**CH1. SELECT 문을 사용하는 데이터 검색**

당신은 베스트 기업의 SQL 개발자로 채용되었습니다. 첫 작업은 HR 업무관련 테이블들에 대한 데이터를 기반으로 몇 가지 보고서를 만드는 것입니다. 개발 툴은 SQL\*PLUS를 사용합니다.

1. 먼저 DEPARTMENTS 테이블의 구조와 해당 테이블의 모든 데이터를 조회합니다.

a. DEPARTMENTS 테이블 구조 보기:

DESCRIBE departments

b. DEPARTMENTS 테이블의 모든 자료 보기:

SELECT \*

FROM departments;

2. EMPLOYEES 테이블 구조 보기.

DESCRIBE employees

인사부서에서는 직원들의 employee ID, last name, job ID, hire date 들을 조회할 수 있는 SQL문을 원합니다. 이때 HIRE\_DATE 컬럼을 STARTDATE라는 별칭을 사용해라. 해당 SQL문을 lab\_01\_02.sql로 저장할 것.

select employee\_id, last\_name, job\_id, hire\_date "STARTDATE" from employees;

save lab\_01\_02.sql

3. 에러 없이 잘 조회되는지 lab\_01\_02.sql 파일을 수행합니다.

get lab\_01\_02.sql (코드 불러오기) 후 /(실행)

start lab\_01\_02.sql (결과만)

@lab\_01\_02.sql (결과만)

안이쁘기 때문에 글자 수를 조절해서 이쁘게 조절

col last\_name

@lab\_01\_02.sql (결과만) 다시보면 한 줄로 나옴

4. EMPLOYEES 테이블을 조회해서 유일한 job ID만을 모두 볼 수 있는 SQL문을 작성합니다.

select distinct job\_id from employees; (distinct 활용해서 간략하게 출력)

5. lab\_01\_02.sql 파일을 이용합니다. 컬럼명을 다음과 같이 별칭을 사용하여 수정합니다.

Emp #, Employee, Job, Hire Date.

ed lab\_01\_02.sql

select employee\_id "Emp\_#", last\_name "Employee", job\_id "Job", hire\_date "Hire Date" from employees

/

6. 모든 직원들의 last name 과 job ID를 조회합니다. 이때 두 컬럼을 콤마와 스페이스로 구분하여 연결하고, Employee and Title이라는 별칭을 사용합니다.

select last\_name ||', ' || job\_id as "Employees and Title" from employees;

7. 직원 테이블의 모든 데이터를 조회하는 SQL문을 작성합니다. 이때 모든 컬럼들을 콤마로 구분해서 보이고, THE\_OUTPUT이라는 별칭을 사용합니다.

**CH2. 데이터 제한 및 정렬**

1. 회사내 예산문제 때문에, 인사부서에서는 급여 $12,000 보다 많이 받는 직원들을 알려고 합니다. Last name, salary만 우선 검색하는 SQL문을 작성하고, lab\_02\_01.sql 파일로 저장합니다.

select last\_name, salary from employees where salary > 12000;

save lab\_02\_01.sql

2. 사번이 176인 직원의 last name, department number를 조회합니다.

select employee\_id, last\_name, salary from employees where employee\_id = 176;

3. lab\_02\_01.sql 파일을 사용해서 급여가 $5,000와 $12,000 사이에 해당되지 않는 직원들을 조회합니다. Last name, salary만 표시. 해당 SQL문을 lab\_02\_03.sql로 저장합니다.

select last\_name, salary

from employees

where salary not between 5000 and 12000

4. last name이 Matos 와 Taylor인 직원의 last name, job ID, hiredate를 구하고 hiredate로 정렬합니다.

select last\_name, job\_id, hire\_date "hiredate" from employees where last\_name = 'Matos' or last\_name = 'Taylor' order by 3;

select last\_name, job\_id, hire\_date "hiredate" from employees where last\_name in (’Matos’,’Taylor’) order by hiredate

