6-1. 서브쿼리, 세미조인, 안티조인

홍형경
chariehong@gmail.com
2020.01

1. 서브쿼리 (Subquery) - 개요

· 일반적인 쿼리(메인, 주 쿼리) 안에 있는 또 다른 쿼리 ➔ 보조, 하위 쿼리

· 메인 쿼리와 서브쿼리가 합쳐져 한 문장을 이룸

· 서브쿼리는 하나의 SELECT 문장으로, 괄호로 둘러싸인 형태

· 메인 쿼리 기준으로 여러 개의 서브 쿼리 사용 가능

1. 서브쿼리 (Subquery) - 종류

· 서브 쿼리 위치에 따라
  - 스칼라 서브쿼리 (Scalar Subquery)
  - 인라인 뷰 (Inline View)
  - 중첩 서브쿼리 (Nested Subquery)

· 메인쿼리와의 연관성
  - 연관성 있는(Correlated) 서브쿼리 : 메인쿼리와 조인
  - 연관성 없는(Noncorrealted) 서브쿼리 : 메인쿼리와 독립적

· 주로 서브쿼리 위치에 따른 분류를 사용

2. 스칼라 서브쿼리 (Scalar Subquery)

· 메인쿼리의 SELECT 절에 위치한 서브쿼리

· SELECT 절에서 마치 하나의 컬럼이나 표현식 처럼 사용

· 스칼라(Scalar) : 크기만 가지는 값, 양을 의미 (수학, 물리)

· 서브쿼리 수행 결과가 하나의 값이 되므로 스칼라 서브쿼리라고 함(?)

2. 스칼라 서브쿼리 (Scalar Subquery)

· 서브쿼리가 최종 반환하는 로우 수는 1개

· 서브쿼리가 최종 반환하는 컬럼이나 표현식도 1개

· 서브쿼리에 별칭(Alias)을 주는 것이 일반적 ➔ 하나의 컬럼 역할을 하므로

· 서브쿼리 내에서 메인 쿼리와 조인 가능
  - 조인 하는 것이 일반적
  - 조인을 안하면 여러 건이 조회될 가능성이 많음
  - 조인을 한다는 것은 연관성 있는 서브쿼리란 뜻

2. 스칼라 서브쿼리 (Scalar Subquery)

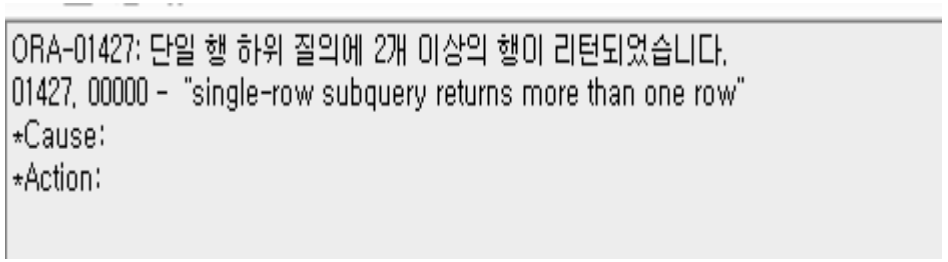
· 사용 예

```
SELECT a.employee_id,
        a.first_name || a.last_name emp_name,
       a.department_id,
     ( SELECT b.department_name
         FROM departments b
        WHERE a.department_id = b.department_id ) dept_name
  FROM employees a
 ORDER BY 1;
```

EMPLOYEE_ID	EMP_NAME	DEPARTMENT_ID	DEPT_NAME
100	StevenKing	90	Executive
101	NeenaKochhar	90	Executive
102	LexDe Haan	90	Executive
103	AlexanderHunold	60	IT
104	BruceErnst	60	IT
105	DavidAustin	60	IT
106	ValliPataballa	60	IT
107	DianaLorentz	60	IT
108	NancyGreenberg	100	Finance
109	DanielFaviet	100	Finance
110	JohnChen	100	Finance
111	IsmaelSciarra	100	Finance
112	Jose ManuelUrman	100	Finance
113	LuisPopp	100	Finance
114	DenRaphaely	30	Purchasing
115	AlexanderKhoo	30	Purchasing
116	ShelliBaida	30	Purchasing
117	SigalTobias	30	Purchasing

