

Hardware and Software
Engineered to Work Together



Oracle Database 12c: SQL Workshop I

Activity Guide
D80190KR11
Edition 1.1 | November 2014 | D88836

Learn more from Oracle University at oracle.com/education/

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Disclaimer

본 문서는 독점적 정보를 포함하고 있으며 저작권법 및 기타 지적 재산법에 의해 보호됩니다. 본 문서는 오라클 교육 과정에서 자신이 사용할 목적으로만 복사하고 인쇄할 수 있습니다. 어떤 방법으로도 본 문서를 수정하거나 변경할 수 없습니다. 저작권법에 따라 "공정"하게 사용하는 경우를 제외하고, 오라클의 명시적 허가 없이 본 문서의 전체 또는 일부를 사용, 공유, 다운로드, 업로드, 복사, 인쇄, 표시, 실행, 재생산, 게시, 라이선스, 우편 발송, 전송 또는 배포할 수 없습니다.

본 문서의 내용은 사전 공지 없이 변경될 수 있습니다. 만일 본 문서의 내용상 문제점을 발견하면 서면으로 통지해 주기 바랍니다. Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. 오라클은 본 문서에 오류가 존재하지 않음을 보증하지 않습니다.

Restricted Rights Notice

만일 본 문서를 미국 정부나 또는 미국 정부를 대신하여 문서를 사용하는 개인이나 법인에게 배송하는 경우, 다음 공지 사항이 적용됩니다.

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle 및 Java는 Oracle Corporation 또는 그 자회사의 등록 상표입니다. 기타의 명칭들은 각 해당 명칭을 소유한 회사의 상표일 수 있습니다.

저자

Dimpi Rani Sarmah

기술 제공자 및 검토자

Nancy Greenberg, Swarnapriya Shridhar, Bryan Roberts, Laszlo Czinkoczki, KimSeong Loh, Brent Dayley, Jim Spiller, Christopher Wensley, Manish Pawar, Clair Bennett, Yanti Chang, Joel Goodman, Gerlinde Frenzen, Madhavi Siddireddy

이 책을 게시하는 데 사용된 프로그램: **Oracle Tutor**

목차

단원 1의 연습: 소개	1-1
단원 1의 연습: 개요	1-2
연습 1-1: 소개	1-3
해답 1-1: 소개	1-4
단원 2의 연습: SQL SELECT 문을 사용하여 데이터 검색	2-1
단원 2의 연습: 개요	2-2
연습 2-1: SQL SELECT 문을 사용하여 데이터 검색	2-3
해답 2-1: SQL SELECT 문을 사용하여 데이터 검색	2-8
단원 3의 연습: 데이터 제한 및 정렬	3-1
단원 3의 연습: 개요	3-2
연습 3-1: 데이터 제한 및 정렬	3-3
해답 3-1: 데이터 제한 및 정렬	3-7
단원 4의 연습: 단일 행 함수를 사용하여 출력 커스터마이징	4-1
단원 4의 연습: 개요	4-2
연습 4-1: 단일 행 함수를 사용하여 출력 커스터마이징	4-3
해답 4-1: 단일 행 함수를 사용하여 출력 커스터마이징	4-9
단원 5의 연습: 변환 함수 및 조건부 표현식 사용	5-1
단원 5의 연습: 개요	5-2
연습 5-1: 변환 함수 및 조건부 표현식 사용	5-3
해답 5-1: 변환 함수 및 조건부 표현식 사용	5-9
단원 6의 연습: 그룹 함수를 사용하여 집계 데이터 보고	6-1
단원 6의 연습: 개요	6-2
연습 6-1: 그룹 함수를 사용하여 집계 데이터 보고	6-3
해답 6-1: 그룹 함수를 사용하여 집계 데이터 보고	6-6
단원 7의 연습: 조인을 사용하여 다중 테이블의 데이터 표시	7-1
단원 7의 연습: 개요	7-2
연습 7-1: 조인을 사용하여 다중 테이블의 데이터 표시	7-3
해답 7-1: 조인을 사용하여 다중 테이블의 데이터 표시	7-8
단원 8의 연습: Subquery를 사용하여 Query 해결	8-1
단원 8의 연습: 개요	8-2
연습 8-1: Subquery를 사용하여 query 해결	8-3
해답 8-1: Subquery를 사용하여 query 해결	8-6
단원 9의 연습: 집합 연산자 사용	9-1
단원 9의 연습: 개요	9-2
연습 9-1: 집합 연산자 사용	9-3
해답 9-1: 집합 연산자 사용	9-5
단원 10의 연습: 데이터 조작	10-1
단원 10의 연습: 개요	10-2
연습 10-1: DML 문을 사용하여 테이블 관리	10-3
해답 10-1: DML 문을 사용하여 테이블 관리	10-7

단원 11의 연습: DDL 문을 사용하여 테이블 생성 및 관리	11-1
단원 11의 연습: 개요	11-2
연습 11-1: 데이터 정의어 소개	11-3
해답 11-1: 데이터 정의어 소개	11-7
추가 연습 및 해답	12-1
단원 1의 연습	12-2
연습 1-1: 추가 연습	12-3
해답 1-1: 추가 연습	12-11
사례 연구: 온라인 서점	12-17
연습 1-2	12-18
해답 1-2	12-23

단원 1의 연습: 소개

1장

단원 1의 연습: 개요

연습 개요

본 연습에서는 **SQL Developer**를 시작하여 새 데이터베이스 연결을 생성하고 **HR** 테이블을 탐색합니다. 일부 **SQL Developer** 환경 설정도 설정합니다.

일부 연습에서 "시간 여유가 있을 경우" 또는 "심화 연습에 도전하려면"으로 시작하는 연습 과정이 제공될 수 있습니다. 이러한 연습은 할당된 시간 내에 다른 연습을 모두 완료한 다음 자신의 실력을 좀 더 테스트해 보고자 하는 경우에만 수행하십시오.

연습은 천천히 정확하게 수행하십시오. 명령 파일을 저장하거나 실행해 볼 수 있습니다. 의문 사항이 있으면 언제든지 강사에게 문의하십시오.

주

- 모든 연습 문제는 **Oracle SQL Developer**를 개발 환경으로 사용합니다. **Oracle SQL Developer**를 사용하는 것이 좋지만 본 과정에서 사용 가능한 **SQL*Plus**를 사용할 수도 있습니다.
- **Query**의 경우 데이터베이스에서 검색된 행의 시퀀스는 표시된 스크린샷과 다를 수 있습니다.

연습 1-1: 소개

개요

본 과정에 나오는 여러 연습 중 첫번째입니다. 필요한 경우 본 연습의 끝 부분에서 해답을 찾아볼 수 있습니다. 본 연습에서는 해당 단원에 나오는 대부분의 주제를 다룹니다.