5. 20번, 50번 부서에 속하는 직원들을 조회합니다. Last name, department number만 표시하고, last name으로 오름차순으로 정렬할 것.

select last\_name, department\_id

from employlees

where department\_id in (20, 50)

order by 1

6. lab\_02\_03.sql 파일을 사용하여 급여 $5,000과 $12,000 사이 받으며 20번 또는 50번 부서에 속하는 직원들을 검색하는 SQL문을 작성합니다. last name, salary만 표시함. 이때 각 컬럼을 Employee와 Monthly Salary로 표시합니다. 해당 SQL문을 lab\_02\_06.sql 이름으로 저장하고, 수행합니다.

select last\_name "Employee", salary "Monthly Salary"

from employees

where salary between 5000 and 12000

and department\_id in (20, 50);

7. 직원들중 2001년도에 입사한 모든 직원의 last name, hiredate를 조회합니다.

(sysdate 값을 통해 교육환경 DB의 날짜 포맷을 확인한 후 수행합니다.)

select sysdate from dual; (먼저 날짜 형식 확인)

select last\_name, hire\_date

from employees

where hire\_date like '01%';

8. 담당 관리자가 없는 모든 직원의 last name, job ID를 조회합니다. (관리자가 없는건 사장)

select last\_name, job\_id

from employees

where manager\_id is null;

9. 커미션을 받는 모든 직원들의 last name, salary, commission 을 조회하는 SQL문을 작성합니다. 이때 급여, 커미션으로 정렬(내림차순)합니다. ORDER BY 절에서 열의 숫자 위치를 사용합니다.

desc employees (구조에서 COMMISSION\_PCT는 not null 을 허용함을 보고 null 이 없어야 커미션 받는건지 확인)

select last\_name, salary, commission\_pct

from employees

where commission\_pct is not null

order by 2 desc, 3 desc;

10. HR 부서의 멤버는 여러분이 작성 중인 query에 유연성이 확대되기를 원합니다.

그들은 유저가 프롬프트에 지정하는 액수보다 많은 급여를 받은 사원이 있을 경우 이들의 성과 급여를 표시하는 보고서를 기대합니다. 이 query를 lab\_02\_10.sql이라는 파일에 저장합니다. 프롬프트가 나오면 값으로 12,000을 입력합니다.

select last\_name, salary

from employees

where salary > &sal\_amt;

11. HR 부서에서 관리자를 기준으로 보고서를 실행하려고 합니다. 유저에게 관리자 ID 입력 프롬프트를 표시하고 해당 관리자에 속한 사원의 사원 ID, 성, 급여 및 부서 id를 생성하는 query를 작성합니다. HR 부서에서 선택한 열을 기준으로 보고서를 정렬하는 기능을 원합니다. 다음 값으로 데이터를 테스트합니다.

manager \_id = 103, last\_name을 기준으로 정렬

manager\_id = 201, salary를 기준으로 정렬

manager\_id = 124, employee\_id를 기준으로 정렬

select employee\_id, last\_name, salary, department\_id

rom employees

where manager\_id = &mgr\_num

order by &order\_col;

12. 성의 세번째 문자가 “a”인 모든 사원의 last name을 표시합니다.

select last\_name

from employees

where last\_name like '\_\_a%';

13. 성에 “a”와 “e”가 모두 포함된 모든 사원의 last name을 표시합니다.

select last\_name

from employees

where last\_name like '%a%'

and last\_name like '%e%';

14. 직무가 판매 사원(sales rep)이나 자재 담당자(st\_clerk)이고 급여가 $2,500, $3,500 또는 $7,000가 아닌 모든 사원의 last name, 직무 및 급여를 표시합니다.

select last\_name, job\_id, salary

from employees

where job\_id in ('SA\_REP', 'ST\_CLERK')

and salary not in (2500, 3500, 7000)