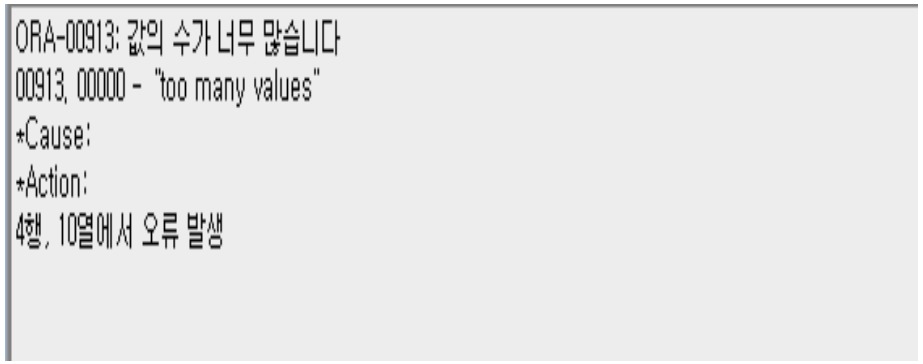
2. 스칼라 서브쿼리 (Scalar Subquery)

```
SELECT a.employee_id,
        a.first_name || a.last_name emp_name,
       a.department_id,
      ( SELECT b.department_name
          FROM departments b
      ) dept_name
  FROM employees a
 ORDER BY 1;
```



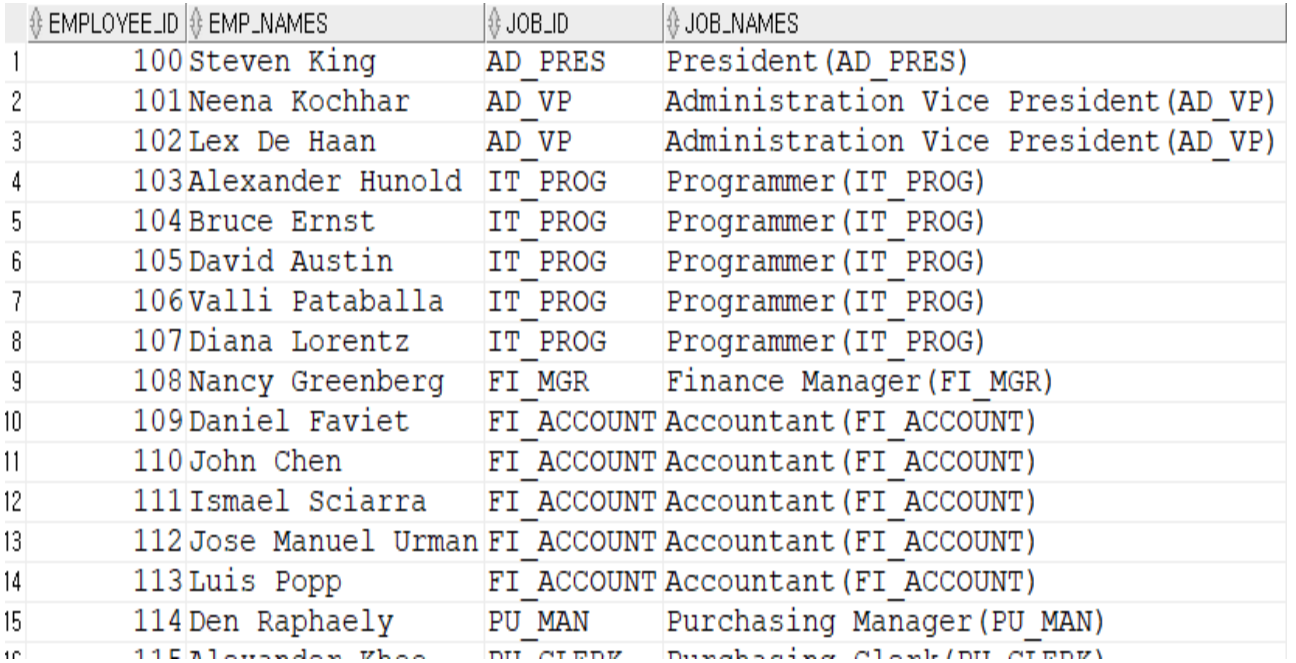
2. 스칼라 서브쿼리 (Scalar Subquery)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.department_id,
       ( SELECT b.department_name, b.location_id
           FROM departments b
          WHERE a.department_id = b.department_id
       ) dept_name
  FROM employees a
 ORDER BY 1;
```



2. 스칼라 서브쿼리 (Scalar Subquery)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_names, a.job_id
      ,( SELECT b.job_title || '(' || b.job_id || ')'
          FROM jobs b
         WHERE a.job_id = b.job_id
       )  job_names
FROM employees a
ORDER BY 1;
```



