SQL Pogramming

- Day 10 -

2023. 04

목차

Day 10. 조인 기본

■ 조인의 필요성

▶ 조인이 필요한 기본적인 이유는 앞서 언급한 정규화에서부터 출발함

▶ 정규화 → 불필요한 데이터의 정합성을 확보하고 이상현상 발생을 피하기 위해 테이블을 분할하여 생성하는 것

▶ 데이터웨어하우스 모델처럼 하나의 테이블에 모든 데이터를 집중시켜놓고(반정규화/비정규화) 그 테이블로부터 필요한 데이터를 조회할 수도 있음.

▶ 그러나 이렇게 하는 경우 가장 중요한 데이터의 정합성에 더 큰 비용을 지불해야 하며 데이터를 추가, 수정, 삭제하는 작업 역시 상당한 노력이 요구될 것.

▶ 성능 측면에서도 간단한 데이터를 조회하는 경우에도 규모가 큰 테이블에서 필요한 데이터를 찾아야 하기 때문에 오히려 검색 속도가 떨어질 수 있음.

▶ 테이블을 정규화하여 분할하게 되면 위와 같은 문제는 자연스럽게 해결됨.

▶ 하지만 특정 요구조건을 만족하는 데이터들을 분할된 테이블로 부터 조회하기 위해서는 테이블 간에 논리적인 연관관계가 필요하고 그런 관계성을 통해 데이터들을 조회할 수 있는 것

▶ 이런 논리적인 관계를 성립시켜 주는 것이 바로 조인의 조건인 것이며 유연한 조인 기능은 관계형 데이터베이스의 가장 큰 장점임.

▶ **본 교육에서 언급하는 기본적인 조인은 STANDARD 조인이며 ,OUTER JOIN, CROSS JOIN 등의 조인은 필요에 따라 별도로 언급하도록 함**

■ 조인의 내부적 수행 유형

▶ **Nested Loop 조인** (NL조인, 중첩루프조인)

▶ Sort Merge 조인

▶ **Hash 조인** (해쉬조인)

Day 10. 조인 기본

■ 조인의 정의

- ▶ 두 개 이상의 테이블 들을 연결 또는 결합하여 데이터를 조회하는 것
- ▶ 일반적으로 SQL 문장의 상당 수가 조인이라고 생각하면 조인의 중요성을 이해할 수 있을 것
- ▶ 조인의 관계형 데이터베이스의 가장 큰 장점이면서 기본적인 기능
- ▶ 일반적인 경우 행(ROW)들은 PK나 FK에 값의 연관에 의해 조인이 성립됨
- ▶ PK, FK의 관계가 없어도 논리적인 값들의 연관성 만으로도 조인 성립이 가능함
- ▶ FROM 절에 여러 테이블이 나열되더라도 특정 시점에 데이터를 처리할 때는 단 두 개의 테이블 간에만 조인이 발생함
- ▶ FROM 절에 A, B, C 테이블이 나열되어 있더라도 특정 2개의 테이블만 먼저 조인이 발생하고 그 결과 집합과 나머지 1개의 테이블이 조인되는 것
- ▶ 4개 이상의 테이블이 나열되더라도 동일함

■ EQUI 조인

- ▶ EQUI(등가) 조인은 두 개의 테이블 간에 컬럼 값들이 서로 정확하게 일치하는 경우에 사용되는 방법으로 대부분 PK←→FK의 관계를 기반으로 함
- ▶ 그러나 일반적으로 테이블 설계시에 나타난 PK←→FK의 관계를 이용하는 것이지 반드시 PK←→FK의 관계로만 EQUI 조인이 성립하는 것은 아님

■ Non EQUI 조인

- ▶ 두 개의 테이블 간에 컬럼 값들이 서로 정확하게 일치하지 않는 경우에는 EQUI 조인을 사용할 수 없음
- ▶ 이런 경우 Non EQUI 조인을 시도하는데, (=) 연산자가 아닌 (BETWEEN, >, >=, <, <= 등)의 연산자들을 사용하여 조인을 시도함
- ▶ 이번 교육에서는 조인의 90% 이상을 차지하는 EQUI 조인에 대해 공부할 것이며, 특수한 상황(데이터 복제)에서도 몇 가지를 적용해 볼 예정임