15. 커미션 금액이 20%인 모든 사원의 last name, 급여 및 커미션을 표시하도록 lab\_02\_06.sql을 수정합니다. lab\_02\_06.sql을 lab\_02\_15.sql로 다시 저장합니다. lab\_02\_15.sql의 명령문을 다시 실행합니다.

select last\_name, salary, commission\_pct

from employees

where commission\_pct = 0.2

1. HR 부서에서 1997년 이후 채용된 모든 clerk에 대한 데이터를 찾으려고 합니다.

select \*

from employees

where job\_id = 'ST\_CLERK'

and hire\_date > '97/12/31';

1. HR 부서에서 커미션을 받는 사원에 대한 보고서를 요구합니다. 해당 사원의 성, 직무, 급여 및 커미션을 표시합니다. 급여의 내림차순으로 데이터를 정렬합니다.

select last\_name, department\_id, salary, commission\_pct from employees

where commission\_pct is not null

order by salary desc;

**CH3. Single-Row 함수를 사용하여 결과물 변경**

1. 시스템 날짜를 표시하기 위한 query를 작성합니다. 열 레이블을 Date로 지정합니다.

SELECT sysdate "Date"

FROM dual;

2. HR 부서에서 각 사원에 대해 사원 번호, 성, 급여 및 15.5% 인상된 급여(정수로 표현)를 표시하는 보고서가 필요합니다. 열 레이블을 New Salary로 지정하고, 작성한 SQL 문을 lab\_03\_02.sql이라는 파일에 저장합니다.

select employee\_id, last\_name, salary, round(salary \* 1.155, 0) "New Salary"

from employees;

save lab\_03\_02.sql

3. lab\_03\_02.sql 파일의 query를 실행합니다.

@lab\_03\_02.sql

4. 새 급여에서 이전 급여를 뺀 열을 추가하도록 lab\_03\_02.sql의 query를 수정합니다. 열 레이블을 Increase로 지정합니다. 파일 내용을 lab\_03\_04.sql로 저장합니다. 수정한 query를 실행합니다.

ed lab\_03\_02

select employee\_id, last\_name, salary, round(salary \* 1.155, 0) "New Salary", round(salary \* 1.155 - salary, 0) "increase"

from employeesget

다른이름으로 저장 ed lab\_03\_04

5. "J", "A" 또는 "M"으로 시작하는 이름을 가진 모든 사원의 last name(첫번째 문자는 대문자, 나머지는 모두 소문자)과 성의 길이를 표시하는 query를 작성합니다. 각 열에 적절한 레이블을 지정하고, 사원의 last name을 기준으로 결과를 정렬합니다.

select initcap(last\_name) "Name", length(last\_name) "Length"

from employees

where last\_name like 'J%'

or last\_name like 'A%'

or last\_name like 'M%'

order by last\_name;

유저에게 last name의 첫 문자를 입력하는 프롬프트를 표시하도록 query를 재작성합니다.

예를 들어, 문자 입력 프롬프트가 표시되었을 때 유저가 H(대문자)를 입력하면 출력에 last name이 "H"로 시작하는 모든 사원이 표시되어야 합니다.

select initcap(last\_name) "Name", length(last\_name) "Length"

from employees

where last\_name like '&start%'

order by last\_name

입력된 문자의 대소문자 여부에 따라 출력이 달라지지 않도록 쿼리를 수정합니다.

입력된 문자는 SELECT query에서 처리되기 전에 대문자로 변경해야 합니다.

SELECT INITCAP(last\_name) AS "Name",

LENGTH(last\_name) AS "Length"

FROM employees

WHERE UPPER(last\_name) LIKE UPPER('&start%')

ORDER BY last\_name;

6. HR 부서에서 각 사원의 근속 기간을 파악하려고 합니다. 각 사원에 대해 last name을 표시하고 채용일부터 오늘까지 경과한 개월 수를 계산합니다. 열 레이블을 MONTHS\_WORKED로 지정하고, 재직 개월 수에 따라 결과를 정렬합니다. 개월 수를 가장 가까운 정수로 반올림합니다.

