핵심정리

BEGIN TRANSACTION (BEGIN TRAN 구문도 가능)으로 트랜잭션을 시작하고 COMMIT TRANSACTION(TRANSACTION은 생략 가능) 또는 ROLLBACK TRANSACTION(TRANSACTION은 생략 가능)으로 트랜잭션을 종료한다. ROLLBACK 구문을 만나면 최초의 BEGIN TRANSACTION 시점까지 모두 ROLLBACK이 수행된다.

저장점(SAVEPOINT)을 정의하면 롤백(ROLLBACK) 할 때 트랜잭션에 포함된 전체 작업을 롤백하는 것이 아니라 현 시점에서 SAVEPOINT까지 트랜잭션의 일부만 롤백할 수 있다

[Oracle]
```
SAVEPOINT  SVPT1;
...
ROLLBACK TO  SVPT1;
```

[SQL Server]
```
SAVE TRANSACTION
   SVTR1;
...
ROLLBACK
   TRANSACTION
   SVTR1;
```

31 아래와 같은 테이블에 SQL구문이 실행되었을 경우 최종 출력 값을 작성하시오.

아 래

[품목]

품목ID	단가
001	1000
002	2000
003	1000
004	2000

[SQL구문]
```
BEGIN TRANSACTION
INSERT INTO 품목(품목ID, 단가)
  VALUES('005',2000)
COMMIT
BEGIN TRANSACTION
DELETE 품목 WHERE 품목ID='002'
BEGIN TRANSACTION
UPDATE 품목 SET 단가=2000 WHERE
  단가=1000
ROLLBACK
SELECT COUNT(품목ID) FROM 품목
  WHERE 단가=2000
```

① 0
② 2
③ 3
④ 4

32 아래의 상품 테이블의 데이터에 대하여 관리자가 아래와 같이 SQL문장을 실행하여 데이터를 변경하였다. 데이터 변경 후의 상품ID '001'의 최종 상품명을 작성하시오.

아 래

[테이블 : 상품]

상품ID	상품명
001	TV

[SQL]
```
BEGIN TRANSACTION;
SAVE TRANSACTION SP1;
UPDATE 상품 SET 상품명 = 'LCD-TV' WHERE 상품ID = '001';
SAVE TRANSACTION SP2;
UPDATE 상품 SET 상품명 = '평면-TV' WHERE 상품ID = '001';
ROLLBACK TRANSACTION SP2;
COMMIT;
```

핵심정리

WHERE 절은 FROM 절 다음에 위치하며,
조건식은 아래 내용으로 구성된다.
- 칼럼(Column)명 (보통 조건식의 좌측에 위치)
- 비교 연산자
- 문자, 숫자, 표현식 (보통 조건식의 우측에 위치)
- 비교 칼럼명 (JOIN 사용시)

연산자의 우선순위
① 괄호로 묶은 연산
② 부정 연산자(NOT)
③ 비교 연산자 (=,>,>=,<,<=)와 SQL 비교 연산자 (BETWEEN a AND b, IN (list), LIKE, IS NULL)
④ 논리 연산자 중 AND, OR의 순으로 처리

NULL의 연산
- NULL 값과의 연산(+,-,*,/ 등)은 NULL 값을 리턴
- NULL 값과의 비교연산 (=,>,>=,<,<=)은 거짓(FALSE)을 리턴
- 특정 값보다 크다, 적다라고 표현할 수 없음

33 아래의 [㉠] 에 들어갈 내용을 적으시오.

아 래

SQL을 사용하여 데이터베이스에서 데이터를 조회할 때 원하는 데이터만을 검색하기 위해서 SELECT, FROM 절과 함께 [㉠] 을(를) 이용하여 조회되는 데이터의 조건을 설정하여 데이터를 제한할 수 있다.

34 다음 중 SQL의 실행 결과로 가장 적절한 것은?

아 래

EMP_TBL

EMPNO	SAL
100	1500
200	3000
300	2000

```
SELECT   COUNT(*)
FROM     EMP_TBL
WHERE    EMPNO > 100 AND SAL >= 3000 OR EMPNO = 200;
```

① 0
② 1
③ 2
④ 3

35 다음 중 SELECT COL1 + COL3 FROM TAB_A; 의 결과로 가장 적절한 것은?

아 래

TAB_A (레코드 3건)

COL1	COL2	COL3
30	NULL	20
NULL	10	40
50	NULL	NULL

① NULL
② 80
10
60
③ 150
④ 50
NULL
NULL

핵심정리

부정 비교 연산자

- != : 같지 않다.
- ^= : 같지 않다.
- <> : 같지 않다. (ISO 표준, 모든 운영체제에서 사용 가능)
- NOT 칼럼명 = : ~와 같지 않다.
- NOT 칼럼명 > : ~보다 크지 않다.

36 다음 SQL 문장 중 COLUMN1의 값이 널(NULL)이 아닌 경우를 찾아내는 문장으로 가장 적절한 것은? (ANSI 표준 기준)

① SELECT * FROM MYTABLE WHERE COLUMN1 IS NOT NULL
② SELECT * FROM MYTABLE WHERE COLUMN1 <> NULL
③ SELECT * FROM MYTABLE WHERE COLUMN1 != NULL
④ SELECT * FROM MYTABLE WHERE COLUMN1 NOT NULL

37 아래와 같은 DDL 문장으로 테이블 생성하고, SQL들을 수행하였을 때 다음 설명 중 옳은 것은?

