

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

문제1. 다음 중 데이터 독립성 구성요소에 대한 설명으로 가장 부적절한 것은 무엇인가?

- ① 외부 스키마(External Schema)는 DB의 사용자 및 응용프로그램이 접근하는 스키마를 말한다.
- ② 개념 스키마(Conceptual Schema)는 모든 사용자 관점을 통합한 조직 전체의 DB를 기술하는 것을 말한다.
- ③ 내부 스키마(Internal Schema)는 물리적 장치에서 데이터가 실제로 저장되는 방법을 표현 하는 것을 말한다.
- ④ 외부/개념/내부 스키마 모두 상호 독립적의 의미를 가지고 있지만 고유한 기능을 가지고 있지는 않다.

정답 ④

※ 데이터베이스 스키마 구조는 외부, 개념, 내부 스키마 3단계로 구분되고 각각은 상호 독립적인 의미를 가지고 고유한 기능을 가진다.

문제2. 다음 중 유형 엔터티이면서 기본/키 엔터티이기도 한 엔터티는 무엇인가?

- ① 수강신청
- ② 교수
- ③ 학부
- ④ 수강신청이력

정답 ②

- ① 수강신청은 사건 엔터티이면서 중심엔터티이다.
- ③ 학부는 개념 엔터티이면서 기본/키 엔터티이다.
- ④ 수강신청이력은 사건 엔터티이면서 행위 엔터티이다.

유형 엔터티 (유무형에 따른 분류)	❖ 물리적인 형태가 있고 안정적이며 지속적으로 활용되는 엔터티(EX. 사원, 물품, 강사, 교수)
기본/키 엔터티 (발생시점에 따른 분류)	❖ 그 업무에 원래 존재하는 정보로써 다른 엔터티와 관계에 의해 생성되지 않고 독립적으로 생성이 가능한 엔터티(EX. 사원, 부서, 고객, 교수)

문제3. 다음 중 엔터티, 인스턴스, 속성, 속성값에 대한 관계에 대한 설명으로 알맞은 것을 2개 고르시오.

- ① 한개의 엔터티는 1개 이상의 인스턴스 집합 이어야 한다.
- ② 한개의 엔터티는 2개 이상의 인스턴스 집합 이어야 한다.
- ③ 한개의 속성은 1개의 속성값을 갖는다.
- ④ 한개의 속성은 2개의 속성값을 갖는다.

정답 ②, ③

- ① 한개의 엔터티는 2개 이상의 인스턴스 집합 이어야 한다.
- ④ 한개의 속성은 1개의 속성값을 갖는다.

문제4. [아래]는 신규로 구축하는 시스템의 업무 내용 중 일부분이다. [아래]의 설명에 부합하는 데이터 모델은 무엇인가?

- 1) 한 명의 고객은 여러 개의 증권계좌 개설이 가능하고, 하나의 증권계좌에는 여러 번에 걸쳐 입금이 가능하다.
- 2) 증권 계좌는 반드시 고객이 있어야 하고, 입금에는 반드시 증권 계좌가 있어야 한다.
- 3) 한 명의 고객은 증권계좌를 단 한 개도 개설하지 않을 수 있다.
- 4) 한개의 증권 계좌는 단 한번도 입금을 하지 않을 수 있다.



정답 ③

※ 관계 선택 사양을 물어보는 문제이다. 3번 보기가 관계의 선택 사양을 정확하게 표현하고 있다.

문제5. 다음 중 주 식별자의 4대 특징에 해당하지 않는 것은 무엇인가?

- ① 유일성
- ② 최소성
- ③ 가변성
- ④ 존재성

정답 ③

※ 가변성이라는 특징은 존재하지 않는다.

➤ 주 식별자의 4대 특징

유일성	❖ 주 식별자에 의해 엔터티 내에 모든 인스턴스들을 유일하게 구분할 수 있음
최소성	❖ 주 식별자를 구성하는 속성의 수는 유일성을 만족하는 최소의 수가 되어야함
불변성	❖ 주 식별자가 한번 특정 엔터티에 지정되면 그 식별자의 값은 변하지 않아야 함
존재성	❖ 주 식별자가 지정되면 반드시 데이터 값이 존재 해야함

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

문제6. [아래]의 일자별매각물건 엔터티는 다음과 같은 <함수적 종속성>을 갖는다. 이러한 경우에 해당 엔터티는 몇 정규형인가?
또한 몇 정규화 대상인가?

<데이터 모델>

일자별매각물건

- ☐ # 매각물건번호
- ☐ # 매각일자
- ☐ * 매각시간
- ☐ * 매각장소
- ☐ * 최저매각가격
- ☐ * 물건상태코드

<함수적 종속성>

매각물건번호 -> (최저매각가격, 물건상태코드)
매각일자 -> (매각시간, 매각장소)

- ① 2정규형, 3정규화대상
- ② 1정규형, 2정규화대상
- ③ 3정규형, 보이스코드정규화대상
- ④ 반정규형, 2정규화대상

정답 ②

② 일자별매각물건 엔터티는 1정규형을 만족하며 2정규화 대상이다. 매각일자가 PK속성 중 하나이므로 부분 함수 종속이 발생하였다.

문제7. 반정규화는 중복성의 원리를 활용하여 데이터 조회 시 성능을 향상 시키는 역할을 한다. 다음 중 반정규화의 중복성의 유형이 아닌 것은 무엇인가?

- ① 속성의 중복성
- ② 테이블의 중복성
- ③ 관계의 중복성
- ④ 칼럼의 중복성

정답 ①

① 속성의 중복성은 존재하지 않는다.

➤ 반정규화의 종류

테이블 반정규화	❖ 테이블 병합, 테이블 분할, 테이블 추가
관계의 반정규화	❖ 중복관계 추가
칼럼 반정규화	❖ 중복칼럼 추가, 파생칼럼 추가, 이력테이블칼럼 추가, PK에 의한 칼럼 추가, 응용시스템 오작동을 위한 칼럼 추가

문제8.

한 테이블에 칼럼이 과도하게 많은 경우 하나의 행이 디스크에 여러 블록에 데이터가 저장되게 된다. 이러한 경우 과도한 블록 I/O가 발생하여 성능이 저하될 수 있다. 해당 현상에 대해 가장 올바르게 설명한 것은 무엇인가?

- ① 로우 길이가 너무 길어서 데이터 블록 하나에 모두 저장되지 않고 두개 이상의 블록에 걸쳐 하나씩 로우가 저장되어 있는 형태가 로우 마이그레이션 현상이다.
- ② 데이터 블록에서 수정이 발생하면 수정된 데이터를 해당 데이터 블록에서 저장하지 못하고 다른 블록의 빈 공간을 찾아가는 방식이 로우 체이닝 이라고 한다.
- ③ 로우 체이닝 현상이 일어나면 많은 블록에 데이터가 저장되면서 불필요한 BLOCK I/O가 발생한다.
- ④ 로우 마이그레이션이 일어나면 많은 블록에 데이터가 저장되지만 공간 효율성은 높아진다.

정답

③

※ 3번 보기가 가장 올바른 설명이다.

- ① 로우 체이닝에 대한 설명이다.
- ② 로우 마이그레이션에 대한 설명이다.
- ④ 로우 마이그레이션이 일어나면서 공간 효율성이 높아지는 것은 아니다.

문제9. 다음 중 슈퍼/서브타입 데이터 모델의 변환 타입에 대한 설명으로 가장 알맞은 것은 무엇인가?

- ① OneToOne 타입은 개별 테이블을 유지하기 때문에 관리용이성이 뛰어나다.
- ② Plus 타입은 슈퍼+서브타입 테이블로 변환하기 때문에 관리용이성이 뛰어나다.
- ③ Single 타입은 하나의 테이블로 통합하기 때문에 관리용이성이 좋지 않다.
- ④ Single 타입은 하나의 테이블로 통합하기 때문에 향후 확장성이 떨어진다.