SELECT last\_name, ROUND(MONTHS\_BETWEEN(SYSDATE, hire\_date)) AS "MONTHS\_WORKED"

FROM employees

ORDER BY "MONTHS\_WORKED";

7. 모든 사원의 last name과 급여를 표시하기 위한 query를 작성합니다. 급여가 15자 길이로 표시되고 왼쪽에 $ 기호가 채워지도록 형식을 지정합니다. 열 레이블을 SALARY로 지정합니다.

select last\_name, lpad(salary, 15, '$') "SALARY"

from employees;

8. 사원의 성에서 처음 8자를 표시하고 급여 액수를 별표로 나타내는 query를 작성합니다. 각 별표는 $1,000을 나타냅니다. 급여의 내림차순으로 데이터를 정렬하고, 열 레이블을 EMPLOYEES\_AND\_THEIR\_SALARIES로 지정합니다.

select rpad(last\_name,8) ||' '||rpad(' ', salary/1000+1, '\*') employees\_and\_their\_salaries

from employees

order by salary desc;

9. 부서 90의 모든 사원에 대해 last name 및 재직 기간(주 단위)을 표시하도록 query를 작성합니다. 주를 나타내는 숫자 열의 레이블을 TENURE로 지정하고, 주를 나타내는 숫자 값을 소수점에서 truncate합니다. 직원 재직 기간의 내림차순으로 레코드를 표시합니다.

select last\_name, trunc((sysdate - hire\_date)/7) “tenure”

from employees

where department\_id = 90

order by tenure desc;

1. HR 부서에서 예산 책정을 위해 예상되는 급여 인상에 대한 보고서를 요구합니다. 이 보고서는 커미션을 받지 않지만 급여가 10% 인상되는 사원을 표시해야 합니다(급여 반올림).

SELECT 'The salary of '||last\_name||' after a 10% raise is '

|| ROUND(salary\*1.10) "New salary"

FROM employees

WHERE commission\_pct IS NULL

1. 사원 및 근속 기간에 대한 보고서를 작성합니다. 모든 사원들의 성 및 근무기간(년, 개월)을 함께 표시합니다. 근속 기간별로 보고서를 정렬합니다. 근속 기간이 가장 긴 사원이 리스트의 맨 위에 나타나야 합니다.

SELECT last\_name, TRUNC(MONTHS\_BETWEEN(SYSDATE, hire\_date) / 12) AS "years", MOD(MONTHS\_BETWEEN(SYSDATE, hire\_date), 12) AS "months"

FROM employees

ORDER BY "years" DESC, "months" DESC, last\_name;

1. 성이 "J", "K", "L" 또는 "M"으로 시작하는 사원을 표시합니다.

select last\_name

from employees

where substr(last\_name, 1, 1) in ('J','K','L','M');

**CH4. 변환 함수 및 조건부 표현식 사용**

1. 각 사원에 대해 다음과 같이 출력하는 보고서를 작성합니다.

<*employee last name*> earns <*salary*> monthly but wants <*3 times salary*> 열 레이블을

Dream Salaries로 지정합니다.

SELECT last\_name || ' earns ' || TO\_CHAR(salary, 'fm$99,999.00') || ' monthly but wants '

|| TO\_CHAR(salary \* 3, 'fm$99,999.00') || '.' "Dream Salaries"

FROM employees;

2. 각 사원의 성, 채용 날짜 및 근무 6개월 후 첫 번째 월요일에 해당하는 급여 심의 날짜를 표시합니다. 열 레이블을 REVIEW로 지정하고, 날짜 형식을 "Monday, the Thirty-First of July, 2000"과 유사한 형식으로 지정합니다.