아 래

```
CREATE TABLE 서비스
(
    서비스번호 VARCHAR2(10) PRIMARY KEY,
    서비스명 VARCHAR2(100) NULL,
    개시일자 DATE NOT NULL
);

[SQL]
㉠ SELECT * FROM 서비스 WHERE 서비스번호 = 1;
㉡ INSERT INTO 서비스 VALUES ('999', '', '2015-11-11');
㉢ SELECT * FROM 서비스 WHERE 서비스명 = '';
㉣ SELECT * FROM 서비스 WHERE 서비스명 IS NULL;
```

① 서비스번호 컬럼에 모든 레코드 중에서 '001'과 같은 숫자형식으로 하나의 레코드만이라도 입력되어 ㉠은 오류 없이 실행된다.
② ORACLE에서 ㉡과같이 데이터를 입력하였을 때, 서비스명 컬럼에 공백문자 데이터가 입력된다.
③ ORACLE에서 ㉡과같이 데이터를 입력하고, ㉢과 같이 조회하였을 때, 데이터는 조회된다.
④ SQL Server에서 ㉡과같이 데이터를 입력하고, ㉣과 같이 조회하였을 때, 데이터는 조회되지 않는다.

핵심정리

BETWEEN a AND b
a와 b의 값 사이에 있으면 된다.(a와 b 값이 포함됨)

IN (list)
리스트에 있는 값 중에서 어느 하나라도 일치하면 된다.

38 아래와 같이 월별매출 테이블에 데이터가 입력되어 있다. 다음 중 2014년 11월부터 2015년 03월까지의 매출금액 합계를 출력하는 SQL 문장으로 옳은 것은?

아 래

[테이블 : 월별매출]

년(PK)	월(PK)	매출금액
2014	01	1000
2014	02	2000
2014	03	3000
2014	11	4000
2014	12	5000
2015	01	6000
2015	02	7000
2015	03	8000
2015	11	9000
2015	12	10000

① SELECT SUM(매출금액) AS 매출금액합계
FROM 월별매출
WHERE 년 BETWEEN '2014' AND '2015'
AND 월 BETWEEN '03' AND '12';

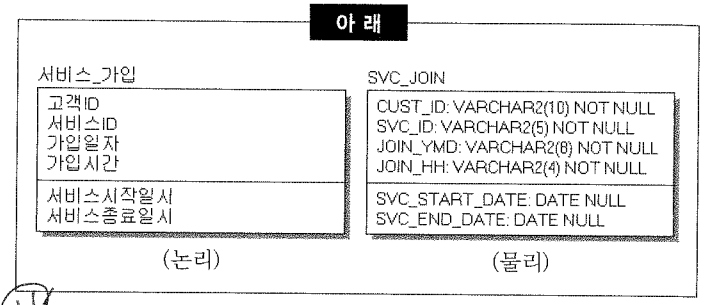
② SELECT SUM(매출금액) AS 매출금액합계
FROM 월별매출
WHERE 년 IN ('2014', '2015')
AND 월 IN ('11', '12', '03', '04', '05');

③ SELECT SUM(매출금액) AS 매출금액합계
FROM 월별매출
WHERE (년 = '2014' OR 년 = '2015')
AND (월 BETWEEN '01' AND '03' OR 월 BETWEEN '11' AND '12');

④ SELECT SUM(매출금액) AS 매출금액합계
FROM 월별매출
WHERE 년 = '2014' AND 월 BETWEEN '11' AND '12'
OR 년 = '2015' AND 월 BETWEEN '01' AND '03';

39 아래 테이블 스키마를 참조하여 SQL 문장을 작성하였다. 다음 중 결과가 다른 SQL 문장은?

아 래



① SELECT SVC_ID, COUNT(*) AS CNT
FROM SVC_JOIN
WHERE SVC_END_DATE >= TO_DATE('20150101000000', 'YYYYMMDDHH24MISS')
AND SVC_END_DATE <= TO_DATE('20150131235959', 'YYYYMMDDHH24MISS')
AND CONCAT(JOIN_YMD, JOIN_HH) = '2014120100'
GROUP BY SVC_ID;

② SELECT SVC_ID, COUNT(*) AS CNT
FROM SVC_JOIN
WHERE SVC_END_DATE >= TO_DATE('20150101', 'YYYYMMDD')
AND SVC_END_DATE < TO_DATE('20150201', 'YYYYMMDD')
AND (JOIN_YMD, JOIN_HH) IN (('20141201', '00'))
GROUP BY SVC_ID;

③ SELECT SVC_ID, COUNT(*) AS CNT
FROM SVC_JOIN
WHERE '201501' = TO_CHAR(SVC_END_DATE, 'YYYYMM')
AND JOIN_YMD = '20141201'
AND JOIN_HH = '00'
GROUP BY SVC_ID;

④ SELECT SVC_ID, COUNT(*) AS CNT
FROM SVC_JOIN
WHERE TO_DATE('201501', 'YYYYMM') = SVC_END_DATE
AND JOIN_YMD||JOIN_HH = '2014120100'
GROUP BY SVC_ID