이 연습에서는 다음을 수행합니다.

- Oracle SQL Developer를 시작하고 ora1 계정에 대해 새 연결을 생성합니다.
- Oracle SQL Developer를 사용하여 ora1 계정에서 데이터 객체를 검사합니다. ora1 계정에는 HR 스키마 테이블이 포함됩니다.

다음 연습 파일 위치를 기록해 두십시오.

/home/oracle/labs/sql1/labs

연습 파일을 저장하라는 요청을 받으면 이전 위치에 해당 파일을 저장하십시오.

작업

1. SQL Developer 바탕 화면 아이콘을 사용하여 Oracle SQL Developer를 시작합니다.
2. 새 Oracle SQL Developer 데이터베이스 연결 생성
 - a. 새 데이터베이스 연결을 생성하려면 **Connections Navigator**에서 **Connections**를 마우스 오른쪽 버튼으로 누르고 컨텍스트 메뉴에서 **New Connection**을 선택합니다. **New/Select Database Connection** 대화상자가 나타납니다.
 - b. 다음 정보를 사용하여 데이터베이스 연결을 생성합니다.

Connection Name: myconnection
 Username: ora1
 Password: ora1
 Hostname: localhost
 Port: 1521
 SID: ORCL

Save Password 체크 박스를 선택합니다.
3. Oracle SQL Developer 데이터베이스 연결 테스트 및 데이터베이스에 연결
 - a. 새 연결을 테스트합니다.
 - b. 상태가 **Success**이면 이 새 연결을 사용하여 데이터베이스에 연결합니다.
4. Connections Navigator에서 테이블 탐색
 - a. Connections Navigator의 **Tables** 노드에서 사용할 수 있는 객체를 확인합니다. 다음 테이블이 있는지 확인합니다.

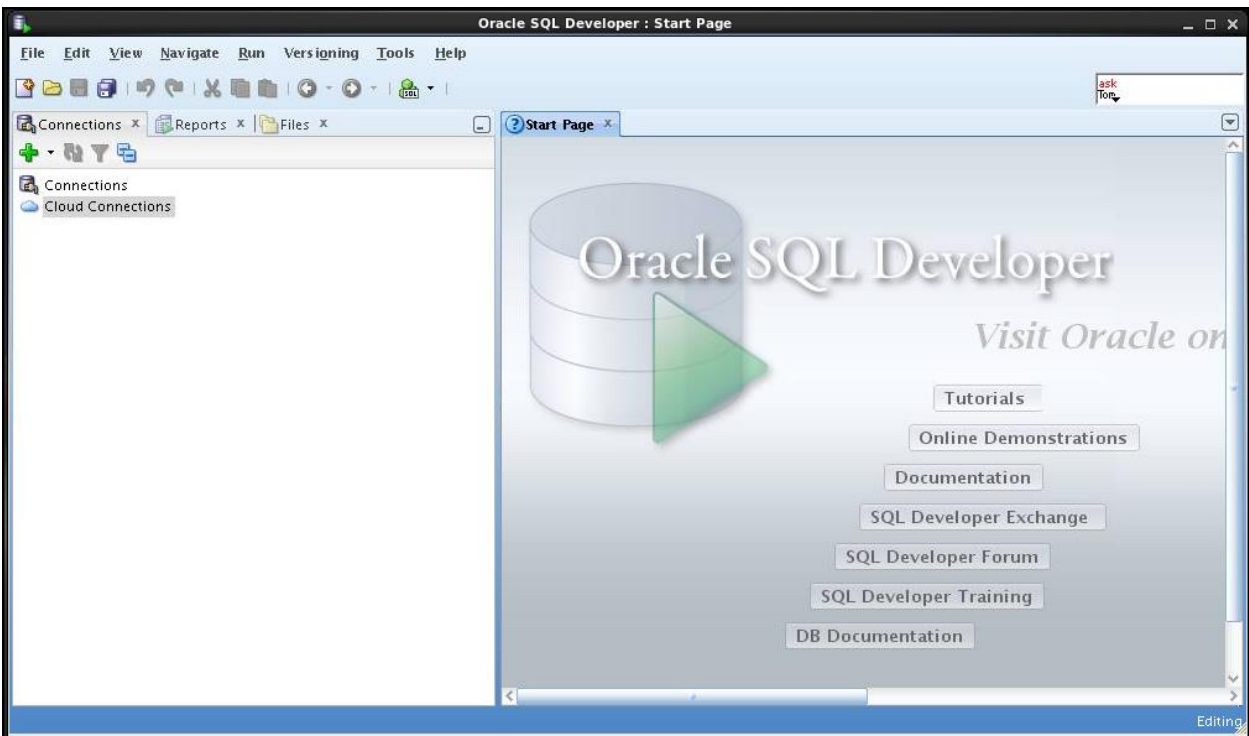
COUNTRIES
 DEPARTMENTS
 EMPLOYEES
 JOB_GRADES
 JOB_HISTORY
 JOBS
 LOCATIONS
 REGIONS
 - b. EMPLOYEES 테이블의 구조를 탐색합니다.
 - c. DEPARTMENTS 테이블의 데이터를 확인합니다.

해답 1-1: 소개

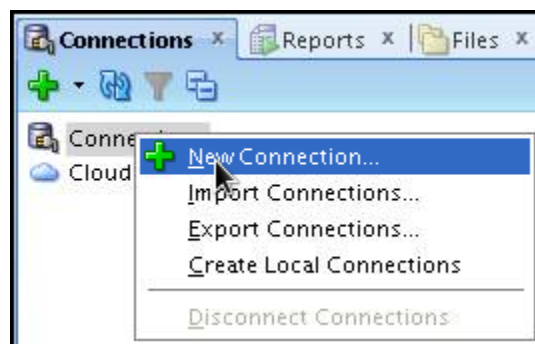
1. SQL Developer 바탕 화면 아이콘을 사용하여 Oracle SQL Developer 시작
Oracle SQL Developer 바탕 화면 아이콘을 두 번 누릅니다.