Day 10. 조인 기본

■ 테이블의 성격

▶ 트랜잭션 테이블

○ 데이터가 지속적으로 대량 발생

○ 주로 테이블의 컬럼에 날짜 컬럼이 존재하여 매일 발생

○ SQL 성능에 지대한 영향을 미치는 테이블

▶ 마스터 테이블

○ 데이터가 지속적으로 발생할 수 있으나 대량 발생하지 않음

○ 코드, 명칭 이외에 여러가지 속성을 갖는 테이블(상품M 등)

▶ 코드 테이블

○ 코드 외 1~3개 정도의 속성을 가지는 코드 그룹을 관리

■ 테이블간 조인 유형

▶ 트랜잭션 + 마스터

▶ 트랜잭션 + 코드

▶ 트랜잭션 + 트랜잭션

▶ 트랜잭션 + 트랜잭션 + 마스터 + 코드

Day 10. 조인 기본

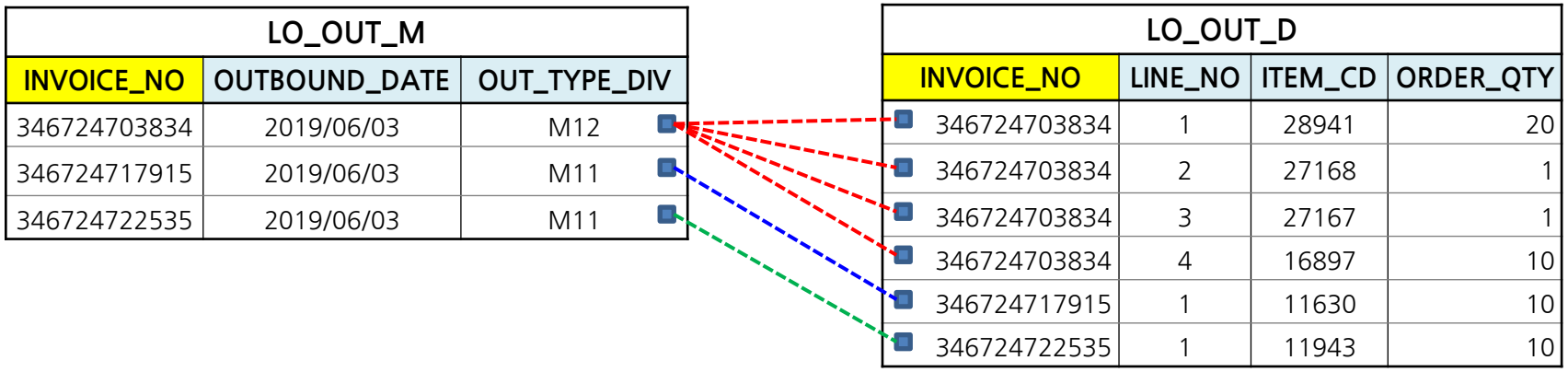
■ 조인 모델

▶ 주문 마스터(LO_OUT_M)와 주문디테일(LO_OUT_D) 테이블을 예시로 설명함

▶ 아래 도시된 모습이 INVOICE_NO라는 컬럼을 매개로 두 테이블을 조인하는 모습임

▶ 조인 문장을 기술하여 실행하면 그 결과는 몇 건이라고 예상하는가?

▶ 조인 문장을 기술하여 실행하면 그 결과는 어떻게 표시될 것이라고 예상하는가?