핵심정리

연산자의 종류

구분	연산자	연산자의 의미
비교 연산자	=	같다.
	〉	보다 크다.
	〉=	보다 크거나 같다.
	〈	보다 작다.
	〈=	보다 작거나 같다.
SQL 연산자	BETWEEN a AND b	a와 b의 값 사이에 있으면 된다.(a와 b 값이 포함됨)
	IN (list)	리스트에 있는 값 중에서 어느 하나라도 일치하면 된다.
	LIKE '비교문자열'	비교문자열과 형태가 일치하면 된다.(%, _ 사용)
	IS NULL	NULL 값인 경우
논리 연산자	AND	앞에 있는 조건과 뒤에 오는 조건이 참(TRUE)이 되면 결과도 참(TRUE)이 된다. 즉, 앞의 조건과 뒤의 조건을 동시에 만족해야 한다.
	OR	앞의 조건이 참(TRUE)이거나 뒤의 조건이 참(TRUE)이 되어야 결과도 참(TRUE)이 된다. 즉, 앞뒤의 조건 중 하나만 참(TRUE)이면 된다.
	NOT	뒤에 오는 조건에 반대되는 결과를 되돌려 준다.
부정 비교 연산자	!=	같지 않다.
	^=	같지 않다.
	◇	같지 않다.(ISO 표준, 모든 운영체제에서 사용 가능)
	NOT 칼럼명 =	~와 같지 않다.
	NOT 칼럼명 〉	~보다 크지 않다.
부정 SQL 연산자	NOT BETWEEN a AND b	a와 b의 값 사이에 있지 않다. (a, b 값을 포함하지 않는다)
	NOT IN (list)	list 값과 일치하지 않는다.
	IS NOT NULL	NULL 값을 갖지 않는다.

함수는 벤더에서 제공하는 함수인 내장 함수(Built-in Function)와 사용자가 정의할 수 있는 함수(User Defined Function)로 나눌 수 있다.
내장 함수는 다시 단일행 함수(Single-Row Function)와 다중행 함수(Multi-Row Function)로 나눌 수 있으며, 다중행 함수는 집계 함수(Aggregate Function), 그룹 함수(Group Function), 윈도우 함수(Window Function) 구분된다.

40 아래와 같은 내장 함수에 대한 설명 중에서 옳은 것을 모두 묶은 것은?

아 래

(가) 함수의 입력 행수에 따라 단일행 함수와 다중행 함수로 구분할 수 있다.
(나) 단일행 함수는 SELECT, WHERE, ORDER BY, UPDATE의 SET 절에 사용이 가능하다.
(다) 1:M 관계의 두 테이블을 조인할 경우 M쪽에 다중행이 출력되므로 단일행 함수는 사용할 수 없다.
(라) 단일행 함수는 다중행 함수와 다르게 여러 개의 인수가 입력되어도 단일 값만을 반환한다.

① 가
② 가, 나
③ 가, 나, 다
④ 가, 나, 다, 라

41 다음 중 아래와 같은 2건의 데이터 상황에서 SQL의 수행 결과로 가장 적절한 것은? (단, 이해를 돕기 위해 ↓는 줄바꿈을 의미 → 실제 저장값이 아님, CHR(10) : ASCII 값 → 줄바꿈을 의미)

아 래

TAB1

ROWNUM	C1
1	A ↓ A
2	B ↓ B ↓ B

```
SELECT SUM(CC)
FROM (
    SELECT(LENGTH(C1) -
    LENGTH(REPLACE(C1,
    CHR(10))) + 1) CC
    FROM TAB1
)
```

① 2
② 3
③ 5
④ 6

단일행 문자형 함수의 종류

문자형 함수	함수 설명
LOWER(문자열)	문자열의 알파벳 문자를 소문자로 바꾸어 준다.
UPPER(문자열)	문자열의 알파벳 문자를 대문자로 바꾸어 준다.
ASCII(문자)	문자나 숫자를 ASCII 코드 번호로 바꾸어 준다.
CHR/CHAR(ASCII번호)	ASCII 코드 번호를 문자나 숫자로 바꾸어 준다.
CONCAT (문자열1, 문자열2)	Oracle, My SQL에서 유효한 함수이며 문자열1과 문자열2를 연결한다. 합성 연산자'||'(Oracle)나 '+'(SQL Server)와 동일하다.
SUBSTR/SUBSTRING (문자열, m[, n])	문자열 중 m위치에서 n개의 문자 길이에 해당하는 문자를 돌려준다. n이 생략되면 마지막 문자까지이다.
LENGTH/LEN(문자열)	문자열의 개수를 숫자값으로 돌려준다.
LTRIM (문자열 [, 지정문자])	문자열의 첫 문자부터 확인해서 지정 문자가 나타나면 해당 문자를 제거한다.(지정 문자가 생략되면 공백 값이 디폴트) SQL Server에서는 LTRIM 함수에 지정문자를 사용할 수 없다. 즉, 공백만 제거할 수 있다.
RTRIM (문자열 [, 지정문자])	문자열의 마지막 문자부터 확인해서 지정 문자가 나타나는 동안 해당 문자를 제거한다.(지정 문자가 생략되면 공백 값이 디폴트) SQL Server에서는 LTRIM 함수에 지정문자를 사용할 수 없다. 즉, 공백만 제거할 수 있다.
TRIM ([leading|trailing|both] 지정문자 FROM 문자열)	문자열에서 머리말, 꼬리말, 또는 양쪽에 있는 지정 문자를 제거한다. (leading | trailing | both 가 생략되면 both가 디폴트) SQL Server에서는 TRIM 함수에 지정문자를 사용할 수 없다. 즉, 공백만 제거할 수 있다.

※ 주: Oracle함수/SQL Server함수 표시, '/' 없는 것은 공통 함수

핵심정리

DUAL 테이블의 특성

- 사용자 SYS가 소유하며 모든 사용자가 액세스 가능한 테이블이다.
- SELECT ~ FROM ~ 의 형식을 갖추기 위한 일종의 DUMMY 테이블이다.
- DUMMY라는 문자열 유형의 칼럼에 'X'라는 값이 들어 있는 행을 1건 포함하고 있다.

42 오라클환경에서 날짜형 데이터를 다룰 경우, 아래 SQL 결과로 가장 적절한 것은?