정답 ④

- ① OneToOne 타입은 관리용이성이 좋지 않다. (테이블 여러 개)
- ② Plus 타입은 관리용이성이 좋지 않다. (테이블 여러 개)
- ③ Single 타입은 관리용이성이 좋다. (테이블 1개)

➤ 슈퍼/서브타입 데이터 모델 변환 타입 비교

구분	OntoOne Type	Plus Type	Single Type
구성 상의 특징	개별테이블유지	슈퍼+서브타입 테이블	하나의 테이블
확장성	우수함	보통	나쁨
조인 성능	나쁨	나쁨	우수함
I/O량 성능	좋음	좋음	나쁨
관리 용이성	좋지않음	좋지않음	좋음
트랜잭션 유형에 따른 선택 방법	개별 테이블로 접근이 많은 경우 선택	슈퍼+서브타입 형식으로 데이터를 처리하는 경우 선택	전체를 일괄적으로 처리하는 경우 선택

문제10. 분산 데이터베이스 적용 시 장단점에 대한 설명으로 가장 올바른 것은 무엇인가?

- ① 여러 곳에 데이터가 분산되므로 응답속도가 느려지고 통신 비용이 증가한다.
- ② 분산 적용되어 데이터의 가용성과 신뢰성이 감소한다.
- ③ 분산 데이터베이스의 특성으로 데이터 무결성을 확실하게 보장한다.
- ④ 분산 환경에 따라 불규칙한 응답속도가 단점이다.

정답 ④

- ① 분산 데이터베이스 적용 시 빠른 응답속도와 통신 비용이 절감된다.
- ② 분산 데이터베이스 적용 시 신뢰성과 가용성이 증대된다.
- ③ 분산 데이터베이스 적용 시 데이터 무결성은 위협을 받을 수 있다.

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

문제1. 다음 중 DML(Data Manipulation Language)의 종류가 아닌 것은 무엇인가?

- ① INSERT
- ② UPDATE
- ③ SELECT
- ④ RENAME

정답 ④

※ RENAME은 DDL(Data Definition Language)의 한 종류이다.

문제2.

CREATE문과 SELECT문을 조합하여 기존 테이블을 복제할 수 있다. (오라클 기준)
[아래]의 <SQL문>의 ㉠안에 들어갈 알맞은 키워드는 무엇인가?

<SQL문>

```
CREATE TABLE TEAM_TEMP ㉠  
SELECT * FROM TEAM;
```

- ① ㉠ : IS
- ② ㉠ : AS
- ③ ㉠ : OF
- ④ ㉠ : IN

정답 ②

※ CREATE TABLE AS SELECT 이며 줄여서 CTAS라고도 한다.

문제3. [아래]와 같은 테이블에서 <SQL1>, <SQL2>, <SQL3> 각각의 SQL문의 결과 집합의 건수를 순서대로 기재한 것은 무엇인가?

SELECT * FROM TB_PLAYER_3;

PLAYER_ID	PLAYER_NM	BIRTH_DE
100001	박찬호	19730629
100002	박찬호	19950605
100003	박지성	19810225
100004	이승우	19980106

<SQL1>

SELECT PLAYER_NM FROM TB_PLAYER_3;

<SQL2>

SELECT ALL PLAYER_NM FROM TB_PLAYER_3;

<SQL3>

SELECT DISTINCT PLAYER_NM FROM TB_PLAYER_3;

- ① 4, 3, 3
- ② 4, 4, 4
- ③ 4, 3, 4
- ④ 4, 4, 3

정답

④

- ※ DISTINCT는 유일 값을 출력하고 ALL은 모든 값을 출력한다.
- ※ 아무것도 기재하지 않으면 ALL을 생략한 것과 같다.

<테스트>

```
DROP TABLE TB_PLAYER_3;
CREATE TABLE TB_PLAYER_3
(
    PLAYER_ID CHAR(6)
, PLAYER_NM VARCHAR2(50)
, BIRTH_DE CHAR(8)
)
;

INSERT INTO TB_PLAYER_3 VALUES ('100001', '박찬호', '19730629');
INSERT INTO TB_PLAYER_3 VALUES ('100002', '박찬호', '19950605');
INSERT INTO TB_PLAYER_3 VALUES ('100003', '박지성', '19810225');
INSERT INTO TB_PLAYER_3 VALUES ('100004', '이승우', '19980106');
COMMIT;
```


문제4. 다음 중 트랜잭션의 특성에 대한 설명으로 가장 부적절한 것은 무엇인가?

- ① 원자성(Atomicity)은 트랜잭션에서 정의된 연산들은 모두 성공적으로 실행되든지 아니면 전혀 실행되지 않은 상태로 남아 있어야 한다.
- ② 일관성(Consistency)은 트랜잭션이 실행되기 전의 데이터베이스 내용이 잘못되어 있지 않다면 트랜잭션이 실행된 이후에도 데이터베이스 내용에 잘못이 있으면 안된다.
- ③ 고립성(Isolation)은 트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을 받으면 즉시 작업을 중지하여 잘못된 결과를 만들지 않는다.
- ④ 지속성(Durability)은 트랜잭션이 성공적으로 수행되면 그 트랜잭션이 갱신한 데이터베이스의 내용은 영구적으로 저장된다.

정답 ③

※ 고립성은 트랜잭션이 실행되는 도중에 다른 트랜잭션의 영향을 받아 잘못된 결과를 만들어서는 안된다.

문제5.

다음 [아래]의 테이블 구성에서 MAJOR(전공)가 '데이터사이언스학'이면서 DEPT_CD(부서코드)가 '101'인 직원 혹은 MAJOR(전공)가 '컴퓨터공학'이면서 DEPT_CD(부서코드)가 '102'인 직원을 추출하는 SQL문으로 옳바르지 않는 것을 2개 고르시오.

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

SELECT A.*
FROM TB_EMP_5 A
WHERE (A.MAJOR, A.DEPT_CD) IN (
('데이터사이언스학', '101'), ('컴퓨터공학', '102')
);

④

SELECT A.*
FROM TB_EMP_5 A
WHERE (A.MAJOR, A.DEPT_CD) NOT IN (
('컴퓨터소프트웨어학', '101'), ('컴퓨터소프트웨어학', '102')
);

①

SELECT A.*
FROM TB_EMP_5 A
WHERE
(A.MAJOR = '데이터사이언스학'
AND A.DEPT_CD = '101'
)
OR
(A.MAJOR = '컴퓨터공학'
AND A.DEPT_CD = '102'
);

②

SELECT A.*
FROM TB_EMP_5 A
WHERE A.MAJOR IN ('데이터사이언스학', '컴퓨터공학')
AND A.DEPT_CD IN ('101', '102');

<테스트>

DROP TABLE TB_EMP_5;
CREATE TABLE TB_EMP_5
(
EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50)
, MAJOR VARCHAR2(150)
, DEPT_CD CHAR(3)
);

INSERT INTO TB_EMP_5 VALUES ('100001', '이경오', '컴퓨터소프트웨어학', '101');
INSERT INTO TB_EMP_5 VALUES ('100002', '이수지', '데이터사이언스학', '101');
INSERT INTO TB_EMP_5 VALUES ('100003', '이지수', '컴퓨터공학', '101');
INSERT INTO TB_EMP_5 VALUES ('100004', '김성민', '컴퓨터공학', '102');
INSERT INTO TB_EMP_5 VALUES ('100005', '김민선', '데이터사이언스학', '102');
INSERT INTO TB_EMP_5 VALUES ('100006', '김선미', '컴퓨터소프트웨어학', '102');
COMMIT;

정답

②, ④

- ② 전공이 '데이터사이언스학' 혹은 '컴퓨터공학'이면서 부서코드가 '101' 혹은 '102'이면 모두 출력된다. (오답)
- ④ 전공이 '컴퓨터소프트웨어학'이면서 부서코드가 '101' 혹은 '102'인 직원은 집합에서 제외된다. (오답)

※ 3번 보기는 다중 리스트를 이용한 IN 연산자의 사용으로써 SQL 문장을 짧게 만들어주고 성능상 유리한 점이 많다.