LO_OUT_M

INVOICE_NO	OUTBOUND_DATE	OUT_TYPE_DIV
346724703834	2019/06/03	M12
346724717915	2019/06/03	M11
346724722535	2019/06/03	M11

LO_OUT_D

INVOICE_NO	LINE_NO	ITEM_CD	ORDER_QTY
346724703834	1	28941	20
346724703834	2	27168	1
346724703834	3	27167	1
346724703834	4	16897	10
346724717915	1	11630	10
346724722535	1	11943	10

Nested Loop 조인 (NL 조인, 중첩루프조인)

Day 10. 조인 기본

■ 조인 결과

조인 결과					
INVOICE_NO	OUTBOUND_DATE	OUT_TYPE_DIV	LINE_NO	ITEM_CD	ORDER_QTY
346724703834	2019/06/03	M12	1	28941	20
346724703834	2019/06/03	M12	2	27168	1
346724703834	2019/06/03	M12	3	27167	1
346724703834	2019/06/03	M12	4	16897	10
346724717915	2019/06/03	M11	1	11630	10
346724722535	2019/06/03	M11	1	11943	10

■ SQL 구현

```
SELECT M1.INVOICE_NO      ,M1.OUTBOUND_DATE      ,M1.OUT_TYPE_DIV
      ,M2.LINE_NO         ,M2.ITEM_CD            ,M2.ORDER_QTY
  FROM LO_OUT_M M1
       JOIN LO_OUT_D M2 ON M2.INVOICE_NO = M1.INVOICE_NO
 WHERE M1.INVOICE_NO IN ('346724703834', '346724722535', '346724717915')
```

Day 10. 조인 기본

■ SQL 문법 순서

▶ SELECT → FROM → JOIN → ON → WHERE → GROUP BY → HAVING → ORDER BY

SELECT	▷	최종 결과로 추출하고 싶은 항목(테이블의 컬럼)들을 순서대로 기술함
FROM	▷	조건을 부여하고 결과를 추출하고 싶은 대상이 되는 첫번째 테이블을 기술함
JOIN	▷	FROM절에 기술된 테이블 이외 추가적으로 참조되는 테이블을 기술함(멀티 가능)
ON	▷	테이블간 연결을 위해 JOIN절에 기술된 테이블의 연결고리를 기술함(멀티 가능)
WHERE	▷	FROM절에 기술된 테이블의 컬럼에 대해 조건을 부여함
GROUP BY	▷	데이터를 그룹핑할 대상 항목(테이블의 컬럼)들을 기술함
HAVING	▷	GROUP BY를 통해 집계한 결과에 대한 조건을 부여함
ORDER BY	▷	최종 결과를 표시할 때 정렬할 순서를 기술함

Day 10. 조인 기본

■ SQL 실행 순서

▶ FROM → **ON** → JOIN → **WHERE** → GROUP BY → HAVING → **SELECT** → DISTINCT → **ORDER BY**

■ SQL 키워드 의미

▶ **FROM** : 조회 테이블 확인

▶ **ON** : 조인 조건 확인 (LEFT JOIN의 대상이 되는 테이블의 컬럼 조건은 여기에 기술 → 키값 조인을 하기 전에 해당 조건을 필터링)

○ *LEFT JOIN 시, 드라이빙 테이블의 조건은 WHERE절, 이너 테이블의 조건은 ON절에 기술*

▶ **JOIN** : 테이블 조인(병합)

▶ **WHERE** : 데이터 추출 조건 확인 (테이블 조인의 결과값에 대한 필터링)

▶ **GROUP BY** : 특정 컬럼 그룹화

▶ **HAVING** : 그룹화 이후 데이터 추출 조건 (*SELECT절의 ALIAS 사용 불가*)

▶ **SELECT** : 데이터 추출

▶ **DISTINCT** : 중복 제거

▶ **ORDER BY** : 데이터 순서 정렬 (SELECT절의 ALIAS 사용 가능)

■ 실제 실행계획 확인 방법 (예상 실행계획 아님)