=> 실습 DB 언어가 한글로 설정되어 있음으로 ‘MONDAY’ 대신 ‘월요일’을 사용할 것.

select last\_name, hire\_date, to\_char(next\_day(add\_months(hire\_date, 6),'월요일'), 'Day, "the" Ddspth "of" Month, YYYY') "REVIEW"

from employees

3. 사원의 성, 채용 날짜, 근무 시작 요일을 표시합니다. 열 레이블을 DAY로 지정하고, 요일순으로 결과를 정렬합니다.

SELECT last\_name, hire\_date, TO\_CHAR(hire\_date, 'DAY') AS "DAY"

FROM employees

ORDER BY "DAY";

select last\_name, hire\_date, to\_char(hire\_date, 'day') "DAY"

from employees

order by 3

4. 사원의 성과 커미션 금액을 표시하는 query를 작성합니다. 사원이 커미션을 받지 않으면 "No Commission"을 표시합니다. 열 레이블을 COMM으로 지정합니다.

select last\_name, nvl(to\_char(commission\_pct), 'No Commission') "COMM"

from employees

5. 다음 데이터를 사용하여 DECODE 함수를 통해 JOB\_ID 열의 값을 기반으로 모든 사원의 등급을 표시하는 query를 작성합니다.

*직책 등급*

AD\_PRES A

ST\_MAN B

IT\_PROG C

SA\_REP D

ST\_CLERK E

None of the above 0

select last\_name, job\_id,

decode(job\_id, 'AD\_PRES', 'A',

'ST\_MAN', 'B',

'IT\_PROG', 'C',

'SA\_REP', 'D',

'ST\_CLERK', 'E',

'None of the above')

from employees

------------------------------------

select job\_id, decode(job\_id, 'AD\_PRES', 'A',

'ST\_MAN', 'B',

'IT\_PROG', 'C',

'SA\_REP', 'D',

'ST\_CLERK', 'E',

'None of the above') "Grade"

from employees

6. CASE 구문을 사용하여 앞의 연습에 나오는 명령문을 재작성합니다.

1. 모든 사원을 표시하고 각 사원이 커미션을 받는지 여부를 *Yes* 또는 *No*로 나타내는 보고서를 작성합니다. query에서 DECODE 식을 사용합니다.

select last\_name, salary, decode(commission\_pct, null, 'No', 'Yes') "commission"

from employees

------------------------------------------------------------

decode 사용안한 버전

select last\_name, salary, nvl2(to\_char(commission\_pct), 'Yes', 'No') "commission"

from employees

1. 각 월의 16일 이전에 채용된 사원을 모두 표시합니다.

select last\_name, hire\_date

from employees

where to\_char(hire\_date, 'dd') < 16

**CH5. Group 함수를 사용한 집계 데이터 보고**

1. 모든 사원의 최고, 최저, 합계 및 평균 급여를 찾습니다. 열 레이블을 각각 Maximum, Minimum, Sum, Average로 지정하고, 결과를 가장 가까운 정수로 반올림합니다. SQL 문을 lab\_05\_01.sql로 저장합니다. query를 실행합니다.

select round(max(salary),0) "Maximum", round(min(salary),0) "Minimum", round(sum(salary),0) "Sum", round(avg(salary),0) "Average"

from employees;

save lab\_05\_01.sql

ed lab\_05\_01

2. 각 직무 유형에 대해 최소, 최대, 합계 및 평균 급여를 표시하도록 lab\_05\_01.sql의 query를 수정합니다. lab\_05\_01.sql을 lab\_05\_02.sql로 다시 저장합니다. lab\_05\_02.sql의 명령문을 실행합니다.

ed lab\_05\_01

select job\_id, round(max(salary),0) "Maximum", round(min(salary),0) "Minimum", round(sum(salary),0) "Sum", round(avg(salary),0) "Average"

from employees

group by job\_id

다른이름 저장

get lab\_05\_02

/

결과값 확인을 위해 19 맞는지 확인 (직무별로 다 잘썼다)

select count(\*)

from jobs;

select job\_id, count(\*)

from employees

group by job\_id

3. 동일한 직무를 수행하는 사람 수를 표시하기 위한 query를 작성합니다.