	EMPLOYEE_ID	EMP_NAMES	JOB_ID	JOB_NAMES
1	100	Steven King	AD_PRES	President(AD_PRES)
2	101	Neena Kochhar	AD_VP	Administration Vice President(AD_VP)
3	102	Lex De Haan	AD_VP	Administration Vice President(AD_VP)
4	103	Alexander Hunold	IT_PROG	Programmer(IT_PROG)
5	104	Bruce Ernst	IT_PROG	Programmer(IT_PROG)
6	105	David Austin	IT_PROG	Programmer(IT_PROG)
7	106	Valli Pataballa	IT_PROG	Programmer(IT_PROG)
8	107	Diana Lorentz	IT_PROG	Programmer(IT_PROG)
9	108	Nancy Greenberg	FI_MGR	Finance Manager(FI_MGR)
10	109	Daniel Faviet	FI_ACCOUNT	Accountant(FI_ACCOUNT)
11	110	John Chen	FI_ACCOUNT	Accountant(FI_ACCOUNT)
12	111	Ismael Sciarra	FI_ACCOUNT	Accountant(FI_ACCOUNT)
13	112	Jose Manuel Urman	FI_ACCOUNT	Accountant(FI_ACCOUNT)
14	113	Luis Popp	FI_ACCOUNT	Accountant(FI_ACCOUNT)
15	114	Den Raphaely	PU_MAN	Purchasing Manager(PU_MAN)

2. 스칼라 서브쿼리 (Scalar Subquery)

-- 조인

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.department_id,
       b.department_name
  FROM employees a,
       departments b
 WHERE a.department_id = b.department_id
 ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
76	175	Alyssa Hutton	80	Sales
77	176	Jonathon Taylor	80	Sales
78	177	Jack Livingston	80	Sales
79	179	Charles Johnson	80	Sales
80	180	Winston Taylor	50	Shipping
81	181	Jean Fleaur	50	Shipping

-- 스칼라 서브쿼리

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.department_id,
       ( SELECT b.department_name
           FROM departments b
          WHERE a.department_id = b.department_id ) dept_name
  FROM employees a
 ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	DEPARTMENT_ID	DEPT_NAME
76	175	Alyssa Hutton	80	Sales
77	176	Jonathon Taylor	80	Sales
78	177	Jack Livingston	80	Sales
79	178	Kimberely Grant	(null)	(null)
80	179	Charles Johnson	80	Sales
81	180	Winston Taylor	50	Shipping

2. 스칼라 서브쿼리 (Scalar Subquery)

-- 조인

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.department_id,
       b.department_name
  FROM employees a,
       departments b
 WHERE a.department_id = b.department_id
 ORDER BY 1;
```

Description	Object owner	Object name	Cost
SELECT STATEMENT, GOAL = ALL_ROWS			7
SORT ORDER BY			7
MERGE JOIN			6
TABLE ACCESS BY INDEX ROWID	HR	DEPARTMENTS	2
INDEX FULL SCAN	HR	DEPT_ID_PK	1
SORT JOIN			4
TABLE ACCESS FULL	HR	EMPLOYEES	3

-- 스칼라 서브쿼리

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.department_id,
       ( SELECT b.department_name
           FROM departments b
          WHERE a.department_id = b.department_id ) dept_name
  FROM employees a
 ORDER BY 1;
```

Description	Object owner	Object name	Cost
SELECT STATEMENT, GOAL = ALL_ROWS			14
TABLE ACCESS BY INDEX ROWID	HR	DEPARTMENTS	1
INDEX UNIQUE SCAN	HR	DEPT_ID_PK	0
TABLE ACCESS BY INDEX ROWID	HR	EMPLOYEES	3
INDEX FULL SCAN	HR	EMP_EMP_ID_PK	1

2. 스칼라 서브쿼리 (Scalar Subquery)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       a.department_id,
       ( SELECT b.department_name
           FROM departments b
          WHERE a.department_id = b.department_id
       ) dept_name,
       ( SELECT d.country_name
           FROM departments b
                ,locations c
                ,countries d
          WHERE a.department_id = b.department_id
            AND b.location_id = c.location_id
            AND c.country_id = d.country_id
       ) country_name
  FROM employees a
ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	DEPARTMENT_ID	DEPT_NAME	COUNTRY_NAME
69	168	Lisa Ozer	80	Sales	United Kingdom
70	169	Harrison Bloom	80	Sales	United Kingdom
71	170	Tayler Fox	80	Sales	United Kingdom
72	171	William Smith	80	Sales	United Kingdom
73	172	Elizabeth Bates	80	Sales	United Kingdom
74	173	Sundita Kumar	80	Sales	United Kingdom
75	174	Ellen Abel	80	Sales	United Kingdom
76	175	Alyssa Hutton	80	Sales	United Kingdom
77	176	Jonathon Taylor	80	Sales	United Kingdom
78	177	Jack Livingston	80	Sales	United Kingdom
79	178	Kimberely Grant	(null)	(null)	(null)
80	179	Charles Johnson	80	Sales	United Kingdom
81	180	Winston Taylor	50	Shipping	United States of America
82	181	Jean Fleaur	50	Shipping	United States of America

