# SQLD 핵심 포인트 30선 — 문제 · 정답 · 해설집

www.sqld.kr · 제1과목 데이터 모델링의 이해 (01~08번) / 제2과목 SQL 기본 및 활용 (09~30번)

## 제1과목: 데이터 모델링의 이해 (01 ~ 08번)

### 문항 01. [데이터모델의이해] 3단계 스키마와 데이터 독립성

**Q.** 쇼핑몰 DB에서 조회 성능을 높이기 위해 ORDERS(주문) 테이블에 주문일자 인덱스를 신설하고 월별 범위 파티셔닝을 적용하였다. 이때 기존 주문내역 조회 화면(View)과 응용프로그램의 SQL은 전혀 수정하지 않고 그대로 사용할 수 있었다. 이와 관련된 데이터 독립성과 변경된 스키마의 연결로 가장 올바른 것은?

① 논리적 데이터 독립성 — 개념 스키마 변경

② 논리적 데이터 독립성 — 내부 스키마 변경

③ 물리적 데이터 독립성 — 개념 스키마 변경

④ 물리적 데이터 독립성 — 내부 스키마 변경

**정답** ④

**해설** 인덱스(Index) 신설이나 파티셔닝, 저장구조 변경은 물리적 저장 방식인 '내부 스키마'를 변경한 것입니다. 내부 스키마가 변경되어도 개념 스키마와 외부 스키마(응용프로그램/View)가 영향을 받지 않는 특성은 '물리적 데이터 독립성'이므로 정답은 4번입니다. (1번은 테이블 분리/컬럼 추가 시 해당)

### 문항 02. [엔터티] 엔터티(Entity)의 성립 조건과 발생 시점별 분류

**Q.** 다음 중 데이터 모델링에서 엔터티(Entity)의 성립 조건과 발생 시점에 따른 분류 설명으로 옳지 않은 것은?

① 엔터티는 업무에서 관리해야 하는 집합이므로 반드시 2개 이상의 인스턴스와 2개 이상의 속성을 가져야 한다 (단, 연관 엔터티 예외).

② 사원, 부서, 고객, 상품처럼 원래 업무에 독립적으로 존재하는 엔터티를 기본 엔터티(Key Entity)라 한다.

③ 주문내역, 결제이력, 접속로그처럼 2개 이상의 엔터티로부터 행위를 통해 자주 발생하는 엔터티를 중심 엔터티(Main Entity)라 한다.

④ 기본 엔터티로부터 발생하고 업무에서 중심적인 역할을 수행하는 주문, 계약, 청구 등은 중심 엔터티에 해당한다.

**정답** ③

**해설** 주문내역, 결제이력, 접속로그처럼 업무 행위 결과로 데이터가 지속적으로 누적·발생하는 엔터티는 '행위 엔터티(Active Entity)'입니다. 중심 엔터티는 주문, 계약, 청구처럼 기본 엔터티와 행위 엔터티의 중간에서 업무 중심 역할을 하는 엔터티이므로 3번이 틀린 설명입니다.

### 문항 03. [속성] 속성(Attribute)의 특성에 따른 분류

**Q.** 온라인 서점의 [도서주문] 엔터티를 설계하면서 ① 도서원가, ② 관리를 위해 부여한 도서분류코드, ③ 실시간 주문수량과 단가를 곱해 화면에 보여주는 예상결제총액 속성을 도출하였다. ①, ②, ③ 속성의 특성에 따른 분류가 순서대로 바르게 짝지어진 것은?

① 도서원가, ② 관리를 위해 부여한 도서분류코드, ③ 실시간 주문수량과 단가를 곱해 화면에 보여주는 예상결제총액 속성을 도출하였다. ①, ②, ③ 속성의 특성에 따른 분류가 순서대로 바르게 짝지어진 것은?

② 관리를 위해 부여한 도서분류코드, ③ 실시간 주문수량과 단가를 곱해 화면에 보여주는 예상결제총액 속성을 도출하였다. ①, ②, ③ 속성의 특성에 따른 분류가 순서대로 바르게 짝지어진 것은?

③ 실시간 주문수량과 단가를 곱해 화면에 보여주는 예상결제총액 속성을 도출하였다. ①, ②, ③ 속성의 특성에 따른 분류가 순서대로 바르게 짝지어진 것은?

④ ① 설계 속성 - ② 파생 속성 - ③ 기본 속성

**정답** ①

**해설** ① 업무로부터 직접 추출된 본래의 속성(도서원가)은 '기본 속성', ② 데이터 모델링과 코드 관리를 위해 새로 만든 속성(도서분류코드)은 '설계 속성', ③ 다른 속성들을 연산하여 얻어지는 속성(예상결제총액)은 '파생 속성'입니다. 파생 속성은 정합성 유지를 위해 가급적 적게 정의해야 합니다.

### 문항 04. [관계] ERD 표기법 비교 (IE vs Barker)

**Q.** '한 명의 고객은 주문을 한 번도 하지 않거나 여러 번 주문할 수 있고, 하나의 주문은 반드시 단 한 명의 고객에 의해서만 발생한다'는 업무 규칙을 ERD로 표현할 때, IE 표기법과 Barker 표기법의 설명으로 옳은 것은?

① IE 표기법에서는 선택 참여(0)를 표현하기 위해 고객과 주문 사이의 관계선을 점선으로 그린다.

② IE 표기법에서는 주문 엔터티 쪽 끝에는 'O'와 까마귀발(<)을, 고객 엔터티 쪽 끝에는 '|' 기호를 표기한다.

③ Barker 표기법에서는 필수/선택 참여 여부를 선 끝의 'O' 기호 유무로 구분한다.