HR 부서의 유저에게 직무를 입력하는 프롬프트를 표시하도록 query를 일반화합니다. 이 스크립트를 lab\_05\_03.sql이라는 파일에 저장합니다. query를 실행합니다. 프롬프트가 나타나면 IT\_PROG를 입력하고 결과를 조회한다.

select job\_id, count(\*)

from employees

where job\_id = '&jobid'

group by job\_id;

it 직무 확인한다

Enter value for jobid: IT\_PROG

결과 잘 나옴 5명

SA\_REP

30 명 잘나옴

4. 관리자를 나열하지 않는 채로 관리자 수를 확인합니다. 열 레이블을 Number of Managers로 지정합니다.

desc employees; 쳐서 MANAGER\_ID가 나오는데 여기에 관리자의 사번이 있기에 여기 번호가 있으면 관리자다

select count(manager\_id)

from employees;

106명 이 나온다. 107 명중사장 1명 뺴고

select count(distinct manager\_id) "Number of Managers"

from employees;

18명 나온다. (107 명 중에 사장을 제외하고 관리자 18명)

5. 최고 급여와 최저 급여의 차이를 알아냅니다. 열 레이블을 DIFFERENCE로 지정합니다.

select max(salary) - min(salary) "Difference"

from employees;

6. 관리자 번호 및 해당 관리자의 부하 사원 중 최저 급여를 받는 사원의 급여를 표시하는 보고서를 작성합니다. 관리자가 알려져 있지 않은 모든 사원을 제외합니다. 최소 급여가 $6,000 이하인 그룹을 제외시킵니다. 급여의 내림차순으로 출력을 정렬합니다.

select manager\_id, min(salary)

from employees

where manager\_id is not null

group by manager\_id

having min(salary) > 6000

order by min(salary) desc

7. 사원의 총 수와 2004년, 2005년, 2006년 및 2007년에 채용된 사원의 수를 표시하는 query를 작성합니다. 적절한 열 머리글을 지정합니다. (decode 함수 사용가능)

select count(\*), sum(decode(to\_char(hire\_date,'yyyy'), 2004, 1, 0)) "2004",

sum(decode(to\_char(hire\_date,'yyyy'), 2005, 1, 0)) "2005",

sum(decode(to\_char(hire\_date,'yyyy'), 2006, 1, 0)) "2006",

sum(decode(to\_char(hire\_date,'yyyy'), 2007, 1, 0)) "2007"

from employees;

8. 부서 20, 50, 80 및 90에 대해 직무, 부서 ID별 해당 직무에 대한 급여 및 해당 직무에 대한 총 급여를 표시하고 각 열에 적절한 머리글을 지정하기 위한 행렬 query를 작성합니다.

(decode 함수 사용 가능)

select job\_id, sum(decode(department\_id, 20, salary)) "Dept 20",

sum(decode(department\_id, 50, salary)) "Dept 50",

sum(decode(department\_id, 80, salary)) "Dept 80",

sum(decode(department\_id, 90, salary)) "Dept 90"

from employees

group by job\_id

1. 성이 "n"으로 끝나는 사원의 수를 알아냅니다.

select count(\*)

from employees

where last\_name like '%n';

**CH6. 조인을 사용하여 여러 테이블의 데이터 표시**

1. HR 부서를 위해 모든 부서의 주소를 생성하는 query를 작성합니다. LOCATIONS 및 COUNTRIES 테이블을 사용합니다. 출력에 location ID, street address, city, state or province 및 국가를 표시합니다. NATURAL JOIN을 사용하여 결과를 생성합니다.