3. 인라인 뷰 (Inline View)

- 메인쿼리의 FROM 절에 위치
- 서브쿼리 자체가 마치 하나의 테이블 처럼 동작
- 서브쿼리가 최종 반환하는 로우와 컬럼, 표현식 수는 1개 이상 가능
- 서브쿼리에 대한 별칭(Alias)은 반드시 명시
- 메인쿼리와 조인조건은 메인 쿼리의 WHERE 절에서 처리가 일반적

3. 인라인 뷰 (Inline View)

· 인라인 뷰가 필요한 이유

- 기존 단일 테이블만 읽어서는 필요한 정보를 가져오기가 어려울 때
  예, 특정 조건으로 집계한 결과와 조인 필요 시
- 인라인 뷰 내 쿼리가 여러 테이블을 읽어오는 경우가 많음

· **LATERAL** 키워드 사용 시 서브쿼리 내에서 조인 가능 ➔ 스칼라 서브쿼리처럼 동작

- 과거 서브쿼리 내에서는 메인 쿼리 참조가 불가능 (조인 불가)
- 12c 부터 추가된 기능
- 서브쿼리 앞에 LATERAL 명시할 경우 메인 쿼리 컬럼 참조 가능

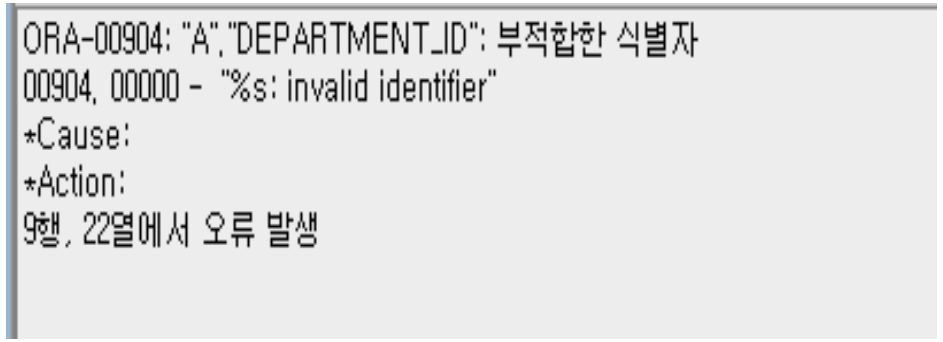
3. 인라인 뷰 (Inline View)

```
SELECT a.employee_id,
        a.first_name || a.last_name emp_name,
       a.department_id,
       c.dept_name
FROM employees a,
       ( SELECT b.department_id,
                b.department_name  dept_name
          FROM departments b ) c
 WHERE a.department_id = c.department_id
 ORDER BY 1;
```

EMPLOYEE_ID	EMP_NAME	DEPARTMENT_ID	DEPT_NAME
100	StevenKing	90	Executive
101	NeenaKochhar	90	Executive
102	LexDe Haan	90	Executive
103	AlexanderHunold	60	IT
104	BruceErnst	60	IT
105	DavidAustin	60	IT
106	ValliPataballa	60	IT
107	DianaLorentz	60	IT
108	NancyGreenberg	100	Finance
109	DanielFaviet	100	Finance
110	JohnChen	100	Finance
111	IsmaelSciarra	100	Finance
112	Jose ManuelUrman	100	Finance
113	LuisPopp	100	Finance
114	DenRaphaely	30	Purchasing
115	AlexanderKhoo	30	Purchasing
116	ShelliBaida	30	Purchasing
117	SigalTobias	30	Purchasing

3. 인라인 뷰 (Inline View)

```
SELECT a.employee_id,
       a.first_name || a.last_name emp_name,
       a.department_id,
       c.dept_name
FROM employees a,
   ( SELECT b.department_id,
                b.department_name  dept_name
       FROM departments b
      WHERE a.department_id = b.department_id
          ) c
 ORDER BY 1;
```