④ Barker 표기법에서 선택 속성은 속성명 앞에 '*' 기호를 붙여 표기한다.

**정답** ②

**해설** 상대방 엔터티 앞에 그려진 기호가 상대방의 참여도를 나타냅니다. 고객 입장에서 주문은 0개 이상(선택·다수)이므로 주문 쪽 끝에 'O<'를 표기하고, 주문 입장에서 고객은 반드시 1명(필수·단수)이므로 고객 쪽 끝에 '||'를 표기합니다(2번 정답). 1번 IE에서 점선은 '비식별자 관계'를 뜻하며, 3번 Barker는 실선/점선으로 필수/선택을 구분하고, 4번 Barker에서 '*'는 필수 속성, 'o'가 선택 속성입니다.

### 문항 05. [식별자] 주식별자의 4대 특성과 후보키 vs 슈퍼키

**Q.** 릴레이션 R(학번, 주민번호, 이름, 학과)에서 '학번'과 '주민번호'는 각각 모든 학생 인스턴스를 유일하게 식별할 수 있으며 NULL 값을 허용하지 않는다. 다음 중 후보키(Candidate Key)와 슈퍼키(Super Key)에 대한 설명으로 옳은 것은?

① (학번, 이름) 조합은 모든 행을 유일하게 식별하므로 후보키(Candidate Key)이다.

② (학번, 주민번호) 조합은 유일성과 최소성을 모두 만족하므로 후보키이다.

③ 후보키는 '학번'과 '주민번호' 각각 2개뿐이며, (학번, 이름)은 최소성을 만족하지 못하는 슈퍼키이다.

④ 학번을 기본키(PK)로 선정하면 주민번호는 슈퍼키에서 제외된다.

**정답** ③

**해설** 후보키(Candidate Key)는 유일성과 최소성을 동시에 만족해야 합니다. '학번' 단독으로 이미 유일성이 충족되므로, 거기에 '이름'이나 '주민번호'를 추가한 (학번, 이름), (학번, 주민번호)는 모두 최소성을 위반하여 후보키가 아닌 '슈퍼키(Super Key)'가 됩니다. 따라서 정답은 3번입니다.

### 문항 06. [식별자] 식별자 관계(실선) vs 비식별자 관계(점선)

**Q.** 부모 엔터티와 자식 엔터티 간의 식별자 관계(Identifying)와 비식별자 관계(Non-Identifying)에 대한 설명으로 가장 부적절한 것은?

① 식별자 관계는 부모의 주식별자(PK)가 자식의 주식별자(PK)의 일부로 상속되는 관계로, IE 표기법에서 실선으로 표시한다.

② 비식별자 관계는 부모의 주식별자가 자식의 일반 외래키(FK) 속성으로만 상속되는 관계로, 부모 없이도 자식 인스턴스가 생성될 수 있는 경우에 적합하다.

③ 모든 관계를 식별자 관계(실선)로만 연결하면 하위 엔터티로 내려갈수록 주식별자 속성 개수가 줄어들어 SQL 조인 문장이 단순해진다.

④ 자식 엔터티가 독자적인 생명주기를 가지거나 PK 속성 증가로 인한 SQL 복잡도를 방지하기 위해 인조식별자를 두고 비식별자 관계로 전환할 수 있다.

**정답** ③

**해설** 식별자 관계(실선)를 1대 → 2대 → 3대 하위 엔터티로 계속 이어가면, 상위 엔터티들의 PK가 자식 PK에 계속 누적되어 하위 엔터티의 PK 컬럼 수가 눈덩이처럼 늘어납니다. 이로 인해 SQL 조인 시 ON 절에 나열해야 할 조건이 많아져 개발 복잡도와 오류 가능성이 커지므로 3번이 틀린 설명입니다.

### 문항 07. [정규화] 정규화 단계 판별 (1NF · 2NF · 3NF · BCNF)

**Q.** 복합 주식별자 (주문번호, 상품코드)를 가진 [주문상세] 엔터티에 '상품명', '상품단가', '주문수량' 속성이 포함되어 있다. 이때 '상품명'과 '상품단가'가 주식별자 전체가 아닌 '상품코드'에만 종속되어 데이터 중복과 갱신 이상이 발생하고 있다면, 이를 해결하기 위해 수행해야 할 정규화 단계는?

```
주문상세 ( #주문번호, #상품코드, 상품명, 상품단가, 주문수량 )
함수 종속성: (주문번호, 상품코드) -> 주문수량  /  상품코드 -> 상품명, 상품단가
```

① 제1정규화 (1NF) — 원자값 확보

② 제2정규화 (2NF) — 부분 함수 종속 제거

③ 제3정규화 (3NF) — 이행적 함수 종속 제거

④ 보이스-코드 정규화 (BCNF) — 결정자 후보키 확보

**정답** ②

**해설** 주식별자가 (주문번호, 상품코드) 복합키일 때, 일반 속성인 상품명·상품단가가 복합키의 일부인 '상품코드'에만 종속되는 현상을 '부분 함수 종속'이라 합니다. 부분 함수 종속을 제거하여 [상품] 엔터티로 분리하는 단계는 '제2정규화(2NF)'입니다. (일반 속성끼리의 종속은 제3정규화)

### 문항 08. [null] NULL의 정의와 3값 논리 연산 (TRUE / FALSE / UNKNOWN)

**Q.** 아래 EMP 테이블의 한 행에서 SAL = 2000, COMM = NULL 일 때, 다음 세 개의 논리 조건식 (가), (나), (다)의 평가 결과(TRUE / FALSE / UNKNOWN)가 순서대로 바르게 짝지어진 것은?

```
(가) COMM = NULL OR SAL = 2000
(나) COMM > 0 AND SAL = 1000
(다) NOT (COMM <> 500)
```