SQLD 출제 예상문제 - 1회
과목 2. SQL 기본 및 활용
2-2. 6번~10번

문제6. [아래]와 같이 TB_CHAR_VARCHAR_6 테이블을 생성하고 데이터를 입력하였다. <SQL문>의 실행 결과는 무엇인가?

```
DROP TABLE TB_CHAR_VARCHAR_6;

CREATE TABLE TB_CHAR_VARCHAR_6
(
  CHAR_VAL CHAR(10)
, VARCHAR_VAL VARCHAR2(10)
)
;

INSERT INTO TB_CHAR_VARCHAR_6 VALUES ( 'ABC'      , 'ABC'      ); --공백없음
INSERT INTO TB_CHAR_VARCHAR_6 VALUES ( 'ABCDE'    , 'ABCDE'    ); --공백없음
INSERT INTO TB_CHAR_VARCHAR_6 VALUES ( 'ABCDEFGH I J', 'ABCDEFGH I J'); --공백없음
INSERT INTO TB_CHAR_VARCHAR_6 VALUES ( 'ABC '     , 'ABC '     ); --공백이 2칸씩
INSERT INTO TB_CHAR_VARCHAR_6 VALUES ( 'ABCDE '   , 'ABCDE '   ); --공백이 5칸씩

COMMIT;
```

<SQL문>

```
SELECT COUNT(*) CNT
FROM TB_CHAR_VARCHAR_6 A
WHERE A.CHAR_VAL = A.VARCHAR_VAL
;
```

- ① 2
- ② 3
- ③ 4
- ④ 5

정답 ①

※ CHAR와 VARCHAR 비교 시 길이가 다르면 다른 값으로 판단한다. 즉 두 문자열의 길이가 동일하고 문자열도 같아야 같다고 판단한다.

※ 즉 아래의 데이터가 같다고 판단한다.

```
INSERT INTO TB_CHAR_VARCHAR_6 VALUES ( 'ABCDEFGH I J', 'ABCDEFGH I J'); --공백없음
INSERT INTO TB_CHAR_VARCHAR_6 VALUES ( 'ABCDE '   , 'ABCDE '   ); --공백이 5칸씩
```

문제7. [아래]는 오라클의 DUAL 테이블을 이용하여 내장 함수를 호출하는 <SQL문>이다. 이 <SQL문>이 출력하는 결과는 무엇인가?

<SQL문>

```
SELECT ABS(CEIL(3.14) + FLOOR(3.14) * SIGN(-3.14)) AS RESULT_VAL  
FROM DUAL  
;
```

- ① -1
- ② 1
- ③ 7
- ④ -7

정답 ②

※ CEIL(3.14) : 4
※ FLOOR(3.14) : 3
※ SIGN(-3.14) : -1
※ 연산자 우선 순위에 의해 $3 * -1 = -3$ 이 먼저 계산됨
※ 그후 $4 + -3 = 1$
※ 즉 결과는 $1 = 4 + (3*-1)$ 이 됨

<테스트>

```
SELECT ABS(CEIL(3.14) + FLOOR(3.14) * SIGN(-3.14)) AS RESULT_VAL  
      , CEIL(3.14) — 4  
      , FLOOR(3.14) — 3  
      , SIGN(-3.14) — -1  
FROM DUAL  
;
```

문제8.

[아래]와 같이 DUAL 테이블에는 DUMMY라는 칼럼 값이 있고 'X'라는 값이 저장되어 있다. [아래] <SQL문>의 실행 결과 값으로 가장 적절한 것은 무엇인가?

```
SELECT DUMMY FROM DUAL;
```

DUMMY

X

<SQL문>

```
SELECT NVL(MAX(DUMMY), 'DUMMY') AS RESULT_VAL  
FROM DUAL  
WHERE 1=0  
;
```

- ① 공집합
- ② DUMMY
- ③ X
- ④ NULL리턴

정답

②

- ※ WHERE절의 1=0 조건으로 인해 공집합이 리턴 되는 상황에서 MAX(DUMMY)의 값은 NULL이 리턴 된다.
- ※ NVL함수로 인해 नी이면 'DUMMY'라는 값을 리턴 하게 된다.

문제9.

[아래]와 같이 TB_EMP_9 테이블을 생성하고 데이터를 입력하였다. 아래의 <SQL문>의 실행 결과를 순서대로 올바르게 기재한 것은 무엇인가?

```
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 COUNT(DEPT_CD)
      , COUNT(*)
      , COUNT(DISTINCT DEPT_CD)
FROM TB_EMP_9
;
```

- ① 10, 10, 5
- ② 10, 10, 4
- ③ 8, 10, 4
- ④ 8, 10, 5

정답

③

- ※ COUNT(DEPT_CD)는 NULL이 아닌 모든 건수를 리턴 8
- ※ COUNT(*)은 모든 행의 건수를 리턴 10
- ※ COUNT(DISTINCT DEPT_CD)는 DEPT_CD의 유일 값의 개수를 리턴 4

문제10.

[아래]와 같이 TB_PLAYER_10 테이블을 생성하고 데이터를 입력하였다. 해당 테이블에서 포지션(POS_NM)별 평균 신장이 180이상인 포지션의 정보를 추출하고 평균신장값을 기준으로 역순 정렬 하고자 한다. 다음 중 결과 집합이 다른 하나를 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;
```

정답

④

- ④ 가장 마지막에 'ORDER BY 1 DESC'을 함으로써 POS_NM기준으로 역순 정렬을 하게 된다. 그래서 다른 결과 집합이 도출된다.
- ② HAVING절이 GROUP BY절보다 먼저 나와도 아무런 상관이 없다.

①

```
SELECT A.POS_NM, AVG(HEIGHT) HEIGHT_AVG
FROM TB_PLAYER_10 A
GROUP BY A.POS_NM
HAVING AVG(A.HEIGHT) >= 180
ORDER BY AVG(A.HEIGHT) DESC;
```

②

```
SELECT A.POS_NM, AVG(HEIGHT) HEIGHT_AVG
FROM TB_PLAYER_10 A
HAVING AVG(A.HEIGHT) >= 180
GROUP BY A.POS_NM
ORDER BY AVG(A.HEIGHT) DESC;
```

③

```
SELECT A.POS_NM, A.HEIGHT_AVG
FROM
(
  SELECT A.POS_NM, AVG(HEIGHT) HEIGHT_AVG
  FROM TB_PLAYER_10 A
  GROUP BY A.POS_NM
  ORDER BY AVG(A.HEIGHT)
) A
WHERE A.HEIGHT_AVG >= 180
ORDER BY 2 DESC;
```

④

```
SELECT A.POS_NM, A.HEIGHT_AVG
FROM
(
  SELECT A.POS_NM, AVG(HEIGHT) HEIGHT_AVG
  FROM TB_PLAYER_10 A
  GROUP BY A.POS_NM
  ORDER BY AVG(A.HEIGHT) DESC
) A
WHERE A.HEIGHT_AVG >= 180
ORDER BY 1 DESC;
```


SQLD 출제 예상문제 - 1회
과목 2. SQL 기본 및 활용
2-3. 11번~15번

문제11.

아래의 <SQL문>은 TB_PLAYER_11 테이블에서 포지션(POS_NM)별 평균키를 출력하면서 각 포지션별 선수 중 키가 185CM이상인 선수를 보유한 포지션 및 평균키를 추출하는 <SQL문>이다. ㉠에 들어갈 알맞은 키워드를 기재 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 A.POS_NM AS 포지션
      , AVG(HEIGHT) AS 평균키
FROM TB_PLAYER_11 A
GROUP BY A.POS_NM
HAVING ㉠ (A.HEIGHT) >= 185
;
```

정답 ㉠ : MAX

※ MAX 함수를 이용해서 각 포지션에서 최대 키가 185이상인 존재하는 포지션을 구한다.

