
1    0      TABLE ACCESS (BY INDEX ROWID BATCHED) OF 'EMP' (TABLE) (Cost=2 Card=2 Bytes=36)
2    1      INDEX (RANGE SCAN) OF 'IDX_EMP_01' (INDEX) (Cost=1 Card=3)
-----

Predicate information (identified by operation id):
-----
1 - filter("SAL">=3000 AND "SAL"<=6000)
2 - access("JOB"='SALESMAN')
-----
```

- ① 3번째 줄의 내용으로 봤을 때 IDX_EMP_01 인덱스를 인덱스 범위 스캔을 하고 있다.
- ② 2번째 줄의 'BY INDEX ROWID'로 인덱스 스캔을 통한 Table Random Access를 하고 있음을 알 수 있다.
- ③ 1번째 줄의 'ALL_ROWS'로 봤을 때 결과 집합의 건수는 2건 이상임을 알 수 있다.
- ④ 'Predicate information' 의 access("JOB"='SALESMAN') 로 봤을 때 JOB칼럼을 선두로 하는 인덱스를 스캔하고 있음을 알 수 있다.

문제37.

아래와 같이 TB_EMP_37 테이블을 생성 후 데이터를 입력하였다. 빈번하게 수행되는 SQL문의 성능을 향상 시키기 위해서 인덱스 생성을 고려 중이다. 다음 중 이 SQL문을 실행하는데 가장 효율적인 인덱스 구성은 무엇인가?

```
DROP TABLE TB_EMP_37;

CREATE TABLE TB_EMP_37
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, SEX_CD CHAR(2) NOT NULL
, BIRTH_DT CHAR(8) NOT NULL
, JOB_CD CHAR(4) NULL
, DEPT_CD CHAR(4) NULL
, CONSTRAINT TB_EMP_37_PK PRIMARY KEY (EMP_NO)
)
;

INSERT INTO TB_EMP_37
SELECT
  LPAD(ROWNUM, 6, '0') AS EMP_NO
, DBMS_RANDOM.STRING('U', 6)
, TO_CHAR(CEIL(MOD(DBMS_RANDOM.VALUE(1, 1000), 2)))
, TO_CHAR(SYSDATE - DBMS_RANDOM.VALUE(1, 3650), 'YYYYMMDD')
, LPAD(TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(1, 10))), 4, '0')
, LPAD(TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(1, 10))), 4, '0')
FROM DUAL CONNECT BY LEVEL <= 100000;

COMMIT;
```

<SQL문>

```
SELECT *
FROM TB_EMP_37
WHERE BIRTH_DT BETWEEN '20170101' AND '20191231'
AND DEPT_CD = '0001'
AND JOB_CD = '0002'
AND SEX_CD <> '1'
;
```

- ① SEX_CD+JOB_CD+DEPT_CD+BIRTH_DT
- ② JOB_CD+BIRTH_DT+DEPT_CD+SEX_CD
- ③ BIRTH_DT+DEPT_CD+JOB_CD+SEX_CD
- ④ DEPT_CD+JOB_CD+BIRTH_DT

문제38. [아래]와 같이 TB_EMP_38 테이블을 구성하고 인덱스를 생성하였다. 이러한 상황에서 아래와 같은 <SQL문>에 대한 실행계획을 출력하였다. 해당 실행계획을 보고 'IDX_TB_EMP_38_01' 인덱스의 인덱스 구성칼럼으로 가장 적절한 것은 무엇인가?

```
DROP TABLE TB_EMP_38;

CREATE TABLE TB_EMP_38
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, SEX_CD CHAR(2) NOT NULL
, BIRTH_DT CHAR(8) NOT NULL
, JOB_CD CHAR(4) NULL
, DEPT_CD CHAR(4) NULL
, CONSTRAINT TB_EMP_38_PK PRIMARY KEY (EMP_NO)
)
;

INSERT INTO TB_EMP_38
SELECT
  LPAD(ROWNUM, 6, '0') AS EMP_NO
, DBMS_RANDOM.STRING('U', 6)
, TO_CHAR(CEIL(MOD(DBMS_RANDOM.VALUE(1, 1000), 2)))
, TO_CHAR(SYSDATE - DBMS_RANDOM.VALUE(1, 3650), 'YYYYMMDD')
, LPAD(TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(1, 10))), 4, '0')
, LPAD(TO_CHAR(TRUNC(DBMS_RANDOM.VALUE(1, 10))), 4, '0')
FROM DUAL CONNECT BY LEVEL <= 100000;