① (가) TRUE / (나) FALSE / (다) UNKNOWN

② (가) UNKNOWN / (나) UNKNOWN / (다) TRUE

③ (가) TRUE / (나) UNKNOWN / (다) FALSE

④ (가) FALSE / (나) FALSE / (다) UNKNOWN

**정답** ①

**해설** - (가) `COMM = NULL`은 UNKNOWN이지만 뒤의 `SAL = 2000`이 TRUE이므로, `UNKNOWN OR TRUE` = TRUE입니다.
- (나) `COMM > 0`은 UNKNOWN이지만 뒤의 `SAL = 1000`이 FALSE이므로, `UNKNOWN AND FALSE` = FALSE입니다.
- (다) `COMM <> 500`은 UNKNOWN이며, `NOT(UNKNOWN)`은 여전히 UNKNOWN입니다. 따라서 정답은 1번입니다.

## 제2과목: SQL 기본 및 활용 (09 ~ 30번)

### 문항 09. [select문] SELECT 문장의 논리적 실행 순서와 별칭(Alias)

**Q.** 다음 SQL 문장 중 문법적 오류(Error) 없이 정상적으로 실행되는 것은?

① SELECT DEPTNO AS 부서번호, SUM(SAL) AS 급여합 FROM EMP WHERE 급여합 >= 5000 GROUP BY DEPTNO;

② SELECT DEPTNO AS 부서번호, AVG(SAL) AS 평균급여 FROM EMP GROUP BY 부서번호 HAVING AVG(SAL) >= 2000;

③ SELECT DEPTNO AS 부서번호, SUM(SAL) AS 급여합 FROM EMP GROUP BY DEPTNO HAVING 급여합 >= 5000;

④ SELECT DEPTNO AS 부서번호, AVG(SAL) AS 평균급여 FROM EMP WHERE SAL >= 1000 GROUP BY DEPTNO HAVING SUM(SAL) >= 3000 ORDER BY 평균급여 DESC;

**정답** ④

**해설** SQL은 `FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY` 순서로 실행됩니다. 따라서 SELECT 절에서 정의한 별칭(부서번호, 급여합, 평균급여)은 WHERE 절(1번), GROUP BY 절(2번), HAVING 절(3번)에서는 아직 생성되기 전이므로 사용할 수 없어 에러가 발생합니다. 오직 ORDER BY 절에서만 SELECT 별칭을 사용할 수 있으므로 4번만 정상 실행됩니다.

### 문항 10. [where절] 연산자 우선순위 (NOT > AND > OR)

**Q.** 다음과 같은 4건의 데이터가 저장된 EMP 테이블에서 아래 SQL을 실행했을 때 출력되는 결과(CNT)는?

```
[EMP 데이터]
1) SMITH (DEPTNO: 10, SAL: 1000)   2) ALLEN (DEPTNO: 20, SAL: 1500)
3) WARD  (DEPTNO: 20, SAL: 3500)   4) KING  (DEPTNO: 30, SAL: 5000)

[실행 SQL]
SELECT COUNT(*) AS CNT FROM EMP
WHERE DEPTNO = 10 OR DEPTNO = 20 AND SAL >= 3000;
```

① 1

② 2

③ 3

④ 4

**정답** ②

**해설** 논리 연산자 우선순위는 `NOT > AND > OR` 이므로, 괄호가 없어도 AND가 먼저 묶여 `WHERE DEPTNO = 10 OR (DEPTNO = 20 AND SAL >= 3000)`으로 실행됩니다. 따라서 `DEPTNO = 10`인 SMITH(급여 무관 통과!) 1명과, `DEPTNO = 20`이면서 `SAL >= 3000`인 WARD 1명이 선택되어 총 2건이 출력됩니다.

### 문항 11. [함수] Oracle vs SQL Server 빈 문자열('') 처리 차이

**Q.** Oracle DBMS 환경에서 아래 SQL 문장들을 순서대로 실행했을 때, 마지막 SELECT 문에서 조회되는 행의 개수는?

```
CREATE TABLE T_STR ( ID NUMBER, VAL VARCHAR2(10) );
INSERT INTO T_STR VALUES (1, '');
INSERT INTO T_STR VALUES (2, NULL);
INSERT INTO T_STR VALUES (3, 'SQL');

SELECT COUNT(*) FROM T_STR WHERE VAL = '';
```

① 0

② 1

③ 2

④ 3

**정답** ①

**해설** Oracle에서는 빈 문자열(`''`)을 입력하면 즉시 `NULL`로 변환되어 저장됩니다. 또한 `WHERE VAL = ''` 조건 역시 `WHERE VAL = NULL` 비교가 되어 UNKNOWN을 반환하므로 단 한 건도 통과하지 못합니다(NULL을 찾으려면 `WHERE VAL IS NULL`을 써야 함). 따라서 결과는 0건(1번)입니다. (만약 SQL Server였다면 1건 조회됨)

### 문항 12. [함수] NULL 관련 함수 (NVL, NVL2, NULLIF, COALESCE)

**Q.** 테이블 T1의 한 행에 `COL1 = 50, COL2 = 50, COL3 = NULL, COL4 = 100` 이 저장되어 있을 때, 아래 SQL의 실행 결과는?

```
SELECT NVL2(NULLIF(COL1, COL2), 10, COALESCE(COL3, COL4, 0)) AS RESULT
FROM T1;
```

① 0

② 10

③ 50

④ 100

**정답** ④

**해설** 안쪽 함수부터 차례로 계산합니다.
1. `NULLIF(COL1, COL2)` = `NULLIF(50, 50)` → 두 값이 같으므로 `NULL`을 반환합니다.
2. `COALESCE(COL3, COL4, 0)` = `COALESCE(NULL, 100, 0)` → 첫 번째 NOT NULL 값인 `100`을 반환합니다.
3. `NVL2(첫번째인자, 두번째, 세번째)` = `NVL2(NULL, 10, 100)` → 첫 번째 인자가 NULL이므로 세 번째 인자인 100(4번)을 반환합니다!