3. 인라인 뷰 (Inline View)

```
SELECT a.employee_id,
        a.first_name || a.last_name emp_name,
       a.department_id,
       c.dept_name
FROM employees a,
     LATERAL
      ( SELECT b.department_name  dept_name
        FROM departments b
       WHERE a.department_id = b.department_id ) c
 ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	DEPARTMENT_ID	DEPT_NAME
73	172	ElizabethBates	80	Sales
74	173	SunditaKumar	80	Sales
75	174	EllenAbel	80	Sales
76	175	AlyssaHutton	80	Sales
77	176	JonathonTaylor	80	Sales
78	177	JackLivingston	80	Sales
79	179	CharlesJohnson	80	Sales
80	180	WinstonTaylor	50	Shipping
81	181	JeanFleaur	50	Shipping
82	182	MarthaSullivan	50	Shipping
83	183	GirardGeoni	50	Shipping
84	184	NanditaSarchand	50	Shipping
85	185	AlexisBull	50	Shipping
86	186	JuliaDellinger	50	Shipping
87	187	AnthonyCabrio	50	Shipping
88	188	KellyChung	50	Shipping
89	189	JenniferDilly	50	Shipping
90	190	TimothyGates	50	Shipping

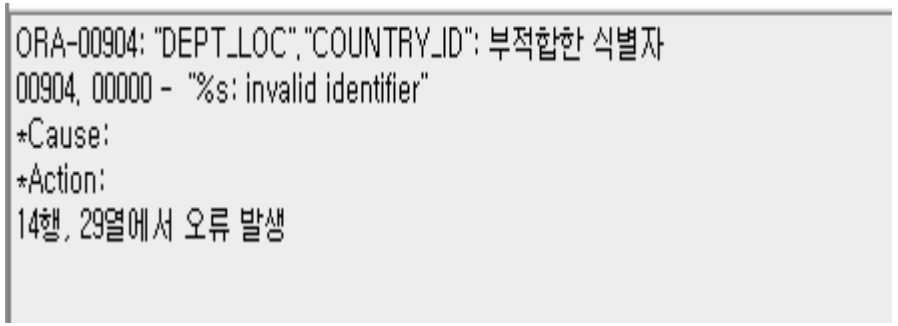
3. 인라인 뷰 (Inline View)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name emp_name,
       dept.department_name,
       loc.street_address, loc.city, loc.country_name
FROM employees a
   ,( SELECT *
       FROM departments b ) dept
   ,( SELECT l.location_id, l.street_address,
             l.city, c.country_name
       FROM locations l,
            countries c
      WHERE l.country_id = c.country_id
     ) loc
 WHERE a.department_id = dept.department_id
   AND dept.location_id = loc.location_id
 ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	DEPARTMENT_NAME	STREET_ADDRESS	CITY	COUNTRY_NAME
76	175	Alyssa Hutton	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom
77	176	Jonathon Taylor	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom
78	177	Jack Livingston	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom
79	179	Charles Johnson	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom
80	180	Winston Taylor	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
81	181	Jean Fleaur	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
82	182	Martha Sullivan	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
83	183	Girard Geoni	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
84	184	Nandita Sarchand	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
85	185	Alexis Bull	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
86	186	Julia Dellinger	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
87	187	Anthony Cabrio	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
88	188	Kelly Chung	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
89	189	Jennifer Dilly	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
90	190	Timothy Gates	Shipping	2011 Interiors Blvd	South San Francisco	United States of America
91	191	Randall Perkins	Shipping	2011 Interiors Blvd	South San Francisco	United States of America

3. 인라인 뷰 (Inline View)

```
SELECT a.employee_id,   a.first_name || ' ' || a.last_name emp_name,
       dept_loc.department_name,
       dept_loc.street_address, dept_loc.city,
       reg.country_name, reg.region_name
FROM employees a
   ,( SELECT b.department_id, b.department_name,
             l.street_address, l.city, l.country_id
        FROM departments b,  locations l
       WHERE b.location_id = l.location_id   ) dept_loc
   ,( SELECT c.country_id, c.country_name,
             r.region_name
        FROM countries c,    regions r
       WHERE c.region_id = r.region_id
         AND c.country_id = dept_loc.country_id  ) reg
 WHERE a.department_id = dept_loc.department_id
 ORDER BY 1;
```



3. 인라인 뷰 (Inline View)