아 래

```
SELECT TO_CHAR(TO_DATE('2015.01.10 10', 'YYYY.MM.DD HH24')
+ 1/24/(60/10), 'YYYY.MM.DD HH24:MI:SS') FROM DUAL;
```

① 2015.01.10 11:01:00
② 2015.01.10 10:05:00
③ 2015.01.10 10:10:00
④ 2015.01.10. 10:30:00

단일행 함수의 종류

종류	내용	함수의 예
문자형 함수	문자를 입력하면 문자나 숫자 값을 반환한다.	LOWER, UPPER, SUBSTR/SUBSTRING, LENGTH/ LEN, LTRIM, RTRIM, TRIM, ASCII,
숫자형 함수	숫자를 입력하면 숫자 값을 반환한다.	ABS, MOD, ROUND, TRUNC, SIGN, CHR/CHAR, CEIL/CEILING, FLOOR, EXP, LOG, LN, POWER, SIN, COS, TAN
날짜형 함수	DATE 타입의 값을 연산한다.	SYSDATE/GETDATE, EXTRACT/DATEPART, TO_NUMBER(TO_CHAR(d,'YYYY'|'MM'|'DD')) / YEAR|MONTH|DAY
변환형 함수	문자, 숫자, 날짜형 값의 데이터 타입을 변환한다.	TO_NUMBER, TO_CHAR, TO_DATE / CAST, CONVERT
NULL 관련 함수	NULL을 처리하기 위한 함수	NVL/ISNULL, NULLIF, COALESCE

43 아래는 SEARCHED_CASE_EXPRESSION SQL문장이다. 이때 사용된 SEARCHED_CASE_EXPRESSION은 SIMPLE_CASE_EXPRESSION을 이용해 똑 같은 기능을 표현할 수 있다. 아래 SQL 문장의 ㉠ 안에 들어갈 표현을 작성하시오. (스칼라 서브쿼리는 제외함)

아 래

```
[SEARCHED_CASE_EXPRESSION 문장 사례]
SELECT LOC,
   CASE WHEN LOC = 'NEW YORK' THEN 'EAST'
    ELSE 'ETC'
   END as AREA
FROM DEPT;

[SIMPLE_CASE_EXPRESSION 문장 사례]
SELECT LOC,
   CASE  ㉠
     ELSE 'ETC'
   END as AREA
FROM DEPT;
```

44 팀별 포지션별 FW, MF, DF, GK 포지션의 인원수와 팀별 전체 인원수를 구하는 SQL을 작성할 때 결과가 다른 것은? (보기 1은 SQL Server 환경이고, 보기 2,3,4는 ORACLE 환경이다)

①
```
SELECT TEAM_ID,
  ISNULL(SUM(CASE  WHEN POSITION = 'FW'  THEN 1  END), 0) FW,
  ISNULL(SUM(CASE  WHEN POSITION = 'MF'  THEN 1  END), 0) MF,
  ISNULL(SUM(CASE  WHEN POSITION = 'DF'  THEN 1  END), 0) DF,
  ISNULL(SUM(CASE  WHEN POSITION = 'GK'  THEN 1  END), 0) GK,
      COUNT(*) SUM
      FROM  PLAYER
      GROUP BY TEAM_ID;
```

②
```
SELECT TEAM_ID,
  NVL(SUM(CASE  POSITION  WHEN 'FW'  THEN 1  END),0) FW,
  NVL(SUM(CASE  POSITION  WHEN 'MF'  THEN 1  END),0) MF,
  NVL(SUM(CASE  POSITION  WHEN 'DF'  THEN 1  END),0) DF,
  NVL(SUM(CASE  POSITION  WHEN 'GK'  THEN 1  END),0) GK,
      COUNT(*) SUM
      FROM  PLAYER
      GROUP BY TEAM_ID;
```

③
```
SELECT TEAM_ID,
  NVL(SUM(CASE  WHEN POSITION = 'FW'  THEN 1  END), 0) FW,
  NVL(SUM(CASE  WHEN POSITION = 'MF'  THEN 1  END), 0) MF,
  NVL(SUM(CASE  WHEN POSITION = 'DF'  THEN 1  END), 0) DF,
  NVL(SUM(CASE  WHEN POSITION = 'GK'  THEN 1  END), 0) GK,
      COUNT(*) SUM
      FROM  PLAYER
      GROUP BY TEAM_ID;
```

④
```
SELECT TEAM_ID,
  NVL(SUM(CASE  POSITION  WHEN 'FW'  THEN 1  ELSE 1 END),0) FW,
  NVL(SUM(CASE  POSITION  WHEN 'MF'  THEN 1  ELSE 1 END),0) MF,
  NVL(SUM(CASE  POSITION  WHEN 'DF'  THEN 1  ELSE 1 END),0) DF,
  NVL(SUM(CASE  POSITION  WHEN 'GK'  THEN 1  ELSE 1 END),0) GK,
      COUNT(*) SUM
      FROM  PLAYER
      GROUP BY TEAM_ID;
```

핵심정리

NULL의 특성

- 널 값은 아직 정의되지 않은 값으로 0 또는 공백과 다르다. 0은 숫자이고, 공백은 하나의 문자이다.
- 테이블을 생성할 때 NOT NULL 또는 PRIMARY KEY로 정의되지 않은 모든 데이터 유형은 널 값을 포함할 수 있다.
- 널 값을 포함하는 연산의 경우 결과 값도 널 값이다. 모르는 데이터에 숫자를 더하거나 빼도 결과는 마찬가지로 모르는 데이터인 것과 같다.
- 결과값을 NULL이 아닌 다른 값을 얻고자 할 때 NVL/ISNULL 함수를 사용한다. NULL 값의 대상이 숫자 유형 데이터인 경우는 주로 0(Zero)으로, 문자 유형 데이터인 경우는 블랭크보다는 'X' 같이 해당 시스템에서 의미 없는 문자로 바꾸는 경우가 많다.