### 문항 13. [group by & having] 집계 함수의 NULL 무시 규칙과 가로 연산 vs 세로 연산

**Q.** 아래 4개 행이 저장된 EXAM 테이블에서 다음 SQL을 실행했을 때 (가), (나), (다)의 출력 결과로 올바른 것은?

```
[EXAM 테이블 데이터 (4행)]
1행: A=10, B=10  |  2행: A=20, B=NULL  |  3행: A=30, B=10  |  4행: A=NULL, B=NULL

SELECT AVG(A) AS (가), AVG(NVL(A, 0)) AS (나), SUM(A + B) AS (다) FROM EXAM;
```

① (가) 15 / (나) 15 / (다) 80

② (가) 20 / (나) 15 / (다) 60

③ (가) 20 / (나) 15 / (다) 80

④ (가) 20 / (나) 20 / (다) 60

**정답** ②

**해설** - (가) `AVG(A)`: NULL인 4행을 제외하고 3행만으로 나누므로 `(10+20+30)/3 = 20`입니다.
- (나) `AVG(NVL(A,0))`: 4행의 NULL이 0으로 치환되어 분모가 4가 되므로 `(10+20+30+0)/4 = 15`입니다.
- (다) `SUM(A + B)`: 가로 사칙연산(`A+B`)을 먼저 수행하면 1행(`10+10=20`), 2행(`20+NULL = NULL!`), 3행(`30+10=40`), 4행(`NULL`)이 되어 2행의 20이 통째로 소멸합니다! 따라서 `20 + 40 = 60`이 됩니다 (만약 `SUM(A)+SUM(B)`였다면 `60+20=80`). 정답은 2번입니다.

### 문항 14. [group by & having] GROUP BY와 HAVING 절의 문법 제약

**Q.** 다음 중 GROUP BY 절과 HAVING 절의 사용에 대한 설명으로 옳은 것은?

① WHERE 절과 HAVING 절 모두 집계 함수(SUM, AVG 등)를 사용하여 조건을 지정할 수 있다.

② SELECT DEPTNO FROM EMP GROUP BY DEPTNO;` 처럼 SELECT 절에 집계 함수가 없으면 문법 에러가 발생한다.

③ GROUP BY DEPTNO`를 사용한 쿼리의 SELECT 절에는 DEPTNO와 집계 함수 외에 개별 사원명(ENAME)을 함께 기술할 수 없다.

④ HAVING 절은 FROM 절 직후, GROUP BY 절 이전에 실행되어 원본 행을 먼저 제거한다.

**정답** ③

**해설** GROUP BY 절을 사용할 때 SELECT 절과 ORDER BY 절에는 GROUP BY 기준 컬럼(DEPTNO)이나 집계 함수만 올 수 있으며, 그룹화되지 않은 일반 컬럼(ENAME)을 적으면 `ORA-00979` 에러가 발생합니다(3번 정답). 1번 WHERE 절에는 집계 함수를 쓸 수 없으며, 2번 집계 함수 없이도 GROUP BY 단독 사용이 가능(DISTINCT 효과)하고, 4번 HAVING은 GROUP BY 이후에 실행됩니다.

### 문항 15. [order by] ORDER BY 절의 NULL 정렬 위치 (Oracle vs SQL Server)

**Q.** BONUS 컬럼에 `[100, NULL, 300, 200]` 4건의 데이터가 들어있는 테이블에서 `SELECT BONUS FROM EMP ORDER BY BONUS DESC;` 쿼리를 Oracle과 SQL Server(MSSQL)에서 각각 실행했을 때 첫 번째 행에 출력되는 값은?

① Oracle: 300 / SQL Server: 300
② Oracle: NULL / SQL Server: 300
③ Oracle: 300 / SQL Server: NULL
④ Oracle: NULL / SQL Server: NULL

**정답** ②

**해설** Oracle은 정렬 시 NULL을 가장 큰 값(무한대 $\infty$)으로 취급하므로 내림차순(DESC) 정렬 시 `NULL`이 맨 첫 번째 줄에 출력됩니다. 반면 SQL Server(MSSQL)는 NULL을 가장 작은 값($-\infty$)으로 취급하므로 내림차순(DESC) 정렬 시 가장 큰 숫자인 `300`이 맨 첫 번째 줄에 출력되고 NULL은 맨 마지막에 출력됩니다.

### 문항 16. [표준 조인] NATURAL JOIN vs USING 절의 별칭(Alias) 금지 규칙

**Q.** EMP(EMPNO, ENAME, DEPTNO) 테이블과 DEPT(DEPTNO, DNAME) 테이블을 조인하는 다음 SQL 중 문법 오류가 발생하는 것은?

① SELECT E.DEPTNO, E.ENAME, D.DNAME FROM EMP E JOIN DEPT D ON E.DEPTNO = D.DEPTNO;
② SELECT DEPTNO, E.ENAME, D.DNAME FROM EMP E JOIN DEPT D USING (DEPTNO);
③ SELECT DEPTNO, E.ENAME, D.DNAME FROM EMP E NATURAL JOIN DEPT D;
④ SELECT E.DEPTNO, E.ENAME, D.DNAME FROM EMP E JOIN DEPT D USING (DEPTNO);

**정답** ④

**해설** `NATURAL JOIN`과 `USING` 절에서 조인 기준이 된 공통 컬럼(`DEPTNO`)은 두 테이블의 컬럼이 하나로 통합되므로, SELECT 절이나 WHERE 절, ORDER BY 절 어디에서도 테이블 별칭(`E.DEPTNO` 또는 `D.DEPTNO`)을 붙일 수 없습니다! 따라서 `E.DEPTNO`라고 별칭을 붙인 4번에서 문법 에러가 발생합니다.