```
SELECT a.employee_id,   a.first_name || ' ' || a.last_name emp_name,
       dept_loc.department_name,
       dept_loc.street_address, dept_loc.city,
       reg.country_name, reg.region_name
FROM employees a
  ,( SELECT b.department_id, b.department_name,
            l.street_address, l.city, l.country_id
       FROM departments b,  locations l
      WHERE b.location_id = l.location_id   ) dept_loc
  ,LATERAL ( SELECT c.country_id, c.country_name,
            r.region_name
       FROM countries c,    regions r
      WHERE c.region_id = r.region_id
        AND c.country_id = dept_loc.country_id  ) reg
 WHERE a.department_id = dept_loc.department_id
 ORDER BY 1;
```

	EMPLOY...	EMP_NAME	DEPARTMENT_NAME	STREET_ADDRESS	CITY	COUNTRY_NAME	REGION_NAME
73	172	Elizabeth Bates	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
74	173	Sundita Kumar	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
75	174	Ellen Abel	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
76	175	Alyssa Hutton	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
77	176	Jonathon Taylor	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
78	177	Jack Livingston	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
79	179	Charles Johnson	Sales	Magdalen Centre, The Oxford Science Park	Oxford	United Kingdom	Europe
80	180	Winston Taylor	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
81	181	Jean Fleaur	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
82	182	Martha Sullivan	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
83	183	Girard Geoni	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas
84	184	Nandita Sarchand	Shipping	2011 Interiors Blvd	South San Francisco	United States of America	Americas

4. 중첩 서브쿼리 (Nested Subquery)

- 메인쿼리의 WHERE 절에 위치
- 서브쿼리가 조건절의 일부로 사용됨
- 서브쿼리 최종 반환 값과 메인쿼리 테이블의 특정 컬럼 값을 비교 시 사용
- 서브쿼리가 최종 반환하는 로우와 컬럼, 표현식 수는 1개 이상 가능
- 조건절의 일부이므로 서브쿼리에 대한 별칭(Alias) 사용 불가
- 서브쿼리 내에서 메인쿼리와 조인 가능

4. 중첩 서브쿼리 (Nested Subquery)

```
SELECT *
  FROM departments
 WHERE department_id IN ( SELECT   department_id
                          FROM employees
                        ) ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700

4. 중첩 서브쿼리 (Nested Subquery)

```
SELECT *
  FROM departments a
 WHERE EXISTS
        ( SELECT  1
            FROM employees b
           WHERE a.department_id = b.department_id
        );
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700

4. 중첩 서브쿼리 (Nested Subquery)

```
SELECT *
  FROM departments a
 WHERE EXISTS ( SELECT 'A'
                  FROM employees b
                 WHERE a.department_id = b.department_id
                   AND b.salary > 10000 );
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	20	Marketing	201	1800
2	30	Purchasing	114	1700
3	80	Sales	145	2500
4	90	Executive	100	1700
5	100	Finance	108	1700
6	110	Accounting	205	1700

4. 중첩 서브쿼리 (Nested Subquery)

```
SELECT employee_id,
       first_name || ' ' || last_name emp_name,
       job_id,
       salary
  FROM employees
 WHERE (job_id, salary ) IN ( SELECT job_id, min_salary
                              FROM jobs)
ORDER BY 1;
```

	EMPLOYEE_ID	EMP_NAME	JOB_ID	SALARY
1	119	Karen Colmenares	PU_CLERK	2500
2	182	Martha Sullivan	SH_CLERK	2500
3	191	Randall Perkins	SH_CLERK	2500

4. 중첩 서브쿼리 (Nested Subquery)

```
SELECT last_name, employee_id
            ,salary + NVL(commission_pct, 0)
            ,job_id, e.department_id
  FROM employees e
            ,departments d
WHERE e.department_id = d.department_id
  AND salary + NVL(commission_pct,0)
            > ( SELECT salary + NVL(commission_pct,0)
                  FROM employees
                 WHERE last_name = 'Pataballa')
ORDER BY last_name, employee_id;
```

	LAST_NAME	EMPLOYEE_ID	TOT_SALARY	JOB_ID	DEPARTMENT_ID
1	Abel	174	11000.3	SA_REP	80
2	Ande	166	6400.1	SA_REP	80
3	Baer	204	10000	PR_REP	70
4	Banda	167	6200.1	SA_REP	80
5	Bates	172	7300.15	SA_REP	80
6	Bernstein	151	9500.25	SA_REP	80
7	Bloom	169	10000.2	SA_REP	80
8	Cambrault	148	11000.3	SA_MAN	80
9	Cambrault	154	7500.2	SA_REP	80
10	Chen	110	8200	FI_ACCOUNT	100
11	De Haan	102	17000	AD_VP	90
12	Doran	160	7500.3	SA_REP	80
13	Ernst	104	6000	IT_PROG	60
14	Errazuriz	147	12000.3	SA_MAN	80
15	Faviet	109	9000	FI_ACCOUNT	100
16	Fay	202	6000	MK_REP	20
17	Fox	170	9600.2	SA_REP	80
18	Fripp	121	8200	ST_MAN	50
19	Gietz	206	8300	AC_ACCOUNT	110
20	Greenberg	108	12008	FI_MGR	100
21	Greene	163	9500.15	SA_REP	80
22	Hall	152	9000.25	SA_REP	80