문제12.

TB_EMP_SAL_12 테이블을 생성하고 각 사원의 연도별 연봉정보 데이터를 입력하였다. 다음 보기의 SQL문 중 2019년도 기준 연봉이 높은 순서로 해당 테이블을 조회하는 SQL문으로 가장 적절한 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 *  
    FROM TB_EMP_SAL_12 A  
    WHERE A.STD_YYYY = '2019'  
    ORDER BY A.SAL DESC  
;
```

②

```
SELECT *  
    FROM TB_EMP_SAL_12 A  
    WHERE A.STD_YYYY = '2019'  
    ORDER BY A.STD_YYYY, A.SAL DESC  
;
```

③

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

④

```
SELECT *  
    FROM TB_EMP_SAL_12 A  
    WHERE A.STD_YYYY = '2019'  
    ORDER BY NVL(A.SAL, 0) ASC  
;
```

정답 ③

※ SAL 칼럼에 NULL인 데이터가 존재하며 오라클은 NULL이 가장 큰 값이라고 생각한다.
※ 그러므로 NVL(A.SAL, 0)로 하여 널인 경우에 0으로 치환하고 역순 정렬하는 보기 3번이 정답이다.

문제13. 아래와 같이 TB_EMP_SAL_13 테이블을 생성 후 데이터를 입력하였다. 다음 중 SQL문의 결과 집합이 다른 하나는 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', '70000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100001', '2019', '65000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100001', '2018', '60000000');

INSERT INTO TB_EMP_SAL_13 VALUES ('100002', '2020', '60000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100002', '2019', '55000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100002', '2018', '50000000');

INSERT INTO TB_EMP_SAL_13 VALUES ('100003', '2020', '50000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100003', '2019', '45000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100003', '2018', '40000000');

INSERT INTO TB_EMP_SAL_13 VALUES ('100004', '2020', '40000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100004', '2019', '35000000');
INSERT INTO TB_EMP_SAL_13 VALUES ('100004', '2018', '30000000');

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;
```

- ①
SELECT STD_YYYY, SAL
FROM TB_EMP_SAL_13
GROUP BY STD_YYYY, SAL
HAVING SAL IS NOT NULL
ORDER BY STD_YYYY, SAL
;
- ②
SELECT STD_YYYY, SAL
FROM TB_EMP_SAL_13
GROUP BY STD_YYYY, SAL
HAVING COUNT(SAL) >= 1
ORDER BY STD_YYYY, SAL
;
- ③
SELECT STD_YYYY, SAL
FROM TB_EMP_SAL_13
GROUP BY STD_YYYY, SAL
HAVING COUNT(SAL) >= 2
ORDER BY STD_YYYY, SAL
;
- ④
SELECT STD_YYYY, SAL
FROM TB_EMP_SAL_13
GROUP BY STD_YYYY, SAL
HAVING SAL > 0
ORDER BY STD_YYYY, SAL
;

정답 ③

- ※ 그룹핑의 기준이 'GROUP BY STD_YYYY, SAL' 이므로 COUNT(SAL)의 값은 1 혹은 0이 나온다.
- ※ 보기 3번은 COUNT(SAL)이 2이상인 집합을 구하므로 아무런 집합도 출력되지 않는다. 즉 정답이다.
- ※ 보기 4번은 SAL > 0 조건으로 비교하고 있으므로 자연스레 SAL칼럼의 값이 NULL인 행은 집합에서 제외된다.

문제14. 다음 중 조인에 대한 설명으로 가장 부적절한 것을 2개 고르시오.

- ① 3개 이상의 집합을 조인 시 3개 이상의 집합에 대하여 동시에 조인 연산을 수행하여 결과 집합을 출력한다.
- ② 3개 이상의 집합을 조인 시 특정 시점에는 반드시 2개의 집합에 대해서만 조인 연산을 수행하여 결과 집합을 출력한다.
- ③ 2개 이상의 집합을 결합하여 데이터를 출력하는 것을 조인이라고 한다.
- ④ 조인은 PK, FK의 연관관계에 의해서만 이루어지며 PK, FK가 아니면 조인이 성립되지 않는다.

정답 ①, ④

※ 조인은 특정 시점에 단 2개의 집합 끼리 조인 연산이 이루어지며 일반 칼럼이라도 논리적으로 성립되면 조인 연산이 가능하다.

문제15.

아래와 같이 TB_EMP_15, TB_EMP_SAL_15, TB_SAL_GRADE_15 테이블을 생성하고 데이터를 입력하였다. 아래 SQL문의 결과는 무엇인가?

```
DROP TABLE TB_EMP_SAL_15;
DROP TABLE TB_EMP_15;
DROP TABLE TB_SAL_GRADE_15;

CREATE TABLE TB_EMP_15
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50) NOT NULL
, CONSTRAINT TB_EMP_15_PK PRIMARY KEY (EMP_NO)
);

INSERT INTO TB_EMP_15 VALUES ('100001', '이경오');
INSERT INTO TB_EMP_15 VALUES ('100002', '이수지');
COMMIT;

CREATE TABLE TB_EMP_SAL_15
(
  EMP_NO CHAR(6)
, SAL_STD_YYYY CHAR(8)
, SAL NUMBER
, CONSTRAINT TB_EMP_SAL_15_PK
  PRIMARY KEY (EMP_NO, SAL_STD_YYYY)
);

ALTER TABLE TB_EMP_SAL_15
ADD CONSTRAINTS TB_EMP_SAL_15_FK FOREIGN KEY (EMP_NO)
REFERENCES TB_EMP_15(EMP_NO);
```

```
INSERT INTO TB_EMP_SAL_15 VALUES ('100001', '2020', 75000000);
INSERT INTO TB_EMP_SAL_15 VALUES ('100001', '2019', 65000000);
INSERT INTO TB_EMP_SAL_15 VALUES ('100001', '2018', 55000000);
INSERT INTO TB_EMP_SAL_15 VALUES ('100002', '2020', 55000000);
INSERT INTO TB_EMP_SAL_15 VALUES ('100002', '2019', 35000000);
INSERT INTO TB_EMP_SAL_15 VALUES ('100002', '2018', 25000000);
COMMIT;

CREATE TABLE TB_SAL_GRADE_15
(
  SAL_GRADE CHAR(1)
, LOW_SAL NUMBER(10) NOT NULL
, HIGH_SAL NUMBER(10) NOT NULL
, CONSTRAINT TB_SAL_GRADE_15_PK PRIMARY KEY (SAL_GRADE)
);

INSERT INTO TB_SAL_GRADE_15 VALUES ('S', 100000000, 999999999);
INSERT INTO TB_SAL_GRADE_15 VALUES ('A', 80000000, 99999999 );
INSERT INTO TB_SAL_GRADE_15 VALUES ('B', 60000000, 79999999 );
INSERT INTO TB_SAL_GRADE_15 VALUES ('C', 40000000, 59999999 );
INSERT INTO TB_SAL_GRADE_15 VALUES ('D', 30000000, 39999999 );
INSERT INTO TB_SAL_GRADE_15 VALUES ('E', 20000000, 29999999 );
INSERT INTO TB_SAL_GRADE_15 VALUES ('F', 0, 19999999 );
COMMIT;
```

<SQL문>

```
SELECT COUNT(DISTINCT C.SAL_GRADE)
      AS CNT_SAL_GRADE
FROM TB_EMP_15 A
      , TB_EMP_SAL_15 B
      , TB_SAL_GRADE_15 C
WHERE A.EMP_NO = B.EMP_NO
      AND B.SAL BETWEEN C.LOW_SAL
                        AND C.HIGH_SAL
;
```

- ① 4
- ② 6
- ③ 42
- ④ 0

정답

①

※ 매칭되는 연봉등급(SAL_GRADE)은 B, B, C, C, D, E가 출력된다.
※ 이 중 유일 값의 개수는 B, C, D, E 총 4개이다.