SELECT l.location\_id, l.street\_address, l.city, l.state\_province, c.country\_name

FROM locations l

NATURAL JOIN countries c;

2. HR 부서에서 모든 사원에 대한 보고서를 요구합니다. 모든 사원의 last name, 부서 ID 및 부서 이름을 표시하는 query를 작성합니다. USING을 사용합니다.

SELECT e.last\_name, e.department\_id, d.department\_name

FROM employees e

JOIN departments d

USING (department\_id);

3. HR 부서에서 Toronto에 근무하는 사원에 대한 보고서를 요구합니다. Toronto에서 근무하는 모든 사원의 last name, 직무, 부서 ID 및 부서 이름을 표시합니다.

ON을 사용합니다.

select e.last\_name, e.job\_id, e.department\_id, d.department\_name, l.city

from employees e join departments d

on (e.department\_id = d.department\_id)

join locations l

on (d.location\_id = l.location\_id)

where lower(l.city) = 'toronto'

col city format a10

4. 사원의 last name 및 사원 번호를 해당 관리자의 last name 및 관리자 번호와 함께 표시하는 보고서를 작성합니다. 열 레이블을 각각 Employee, Emp#, Manager, Mgr#으로 지정하고, SQL 문을 lab\_06\_04.sql로 저장합니다. query를 실행합니다.

5. King을 비롯하여 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 lab\_06\_04.sql을 수정합니다. 사원 번호순으로 결과를 정렬합니다. SQL문을 lab\_06\_05.sql로 저장합니다. lab\_06\_05.sql의 query를 실행합니다.

6. HR 부서를 위해 사원의 last name과 부서 ID 및 주어진 사원과 동일한 부서에 근무하는 모든 사원을 표시하는 보고서를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 이 스크립트를 lab\_06\_06.sql이라는 파일에 저장합니다. query를 실행합니다.

7. HR 부서에서 직책 등급 및 급여에 대한 보고서를 요구합니다. JOB\_GRADES 테이블에 익숙해지도록 먼저 JOB\_GRADES 테이블의 구조를 표시합니다. 그런 다음 모든 사원의 이름, 직무, 부서 이름, 급여 및 등급을 표시하는 query를 작성합니다.

8. HR 부서에서 Davies 이후에 채용된 모든 사원의 이름을 파악하려고 합니다. 사원 Davies 이후로 채용된 모든 사원의 이름과 채용 날짜를 표시하기 위한 query를 작성합니다.

SELECT e.last\_name, e.hire\_date

FROM employees e JOIN employees davies

ON (davies.last\_name = 'Davies')

WHERE davies.hire\_date < e.hire\_date;

9. HR 부서에서 관리자보다 먼저 채용된 모든 사원의 이름과 채용 날짜 및 해당 관리자의 이름과 채용 날짜를 찾으려고 합니다. 이 스크립트를 lab\_06\_09.sql이라는 파일에 저장합니다.

1. 특정 위치에서 근무하는 사원의 부서 이름, 위치 ID, 이름, 직책 및 급여를 표시하는 보고서를 작성합니다. 유저에게 위치를 입력하는 프롬프트를 표시합니다.

* 프롬프트가 표시되면 location\_id로 1800을 입력합니다.

1. 각 부서에 대한 이름, 위치 및 사원 수를 보여주는 보고서를 작성합니다. 보고서에 사원이 없는 부서도 포함되는지 확인합니다.
2. Administration 및 Executive 부서에서 찾은 직무를 표시하는 보고서를 작성합니다. 또한 해당 직무에 대한 사원 수도 표시합니다. 사원 수가 가장 많은 직무를 가장 먼저 표시합니다.
3. 급여가 $15,000 이상인 관리자 휘하의 모든 사원을 표시합니다. 사원 이름, 관리자 이름, 관리자 급여 및 관리자의 급여 등급을 표시합니다.
4. 모든 부서의 부서 번호, 부서 이름, 사원 수, 평균 급여와 각 부서에서 일하는 사원의 이름, 급여 및 직무를 표시합니다.
5. 모든 사원에 대해 사원 번호, 성, 급여, 부서 번호 및 해당 부서의 평균 급여를 표시하는 보고서를 작성합니다.