4. 중첩 서브쿼리 (Nested Subquery)

```
SELECT department_id, employee_id, last_name, salary
  FROM employees a
 WHERE salary > (SELECT AVG(salary)
                   FROM employees b
                  WHERE a.department_id = b.department_id)
ORDER BY department_id;
```

※ EMPLOYEES 테이블에서 자신이 속한 부서의 평균 급여보다 많이 받는 사원 정보

	DEPARTMENT_ID	EMPLOYEE_ID	LAST_NAME	SALARY
1	20	201	Hartstein	13000
2	30	114	Raphaely	11000
3	50	141	Rajs	3500
4	50	189	Dilly	3600
5	50	137	Ladwig	3600
6	50	188	Chung	3800
7	50	193	Everett	3900
8	50	192	Bell	4000
9	50	185	Bull	4100
10	50	184	Sarchand	4200
11	50	124	Mourgos	5800
12	50	123	Vollman	6500
13	50	122	Kaufling	7900
14	50	120	Weiss	8000
15	50	121	Fripp	8200
16	60	104	Ernst	6000
17	60	103	Hunold	9000

4. 중첩 서브쿼리 (Nested Subquery)

```
SELECT a.department_id, a.employee_id,
       a.last_name, a.salary,
       k.department_id second_dept_id,
       k.avg_salary
  FROM employees a,
      ( SELECT b.department_id, AVG(b.salary) avg_salary
          FROM employees b
         GROUP BY b.department_id
       ) k
 WHERE a.department_id = k.department_id
ORDER BY a.department_id;
```

	DEPARTMENT_ID	EMPLOYEE_ID	LAST_NAME	SALARY	SECOND_DEPT_ID	AVG_SALARY
1	10	200	Whalen	4400	10	4400
2	20	201	Hartstein	13000	20	9500
3	20	202	Fay	6000	20	9500
4	30	114	Raphaely	11000	30	4150
5	30	115	Khoo	3100	30	4150
6	30	116	Baida	2900	30	4150
7	30	117	Tobias	2800	30	4150
8	30	118	Himuro	2600	30	4150
9	30	119	Colmenares	2500	30	4150
10	40	203	Mavris	6500	40	6500
11	50	120	Weiss	8000	50	3475.555555555555555555555555555555555556
12	50	121	Fripp	8200	50	3475.555555555555555555555555555555555556
13	50	122	Kaufling	7900	50	3475.555555555555555555555555555555555556
14	50	123	Vollman	6500	50	3475.555555555555555555555555555555555556
15	50	124	Mourgos	5800	50	3475.555555555555555555555555555555555556
16	50	125	Nayer	3200	50	3475.555555555555555555555555555555555556
17	50	126	Mikkilineni	2700	50	3475.555555555555555555555555555555555556
18	50	127	Landry	2400	50	3475.555555555555555555555555555555555556
19	50	128	Markle	2200	50	3475.555555555555555555555555555555555556
20	50	129	Bissot	3300	50	3475.555555555555555555555555555555555556
21	50	130	Atkinson	2800	50	3475.555555555555555555555555555555555556
22	50	131	Marlow	2500	50	3475.555555555555555555555555555555555556
23	50	132	Olson	2100	50	3475.555555555555555555555555555555555556
24	50	133	Mallin	3300	50	3475.555555555555555555555555555555555556
25	50	134	Rogers	2900	50	3475.555555555555555555555555555555555556
26	50	135	Gee	2400	50	3475.555555555555555555555555555555555556
27	50	136	Philtanker	2200	50	3475.555555555555555555555555555555555556

5. 세미 조인 (Semi Join)

· 두 번째 테이블에 있는 로우와 조건이 맞는 첫 번째 테이블의 로우 반환

· 메인 쿼리와 중첩 서브쿼리를 사용할 때 사용하는 조인

· WHERE 절에서 IN, EXISTS 연산자를 사용

· IN 연산자

```
SELECT *
  FROM departments
 WHERE department_id  IN
         ( SELECT   department_id
             FROM employees
         ) ;
```

· EXISTS 연산자

```
SELECT *
  FROM departments a
 WHERE EXISTS
          ( SELECT  1
              FROM employees b
             WHERE a.department_id = b.department_id
         ) ;
```


6. 안티 조인 (Anti Join)