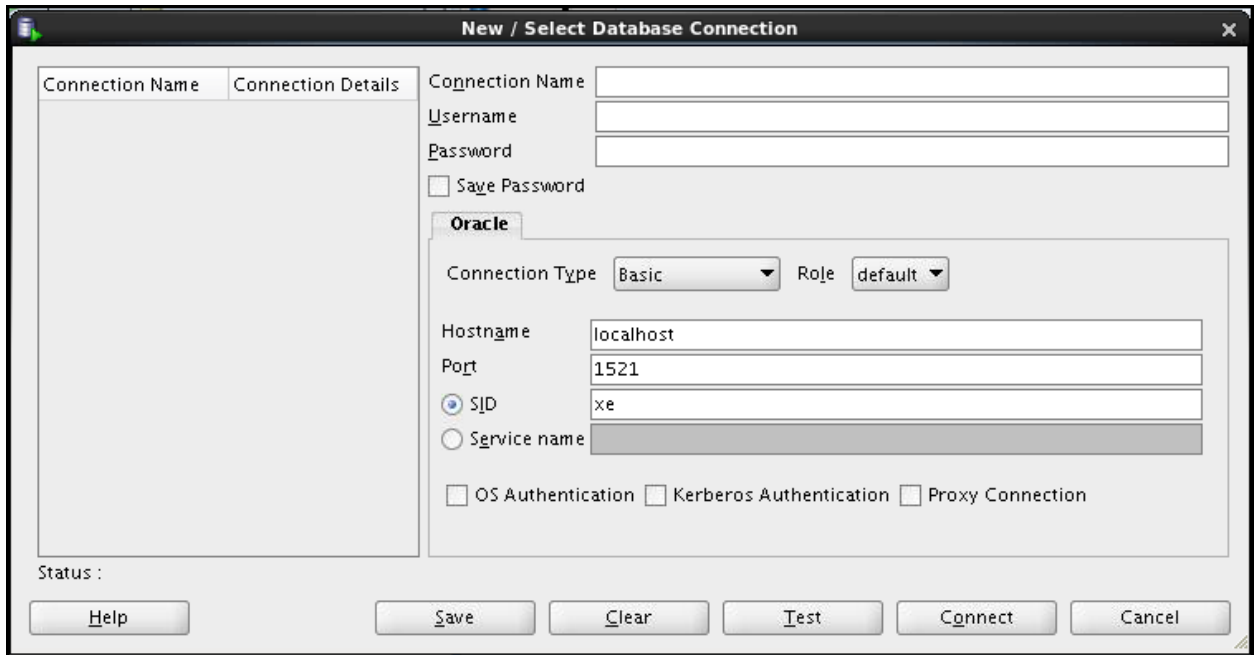
SQL Developer 인터페이스가 나타납니다.



2. 새 Oracle SQL Developer 데이터베이스 연결 생성
 - a. 새 데이터베이스 연결을 생성하려면 **Connections Navigator**에서 **Connections**를 마우스 오른쪽 버튼으로 누르고 컨텍스트 메뉴에서 **New Connection**을 선택합니다.



New/Select Database Connection 대화상자가 나타납니다.

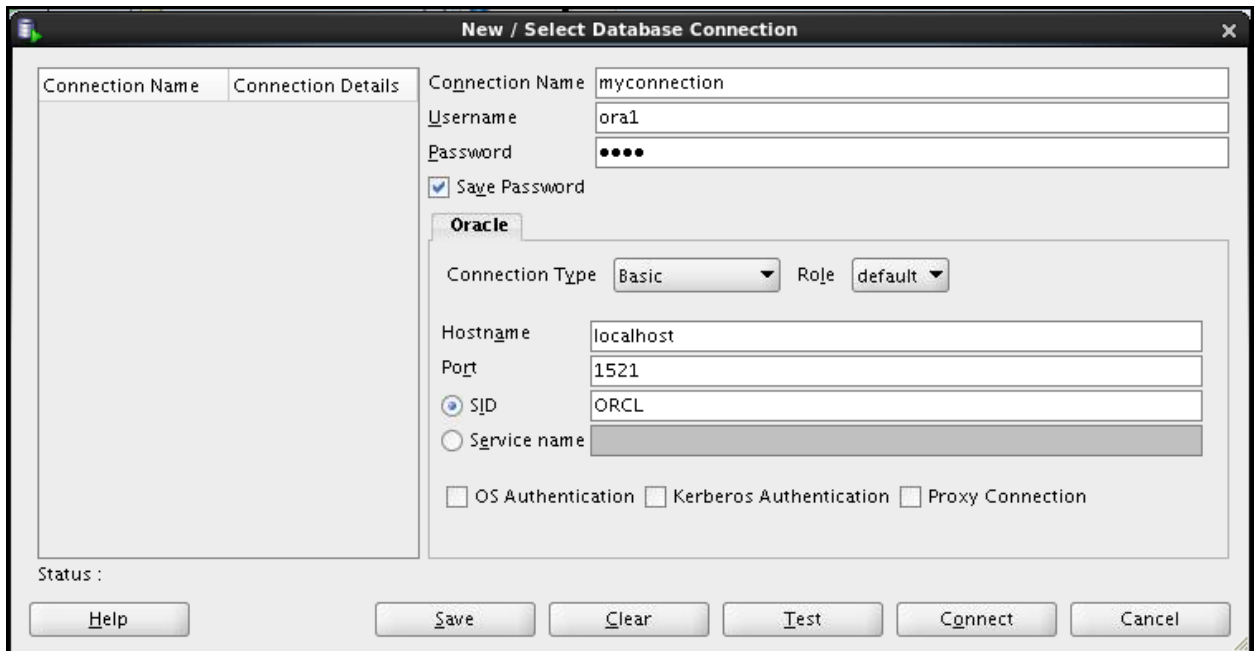


The dialog box titled "New / Select Database Connection" has a "Connection Name" tab and a "Connection Details" tab. The "Connection Details" tab is active, showing fields for "Connection Name", "Username", "Password", and a "Save Password" checkbox. Below these is the "Oracle" section with "Connection Type" set to "Basic" and "Role" set to "default". The "Hostname" is "localhost", "Port" is "1521", and "SID" is "xe". The "Service name" field is empty. At the bottom of the Oracle section are checkboxes for "OS Authentication", "Kerberos Authentication", and "Proxy Connection". The "Status:" label is at the bottom left, and buttons for "Help", "Save", "Clear", "Test", "Connect", and "Cancel" are at the bottom right.

b. 다음 정보를 사용하여 데이터베이스 연결을 생성합니다.