### 문항 17. [조인] LEFT OUTER JOIN에서 ON 절 조건 vs WHERE 절 조건

**Q.** 좌측 테이블 A(ID: 1, 2, 3 총 3건)와 우측 테이블 B(ID: 1[LOC='서울'], ID: 2[LOC='부산'] 총 2건)에 대해 아래 (가), (나) 두 쿼리를 실행했을 때 조회되는 행의 개수로 올바른 것은?

```
(가) SELECT A.ID, B.LOC FROM A LEFT OUTER JOIN B ON A.ID = B.ID AND B.LOC = '서울';
(나) SELECT A.ID, B.LOC FROM A LEFT OUTER JOIN B ON A.ID = B.ID WHERE B.LOC = '서울';
```

① (가) 3건 / (나) 3건

② (가) 3건 / (나) 1건

③ (가) 1건 / (나) 3건

④ (가) 1건 / (나) 1건

**정답** ②

**해설** - (가) 우측 테이블 조건(`B.LOC = '서울'`)을 `ON` 절에 작성하면 조인 매칭 시에만 조건을 따지고, LEFT OUTER JOIN의 주인공인 좌측 테이블 A의 3건(1, 2, 3)은 모두 보존되어 3건이 출력됩니다(2번, 3번 행의 LOC는 NULL).
- (나) 반면 똑같은 조건을 `WHERE` 절에 작성하면, 조인이 끝난 뒤 NULL로 채워진 2번, 3번 행이 `WHERE B.LOC = '서울'` 조건에서 모두 탈락(INNER JOIN화)하여 ID=1인 1건만 출력됩니다!

### 문항 18. [조인] Oracle 전용 외부 조인 `( + )` 기호의 위치와 규칙

**Q.** 다음 ANSI 표준 외부 조인 SQL과 100% 동일한 결과를 반환하는 Oracle 전용 조인 구문은?

```
SELECT E.EMPNO, D.DNAME
FROM EMP E LEFT OUTER JOIN DEPT D
  ON E.DEPTNO = D.DEPTNO AND D.LOC = 'SEOUL';
```

① SELECT E.EMPNO, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO(+) = D.DEPTNO AND D.LOC(+) = 'SEOUL';

② SELECT E.EMPNO, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO = D.DEPTNO(+) AND D.LOC = 'SEOUL';

③ SELECT E.EMPNO, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO = D.DEPTNO(+) AND D.LOC(+) = 'SEOUL';

④ SELECT E.EMPNO, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO(+) = D.DEPTNO AND D.LOC = 'SEOUL';

**정답** ③

**해설** `LEFT OUTER JOIN`은 왼쪽(E)이 기준이고 오른쪽(D) 데이터가 부족하므로 오른쪽 컬럼에 `(+)`를 붙여야 합니다. 특히 `ON` 절에 들어간 우측 테이블 부가 조건(`D.LOC = 'SEOUL'`)을 Oracle 구문으로 옮길 때는 반드시 부가 조건에도 `D.LOC(+) = 'SEOUL'`처럼 `(+)`를 붙여야 아우터 조인이 유지됩니다(3번 정답). 2번처럼 `(+)`를 빼먹으면 INNER JOIN으로 변질됩니다.

### 문항 19. [서브쿼리] `NOT IN`과 `NULL`의 치명적 함정

**Q.** EMP 테이블은 총 10건이며, 상사 사번(MGR) 컬럼에는 `[7839, 7698, NULL]` 값들이 포함되어 있다. 아래 (가), (나) 두 SQL을 실행했을 때 반환되는 행의 수에 대한 설명으로 옳은 것은?

```
(가) SELECT COUNT(*) FROM EMP WHERE EMPNO IN (SELECT MGR FROM EMP);
(나) SELECT COUNT(*) FROM EMP WHERE EMPNO NOT IN (SELECT MGR FROM EMP);
```

① (가)와 (나) 모두 0을 반환한다.

② (가)는 상사인 사원 수를 정상 반환하지만, (나)는 서브쿼리의 NULL 때문에 0을 반환한다.

③ (가)는 0을 반환하지만, (나)는 상사가 아닌 말단 사원 수를 정상 반환한다.

④ (가)의 결과와 (나)의 결과를 더하면 전체 사원 수인 10이 된다.

**정답** ②

**해설** - (가) `IN (7839, 7698, NULL)`은 `EMPNO = 7839 OR EMPNO = 7698 OR EMPNO = NULL(UNKNOWN)`이므로 OR 연산에 의해 일치하는 상사 사원들이 정상 출력됩니다.
- (나) 그러나 `NOT IN (7839, 7698, NULL)`은 `EMPNO <> 7839 AND EMPNO <> 7698 AND EMPNO <> NULL(UNKNOWN)`으로 변환되어, 마지막 `AND UNKNOWN` 때문에 모든 행의 조건 결과가 UNKNOWN이 되어 무조건 0건(COUNT = 0)이 됩니다! 따라서 정답은 2번입니다.