▶ 실제 실행계획을 확인하고자 하는 SQL(메인 SQL)의 SELECT LIST절에 **GATHER_PLAN_STATISTICS** 힌트를 추가함.

```
SELECT --+ GATHER_PLAN_STATISTICS
       WF.GET_PROCESS_CD(M1.OUTBOUND_STATE)                              AS PROCESS_CD
      ,COUNT(DISTINCT TO_CHAR(M1.ORDER_DATE ,'YYYYMMDD') || M1.ORDER_NO) AS BILL_CNT
  FROM LO010NM M1
       JOIN LO010ND M2 ON M2.CENTER_CD  = M1.CENTER_CD
                      AND M2.BRAND_CD   = M1.BRAND_CD
                      AND M2.ORDER_DATE = M1.ORDER_DATE
                      AND M2.ORDER_NO   = M1.ORDER_NO
       JOIN CMITEM  T1 ON T1.BRAND_CD   = M2.BRAND_CD
                      AND T1.ITEM_CD    = M2.ITEM_CD
 WHERE M1.CENTER_CD        = :P_CENTER_CD
   AND M1.BRAND_CD         = :P_BRAND_CD
   AND M1.ORDER_DATE BETWEEN :P_ORDER_DATE1 AND :P_ORDER_DATE2
   AND M1.INOUT_CD      LIKE :P_INOUT_CD
   AND M1.DELIVERY_CD   LIKE :P_DELIVERY_CD
   AND M2.ITEM_CD       LIKE :P_ITEM_CD || '%'
   AND T1.ITEM_NM       LIKE '%' || :P_ITEM_NM || '%'
   AND M1.OUTBOUND_STATE   < WF.GET_PROCESS_STATE_NXT('A')
 GROUP BY WF.GET_PROCESS_CD(M1.OUTBOUND_STATE);
```

▶ 가장 최근에 실행한 SQL의 실제 실행계획을 확인함.

《 V$SESSION 권한 필요 / STATISTICS_LEVEL = ALL 》

```
SELECT *
  FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(NULL, NULL, 'ALLSTATS LAST'));
```

Day 10. 조인 기본

■ 실제 실행계획 확인 방법 (예상 실행계획 아님)

▶ 실행계획 확인

```
-----------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Starts | E-Rows | A-Rows |   A-Time   | Buffers | OMem | 1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |      1 |        |      1 |00:00:06.14 |    3369 |      |      |          |
|   1 |  HASH GROUP BY               |                   |      1 |      2 |      1 |00:00:06.14 |    3369 |  879K|  879K|  403K (0)|
|   2 |   VIEW                       | VM_NWVW_1         |      1 |      2 |    354 |00:00:06.14 |    3369 |      |      |          |
|   3 |    HASH GROUP BY             |                   |      1 |      2 |    354 |00:00:06.14 |    3369 |  848K|  848K| 1270K (0)|
|*  4 |     FILTER                   |                   |      1 |        |   400K |00:00:03.77 |    3369 |      |      |          |
|*  5 |      HASH JOIN               |                   |      1 |      2 |   400K |00:00:03.07 |    3369 |  842K|  842K| 1327K (0)|
|*  6 |       VIEW                   | index$_join$_004  |      1 |     13 |   5003 |00:00:00.05 |      91 |      |      |          |
|*  7 |        HASH JOIN             |                   |      1 |        |   5003 |00:00:00.04 |      91 |  905K|  905K| 1345K (0)|
|*  8 |         INDEX RANGE SCAN     | CMITEM_IDX01      |      1 |     13 |   5003 |00:00:00.01 |      52 |      |      |          |
|*  9 |         INDEX FAST FULL SCAN | CMITEM_IDXPK      |      1 |     13 |   5003 |00:00:00.01 |      39 |      |      |          |
|  10 |       NESTED LOOPS           |                   |      1 |    873 |   400K |00:00:01.56 |    3278 |      |      |          |
|* 11 |        TABLE ACCESS FULL     | LO010NM           |      1 |     21 |    354 |00:00:00.01 |      15 |      |      |          |
|* 12 |        INDEX RANGE SCAN      | LO010ND_IDX03     |    354 |     42 |   400K |00:00:00.99 |    3263 |      |      |          |
-----------------------------------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   4 - filter(:P_ORDER_DATE1<=:P_ORDER_DATE2)
   5 - access("T1"."BRAND_CD"="M2"."BRAND_CD" AND "T1"."ITEM_CD"="M2"."ITEM_CD")
   6 - filter("T1"."ITEM_NM" LIKE '%'||:P_ITEM_NM||'%')
   7 - access(ROWID=ROWID)
   8 - access("T1"."BRAND_CD"=:P_BRAND_CD AND "T1"."ITEM_NM" LIKE '%'||:P_ITEM_NM||'%')
   9 - filter(("T1"."BRAND_CD"=:P_BRAND_CD AND "T1"."ITEM_CD" LIKE :P_ITEM_CD||'%'))
  11 - filter(("M1"."CENTER_CD"=:P_CENTER_CD AND "M1"."DELIVERY_CD" LIKE :P_DELIVERY_CD AND "M1"."BRAND_CD"=:P_BRAND_CD AND
              "M1"."ORDER_DATE">=:P_ORDER_DATE1 AND "M1"."ORDER_DATE"<=:P_ORDER_DATE2 AND "M1"."INOUT_CD" LIKE :P_INOUT_CD))
  12 - access("M2"."CENTER_CD"=:P_CENTER_CD AND "M2"."BRAND_CD"=:P_BRAND_CD AND "M2"."ORDER_DATE"="M1"."ORDER_DATE" AND
              "M2"."ORDER_NO"="M1"."ORDER_NO" AND "M2"."ITEM_CD" LIKE :P_ITEM_CD||'%')
       filter(("M2"."ITEM_CD" LIKE :P_ITEM_CD||'%' AND "M2"."ORDER_DATE">=:P_ORDER_DATE1 AND
              "M2"."ORDER_DATE"<=:P_ORDER_DATE2))
```