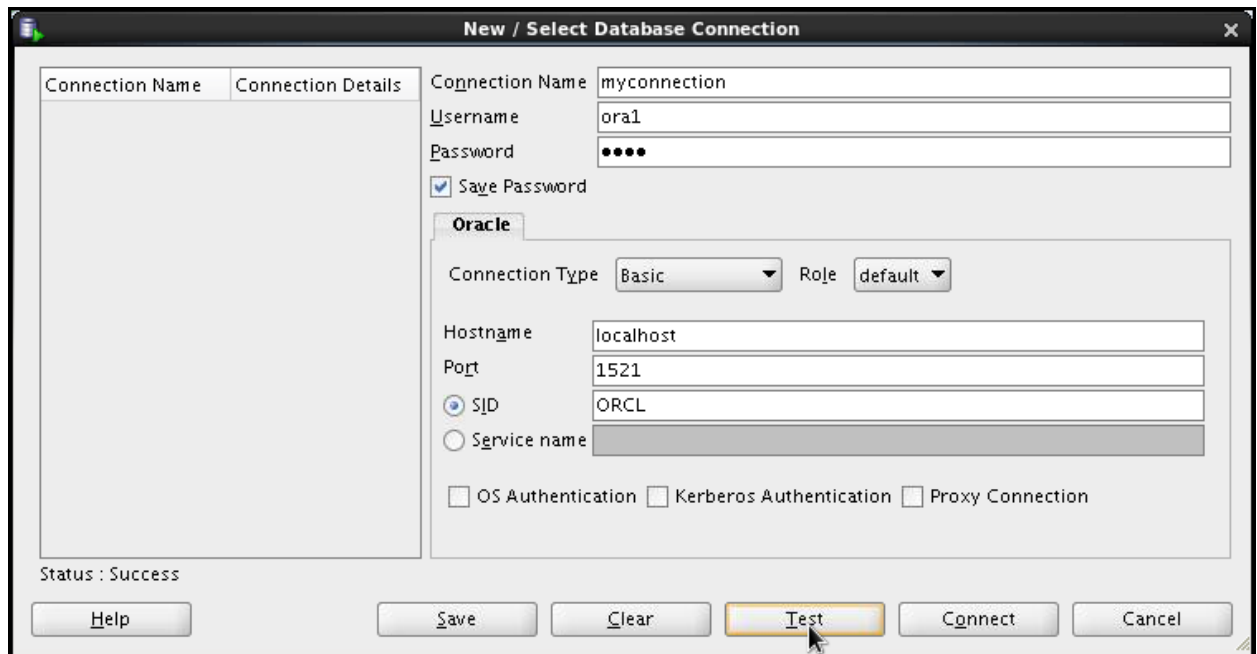
- i. Connection Name: myconnection
- ii. Username: ora1
- iii. Password: ora1
- iv. Hostname: localhost
- v. Port: 1521
- vi. SID: ORCL

Save Password 체크 박스를 선택합니다.

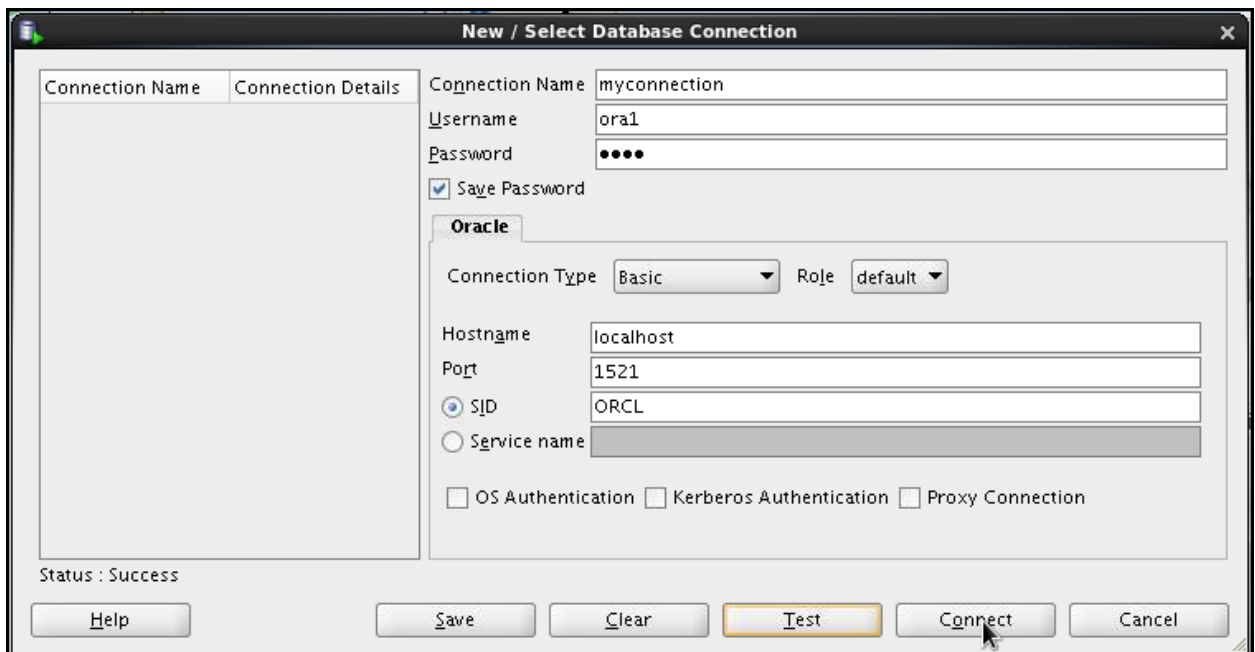


The dialog box is now filled with the information from the list. "Connection Name" is "myconnection", "Username" is "ora1", "Password" is "ora1" (masked with dots), and "Save Password" is checked. In the "Oracle" section, "Connection Type" is "Basic", "Role" is "default", "Hostname" is "localhost", "Port" is "1521", and "SID" is "ORCL". The "Service name" field is empty. The "Status:" label and buttons are at the bottom.