<테스트>

```
SELECT C.SAL_GRADE AS CNT_SAL_GRADE
      , B.SAL
FROM TB_EMP_15 A
      , TB_EMP_SAL_15 B
      , TB_SAL_GRADE_15 C
WHERE A.EMP_NO = B.EMP_NO
      AND B.SAL BETWEEN C.LOW_SAL AND C.HIGH_SAL
ORDER BY 1;
```

SQLD 출제 예상문제 - 1회
과목 2. SQL 기본 및 활용
2-4. 16번~20번

문제16.

다음 중 E.F.CODD 박사의 논문에서 언급된 일반 집합 연산자와 현재 SQL언어의 기능을 매핑한 것으로 가장 적절한 것은 무엇인가?

- ① UNION 연산 -> JOIN 기능으로
- ② INTERSECT 연산 -> JOIN 기능으로
- ③ DIFFERENCE 연산 -> NOT EXISTS 기능으로
- ④ PRODUCT 연산 -> CROSS JOIN 기능으로

정답

④

- ※ UNION 연산 -> UNION 기능
- ※ INTERSECT 연산 -> INTERSECT 기능
- ※ DIFFERENCE 연산 -> EXCEPT/MINUS 기능
- ※ PRODUCT 연산 -> CROSS JOIN 기능

문제17. 다음 중 아래의 <SQL1>, <SQL2>, <SQL3>에 대한 설명으로 가장 적절한 것은 무엇인가?

<SQL1>

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

<SQL2>

```
SELECT EMP.DEPTNO, EMPNO, ENAME, DNAME
FROM EMP INNER JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO
;
```

<SQL3>

```
SELECT EMP.DEPTNO, EMPNO, ENAME, DNAME
FROM EMP JOIN DEPT
ON EMP.DEPTNO = DEPT.DEPTNO
;
```

- ① <SQL1>, <SQL2>, <SQL3>의 결과 집합은 데이터 상황에 따라 모두 같거나 다를 수 있다.
- ② <SQL2>는 'INNER JOIN'을 기재했기 때문에 EMP 테이블 먼저 읽은 후 DEPT 테이블과 조인을 수행한다.
- ③ <SQL2>는 'JOIN'을 기재했기 때문에 EMP 테이블 먼저 읽은 후 DEPT 테이블과 조인을 수행한다.
- ④ <SQL3>는 'INNER JOIN'에서 'INNER'를 생략한 상황이며 두개의 테이블 중 어떤 테이블을 먼저 읽을지 는 알 수 없다.

정답

④

※ <SQL3>은 INNER JOIN시 INNER를 생략한 SQL문이고 조인 순서는 옵티마이저의 선택에 따르게 된다.

문제18. 다음 중 집합연산자에 대한 설명으로 가장 적절한 것은 무엇인가?

- ① 여러 개의 질의 결과를 연결하여 하나로 결합하는 방식이므로 대용량 결과 질의에 적합하다.
- ② 집합연산자를 사용하는 상황은 반드시 서로 다른 테이블끼리의 연산이 이루어져야 한다.
- ③ SQL 튜닝 관점에서 실행계획을 분리하고자 하는 목적으로도 사용한다.
- ④ SELECT절의 칼럼 수가 동일하지 않아도 각 결과의 사이즈가 동일하면 된다.

정답 ③

- ① 대용량 결과 질의와는 상관없이 모두 사용가능하다.
- ② 같은 테이블끼리의 연산도 가능하다.
- ④ SELECT절의 칼럼 수가 동일해야 한다.

문제19. 아래와 같은 테이블 및 데이터 구성에서 <SQL문>의 결과집합으로 가장 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 DISTINCT A.DEPT_NO  
FROM TB_EMP_19 A  
WHERE A.DEPT_NO = 'D00002'  
UNION ALL  
SELECT A.DEPT_NO  
FROM TB_EMP_19 A  
WHERE A.DEPT_NO = 'D00001'  
ORDER BY DEPT_NO  
;
```

①

DEPT_NO

D00001
D00002

②

DEPT_NO

D00001
D00001
D00002

③

SQL
문법 에러

④

DEPT_NO

D00001
D00001
D00002
D00002

정답

②

- ※ 'DISTINCT A.DEPT_NO'의 DISTINCT는 첫번째 SELECT에서만 유효하다.
- ※ 즉 두번째 SELECT문에서는 DEPT_NO가 'D00001'에 해당하는 직원이 2명이므로 2건이 출력된다.

문제20. 다음 중 계층형 질의의 구문에 관한 설명 중 가장 부적절한 것은 무엇인가?

- ① START WITH절은 계층 구조 전개 of 시작 위치를 지정하는 구문이다.
- ② CONNECT BY절은 다음에 전개될 자식 데이터를 지정하는 구문이다.
- ③ ORDER SIBLINGS BY는 계층형 쿼리의 결과 집합에 대한 정렬 순서를 지정하는 구문이다.
- ④ WHERE은 계층형 쿼리의 모든 전개를 수행한 후에 만족하는 데이터를 추출한다.

정답 ③

※ ORDER SIBLINGS BY는 형제 노드(동일 LEVEL)사이에서 정렬을 수행한다.

SQLD 출제 예상문제 - 1회
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;
```

<테스트>

```
SELECT DEPT_NO
, SYS_CONNECT_BY_PATH(DEPT_NM, '/') AS DEPT_NM
, UPPER_DEPT_NO
FROM TB_DEPT_21
START WITH UPPER_DEPT_NO IS NULL
CONNECT BY PRIOR DEPT_NO = UPPER_DEPT_NO;
```

<SQL문>

```
SELECT DEPT_NO
, ㉠ (DEPT_NM, '/') AS DEPT_NM
, UPPER_DEPT_NO
FROM TB_DEPT_21
START WITH UPPER_DEPT_NO IS NULL
CONNECT BY PRIOR DEPT_NO = UPPER_DEPT_NO
;
```

<실행결과>

DEPT_NO	DEPT_NM	UPPER_DEPT_NO
D00001	/회장실	
D00002	/회장실/영업본부	D00001
D00004	/회장실/영업본부/국내영업부	D00002
D00008	/회장실/영업본부/국내영업부/기업영업팀	D00004
D00009	/회장실/영업본부/국내영업부/공공영업팀	D00004
D00005	/회장실/영업본부/해외영업부	D00002
D00010	/회장실/영업본부/해외영업부/북미영업팀	D00005
D00011	/회장실/영업본부/해외영업부/남미영업팀	D00005
D00003	/회장실/기술본부	D00001
D00006	/회장실/기술본부/개발사업부	D00003
D00012	/회장실/기술본부/개발사업부/서버개발팀	D00006
D00013	/회장실/기술본부/개발사업부/화면개발팀	D00006
D00007	/회장실/기술본부/데이터사업부	D00003
D00014	/회장실/기술본부/데이터사업부/오라클기술팀	D00007
D00015	/회장실/기술본부/데이터사업부/오픈소스기술팀	D00007

정답

SYS_CONNECT_BY_PATH

문제22. 다음 중 SQL문 내에서 서브 쿼리가 사용 가능한 위치로 가장 부적절한 것은 무엇인가?

- ① SELECT절
- ② ORDER BY절
- ③ UPDATE문의 SET절
- ④ INSERT문의 INTO절

정답 ④

※ 서브쿼리의 사용 가능 위치는 아래와 같다.