45 다음 중아래 TAB1을 보고 각 SQL 실행 결과를 가장 올바르게 설명한 것을 고르시오.

아 래

[TAB1]

COL1	COL2
a	NULL
b	''
c	3
d	4
e	3

① SELECT COL2 FROM TAB1 WHERE COL1 = 'b' ;
→ 실행 결과가 없다.(공집합)

② SELECT ISNULL(COL2,'X') FROM TAB1 WHERE COL1 = 'a' ;
→ 실행 결과로 'X'를 반환한다.

③ SELECT COUNT(COL1) FROM TAB1 WHERE COL2 = NULL;
→ 실행 결과는 1이다.

④ SELECT COUNT(COL2) FROM TAB1 WHERE COL1 IN ('b','c');
→ 실행 결과는 1이다.

46 사원 테이블에서 MGR의 값이 7698과 같으면 NULL을 표시하고, 같지 않으면 MGR을 표시 하려고 한다. 아래 SQL 문장의 ㉠ 안에 들어갈 함수명을 작성하시오.

아 래

```
SELECT ENAME, EMPNO, MGR, ㉠ (MGR,7698) as NM
FROM EMP;
```

단일행 NULL 관련 함수의 종류

일반형 함수	함수 설명
NVL(표현식1, 표현식2) / ISNULL(표현식1, 표현식2)	표현식1의 결과값이 NULL이면 표현식2의 값을 출력한다. 단, 표현식1과 표현식2의 결과 데이터 타입이 같아야 한다. NULL 관련 가장 많이 사용되는 함수이므로 상당히 중요하다.
NULLIF(표현식1, 표현식2)	표현식1이 표현식2와 같으면 NULL을, 같지 않으면 표현식1을 리턴한다.
COALESCE(표현식1, 표현식2, ……)	임의의 개수 표현식에서 NULL이 아닌 최초의 표현식을 나타낸다. 모든 표현식이 NULL이라면 NULL을 리턴한다.

※ 주: Oracle함수/SQL Server함수 표시, '/' 없는 것은 공통 함수

핵심정리

NULL 포함 연산의 결과

NULL + 2 2 + NULL,
NULL − 2, 2 − NULL,
NULL * 2, 2 * NULL,
NULL / 2, 2 / NULL,
의 결과는 모두 NULL 이다.

47 다음 중 아래 데이터를 가지고 있는 EMP_Q 테이블에서 세개의 SQL 결과로 가장 적절한 것은?

아 래

```
SELECT SAL/COMM FROM EMP_Q WHERE ENAME = 'KING';
SELECT SAL/COMM FROM EMP_Q WHERE ENAME = 'FORD';
SELECT SAL/COMM FROM EMP_Q WHERE ENAME = 'SCOTT';
```

※ 단, SCOTT의 COMM은 NULL 값임

EMP_Q

ENAME (문자타입)	SAL (숫자타입)	COMM (숫자타입)
KING	0	300
FORD	5000	0
SCOTT	1000	

① 0, NULL, NULL
② 0, 에러 발생, 에러 발생
③ 에러 발생, 에러 발생, NULL
④ 0, 에러 발생, NULL

48 다음 중 아래와 같은 데이터 상황에서 SQL의 수행 결과로 가장 적절한 것은?

아 래

TAB1

C1	C2	C3
1	2	3
	2	3
		3

```
SELECT SUM(COALESCE
  (C1, C2, C3))
FROM TAB1
```

① 0
② 1
③ 6
④ 14

핵심정리

NULL 관련 함수
- NVL(표현식1, 표현식2)
  Oracle 함수
  ISNULL(표현식1, 표현식2)
- SQL Server 함수
  NULLIF(표현식1, 표현식2)
  COALESCE(표현식1, 표현식2, ……)

49 아래의 각 함수에 대한 설명 중 ㉠, ㉡, ㉢에 들어갈 함수를 차례대로 작성하시오.

아 래

㉠ (표현식1,표현식2): 표현식1의 결과값이 NULL이면 표현식2의 값을 출력한다.
㉡ (표현식1,표현식2): 표현식1이 표현식2와 같으면 NULL을, 같지 않으면 표현식1을 리턴한다.
㉢ (표현식1,표현식2): 임의의 개수 표현식에서 NULL이 아닌 최초의 표현식을 나타낸다.

50 다음 중 아래 각각 3개의 SQL 수행 결과로 가장 적절한 것은?

아 래

```
SELECT AVG(COL3) FROM TAB_A;
SELECT AVG(COL3) FROM TAB_A WHERE COL1 > 0;
SELECT AVG(COL3) FROM TAB_A WHERE COL1 IS NOT NULL;
```

[TAB_A]

COL1	COL2	COL3
30	NULL	20
NULL	40	0
0	10	NULL

① 20, 20, 20
② 20, 10, 10
③ 10, 20, 20
④ 10, 10, 10

집계 함수의 종류

집계 함수	사용 목적
COUNT(*)	NULL 값을 포함한 행의 수를 출력한다.
COUNT(표현식)	표현식의 값이 NULL 값인 것을 제외한 행의 수를 출력한다.
SUM([DISTINCT | ALL] 표현식)	표현식의 NULL 값을 제외한 합계를 출력한다.
AVG([DISTINCT | ALL] 표현식)	표현식의 NULL 값을 제외한 평균을 출력한다.
MAX([DISTINCT | ALL] 표현식)	표현식의 최대값을 출력한다. (문자,날짜 데이터 타입도 사용가능)
MIN([DISTINCT | ALL] 표현식)	표현식의 최소값을 출력한다. (문자,날짜 데이터 타입도 사용가능)
STDDEV([DISTINCT | ALL] 표현식)	표현식의 표준 편차를 출력한다.
VARIAN([DISTINCT | ALL] 표현식)	표현식의 분산을 출력한다.
기타 통계 함수	벤더별로 다양한 통계식을 제공한다.