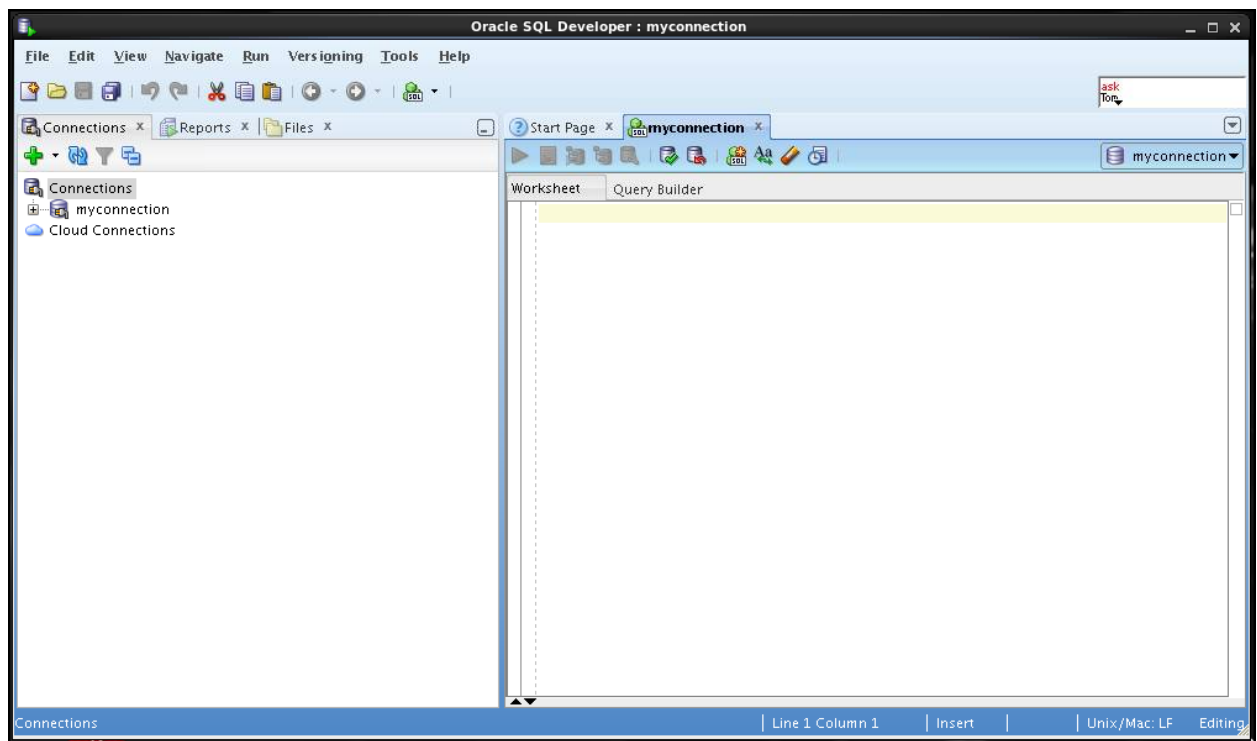
3. Oracle SQL Developer 데이터베이스 연결을 사용하여 테스트 및 연결
- a. 새 연결을 테스트합니다.



- b. 상태가 Success이면 이 새 연결을 사용하여 데이터베이스에 연결합니다.



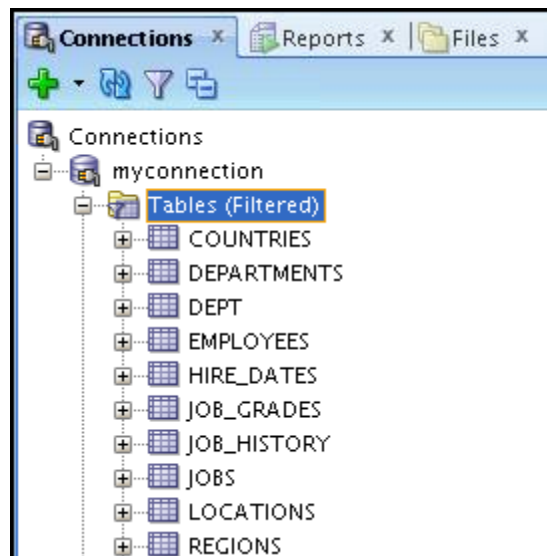
연결을 생성하면 해당 연결에 대한 SQL Worksheet가 자동으로 열립니다.



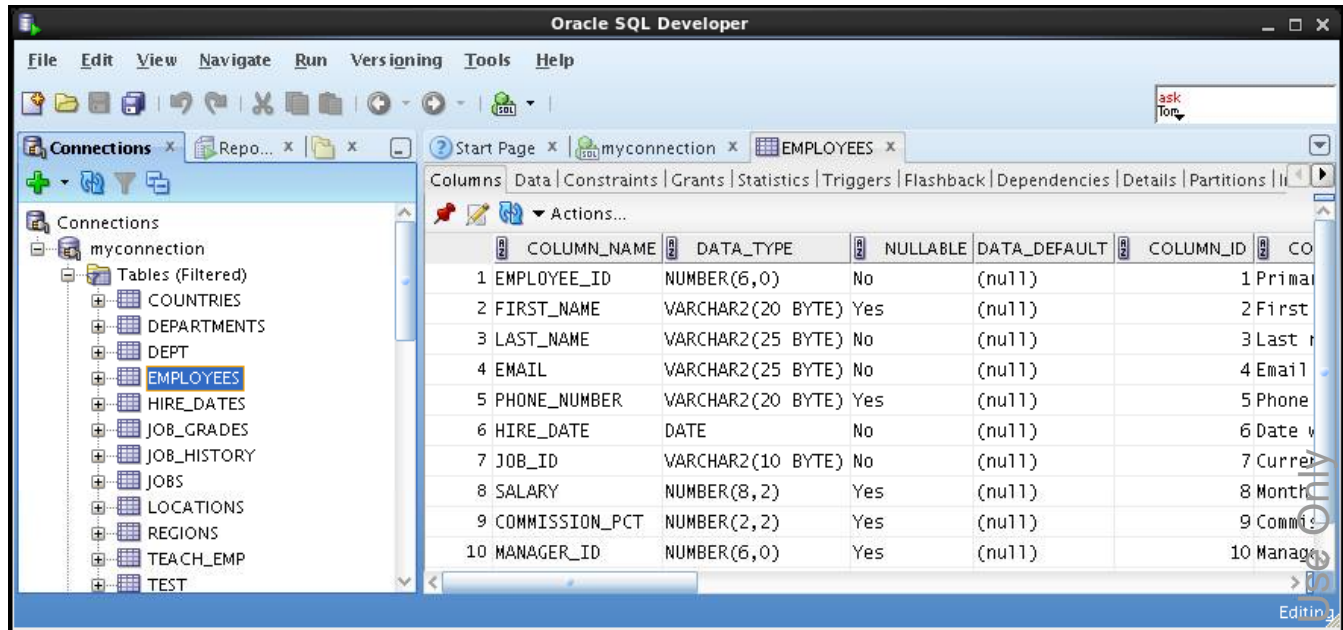
4. Connections Navigator에서 테이블 탐색

- a. Connections Navigator의 Tables 노드에서 사용할 수 있는 객체를 확인합니다.
다음 테이블이 있는지 확인합니다.