### 문항 20. [서브쿼리] 스칼라 서브쿼리와 다중행 연산자 (ANY / ALL)

**Q.** 다음 중 서브쿼리(Subquery)에 대한 설명으로 가장 부적절한 것은?

① SELECT 절에 사용하는 스칼라 서브쿼리의 조회 결과가 0건(공집합)이면 에러가 발생하지 않고 NULL을 반환한다.

② 스칼라 서브쿼리의 조회 결과가 2건 이상이면 단일 행 반환 제약 위반으로 런타임 에러가 발생한다.

③ WHERE SAL > ALL (SELECT SAL FROM EMP WHERE DEPTNO = 20)` 조건은 20번 부서의 최소 급여(MIN)보다 크면 참이 된다.

④ FROM 절에 사용하는 인라인 뷰(Inline View) 내부에서는 ORDER BY 절을 사용할 수 있다.

**정답** ③

**해설** `> ALL (집합)`은 집합 내의 모든 값보다 커야 하므로 가장 큰 값인 최대값(`> MAX`)보다 커야 참이 됩니다! 최소값(`> MIN`)보다 크면 참이 되는 연산자는 `> ANY` (또는 `> SOME`)이므로 3번이 틀린 설명입니다.

### 문항 21. [집합연산자] 집합 연산자 (UNION vs UNION ALL, INTERSECT, MINUS)

**Q.** 테이블 R1에 `[10, 20, 20]` 총 3개 행이 있고, 테이블 R2에 `[20, 30]` 총 2개 행이 있을 때, 아래 (가), (나), (다) 쿼리 실행 결과의 행 개수가 바르게 연결된 것은?

```
(가) SELECT COL FROM R1 UNION ALL SELECT COL FROM R2;
(나) SELECT COL FROM R1 UNION SELECT COL FROM R2;
(다) SELECT COL FROM R1 UNION SELECT COL FROM R1;
```

① (가) 5건 / (나) 4건 / (다) 3건

② (가) 5건 / (나) 3건 / (다) 2건

③ (가) 3건 / (나) 3건 / (다) 3건

④ (가) 5건 / (나) 3건 / (다) 3건

**정답** ②

**해설** - (가) `UNION ALL`은 중복을 제거하지 않고 단순 결합하므로 `3 + 2 = 5건` `[10, 20, 20, 20, 30]`입니다.
- (나) `UNION`은 두 테이블 간 중복뿐만 아니라 R1 내부 자체 중복까지 모두 제거하여 `[10, 20, 30]` 총 3건이 됩니다.
- (다) R1을 자기 자신과 `UNION` 하더라도 중복 제거(Distinct Sort)가 수행되므로 중복된 20이 하나로 합쳐져 `[10, 20]` 총 2건만 남습니다! 따라서 정답은 2번입니다.

### 문항 22. [그룹함수] 그룹 함수 (ROLLUP, CUBE, GROUPING SETS)

**Q.** 다음 중 `GROUP BY CUBE(DEPTNO, JOB)` 구문과 완전히 동일한 집계 결과를 생성하는 SQL 구문은?

① GROUP BY ROLLUP(DEPTNO, JOB)

② GROUP BY GROUPING SETS ((DEPTNO, JOB), (DEPTNO), (JOB))

③ GROUP BY GROUPING SETS ((DEPTNO, JOB), (DEPTNO), (JOB), ())

④ GROUP BY DEPTNO, ROLLUP(JOB)

**정답** ③

**해설** `CUBE(A, B)`는 $2^2 = 4$개의 그룹 조합인 ① `(A, B)` 상세, ② `(A)` 소계, ③ `(B)` 소계, ④ `()` 전체 총계를 모두 생성합니다. 이를 `GROUPING SETS`로 똑같이 표현하려면 반드시 마지막에 전체 총계를 뜻하는 빈 괄호 `()`까지 포함해야 하므로 3번이 정답입니다. (2번은 전체 총계가 빠져 있어 가장 많이 속는 함정 보기입니다!)

### 문항 23. [윈도우함수] 순위 함수 3총사 (RANK vs DENSE_RANK vs ROW_NUMBER)

**Q.** 4명의 사원 급여가 `[5000, 3000, 3000, 2000]`일 때, 아래 SQL 실행 결과에서 급여가 2000인 사원의 (가), (나), (다) 값으로 올바른 것은?

```
SELECT SAL,
       RANK() OVER (ORDER BY SAL DESC) AS (가),
       DENSE_RANK() OVER (ORDER BY SAL DESC) AS (나),
       ROW_NUMBER() OVER (ORDER BY SAL DESC) AS (다)
FROM EMP;
```

① (가) 4 / (나) 3 / (다) 4

② (가) 3 / (나) 4 / (다) 4

③ (가) 4 / (나) 4 / (다) 3

④ (가) 3 / (나) 3 / (다) 4

**정답** ①

**해설** 급여가 `[5000, 3000, 3000, 2000]`으로 공동 2등이 2명일 때:
- (가) `RANK()`는 공동 순위만큼 건너뛰므로 1, 2, 2, 4가 됩니다.
- (나) `DENSE_RANK()`는 순위를 건너뛰지 않고 빽빽하게 이어가므로 1, 2, 2, 3이 됩니다.
- (다) `ROW_NUMBER()`는 동점이어도 무조건 고유 일련번호를 부여하므로 1, 2, 3, 4가 됩니다. 따라서 2000인 사원의 결과는 `4, 3, 4`(1번)입니다.