COMMIT;
```

<SQL문>

```
SELECT /*+ INDEX(TB_EMP_38 IDX_TB_EMP_38_01) */
      *
FROM TB_EMP_38
WHERE BIRTH_DT BETWEEN '20200101' AND '20201231'
      AND SEX_CD = '1'
      AND JOB_CD = '0007'
      AND DEPT_CD = '0007'
;
```

```
*****[Explain Plan Time: 2020/09/08 09:23:20]*****
Execution Plan
-----
0      SELECT STATEMENT Optimizer=HINT: FIRST_ROWS (Cost=7K Card=49 Bytes=3K)
1    0      TABLE ACCESS (BY INDEX ROWID BATCHED) OF 'TB_EMP_38' (TABLE) (Cost=7K Card=49 Bytes=3K)
2    1      INDEX (RANGE SCAN) OF 'IDX_TB_EMP_38_01' (INDEX) (Cost=22 Card=6K)
-----

Predicate information (identified by operation id):
-----
1 - filter("SEX_CD"='1' AND "JOB_CD"='0007' AND "DEPT_CD"='0007')
2 - access("BIRTH_DT">='20200101' AND "BIRTH_DT"<='20201231')
```

- ① SEX_CD+JOB_CD+DEPT_CD
- ② SEX_CD+JOB_CD+DEPT_CD+BIRTH_DT
- ③ BIRTH_DT+SEX_CD+JOB_CD+DEPT_CD
- ④ BIRTH_DT

문제39. 아래는 HASH조인의 연산과정을 설명하고 있다. 다음 중 HASH조인의 연산 순서로 가장 적절한 것은 무엇인가?

- (1) OUTER 집합의 조인 키를 기준으로 HASH 함수를 적용하여 HASH 테이블 생성 (조인칼럼과 필요 칼럼 함께 저장)
- (2) OUTER 집합에서 주어진 조건을 만족하는 행을 찾음
- (3) HASH 테이블에 모든 대상 집합이 들어갈때까지 반복함
- (4) INNER 집합의 조인 키를 기준으로 HASH 함수를 적용하여 해당 버킷을 찾음
- (5) INNER 집합에서 주어진 조건을 만족하는 행을 찾음
- (6) 조인에 성공하면 해당 ROW를 결과집합에 포함
- (7) INNER 집합에서 모든 대상건을 찾을때까지 해당 과정 반복

- ① (2) -> (1) -> (3) -> (5) -> (4) -> (6) -> (7)
- ② (1) -> (2) -> (3) -> (5) -> (4) -> (6) -> (7)
- ③ (1) -> (5) -> (2) -> (3) -> (4) -> (6) -> (7)
- ④ (5) -> (1) -> (2) -> (3) -> (4) -> (6) -> (7)

문제40. 다음 아래의 실행 계획에 대한 설명 중 가장 부적절한 것을 고르시오.

```
*****[Explain Plan Time: 2020/09/08 17:41:09]*****
Execution Plan
-----
0      SELECT STATEMENT Optimizer=ALL_ROWS (Cost=2 Card=1 Bytes=58)
1    0      NESTED LOOPS (Cost=2 Card=1 Bytes=58)
2      1      TABLE ACCESS (BY INDEX ROWID) OF 'EMP' (TABLE) (Cost=1 Card=1 Bytes=38)
3      2      INDEX (UNIQUE SCAN) OF 'PK_EMP' (INDEX (UNIQUE)) (Cost=0 Card=1)
4      1      TABLE ACCESS (BY INDEX ROWID) OF 'DEPT' (TABLE) (Cost=1 Card=1 Bytes=20)
5      4      INDEX (UNIQUE SCAN) OF 'PK_DEPT' (INDEX (UNIQUE)) (Cost=0 Card=1)
-----

Predicate information (identified by operation id):
-----
3 - access("A"."EMPNO"=7369)
5 - access("A"."DEPTNO"="B"."DEPTNO")
-----
```

- ① NL조인의 OUTER 집합과 INNER 집합을 조회 시 모두 UNIQUE SCAN을 한 것으로 보아 결과 건수는 1건 이하이다.
- ② NL조인을 수행하고 있으며 EMP 테이블이 드라이빙 테이블이다.
- ③ 실행 순서상 가장 먼저 PK_EMP 인덱스에 대한 UNIQUE INDEX 스캔을 한다.
- ④ 'Optimizer=ALL_ROWS' 인 것으로 보아 NL조인에서 매칭되는 모든 집합이 출력된다.