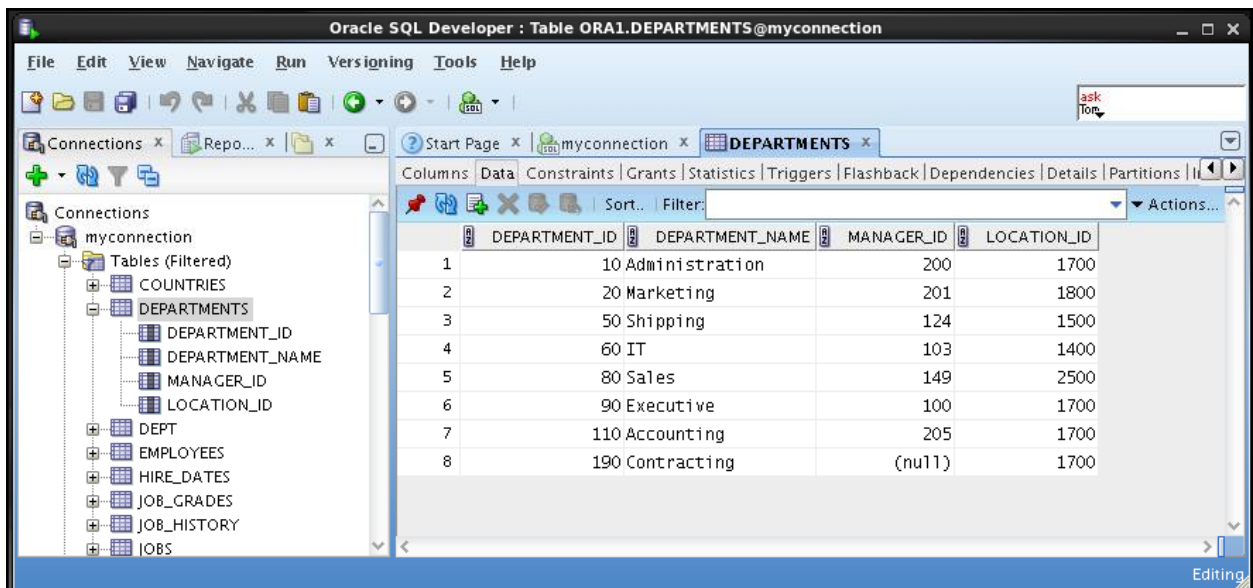
COUNTRIES
DEPARTMENTS
EMPLOYEES
JOB_GRADES
JOB_HISTORY
JOBS
LOCATIONS
REGIONS



b. EMPLOYEES 테이블의 구조를 탐색합니다.



c. DEPARTMENTS 테이블의 데이터를 확인합니다.



단원 2의 연습:
SQL SELECT 문을 사용하여
데이터 검색

2장

단원 2의 연습: 개요

연습 개요

이 연습에서는 다음 내용을 다룹니다.

- 다른 테이블에서 모든 데이터 선택
- 테이블의 구조 설명
- 산술식을 수행하고 열 이름 지정

연습 2-1: SQL SELECT 문을 사용하여 데이터 검색

개요

이 연습에서는 간단한 SELECT query를 작성합니다. 이러한 query에는 본 단원에서 학습한 대부분의 SELECT 절과 연산이 포함됩니다.

작업 1

지식을 테스트해 보십시오.

1. 다음 SELECT 문이 성공적으로 실행됩니다.

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

맞음/틀림

2. 다음 SELECT 문이 성공적으로 실행됩니다.

```
SELECT *
FROM   job_grades;
```

맞음/틀림

3. 다음 명령문에 네 개의 코딩 오류가 있습니다. 식별할 수 있습니까?

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

작업 2

연습을 시작하기 전에 다음과 같은 점에 주의합니다.

- 모든 연습 파일을 다음 위치에 저장하십시오.
/home/oracle/labs/sql1/labs
- SQL Worksheet에 SQL 문을 입력합니다. SQL Developer에서 스크립트를 저장하려면 필요한 SQL Worksheet가 활성화되어 있는지 확인한 다음 File 메뉴에서 Save As를 선택하여 SQL 문을 lab_<lessonno>_<stepno>.sql 스크립트로 저장합니다. 기존 스크립트를 수정하는 경우 Save As를 사용하여 스크립트를 다른 파일 이름으로 저장해야 합니다.
- query를 실행하려면 SQL Worksheet에서 Execute Statement 아이콘을 누릅니다. 또는 F9를 누를 수 있습니다. DML 및 DDL 문의 경우에는 Run Script 아이콘을 사용하거나 F5를 누릅니다.
- query를 실행한 후 동일한 워크시트에 다음 query를 입력하지 않도록 합니다. 새 워크시트를 엽니다.

여러분은 Acme Corporation의 SQL 프로그래머로 채용되었습니다. 첫 작업은 Human Resources 테이블의 데이터를 기반으로 몇 가지 보고서를 작성하는 것입니다.

4. 첫 작업은 DEPARTMENTS 테이블의 구조와 해당 내용을 파악하는 것입니다.

```
DESCRIBE departments
Name                Null      Type
-----
DEPARTMENT_ID       NOT NULL  NUMBER(4)
DEPARTMENT_NAME      NOT NULL  VARCHAR2(30)
MANAGER_ID           NUMBER(6)
LOCATION_ID            NUMBER(4)
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

5. EMPLOYEES 테이블의 구조와 해당 내용을 파악합니다.

- a. EMPLOYEES 테이블의 구조를 확인합니다.