- 1) SELECT절
- 2) FROM절
- 3) WHERE절
- 4) HAVING절
- 5) ORDER BY절
- 6) INSERT문의 VALUES절
- 7) UPDATE문의 SET절

문제23.

아래와 같이 TB_EMP_23 및 TB_DEPT_23 테이블을 생성 하고 데이터를 입력하였다. 각 부서별 생일이 가장 빠른 사람(나이가 많은 사람)의 사원번호, 사원명, 부서명, 생일 출력하고 정렬 순서는 생일이 빠른 순으로 출력하고자 한다.
<실행결과>가 아래와 같을 때 다음 보기에서 제시하는 SQL문 중 조건에 부합하는 SQL문을 1개 고르시오.

```

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. 각 부서별 생일이 가장 빠른 사람(나이가 많은 사람)의 사원번호, 사원명, 부서명, 생일 출력하고 정렬 순서는 생일이 빠른 순으로 출력하고자 한다.
<실행결과>가 아래와 같을 때 다음 보기에서 제시하는 SQL문 중 조건에 부합하는 SQL문을 1개 고르시오.

<실행결과>

EMP_NO	EMP_NM	DEPT_NM	BIRTH_DE
E00003	박경민	Data플랫폼팀	19830414
E00001	이경오	Data시각화팀	19840718
E00005	최철순	Data분석팀	19860112

정답

④

※ 보기 4번 SQL문이 DEPT_CD별 MIN(BIRTH_DE) 값을 구해서 부서별 생일이 가장 빠른 ROW를 찾은 후에 BIRTH_DE 기준으로 정렬한 집합을 출력하고 있다.

② 결과 ROW는 동일하나 BIRTH_DE 순으로 순차 정렬 되지 않았다.

①

```
SELECT A.EMP_NO, A.EMP_NM
      , (SELECT L.DEPT_NM FROM TB_DEPT_23 L WHERE L.DEPT_CD = A.DEPT_CD) AS DEPT_NM
      , A.BIRTH_DE
FROM TB_EMP_23 A
WHERE (A.DEPT_CD, BIRTH_DE) IN
      (SELECT B.DEPT_CD, MAX(BIRTH_DE)
       FROM TB_EMP_23 B GROUP BY B.DEPT_CD
      )
ORDER BY A.BIRTH_DE;
```

②

```
SELECT A.EMP_NO, A.EMP_NM
      , (SELECT L.DEPT_NM FROM TB_DEPT_23 L WHERE L.DEPT_CD = A.DEPT_CD) AS DEPT_NM
      , A.BIRTH_DE
FROM TB_EMP_23 A
WHERE (A.DEPT_CD, BIRTH_DE) IN
      (SELECT B.DEPT_CD, MIN(BIRTH_DE)
       FROM TB_EMP_23 B GROUP BY B.DEPT_CD
      )
ORDER BY A.DEPT_CD, A.BIRTH_DE;
```

③

```
SELECT A.EMP_NO, A.EMP_NM
      , C.DEPT_NM AS DEPT_NM
      , A.BIRTH_DE
FROM TB_EMP_23 A
      , (
          SELECT B.DEPT_CD, MIN(BIRTH_DE) AS BIRTH_DE
          FROM TB_EMP_23 B
          GROUP BY B.DEPT_CD
        ) B
      , TB_DEPT_23 C
WHERE A.DEPT_CD = B.DEPT_CD(+)
AND A.BIRTH_DE = B.BIRTH_DE(+)
AND A.DEPT_CD = C.DEPT_CD(+)
ORDER BY A.BIRTH_DE;
```

④

```
SELECT A.EMP_NO, A.EMP_NM
      , (SELECT L.DEPT_NM FROM TB_DEPT_23 L WHERE L.DEPT_CD = A.DEPT_CD) AS DEPT_NM
      , A.BIRTH_DE
FROM TB_EMP_23 A
      , (
          SELECT B.DEPT_CD, MIN(BIRTH_DE) AS BIRTH_DE
          FROM TB_EMP_23 B
          GROUP BY B.DEPT_CD
        ) B
WHERE A.DEPT_CD = B.DEPT_CD
AND A.BIRTH_DE = B.BIRTH_DE
ORDER BY A.BIRTH_DE
;
```

문제24. [아래]의 설명은 반환되는 데이터 형태에 따른 서브 쿼리 분류 중 어떤 서브 쿼리에 대한 설명인지 기재 하시오.

- 1) 서브 쿼리의 실행 결과로 여러 칼럼을 반환한다. 메인 쿼리의 조건절에 여러 칼럼을 동시에 비교할 수 있다.
- 2) 서브 쿼리와 메인쿼리에서 비교하고자하는 칼럼 개수와 칼럼의 위치가 동일해야 한다.

정답 다중 칼럼 서브 쿼리(Multi Column 서브 쿼리)

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', '프로시저개발자', 40000000, 'D101');

INSERT INTO TB_EMP_25 VALUES ('100003', '김철수', '리눅스엔지니어', 40000000, 'D102');
INSERT INTO TB_EMP_25 VALUES ('100004', '박상진', '윈도우엔지니어', 35000000, 'D102');

INSERT INTO TB_EMP_25 VALUES ('100005', '박은정', 'R개발자' , 50000000, 'D103');
INSERT INTO TB_EMP_25 VALUES ('100006', '김다연', '파이썬개발자' , 45000000, 'D103');

INSERT INTO TB_EMP_25 VALUES ('100007', '박수진', '오라클튜너' , 65000000, 'D104');
INSERT INTO TB_EMP_25 VALUES ('100008', '김성수', '오픈소스튜너' , 60000000, 'D104');

INSERT INTO TB_EMP_25 VALUES ('100009', '추상미', '웹개발자' , 35000000, 'D105');
INSERT INTO TB_EMP_25 VALUES ('100010', '박나래', '자바개발자' , 30000000, 'D105');

COMMIT;
```

<SQL문>

```
SELECT 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 (B.DEPT_NM)
;
```

- ① 5
- ② 10
- ③ 50
- ④ 6

정답 ④

※ DEPT_NM 칼럼 만을 ROLLUP했기 때문에 DEPT_NM의 유일 값 5개, 전체 합계를 위한 행 1개 총 6건의 ROW가 출력된다.

SQLD 출제 예상문제 - 1회
과목 2. SQL 기본 및 활용
2-6. 26번~30번

문제26.

아래와 같이 TB_EMP_26, TB_DEPT_26 테이블을 생성하고 데이터를 입력하였다. 아래와 같이 <SQL1>, <SQL2>를 실행하고자 한다. 실행하는 2개의 SQL문의 결과 집합이 동일하다고 가정했을 때 ㉠, ㉡에 들어갈 내용으로 가장 적절한 것은?

```
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', '박찬진', '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;
```

<SQL1>

```
SELECT A.JOB_NM
      , B.DEPT_NM
      , AVG(A.CUR_SAL) AS CUR_SAL
FROM TB_EMP_26 A
      , TB_DEPT_26 B
WHERE A.DEPT_CD = B.DEPT_CD
㉠
ORDER BY A.JOB_NM, B.DEPT_NM, CUR_SAL;
```

<SQL2>

```
SELECT A.JOB_NM
      , B.DEPT_NM
      , AVG(A.CUR_SAL) AS CUR_SAL
FROM TB_EMP_26 A
      , TB_DEPT_26 B
WHERE A.DEPT_CD = B.DEPT_CD
㉡
ORDER BY A.JOB_NM, B.DEPT_NM, CUR_SAL;
```

- ① ㉠ : GROUP BY ROLLUP(A.JOB_NM, B.DEPT_NM) ㉡ : GROUP BY GROUPING SETS(A.JOB_NM, B.DEPT_NM, ())
- ② ㉠ : GROUP BY ROLLUP(A.JOB_NM, B.DEPT_NM) ㉡ : GROUP BY GROUPING SETS((A.JOB_NM, B.DEPT_NM), A.JOB_NM, ())
- ③ ㉠ : GROUP BY ROLLUP(B.DEPT_NM, A.JOB_NM) ㉡ : GROUP BY GROUPING SETS((A.JOB_NM, B.DEPT_NM), A.JOB_NM, ())
- ④ ㉠ : GROUP BY ROLLUP(B.DEPT_NM, A.JOB_NM) ㉡ : GROUP BY GROUPING SETS(B.DEPT_NM, A.JOB_NM, ())

정답 ②

※ 보기 2번이 두개의 SQL문의 결과 집합이 같은 SQL문이다.
※ 만약 ③번 혹은 ④번이 정답이 되려면 ㉡이 GROUPING SETS((B.DEPT_NM, A.JOB_NM), B.DEPT_NM, ()) 바뀌어야 한다.