핵심정리

GROUP BY 문장

SELECT [DISTINCT]
  칼럼명 [ALIAS명]
FROM 테이블명
[WHERE 조건식]
[GROUP BY
  칼럼(Column)이나
  표현식]
[HAVING 그룹조건식] ;

51 어느 기업의 직원 테이블(EMP)이 직급(GRADE) 별로 사원 500명, 대리 100명, 과장 30명, 차장 10명, 부장 5명, 직급이 정해지지 않은(NULL) 사람 25명으로 구성되어 있을 때, 다음 중 SQL문을 SQL1)부터 SQL3)까지 순차적으로 실행한 결과 건수를 순서대로 나열한 것으로 가장 적절한 것은?

아 래

```
SQL1) SELECT COUNT(GRADE) FROM EMP;
SQL2) SELECT GRADE FROM EMP WHERE GRADE IN
      ('차장', '부장', '널');
SQL3) SELECT GRADE, COUNT(*) FROM EMP GROUP
      BY GRADE;
```

① 670, 15, 5　　② 645, 40, 5
③ 645, 15, 6　　④ 670, 40, 6

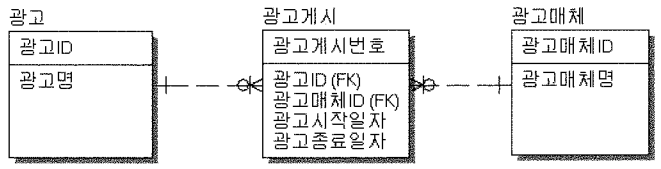
핵심정리

GROUP BY 절과 HAVING 절의 특성

- GROUP BY 절을 통해 소그룹별 기준을 정한 후, SELECT 절에 집계 함수를 사용한다.
- 집계 함수의 통계 정보는 NULL 값을 가진 행을 제외하고 수행한다.
- GROUP BY 절에서는 SELECT 절과는 달리 ALIAS 명을 사용할 수 없다.
- 집계 함수는 WHERE 절에는 올 수 없다. (집계 함수를 사용할 수 있는 GROUP BY 절보다 WHERE 절이 먼저 수행된다)
- WHERE 절은 전체 데이터를 GROUP으로 나누기 전에 행들을 미리 제거시킨다.
- HAVING 절은 GROUP BY 절의 기준 항목이나 소그룹의 집계 함수를 이용한 조건을 표시할 수 있다.
- GROUP BY 절에 의한 소그룹별로 만들어진 집계 데이터 중, HAVING 절에서 제한 조건을 두어 조건을 만족하는 내용만 출력한다.
- HAVING 절은 일반적으로 GROUP BY 절 뒤에 위치한다.

52 아래는 어느 회사의 광고에 대한 데이터 모델이다. 다음 중 광고매체 ID별 최초로 게시한 광고명과 광고시작일자를 출력하기 위하여 아래 ㉠ 에 들어갈 SQL로 옳은 것은?

아 래




```
[SQL]
SELECT C.광고매체명, B.광고명, A.광고시작일자
FROM   광고게시 A, 광고 B, 광고매체 C,
       ( [ ㉠ ] ) D
WHERE A.광고시작일자 = D.광고시작일자
AND   A.광고매체ID = D.광고매체ID
AND   A.광고ID = B.광고ID
AND   A.광고매체ID = C.광고매체ID
ORDER BY C.광고매체명;
```

① SELECT D.광고매체ID, MIN(D.광고시작일자) AS 광고시작일자
FROM 광고게시 D
WHERE D.광고매체ID = C.광고매체ID
GROUP BY D.광고매체ID

② SELECT 광고매체ID, MIN(광고시작일자) AS 광고시작일자
FROM 광고게시
GROUP BY 광고매체ID

③ SELECT MIN(광고매체ID) AS 광고매체ID, MIN(광고시작일자) AS 광고시작일자
FROM 광고게시
GROUP BY 광고ID

④ SELECT MIN(광고매체ID) AS 광고매체ID, MIN(광고시작일자) AS 광고시작일자
FROM 광고게시

53 다음 중 오류가 발생하는 SQL 문장인 것은?

① SELECT 회원ID, SUM(주문금액) AS 합계
FROM 주문
GROUP BY 회원ID
HAVING COUNT(*) > 1;

② SELECT SUM(주문금액) AS 합계
FROM 주문
HAVING AVG(주문금액) > 100;

③ SELECT 메뉴ID, 사용유형코드, COUNT(*) AS CNT
FROM 시스템사용이력
WHERE 사용일시 BETWEEN SYSDATE - 1 AND SYSDATE
GROUP BY 메뉴ID, 사용유형코드
HAVING 메뉴ID = 3 AND 사용유형코드 = 100;

④ SELECT 메뉴ID, 사용유형코드, AVG(COUNT(*)) AS AVGCNT
FROM 시스템사용이력
GROUP BY 메뉴ID, 사용유형코드;

54 다음 중 아래와 같은 테이블 A에 대해서 SQL을 수행하였을 때의 결과로 가장 적절한 것은?