```
DESCRIBE employees
Name                Null      Type
-----
EMPLOYEE_ID         NOT NULL  NUMBER(6)
FIRST_NAME           VARCHAR2(20)
LAST_NAME            NOT NULL  VARCHAR2(25)
EMAIL                NOT NULL  VARCHAR2(25)
PHONE_NUMBER         VARCHAR2(20)
HIRE_DATE            NOT NULL  DATE
JOB_ID               NOT NULL  VARCHAR2(10)
SALARY               NUMBER(8,2)
COMMISSION_PCT       NUMBER(2,2)
MANAGER_ID           NUMBER(6)
DEPARTMENT_ID        NUMBER(4)
```


- b. HR 부서에서는 **query**를 통해 사원 ID를 먼저 표시한 후 이어서 각 사원에 대한 성, 직무 ID, 채용 날짜 및 사원 ID를 표시하려고 합니다. HIRE_DATE 열에 대한 **alias**로 STARTDATE를 입력하십시오. SQL 문을 lab_02_5b.sql이라는 파일에 저장하여 이 파일을 HR 부서에 전달할 수 있도록 합니다. lab_02_5b.sql 파일의 **query**가 제대로 실행되는지 테스트합니다.

주: query를 실행한 후 동일한 워크시트에 다음 query를 입력하지 않도록 합니다. 새 워크시트를 엽니다.

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	100	King	AD_PRES	17-JUN-03
2	101	Kochhar	AD_VP	21-SEP-05
3	102	De Haan	AD_VP	13-JAN-01
4	103	Hunold	AC_MGR	03-JAN-06
5	104	Ernst	IT_PROG	21-MAY-07
6	107	Lorentz	IT_PROG	07-FEB-07
7	124	Mourgos	ST_MAN	16-NOV-07
8	141	Rajs	ST_CLERK	17-OCT-03
9	142	Davies	ST_CLERK	29-JAN-05
10	143	Matos	ST_CLERK	15-MAR-06
11	144	Vargas	ST_CLERK	09-JUL-06
12	149	Zlotkey	SA_MAN	29-JAN-08
13	174	Abel	SA_REP	11-MAY-04
14	176	Taylor	SA_REP	24-MAR-06
15	178	Grant	SA_REP	24-MAY-07
16	200	Whalen	AD_ASST	17-SEP-03
17	201	Hartstein	MK_MAN	17-FEB-04
18	202	Fay	MK_REP	17-AUG-05
19	205	Higgins	AC_MGR	07-JUN-02
20	206	Gietz	AC_ACCOUNT	07-JUN-02

6. HR 부서에서 EMPLOYEES 테이블의 모든 고유 직무 ID를 표시하는 query를 요구합니다.

A2	JOB_ID
1	AC_ACCOUNT
2	AC_MGR
3	AD_ASST
4	AD_PRES
5	AD_VP
6	IT_PROG
7	MK_MAN
8	MK_REP
9	SA_MAN
10	SA_REP
11	ST_CLERK
12	ST_MAN

작업 3

시간 여유가 있을 경우 다음 연습을 완료하십시오.

7. HR 부서에서 보고에 적합하도록 열 머리글의 사원 정보를 쉽게 표시해 달라고 합니다. lab_02_5b.sql의 명령문을 새 **SQL Worksheet**로 복사합니다. 열 이름을 각각 Emp #, Employee, Job 및 Hire Date로 지정합니다. 그런 다음 query를 다시 실행합니다.

A2	Emp #	A2	Employee	A2	Job	A2	Hire Date
1	100	King	AD_PRES	17-JUN-03			
2	101	Kochhar	AD_VP	21-SEP-05			
3	102	De Haan	AD_VP	13-JAN-01			
4	103	Hunold	AC_MGR	03-JAN-06			
5	104	Ernst	IT_PROG	21-MAY-07			
6	107	Lorentz	IT_PROG	07-FEB-07			
7	124	Mourgos	ST_MAN	16-NOV-07			
8	141	Rajs	ST_CLERK	17-OCT-03			
9	142	Davies	ST_CLERK	29-JAN-05			
10	143	Matos	ST_CLERK	15-MAR-06			
11	144	Vargas	ST_CLERK	09-JUL-06			
12	149	Zlotkey	SA_MAN	29-JAN-08			
13	174	Abel	SA_REP	11-MAY-04			
14	176	Taylor	SA_REP	24-MAR-06			
15	178	Grant	SA_REP	24-MAY-07			
16	200	Whalen	AD_ASST	17-SEP-03			
17	201	Hartstein	MK_MAN	17-FEB-04			
18	202	Fay	MK_REP	17-AUG-05			
19	205	Higgins	AC_MGR	07-JUN-02			
20	206	Gietz	AC_ACCOUNT	07-JUN-02			

8. HR 부서에서 모든 사원과 해당 직무 ID에 대한 보고서를 요청했습니다. 성과 직무 ID를
이어서 표시하고(첨표와 공백으로 구분) 열 이름을 Employee and Title로 지정합니다.

R	Employee and Title
1	Abel, SA_REP
2	Davies, ST_CLERK
3	De Haan, AD_VP
4	Ernst, IT_PROG
5	Fay, MK_REP
6	Gietz, AC_ACCOUNT

...

19	Whalen, AD_ASST
20	Zlotkey, SA_MAN

다른 작업을 수행하려면 다음 연습을 완료하십시오.

9. EMPLOYEES 테이블의 데이터에 익숙해지도록 해당 테이블의 모든 데이터를 표시하는
query를 생성합니다. 각 열 출력은 첨표로 구분합니다. 열 이름을 THE_OUTPUT으로
지정합니다.

R	THE_OUTPUT
1	100,Steven,King,SKING,515.123.4567,AD_PRES,,17-JUN-03,24000,,90
2	101,Neena,Kochhar,NKOCHHAR,515.123.4568,AD_VP,100,21-SEP-05,17000,,90
3	102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-01,17000,,90
4	103,Alexander,Hunold,AHUNOLD,590.423.4567,AC_MGR,102,03-JAN-06,12008,,60
5	104,Bruce,Ernst,BERNST,590.423.4568,IT_PROG,103,21-MAY-07,6000,,60
6	107,Diana,Lorentz,DLORENTZ,590.423.5567,IT_PROG,103,07-FEB-07,4200,,60

...

18	202,Pat,Fay,PFAY,603.123.6666,MK_REP,201,17-AUG-05,6000,,20
19	205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-02,12008,,110
20	206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-02,8300,,110

해답 2-1: SQL SELECT 문을 사용하여 데이터 검색

작업 1

지식을 테스트해 보십시오.

1. 다음 SELECT 문이 성공적으로 실행됩니다.

```
SELECT last_name, job_id, salary AS Sal
FROM employees;
```

맞음/틀림

2. 다음 SELECT 문이 성공적으로 실행됩니다.

```
SELECT *
FROM job_grades;
```

맞음/틀림

3. 다음 명령문에 네 개의 코딩 오류가 있습니다. 식별할 수 있습니까?

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

- **EMPLOYEES** 테이블에 **sal**이라는 열이 없습니다. 열 이름은 **SALARY**입니다.
- 두 번째 행의 곱하기 연산자는 **x**가 아니라 *****입니다.
- **ANNUAL SALARY alias**에 공백이 포함될 수 없습니다. **alias**는 **ANNUAL_SALARY**로 표기하거나 큰따옴표로 묶어야 합니다.
- **LAST_NAME** 열 뒤에 쉼표가 누락되었습니다.

작업 2

여러분은 Acme Corporation의 SQL 프로그래머로 채용되었습니다. 첫 작업은 Human Resources 테이블의 데이터를 기반으로 몇 가지 보고서를 작성하는 것입니다.

4. 첫 작업은 DEPARTMENTS 테이블의 구조와 해당 내용을 파악하는 것입니다.
- a. DEPARTMENTS 테이블의 구조를 파악하려면 다음 명령문을 사용합니다.

```
DESCRIBE departments
```

- b. DEPARTMENTS 테이블에 포함된 데이터를 확인하려면 다음 명령문을 사용합니다.

```
SELECT *
FROM departments;
```

5. EMPLOYEES 테이블의 구조와 해당 내용을 파악합니다.
- a. EMPLOYEES 테이블의 구조를 확인합니다.