· 세미 조인에서 NOT 연산자 사용하는 조인

· 서브쿼리와의 조인조건에 부합하지 않는 건을 조회

```
SELECT *
  FROM departments a
 WHERE NOT EXISTS
                        ( SELECT  1
                            FROM employees b
                           WHERE a.department_id = b.department_id
                        ) ;
```


6. 안티 조인 (Anti Join)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name
  FROM employees a
 WHERE a.employee_id
              NOT IN ( SELECT employee_id
                         FROM job_history )
 ORDER BY 1;
```

	EMPLOY...	A.FIRST_NAME||' '||A.LAST_NAME
1	100	Steven King
2	103	Alexander Hunold
3	104	Bruce Ernst
4	105	David Austin
5	106	Valli Pataballa
6	107	Diana Lorentz
7	108	Nancy Greenberg
8	109	Daniel Faviet
9	110	John Chen
10	111	Ismael Sciarra
11	112	Jose Manuel Urman
12	113	Luis Popp
13	115	Alexander Khoo
14	116	Shelli Baida
15	117	Sigal Tobias
16	118	Guy Himuro
17	119	Karen Colmenares
18	120	Matthew Weiss
19	121	Adam Fripp
20	123	Shanta Vollman
21	124	Kevin Mourgos
22	125	Julia Nayer
23	126	Irene Mikkilineni

6. 안티 조인 (Anti Join)

```
SELECT a.employee_id,
       a.first_name || ' ' || a.last_name
  FROM employees a
 WHERE NOT EXISTS ( SELECT 0
                      FROM job_history  b
                     WHERE a.employee_id = b.employee_id
                   )
 ORDER BY 1;
```

	EMPLOYEE_ID	A.FIRST_NAME||' '||A.LAST_NAME
1	100	Steven King
2	103	Alexander Hunold
3	104	Bruce Ernst
4	105	David Austin
5	106	Valli Pataballa
6	107	Diana Lorentz
7	108	Nancy Greenberg
8	109	Daniel Faviet
9	110	John Chen
10	111	Ismael Sciarra
11	112	Jose Manuel Urman
12	113	Luis Popp
13	115	Alexander Khoo
14	116	Shelli Baida
15	117	Sigal Tobias
16	118	Guy Himuro
17	119	Karen Colmenares
18	120	Matthew Weiss
19	121	Adam Fripp
20	123	Shanta Vollman
21	124	Kevin Mourgos

학습정리

· 서브쿼리는 메인쿼리에 포함된 독립적인 SELECT 문장으로 괄호로 둘러싸인 쿼리를 말한다.

· 스칼라 서브쿼리는 메인 쿼리의 SELECT 절에 위치한 서브쿼리이다.

· 인라인 뷰는 메인 쿼리의 FROM 절에 위치한 서브쿼리이다.

· 중첩 서브쿼리는 메인 쿼리의 WHERE 절에 위치해 조건절의 일부로 사용된다.

· 세미조인은 중첩 서브쿼리와의 조인을 말한다.

· 안티조인은 세미조인에 NOT 연산자를 사용한 조인이다.

Quiz

1. 다음 문장은 어떤 정보를 조회하는지 설명해 보세요.

```
SELECT a.employee_id
     ,a.first_name || ' ' || a.last_name emp_name
     ,a.job_id
     ,a.salary
     ,( SELECT AVG(b.salary)
         FROM employees b
        WHERE a.job_id = b.job_id
        GROUP BY b.job_id
      ) avg_salary
FROM employees a;
```


Quiz

2. 다음 쿼리를 LATERAL 키워드를 사용해 같은 결과를 조회하도록 변경해 보세요.

```
SELECT b.department_name, loc.street_address, loc.country_name
  FROM departments b
     ,( SELECT l.location_id, l.street_address, c.country_name
          FROM locations l,   countries c
         WHERE l.country_id = c.country_id ) loc
 WHERE b.location_id = loc.location_id;
```


Quiz

3. 다음 문장을 IN 대신 EXISTS 연산자를 사용해 같은 결과를 조회하도록 변경해 보세요.

```
SELECT employee_id,
       job_id, salary
  FROM employees
 WHERE (job_id, salary ) IN ( SELECT job_id, min_salary
                                FROM jobs);
```


Quiz

4. 다음은 ANTI 조인 문장입니다. 이 문장은 employees 테이블에 할당되지 않은 부서정보를 조회하려는 문장인데, 실행하면 데이터가 조회되지 않습니다. 해당 부서정보를 조회하도록 이 쿼리를 수정해 보세요.

```
SELECT *
  FROM departments
 WHERE department_id NOT IN
          ( SELECT a.department_id
              FROM employees A
           ) ;
```