문제27. [아래]는 그룹 함수인 CUBE에 대한 설명이다. 아래 설명에서 ㉠에 들어갈 숫자를 기재 하시오.

- 1) ROLLUP에서는 단지 가능한 SUBTOTAL만을 생성하였지만, CUBE는 결합 가능한 모든 값에 대하여 다차원 집계를 생성한다.
- 2) CUBE는 표시된 인수들에 대한 계층별 집계를 구할 수 있으며, 이때 표시된 인수들간에는 계층 구조인 ROLLUP과는 달리 평등한 관계이므로 인수의 순서가 바뀌는 경우 행간의 정렬 순서는 바뀔 수 있어도 데이터 결과집합은 동일하다.
- 3) CUBE는 GROUPING COLUMNS이 가질 수 있는 모든 경우의 수에 대하여 SUBTOTAL을 생성하므로 GROUPING COLUMNS의 수가 N이라고 가정하면 ㉠의 N승 LEVEL의 SUBTOTAL을 생성하게 된다.

정답 ㉠ : 2

※ 만약 GROUPING COLUMNS의 수가 2개 이면 2의 2승 = 4 LEVEL의 SUBTOTAL을 생성하게 된다.

문제28.

[아래]와 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 SUM(DENSE_RANK_CUR_SAL) SUM_DENSE_RANK_CUR_SAL
FROM
(
  SELECT DISTINCT DENSE_RANK() OVER (ORDER BY CUR_SAL) DENSE_RANK_CUR_SAL
  FROM TB_EMP_28
)
;
```

- ① 37
- ② 28
- ③ 40
- ④ SQL문법 오류발생

정답 ②

- ※ 인라인 뷰 내부의 SQL문의 결과는 1, 2, 3, 4, 5, 6, 7이 나온다.
- ※ DENSE_RANK는 중복되는 순위는 하나의 순위로 매겨주게 된다.
- ※ 중복되는 건이 연봉이 3500만 2건, 4000만 2건, 4500만 2건 총 6건이므로 그중 3건이 중복으로 제외된다.
- ※ 즉 10명중 7등까지만 등수가 매겨진다. DISTINCT로 중복된 등수를 제거 했으므로 1,2,3,4,5,6,7이 나오게 되고 그 수를 모두 합하면 28이 정답이다.

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

<SQL문>

```
SELECT EMP_NO, EMP_NM, COUNT(DEPT_CD) OVER(PARTITION BY SEX_CD) AS CNT
FROM TB_EMP_29;
```

①	②	③	④
EMP_NO EMP_NM SEX_CD CNT	EMP_NO EMP_NM SEX_CD CNT	EMP_NO EMP_NM SEX_CD CNT	EMP_NO EMP_NM SEX_CD CNT
E00001 이경오 1 3	E00001 이경오 1 3	E00001 이경오 1 3	E00001 이경오 1 3
E00002 이수지 2 0	E00002 이수지 2 4	E00002 이수지 2 4	E00002 이수지 2 3
E00003 박경민 1 3	E00003 박경민 1 3	E00003 박경민 1 3	E00003 박경민 1 3
E00004 최주연 2 0	E00004 최주연 2 4	E00004 최주연 2 4	E00004 최주연 2 3
E00005 최철순 1 3	E00005 최철순 1 3	E00005 최철순 1 3	E00005 최철순 1 3
E00006 이지연 2 0	E00006 이지연 2 4	E00006 이지연 2 4	E00006 이지연 2 3
E00007 차은영 2 0	E00007 차은영 2 0	E00007 차은영 2 4	E00007 차은영 2 3

정답 ④

- ※ 직원 중 성별코드(SEX_CD)가 '2'(여성)인 직원은 모두 4명이다.
- ※ 하지만 그중 '차은영' 사원의 DEPT_CD가 NULL이므로 성별이 여성인 직원의 DEPT_CD의 개수는 4개가 아니라 3개가 된다.
- ※ 그래서 CNT값은 전부 3이 출력된다.

문제30.

아래와 같이 TB_EMP_30 테이블을 생성하고 데이터를 입력하였다. 아래와 같은 <결과집합>을 도출하는 SQL문을 작성하고자 한다. <SQL문>의 ㉠에 들어갈 알맞은 분석 함수 및 분석 함수내 인자를 작성하시오. (작성형식 : 분석함수명(인자값))

```
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 A.EMP_NO
      , A.EMP_NM
      , ㉠ OVER (ORDER BY CUR_SAL DESC) AS CUR_SAL_SE
      , CUR_SAL
FROM TB_EMP_30 A
```

<결과집합>

EMP_NO	EMP_NM	CUR_SAL_SE	CUR_SAL
100001	이경오	1	80000000
100002	이동민	1	60000000
100003	김철수	2	40000000
100004	박상진	2	20000000

정답 NTILE(2)

※ NTILE함수를 이용해 전체 건수를 인자 값으로 N등분한 결과를 구할 수 있다.

SQLD 출제 예상문제 - 1회
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

CONN SYSTEM/1234 --h
DROP USER DCL; --i
```

- ① 신규로 생성한 DCL계정에서 DBA권한을 주려고 했으므로 ㉔단계에서 스크립트가 실패한다.
- ② ㉔단계에서 CREATE SESSION 권한이 없으므로 DCL 사용자로 접속에 실패한다.
- ③ ①단계에서 DCL 계정으로 테이블 및 데이터를 가지고 있으므로 계정 제거에 실패한다.
- ④ ①단계에서 DCL 계정을 제거하고 DCL 계정이 소유한 테이블 또한 제거한다.

정답 ③

※ DROP USER 시 CASCADE옵션을 주지 않으면 'ORA-01922: 'DCL'(을)를 삭제하려면 CASCADE를 지정하여야 합니다' 에러와 함께 사용자 제거에 실패한다.

문제32.

아래 <SQL문>은 'DCL' 유저가 가지고 있는 모든 오브젝트 및 유저를 제거하는 <SQL문>이다. 이 <SQL문>의 ㉠에 들어
갈 알맞은 키워드를 기재 하시오.

<SQL문>

--SYSTEM계정으로 접속
DROP USER DCL ㉠;

정답

CASCADE

※ CASCADE를 사용하게 되면 사용자 이름과 관련된 모든 데이터베이스 스키마가 데이터 사전 으로부터 삭제되며 모든 스키마 객체들 또한 물리적으로 삭제 된다.

문제33.

오라클 PL/SQL의 블록 구조는 DECLARE, BEGIN, EXECTION, END로 나누어져 있다. 다음 중 PL/SQL의 블록 구조에 대한 설명은 가장 부적절한 것은 무엇인가?