■ 실제 실행계획 확인 방법 (예상 실행계획 아님)

▶ 실행계획 항목 설명

① **E-Rows** : 각 오퍼레이션이 끝났을 때 Return되는 건수 (예측 건수)

② **E-Bytes** : 각 오퍼레이션이 Return한 byte 수 (예측 byte 수)

③ **Stats** : 각 오퍼레이션을 try한 건수 (예를 들어 nested loop join이라면 인덱스를 여러 번 scan)

④ **A-Rows** : 각 오퍼레이션이 Return한 건수 (실제 건수)

⑤ **A-Time** : 각 오페레이션의 실행 시간 (실제 실행 시간)
0.1초까지 표시 (HH:MM:SS.FF) → Child Operation의 값을 합친 누적치

⑥ **Buffers** : 각 오퍼레이션이 메모리에서 읽은 Block 수

⑦ **Reads** : 각 오퍼레이션이 disk에서 읽은 Block 수

⑧ **Writes** : 각 오퍼레이션이 disk에 기록한 Block 수

⑨ **Omem, 1Mem** : optimal execution, one-pass execution에 필요한 메모리 (예측치)

⑩ **Used_Mem** : 마지막 실행시의 사용한 메모리

⑪ **Used_Tmp** : 마지막 실행시 메모리가 부족하여 temporary space를 대신 사용할 때 나타남
보이는 값에 1024를 곱해야 함 (32K → 32MB)

주문 마스터 정보 (A_OUT_M)

BRAND_CD	INVOICE_NO	OUTBOUND_DATE	OUT_TYPE_DIV	ORDER_NM
1001	#01	2023-01-03	M11	윤현수
	#02	2023-01-03	M11	전정훈
	#03	2023-01-04	M12	고선주
	#04	2023-01-05	M12	최재원
	#05	2023-01-05	M21	권민재
2001	#01	2023-01-03	M11	강민규
	#07	2023-01-04	M21	김민기
	#08	2023-01-04	M22	김민기
	#09	2023-01-04	M22	조승완
	#10	2023-01-05	M22	진효인

상품 마스터 정보 (A_ITEM)

BRAND_CD	ITEM_CD	ITEM_NM	QTY_IN_BOX
1001	A	상품A	2
	B	상품B	2
	C	상품C	2
	D	상품D	3
	E	상품E	3
2001	A	상품A	2
	B	상품B	2
	C	상품C	2
	D	상품D	3
	E	상품E	3