아 래

```
CREATE TABLE A
(
  가 VARCHAR(5) PRIMARY KEY,
  나 VARCHAR(5) NOT NULL,
  다 INT NOT NULL
);
```

[테이블 : A]

가	나	다
001	A001	100
002	A001	200
003	A002	100
004	A002	200
005	A002	200
006	A003	100
007	A003	200
008	A003	100
009	A003	200
010	A004	200

[SQL]

```
SELECT MAX(가) AS 가, 나, SUM(다) AS 다
FROM   A
GROUP BY 나
HAVING COUNT(*) > 1
ORDER BY 다 DESC;
```

①

가	나	다
009	A003	600
005	A002	500

②

가	나	다
009	A003	600
005	A002	500
002	A001	300

③

가	나	다
009	A003	600
005	A002	500
002	A001	300
010	A004	200

④ 위의 SQL은 SELECT 절에 COUNT를 사용하지 않았으므로, HAVING 절에서 오류가 발생한다.

핵심정리

ORDER BY 문장

SELECT 칼럼명
[ALIAS명]
FROM 테이블명
[WHERE 조건식]
[GROUP BY
칼럼(Column)이나
표현식]
[HAVING 그룹조건식]
[ORDER BY
칼럼(Column)이나
표현식 [ASC 또는
DESC]] ;

- ASC(Ascending) : 조회한 데이터를 오름차순으로 정렬한다. (기본 값이므로 생략 가능)
- DESC(Descending) : 조회한 데이터를 내림차순으로 정렬한다.

55 다음 중 아래 SQL의 실행결과로 가장 적절한 것은?

아 래

TBL

ID
100
100
200
200
200
999
999

```
SELECT ID FROM TBL
GROUP BY ID
HAVING COUNT(*) = 2
ORDER BY (CASE WHEN ID = 999 THEN 0 ELSE
  ID END)
```

①

ID
100
999

②

ID
999
100

③

ID
100
200
999

④

ID
999
200
100

56 다음 SQL 중 오류가 발생하는 것은?

①
```
SELECT 지역, SUM(매출금액) AS 매출금액
FROM   지역별매출
GROUP BY 지역
ORDER BY 매출금액 DESC;
```

②
```
SELECT 지역, 매출금액
FROM   지역별매출
ORDER BY 년 ASC;
```

③
```
SELECT 지역, SUM(매출금액) AS 매출금액
FROM   지역별매출
GROUP BY 지역
ORDER BY 년 DESC;
```

④
```
SELECT 지역, SUM(매출금액) AS 매출금액
FROM   지역별매출
GROUP BY 지역
HAVING SUM(매출금액) > 1000
ORDER BY COUNT(*) ASC;
```

핵심정리

ORDER BY 절 특징

- 기본적인 정렬 순서는 오름차순(ASC)이다.
- 숫자형 데이터 타입은 오름차순으로 정렬했을 경우에 가장 작은 값부터 출력된다.
- 날짜형 데이터 타입은 오름차순으로 정렬했을 경우 날짜 값이 가장 빠른 값이 먼저 출력된다. 예를 들어 '01-JAN-2012'는 '01-SEP-2012'보다 먼저 출력된다.
- Oracle에서는 NULL 값을 가장 큰 값으로 간주하여 오름차순으로 정렬했을 경우에는 가장 마지막에, 내림차순으로 정렬했을 경우에는 가장 먼저 위치한다.
- 반면, SQL Server에서는 NULL 값을 가장 작은 값으로 간주하여 오름차순으로 정렬했을 경우에는 가장 먼저, 내림차순으로 정렬했을 경우에는 가장 마지막에 위치한다.

57 다음 중 ORDER BY 절에 대한 설명으로 가장 부적절한 것은?

① SQL 문장으로 조회된 데이터들을 다양한 목적에 맞게 특정 컬럼을 기준으로 정렬하는데 사용한다.
② DBMS마다 NULL 값에 대한 정렬 순서가 다를 수 있으므로 주의하여야 한다.
③ ORDER BY 절에서 컬럼명 대신 Alias 명이나 컬럼 순서를 나타내는 정수도 사용이 가능하나, 이들을 혼용하여 사용할 수 없다.
④ GROUP BY 절을 사용하는 경우 ORDER BY 절에 집계 함수를 사용할 수도 있다.

58 다음 SQL의 실행 결과로 가장 적절한 것은?

아 래

TBL

ID	AMT
A	50
A	200
B	300
C	100

[SQL]
```
SELECT ID, AMT
FROM TBL
ORDER BY (CASE WHEN ID = 'A' THEN 1 ELSE 2 END),
         AMT DESC
```

①

ID	AMT
B	300
A	200
C	100
A	50

②

ID	AMT
A	200
A	50
B	300
C	100

③

ID	AMT
A	50
C	100
A	200
B	300

④

ID	AMT
B	300
A	200
A	50
C	100

핵심정리

SELECT 문장 실행 순서

① 발췌 대상 테이블을 참조한다. (FROM)
② 발췌 대상 데이터가 아닌 것은 제거한다. (WHERE)
③ 행들을 소그룹화 한다. (GROUP BY)
④ 그룹핑된 값의 조건에 맞는 것만을 출력한다. (HAVING)
⑤ 데이터 값을 출력/계산한다. (SELECT)
⑥ 데이터를 정렬한다. (ORDER BY)

[TOP ( )예제] 사원 테이블에서 급여가 높은 2명을 내림차순으로 출력하는데 같은 급여를 받는 사원이 있으면 같이 출력한다.

```
SELECT TOP(2) WITH
   TIES ENAME, SAL
FROM EMP
ORDER BY SAL DESC;
```

59 다음 중 SELECT 문장의 실행 순서를 올바르게 나열한 것은?