- ① DECLARE는 선언부로서 사용할 변수나 인수에 대한 정의 및 데이터 타입을 선언한다. 아무런 변수도 사용하지 않을 경우에도 반드시 선언해야 한다.
- ② BEGIN은 실행부로서 개발자가 처리하고 하는 SQL문과 필요한 로직이 정의되는 실행부이다. 반드시 선언해야 하는 필수항목이다.
- ③ EXCEPTION은 BEGIN~END에서 실행되는 SQL문에 발생한 에러를 처리하는 에러처리부이다. 반드시 선언해야 한다.
- ④ END는 실행부의 종료를 명시해주는 기능을 하며 반드시 선언해야 하다.

정답 ③

※ EXECEPTION은 선택항목이다. 생략이 가능하다. 반드시 선언해야 하는 것은 아니다.

문제34.

다음 아래와 같이 TB_EMP_34 테이블을 생성 후 데이터를 입력하였다. 그후 FN_EMP_CNT_BY_EMP_NM_34 라는 사용자 정의 함수를 생성하였다. 아래의 <SQL문>을 수행하면 나오게 되는 결과값을 기재 하시오. (힌트: 결과는 숫자이다.)

```
DROP TABLE TB_EMP_34;

CREATE TABLE TB_EMP_34
(
  EMP_NO CHAR(6)
, EMP_NM VARCHAR2(50)
, DEPT_CD CHAR(4)
, CONSTRAINT TB_EMP_34_PK PRIMARY KEY(EMP_NO)
)
;

INSERT INTO TB_EMP_34 VALUES ('100001', '이경오', 'D101' );
INSERT INTO TB_EMP_34 VALUES ('100002', '이수진', 'D101' );
INSERT INTO TB_EMP_34 VALUES ('100003', '권수철', 'D102' );
INSERT INTO TB_EMP_34 VALUES ('100004', '이지은', 'D102' );
INSERT INTO TB_EMP_34 VALUES ('100005', '정수라', 'D103' );
INSERT INTO TB_EMP_34 VALUES ('100006', '김연정', NULL );

COMMIT;

DROP FUNCTION FN_EMP_CNT_BY_EMP_NM_34;
```

```
CREATE OR REPLACE FUNCTION FN_EMP_CNT_BY_EMP_NM_34(IN_EMP_NM IN VARCHAR)
RETURN NUMBER IS V_EMP_CNT NUMBER;

BEGIN
  SELECT COUNT(*) CNT
    INTO V_EMP_CNT
    FROM TB_EMP_34
   WHERE EMP_NM LIKE IN_EMP_NM || '%';
;

RETURN V_EMP_CNT;
END;
/
;
```

<SQL문>

```
SELECT
  SUM(FN_EMP_CNT_BY_EMP_NM_34(SUBSTR(A.EMP_NM, 1, 1))
    - FN_EMP_CNT_BY_EMP_NM_34(A.EMP_NM)
  ) AS CNT_SUM
  FROM TB_EMP_34 A
 ORDER BY A.EMP_NO
;
```

정답 6

- ※ '이'씨 성을 가진 사람은 3명이다. '권'씨 '정'씨, '김'씨는 각각 1명씩이다.
- ※ '이'씨 성을 가진 사람은 3 - 1(자기자신) = 2가 출력되고
- ※ 그 외 성씨를 가진 사람은 1 - 1(자기자신) = 0이 출력된다.
- ※ '이'씨 성을 가진 사람은 3명이므로 2+2+2는 6이다.

문제35. 다음 중 옵티마이저에 대한 설명 중 가장 부적절한 것은 무엇인가?

- ① 사용자가 질의한 SQL문에 대해 최적의 실행 방법을 결정하는 역할을 수행한다.
- ② 다양한 실행 방법들 중에서 최적의 실행방법을 결정하는 역할을 한다.
- ③ 관계형 데이터베이스는 옵티마이저가 결정한 실행방법대로 실행 엔진이 데이터를 처리하여 결과 데이터를 사용자에게 전달한다.
- ④ 옵티마이저 내부적으로 실제 SQL문을 처리한 후 실행계획을 수립하고 SQL을 실행하여 사용자에게 결과를 전달한다.

정답 ④

※ 옵티마이저는 실제로 SQL문을 처리해보지 않은 상태에서 실행계획을 결정해야 한다.

SQLD 출제 예상문제 - 1회
과목 2. SQL 기본 및 활용
2-8. 36번~40번

문제36. 다음 중 실행계획을 구성하는 요소에 해당하지 않는 것은?

- ① 조인 순서
- ② 조인 기법
- ③ 액세스 기법
- ④ 수행 속도

정답 ④

※ 실행계획을 구성하는 요소에는 조인 순서, 조인 기법, 액세스 기법, 최적화 정보, 연산 등이 있다.

문제37. 다음 중 B-TREE 인덱스의 구조에서 구성요소로 가장 부적절한 것은 무엇인가?

- ① Root Block
- ② Branch Block
- ③ Root Level
- ④ Leaf Block

정답 ③

※ B-TREE 인덱스는 Root Block, Brach Block, Leaf Block으로 이루어져 있다.

문제38. 다음 중 옵티마이저가 풀 테이블 스캔 방식을 선택하는 상황 중 가장 부적절한 것은?

- ① SQL문의 WHERE절에 조건이 존재하지 않는 경우
- ② SQL문의 WHERE절에 조건 중 사용 가능한 인덱스가 존재하지 않는 경우
- ③ 옵티마이저의 판단 결과 테이블 풀 스캔이 비용상 유리하다고 판단 되는 경우
- ④ 적절한 인덱스가 있지만 테이블의 데이터가 대용량인 경우

정답 ④

※ 인덱스 스캔을 할지 테이블 풀 스캔을 할지는 옵티마이저가 판단한다.

문제39. 아래는 NL조인의 연산 과정을 설명하고 있다. 다음 중 NL 조인의 연산 순서로 가장 적절한 것은 무엇인가?

- (1) INNER 집합을 액세스 하고 매칭되는 ROW를 결과 집합에 포함
- (2) OUTER 집합의 조인 키를 가지고 INNER 집합에 조인 키가 존재하는지 찾으러 감
- (3) OUTER 집합에서 조건을 만족하는 첫 번째 ROW를 찾음
- (4) INNER 집합에서 OUTER 집합의 조인 칼럼의 값이 존재하는지 확인
- (5) OUTER 집합의 모든 ROW가 작업을 다 마칠 때 까지 해당 과정 반복

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

정답 ②

➤ NL 조인 연산 순서

- 첫번째 : OUTER 집합에서 조건을 만족하는 첫 번째 ROW를 찾음
- 두번째 : OUTER 집합의 조인 키를 가지고 INNER 집합에 조인 키가 존재하는지 찾으러 감
- 세번째 : INNER 집합에서 OUTER 집합의 조인 칼럼의 값이 존재하는지 확인
- 네번째 : INNER 집합을 액세스 하고 매칭되는 ROW를 결과 집합에 포함
- 다섯번째 : OUTER 집합의 모든 ROW가 작업을 다 마칠 때 까지 해당 과정 반복

문제40. 다음 중 조인 기법에 대한 설명으로 가장 부적절한 것은?

- ① NL조인은 결과 행의 수가 적은 집합을 조인 순서상 선행 집합으로 선택하는 것이 전체 일의 양을 줄인다.
- ② HASH조인은 결과 행의 수가 적은 집합을 선행 집합으로 사용하는 것이 성능 상 유리하다.
- ③ HASH조인에서 선행 집합을 BUILD INPUT이라고 하며 후행 집합을 PROBE INPUT이라고 한다.
- ④ NL조인은 선행 집합을 스캔 시 반드시 인덱스 스캔을 할 필요는 없지만 후행 집합을 스캔 시에는 반드시 인덱스가 필요하다.

정답 ④

※ 후행 집합을 스캔 시 효율적인 인덱스가 있어야 성능상 훨씬 유리하다고 할 수 있다. 하지만 반드시 필요한 것은 아니다.