### 문항 24. [윈도우함수] 윈도우 프레임 (ROWS vs RANGE)와 ORDER BY 기본값

**Q.** 아래와 같이 2행과 3행의 주문일자(DT)가 `'01.02'`로 동일한 T_ORDER 테이블에서 다음 SQL을 실행했을 때, 2행(ID=2)의 S1과 S2 컬럼 값으로 올바른 것은?

```
[T_ORDER 데이터]
ID=1 (DT: '01.01', AMT: 1000)
ID=2 (DT: '01.02', AMT: 1000)  <-- 3행과 DT 동점!
ID=3 (DT: '01.02', AMT: 2000)  <-- 2행과 DT 동점!

SELECT ID,
       SUM(AMT) OVER (ORDER BY DT) AS S1,
       SUM(AMT) OVER (ORDER BY DT ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS
S2
FROM T_ORDER;
```

① S1 = 2000 / S2 = 2000

② S1 = 4000 / S2 = 2000

③ S1 = 2000 / S2 = 4000

④ S1 = 4000 / S2 = 4000

**정답** ②

**해설** 윈도우 함수 OVER 절에 `ORDER BY`만 기술하고 프레임(`ROWS`/`RANGE`)을 생략하면 기본값은 `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW`입니다!
- `S1`(RANGE 기본값): 정렬 기준인 DT 값이 `'01.02'`로 같은 2행(1000)과 3행(2000)을 하나의 논리적 범위로 묶어 일괄 합산하므로, 2행에서도 `1000 + (1000 + 2000) = 4000`이 출력됩니다!
- `S2`(ROWS 지정): 물리적인 행 단위로 한 줄씩만 더하므로 2행에서는 `1000 + 1000 = 2000`이 출력됩니다. 정답은 2번입니다.

### 문항 25. [윈도우함수] 비율/행순서 함수 (PERCENT_RANK, CUME_DIST, NTILE, RATIO_TO_REPORT)

**Q.** 총 10명의 학생 데이터를 성적순으로 정렬하여 아래 윈도우 함수를 적용했을 때의 결과 설명으로 옳은 것은?

① PERCENT_RANK()` 함수를 적용하면 1등 학생의 결과값은 0.1 이다.

② CUME_DIST()` 함수를 적용하면 1등 학생의 결과값은 0 이다.

③ NTILE(4)` 함수를 적용하면 1조부터 4조까지의 인원수는 각각 3명, 3명, 2명, 2명으로 배정된다.

④ RATIO_TO_REPORT(SCORE) OVER (ORDER BY SCORE DESC)` 구문은 성적순 누적 비율을 정상 반환한다.

**정답** ③

**해설** 10명을 `NTILE(4)`로 나누면 $10 \div 4 = \text{몫 } 2\text{명, 나머지 } 2\text{명}$이 발생하며, 나머지 인원은 항상 1번 조부터 차례대로 1명씩 추가 배정되므로 `(2+1), (2+1), 2, 2` → 3명, 3명, 2명, 2명이 됩니다(3번 정답). 1번 `PERCENT_RANK`의 1등은 무조건 `0`이며, 2번 `CUME_DIST`의 1등(단독)은 `1/10 = 0.1`이고, 4번 `RATIO_TO_REPORT`는 OVER 절 안에 `ORDER BY`를 사용할 수 없습니다.

### 문항 26. [TopN쿼리] Oracle ROWNUM의 동작 원리와 Top-N 쿼리

**Q.** Oracle DBMS에서 EMP 테이블(총 14건)의 급여(SAL) 상위 2번째~5번째 사원을 조회하거나 상위 N명을 추출하려는 아래 쿼리 설명 중 옳은 것은?

① SELECT * FROM EMP WHERE ROWNUM = 2;` 를 실행하면 두 번째 행 1건이 정상 출력된다.

② SELECT * FROM EMP WHERE ROWNUM > 1 AND ROWNUM <= 5;` 를 실행하면 2번째부터 5번째까지 총 4건이 출력된다.

③ SELECT * FROM EMP WHERE ROWNUM <= 5 ORDER BY SAL DESC;` 를 실행하면 전사 급여 상위 5명이 정확히 출력된다.

④ SELECT * FROM (SELECT * FROM EMP ORDER BY SAL DESC) WHERE ROWNUM <= 5;` 를 실행해야 급여 상위 5명이 정확히 출력된다.

**정답** ④

**해설** Oracle의 `ROWNUM`은 WHERE 절을 통과하는 행에 1번부터 순차 부여되므로, 인라인 뷰 없이 단독으로 `ROWNUM = 2`(1번)나 `ROWNUM > 1`(2번) 조건을 주면 첫 행(1번)이 계속 탈락하여 결과는 무조건 0건입니다. 또한 3번은 정렬(`ORDER BY`)보다 `WHERE ROWNUM <= 5`가 먼저 실행되어 무작위 5명을 먼저 자른 뒤 정렬하므로 오답입니다. 반드시 4번처럼 인라인 뷰에서 정렬을 먼저 끝내고 바깥에서 ROWNUM으로 잘라야 합니다.