주문 디테일 정보 (A_OUT_D)

BRAND_CD	INVOICE_NO	LINE_NO	ITEM_CD	ORDER_QTY
1001	#01	1	A	1
	#02	1	B	1
		2	C	3
	#03	1	B	2
	#04	1	A	1
		2	D	1
		3	E	2
	#05	1	C	5
2001	#01	1	A	1
		2	B	2
	#07	1	E	1
	#08	1	C	1
	#09	1	B	3
		2	D	1
	#10	1	E	1

■ 1월 4일에 B상품 또는 D상품을 주문한 주문의 [브랜드], [출고일자], [인보이스], [라인번호], [주문수량]을 표시해 줘!

■ 1월 3일에서 1월 4일 사이에 B상품 또는 D상품을 주문한 주문의 [브랜드], [출고일자], [상품코드], [주문수량 합계]를 표시해 줘!

■ 1001 브랜드에서 1월 4일부터 1월 5일 사이에 주문한 인보이스들 중에서 총 주문수량이 가장 많은 [인보이스], [주문자]를 가르쳐 줘!

주문 마스터 정보 (A_OUT_M)

BRAND_CD	INVOICE_NO	OUTBOUND_DATE	OUT_TYPE_DIV	ORDER_NM
1001	#01	2023-01-03	M11	윤현수
	#02	2023-01-03	M11	전정훈
	#03	2023-01-04	M12	고선주
	#04	2023-01-05	M12	최재원
	#05	2023-01-05	M21	권민재
2001	#01	2023-01-03	M11	강민규
	#07	2023-01-04	M21	김민기
	#08	2023-01-04	M22	김민기
	#09	2023-01-04	M22	조승완
	#10	2023-01-05	M22	진효인

상품 마스터 정보 (A_ITEM)

BRAND_CD	ITEM_CD	ITEM_NM	QTY_IN_BOX
1001	A	상품A	2
	B	상품B	2
	C	상품C	2
	D	상품D	3
	E	상품E	3
2001	A	상품A	2
	B	상품B	2
	C	상품C	2
	D	상품D	3
	E	상품E	3

주문 디테일 정보 (A_OUT_D)

BRAND_CD	INVOICE_NO	LINE_NO	ITEM_CD	ORDER_QTY
1001	#01	1	A	1
	#02	1	B	1
		2	C	3
	#03	1	B	2
	#04	1	A	1
		2	D	1
		3	E	2
	#05	1	C	5
2001	#01	1	A	1
		2	B	2
	#07	1	E	1
	#08	1	C	1
	#09	1	B	3
		2	D	1
	#10	1	E	1

■ 1월 1일부터 1월 4일 사이에 3개 이상 주문한 상품이 있는 주문의 [브랜드], [출고일자], [인보이스], [상품코드], [상품명], [주문수량]을 표시해 줘!

■ 1월 1일부터 1월 4일 사이에 [브랜드별] & [상품별] 주문수량 합계를 표시하되, 상품명과 입수는 조인을 이용해 표시해 줘!

■ 위 결과에 인라인뷰를 적용하여 박스수, 낱개수량을 표시해 줘!

■ 위 결과에 인라인뷰를 적용하여 박스수가 가장 많은 TOP3만 표시해 줘!

Day 10. 조인 기본

■ CS_NO 테이블의 활용 방법1 (날짜 연산)

▶ 1~10,000 사이의 값을 가지는 하나의 컬럼을 보유한 테이블

▶ SELECT LEVEL FROM DUAL CONNECT BY LEVEL <= 10

→ 오라클에서만 대체 가능한 구문 (계층형 쿼리)

▶ 필요한 숫자 범위만큼 추출하여 날짜 연산에 활용 가능

→ SELECT '2023-01-01' + 1(NO) = 2023-01-02

▶ 날짜의 차이 일수만큼의 레코드를 만들 수 있음

→ WHERE NO <= '2023-01-10' - '2023-01-01'

■ CS_NO 테이블의 활용 방법2 (레코드 복제)