① SELECT - FROM - WHERE - GROUP BY - HAVING - ORDER BY
② FROM - SELECT - WHERE - GROUP BY - HAVING - ORDER BY
③ FROM - WHERE - GROUP BY - HAVING - ORDER BY - SELECT
④ FROM - WHERE - GROUP BY - HAVING - SELECT - ORDER BY

60 아래의 팀별성적 테이블에서 승리건수가 높은 순으로 3위까지 출력하되 3위의 승리건수가 동일한 팀이 있다면 함께 출력하기 위한 SQL 문장으로 올바른 것은?

아 래

[테이블 : 팀별성적]

팀명	승리건수	패배건수
A팀	120	80
B팀	20	180
C팀	10	190
D팀	100	100
E팀	110	90
F팀	100	100
G팀	70	130

① SELECT TOP(3) 팀명, 승리건수
FROM 팀별성적
ORDER BY 승리건수 DESC;
② SELECT TOP(3) 팀명, 승리건수
FROM 팀별성적;
③ SELECT 팀명, 승리건수
FROM 팀별성적
WHERE ROWNUM <= 3
ORDER BY 승리건수 DESC;
④ SELECT TOP(3) WITH TIES 팀명, 승리건수
FROM 팀별성적
ORDER BY 승리건수 DESC;

핵심정리

EQUI JOIN 문장

```
SELECT 테이블1.칼럼명,
  테이블2.칼럼명, ...
FROM 테이블1, 테이블2
WHERE 테이블1.칼럼명1
  = 테이블2.칼럼명2;
```

→ WHERE 절에 JOIN 조건을 넣는다.

ANSI/ISO SQL 표준 EQUI JOIN 문장

```
SELECT 테이블1.칼럼명,
  테이블2.칼럼명, ...
FROM 테이블1 INNER
  JOIN 테이블2
ON 테이블1.칼럼명1 =
  테이블2.칼럼명2;
```

→ ON 절에 JOIN 조건을 넣는다.

61 다음 중 5개의 테이블로부터 필요한 칼럼을 조회하려고 할 때, 최소 몇 개의 JOIN 조건이 필요한가?

① 2개　② 3개
③ 4개　④ 5개

62 아래의 영화 데이터베이스 테이블의 일부에서 밑줄 친 속성들은 테이블의 기본키이며 출연료가 8888 이상인 영화명, 배우명, 출연료를 구하는 SQL로 가장 적절한 것은?

아 래

배우(배우번호, 배우명, 성별)
영화(영화번호, 영화명, 제작년도)
출연(배우번호, 영화번호, 출연료)

①
```
SELECT 출연.영화명, 영화.배우명, 출연.출연료
  FROM   배우, 영화, 출연
  WHERE 출연료 >= 8888
  AND    출연.영화번호 = 영화.영화번호
  AND    출연.배우번호 = 배우.배우번호;
```

②
```
SELECT 영화.영화명, 배우.배우명, 출연료
  FROM   영화, 배우, 출연
  WHERE 출연.출연료 > 8888
  AND    출연.영화번호 = 영화.영화번호
  AND    영화.영화번호 = 배우.배우번호;
```

③
```
SELECT 영화명, 배우명, 출연료
  FROM   배우, 영화, 출연
  WHERE 출연료 >= 8888
  AND    영화번호 = 영화.영화번호
  AND    배우번호 = 배우.배우번호;
```

④
```
SELECT 영화.영화명, 배우.배우명, 출연료
  FROM   배우, 영화, 출연
  WHERE 출연료 >= 8888
  AND    출연.영화번호 = 영화.영화번호
  AND    출연.배우번호 = 배우.배우번호;
```

핵심정리

두 개 이상의 테이블 들을 연결 또는 결합하여 데이터를 출력하는 것을 JOIN이라고 하며, 일반적인 경우 행들은 PRIMARY KEY(PK)나 FOREIGN KEY(FK) 값의 연관에 의해 JOIN이 성립된다. 하지만 어떤 경우에는 이러한 PK, FK의 관계가 없어도 논리적인 값들의 연관만으로 JOIN이 성립 가능하다.

63 다음 중 아래에서 Join에 대한 설명으로 가장 적절한 것은?

아 래

(가) 일반적으로 Join은 PK와 FK 값의 연관성에 의해 성립된다.
(나) DBMS 옵티마이져는 From 절에 나열된 테이블들을 임의로 3개 정도씩 묶어서 Join을 처리한다.
(다) EQUI Join은 Join에 관여하는 테이블 간의 컬럼 값들이 정확하게 일치하는 경우에 사용되는 방법이다.
(라) EQUI Join은 '=' 연산자에 의해서만 수행되며, 그 이외의 비교 연산자를 사용하는 경우에는 모두 Non EQUI Join이다.
(마) 대부분 Non EQUI Join을 수행할 수 있지만, 때로는 설계상의 이유로 수행이 불가능한 경우도 있다.

① 가, 다, 라
② 가, 나, 다
③ 가, 나, 다, 라
④ 가, 다, 라, 마

64 다음 SQL의 실행 결과로 맞는 것은?

아 래

[EMP_TBL]

EMPNO	ENAME
1000	SMITH
1050	ALLEN
1100	SCOTT

[RULE_TBL]

RULE_NO	RULE
1	S%
2	%T%

[SQL]

```
SELECT COUNT(*) CNT
FROM EMP_TBL A, RULE_TBL B
WHERE A.ENAME LIKE B.RULE
```

① 0
② 2
③ 4
④ 6