### 문항 27. [계층형쿼리] 계층형 질의 (CONNECT BY PRIOR 순방향 vs 역방향 전개)

**Q.** 조직도 테이블 ORG(EMPNO: 사번, ENAME: 성명, MGR: 상사사번)에서 사장(MGR IS NULL)부터 시작하여 부하 직원으로 내려가는 [순방향 전개(Top-Down)] 계층형 질의를 작성할 때, 빈칸 ( ㄱ )에 들어갈 구문으로 올바른 것을 모두 고른 것은?

```
SELECT LEVEL, EMPNO, ENAME, MGR
FROM ORG
START WITH MGR IS NULL
CONNECT BY (        ㄱ        )
ORDER SIBLINGS BY EMPNO;

[보기]
(a) PRIOR EMPNO = MGR      (b) PRIOR MGR = EMPNO
(c) MGR = PRIOR EMPNO      (d) EMPNO = PRIOR MGR
```

① (a), (b)

② (a), (c)

③ (b), (d)

④ (c), (d)

**정답** ②

**해설** 계층형 질의에서 전개 방향은 등호(`=`)의 좌우 위치가 아니라 PRIOR 키워드가 어느 컬럼에 붙어 있는가로 결정됩니다! 부모(사장)에서 자식(부하)으로 내려가는 순방향 전개(Top-Down)는 직전 행(`PRIOR`)의 사번(`EMPNO`, 자식키)이 현재 행의 매니저번호(`MGR`, 부모키)와 같아야 하므로, `PRIOR EMPNO = MGR`(a) 또는 좌우만 바꾼 `MGR = PRIOR EMPNO`(c)가 똑같이 순방향 전개가 됩니다! ((b)와 (d)는 역방향 전개)

### 문항 28. [pivot & unpivot] PIVOT과 UNPIVOT의 구조 및 암묵적 그룹화 규칙

**Q.** 주문실적 테이블 SALES(ORDER_ID, REGION, QUARTER, AMT)에서 지역(REGION)별로 분기(QUARTER: 'Q1', 'Q2') 매출합계를 가로 열로 회전시켜 지역당 1줄씩 총 2줄(서울, 부산)로 요약 출력하려고 한다. 다음 중 가장 올바른 SQL은?

① SELECT * FROM SALES PIVOT (SUM(AMT) FOR QUARTER IN ('Q1', 'Q2'));

② SELECT * FROM (SELECT REGION, QUARTER, AMT FROM SALES) PIVOT (SUM(AMT) FOR QUARTER IN ('Q1', 'Q2'));

③ SELECT * FROM (SELECT REGION, QUARTER, AMT FROM SALES) PIVOT (AMT FOR QUARTER IN ('Q1', 'Q2'));

④ SELECT * FROM (SELECT REGION, QUARTER, AMT FROM SALES) UNPIVOT (SUM(AMT) FOR QUARTER IN ('Q1', 'Q2'));

**정답** ②

**해설** `PIVOT` 절은 내부의 집계 컬럼(`AMT`)과 `FOR` 절 컬럼(`QUARTER`)을 제외한 나머지 모든 원본 컬럼을 암묵적 `GROUP BY` 기준으로 삼습니다. 1번처럼 원본 SALES 테이블을 그대로 넣으면 기본키인 `ORDER_ID`까지 GROUP BY 기준에 포함되어 지역별로 합쳐지지 않고 주문 건수만큼 행이 쪼개집니다! 따라서 2번처럼 인라인 뷰로 필요한 3개 컬럼(`REGION, QUARTER, AMT`)만 추출한 뒤 `PIVOT`(집계함수 필수!)을 수행해야 합니다.

### 문항 29. [정규표현식] 정규표현식 (REGEXP_LIKE, 메타문자 `^` vs `[^]`)

**Q.** 다음 정규표현식 조건 중 문자열 `'A123'`과 매칭(TRUE)되지 않는 것은?

① REGEXP_LIKE('A123', '^[A-Z][0-9]+$')

② REGEXP_LIKE('A123', '[^0-9]')

③ REGEXP_LIKE('A123', '^[0-9]')

④ REGEXP_LIKE('A123', 'A.23')

**정답** ③

**해설** 대괄호 밖의 캐럿(`^[0-9]`)은 '숫자로 시작하는 문자열'을 뜻하므로 영문자 `'A'`로 시작하는 `'A123'`은 매칭에 실패합니다(3번 정답). 반면 2번 대괄호 안의 캐럿(`[^0-9]`)은 '숫자가 아닌 문자가 1개라도 포함되어 있는가?'를 묻는 것이므로 `'A'` 때문에 TRUE가 됩니다!

### 문항 30. [관리구문] DML / DDL / TCL과 트랜잭션, SAVEPOINT, Auto-Commit

**Q.** Oracle DBMS에서 빈 테이블 T(COL NUMBER)를 생성한 직후 아래 SQL 문장들을 순서대로 실행했을 때, 최종적으로 테이블 T에 남아있는 데이터의 행 개수와 값으로 올바른 것은?

```
INSERT INTO T VALUES (10);
SAVEPOINT SP1;
INSERT INTO T VALUES (20);
SAVEPOINT SP2;
DELETE FROM T WHERE COL = 10;
ROLLBACK TO SP1;
INSERT INTO T VALUES (30);
CREATE TABLE T_DUMMY (ID NUMBER); -- DDL 실행!
ROLLBACK;
```

① 0건 (공집합)

② 1건 (10)

③ 2건 (10, 30)

④ 3건 (10, 20, 30)

**정답** ③

**해설** 단계별 트랜잭션 상태를 추적하면:
1. `10` 입력 후 `SP1` 저장 → `20` 입력 및 `10` 삭제 수행.
2. `ROLLBACK TO SP1` 실행 → `SP1` 이후에 수행한 `20` 입력과 `10` 삭제가 모두 취소되어 `[10]`만 남음 (이때 `SP2`는 소멸).
3. `30` 입력 → 현재 미결 데이터는 `[10, 30]`.
4. `CREATE TABLE T_DUMMY`(DDL) 실행! → Oracle에서 DDL이 실행되는 순간 직전까지의 미결 DML(`[10, 30]`)이 암시적으로 자동 COMMIT(Auto-Commit) 됩니다!
5. 따라서 마지막 `ROLLBACK;`은 아무 효과가 없으며 최종적으로 10, 30 총 2건(3번)이 영구 반영됩니다.