▶ 고정배수 복제

→ 원본 레코드를 필요한 만큼 복제하는 방법

→ 조인의 연결고리를 이어주지 않고 정해진 개수만큼 레코드를 복제

▶ 레코드의 특정 컬럼 값을 이용한 변동배수 복제

→ 조인 시, 컬럼 값에 따라 복제되는 개수를 달리하는 방법

→ 조인의 연결고리를 이어주되, 연결고리의 값이 레코드에 따라 달라짐

실전문제① ▶ 3개 테이블간의 조인을 연습하자			
《테이블》	■ LO_OUT_M(출고주문)	■ LO_OUT_D(출고주문상세)	■ CM_ITEM(상품마스터)
《조건》	■ INVOICE_NO(송장번호)	▶346724703834 or 346724722535 or 346724717915	
《정렬》	■ INVOICE_NO(송장번호), LINE_NO(송장라인번호)		
《특징》	■ 조인의 개념을 완벽하게 이해하기 ■ 3개 테이블 이상의 조인도 다르지 않다는 것을 확인하기 ■ ITEM_NM(상품명) 컬럼은 CM_ITEM(상품마스터) 테이블에 있는 컬럼으로 표시하기		

결과 ▼ 총 건수 : 6건

INVOICE_NO	OUTBOUND_DATE	OUT_TYPE_DIV	LINE_NO	ITEM_CD	ITEM_NM	ORDER_QTY
346724703834	2019/06/03	M12	1	28941	동원보성녹차350ml(20개입)	20
346724703834	2019/06/03	M12	2	27168	면발의신 쟁반 막국수 405g	1
346724703834	2019/06/03	M12	3	27167	면발의신 원조 생쫄면 385g	1
346724703834	2019/06/03	M12	4	16897	웬디 양반단호박죽	10
346724717915	2019/06/03	M11	1	11630	마일드참치100g*20캔	10
346724722535	2019/06/03	M11	1	11943	매운고추참치 100g	10

실전문제② ▶ 한 SQL에서 동일한 테이블을 2회 이상 조인에 참여시키는 형태 연습하기				
《테이블》	■ LO_OUT_M(출고주문)	■ LO_OUT_D(출고주문상세)	■ CM_ITEM(상품마스터)	■ CS_CODE(상용코드)
《조건》	■ INVOICE_NO(송장번호)	▶346724703834 or 346724722535 or 346724717915		
《정렬》	■ INVOICE_NO(송장번호), LINE_NO(송장라인번호)			
《특징》	■ 동일 테이블이 조인 절에 여러 번 기술되는 것도 다른 테이블과 다르지 않다는 것을 이해하자 ■ TEMP_DIV(온도구분) → CODE_GRP='LDIV01' / OUT_TYPE_DIV(출고유형구분) → CODE_GRP = 'LDIV03'			

결과 ▼ 총 건수 : 6건

INVOICE_NO	OUTBOUND_DATE	OUT_TYPE_DIV	LINE_NO	ITEM_CD	ITEM_NM	ORDER_QTY	TEMP_NM	OUT_TYPE_NM
346724703834	2019/06/03	M12	1	28941	동원보성녹차350ml(20개입)	20	상온	[상온]DPS
346724703834	2019/06/03	M12	2	27168	면발의신 쟁반 막국수 405g	1	상온	[상온]DPS
346724703834	2019/06/03	M12	3	27167	면발의신 원조 생쫄면 385g	1	상온	[상온]DPS
346724703834	2019/06/03	M12	4	16897	웬디 양반단호박죽	10	상온	[상온]DPS
346724717915	2019/06/03	M11	1	11630	마일드참치100g*20캔	10	상온	[상온]기획
346724722535	2019/06/03	M11	1	11943	매운고추참치 100g	10	상온	[상온]기획

Thank you !

ASETEC Location http://www.asetec.co.kr

본사. 경기도 성남시 분당구 성남대로 331번길 8, 킨스타워 2201호 TEL.031.609.7000 FAX.031.609.7009
부산. 부산광역시 해운대구 센텀동로 99 TEL.051.506.6352 FAX.051.504.8794