```
DESCRIBE employees
```

- b. HR 부서에서 사원 ID가 먼저 나타나고 이어서 각 사원에 대한 성, 직무 ID, 채용 날짜 및 사원 ID를 표시하는 **query**를 요구합니다. HIRE_DATE 열에 대한 **alias**로 STARTDATE를 입력하십시오. SQL 문을 lab_02_5b.sql이라는 파일에 저장하여 이 파일을 HR 부서에 전달할 수 있도록 합니다. lab_02_5b.sql 파일의 **query**가 제대로 실행되는지 테스트합니다.

```
SELECT employee_id, last_name, job_id, hire_date StartDate
FROM employees;
```

6. HR 부서에서 EMPLOYEES 테이블의 모든 고유 직무 ID를 표시하는 **query**를 요구합니다.

```
SELECT DISTINCT job_id
FROM employees;
```

작업 3

시간 여유가 있을 경우 다음 연습을 완료하십시오.

7. HR 부서에서 보고에 적합하도록 열 머리글의 사원 정보를 쉽게 표시해 달라고 합니다. lab_02_5b.sql의 명령문을 새 **SQL Worksheet**로 복사합니다. 열 이름을 각각 Emp #, Employee, Job 및 Hire Date로 지정합니다. 그런 다음 **query**를 다시 실행합니다.

```
SELECT employee_id "Emp #", last_name "Employee",
       job_id "Job", hire_date "Hire Date"
FROM employees;
```

8. HR 부서에서 모든 사원과 해당 직무 ID에 대한 보고서를 요청했습니다. 성과 직무 ID를 이어서 표시하고(쉼표와 공백으로 구분) 열 이름을 Employee and Title로 지정합니다.

```
SELECT last_name||', '||job_id "Employee and Title"
FROM employees;
```

다른 작업을 수행하려면 다음 연습을 완료하십시오.

9. EMPLOYEES 테이블의 데이터에 익숙해지도록 해당 테이블의 모든 데이터를 표시하는 **query**를 생성합니다. 각 열 출력은 쉼표로 구분합니다. 열 이름을 THE_OUTPUT으로 지정합니다.

```
SELECT employee_id || ',' || first_name || ',' || last_name  
       || ',' || email || ',' || phone_number || ',' || job_id  
       || ',' || manager_id || ',' || hire_date || ','  
       || salary || ',' || commission_pct || ',' ||  
department_id  
       THE_OUTPUT  
FROM   employees;
```

단원 3의 연습: 데이터 제한 및 정렬

3장

단원 3의 연습: 개요

연습 개요

이 연습에서는 다음 내용을 다룹니다.

- 데이터 선택 및 표시되는 행의 순서 변경
- WHERE 절을 사용하여 행 제한
- ORDER BY 절을 사용하여 행 정렬
- 치환 변수를 사용하여 **SQL SELECT** 문에 유연성 부여

연습 3-1: 데이터 제한 및 정렬

개요

이 연습에서는 WHERE 절과 ORDER BY 절을 사용하는 명령문을 사용하여 더 많은 보고서를 작성합니다. 앰퍼샌드 치환을 포함시키면 SQL 문의 재사용 및 범용성을 높일 수 있습니다.

작업

HR 부서에서 몇 가지 query 작성과 관련해 여러분의 도움을 요청합니다.

1. HR 부서에서 예산 문제로 인해 급여가 \$12,000가 넘는 사원의 성과 급여를 표시하는 보고서가 필요합니다. 작성한 SQL 문을 lab_03_01.sql이라는 텍스트 파일로 저장합니다. query를 실행합니다.

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12008

2. 새 SQL Worksheet를 엽니다. 사원 번호 176의 성과 부서 ID를 표시하는 보고서를 작성합니다. query를 실행합니다.

	LAST_NAME	DEPARTMENT_ID
1	Taylor	80

3. HR 부서에서 급여가 높은 사원과 급여가 낮은 사원을 찾아야 합니다. 급여가 \$5,000 ~ \$12,000의 범위에 속하지 않는 모든 사원의 성 및 급여를 표시하도록 lab_03_01.sql을 수정합니다. 작성한 SQL 문을 lab_03_03.sql로 저장합니다.

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Lorentz	4200
5	Rajs	3500
6	Davies	3100
7	Matos	2600
8	Vargas	2500
9	Whalen	4400
10	Hartstein	13000
11	Higgins	12008

4. Matos 및 Taylor라는 성을 가진 사원의 성, 직무 ID, 채용 날짜를 표시하는 보고서를 작성합니다. 채용 날짜를 기준으로 오름차순으로 query를 정렬합니다.

	LAST_NAME	JOB_ID	HIRE_DATE
1	Matos	ST_CLERK	15-MAR-06
2	Taylor	SA_REP	24-MAR-06

5. 부서 20 또는 50에 속하는 모든 사원의 성과 부서 ID를 last_name별로 오름차순으로 정렬하여 표시합니다.

	LAST_NAME	DEPARTMENT_ID
1	Davies	50
2	Fay	20
3	Hartstein	20
4	Matos	50
5	Mourgos	50
6	Rajs	50
7	Vargas	50

6. \$5,000 ~ \$12,000의 급여를 받고 부서 20 또는 50에 속하는 사원의 성과 급여를 표시하도록 lab_03_03.sql을 수정합니다. 열 레이블을 각각 Employee 및 Monthly Salary로 지정합니다. lab_03_03.sql을 lab_03_06.sql로 다시 저장합니다. lab_03_06.sql의 명령문을 실행합니다.

	Employee	Monthly Salary
1	Fay	6000
2	Mourgos	5800

7. HR 부서에서 2006년에 채용된 모든 사원의 성과 채용 날짜를 표시하는 보고서를 요구합니다.

	LAST_NAME	HIRE_DATE
1	Hunold	03-JAN-06
2	Matos	15-MAR-06
3	Vargas	09-JUL-06
4	Taylor	24-MAR-06

8. 담당 관리자가 없는 모든 사원의 성과 직책을 표시하는 보고서를 작성합니다.

	LAST_NAME	JOB_ID
1	King	AD_PRES

9. 커미션을 받는 모든 사원의 성, 급여 및 커미션을 표시하는 보고서를 작성합니다. 급여 및 커미션의 내림차순으로 데이터를 정렬합니다.

ORDER BY 절에서 열의 숫자 위치를 사용합니다.

	1	2	3
	LAST_NAME	SALARY	COMMISSION_PCT
1	Abel	11000	0.3
2	Zlotkey	10500	0.2
3	Taylor	8600	0.2
4	Grant	7000	0.15

10. HR 부서의 멤버는 여러분이 작성 중인 query에 유연성이 강화되기를 원합니다. 사용자가 프롬프트에 지정하는 액수보다 많은 급여를 받는 사원의 급여와 성을 표시하는 보고서를 기대합니다. 이 query를 lab_03_10.sql이라는 파일에 저장합니다. (작업 1에서 생성한 query를 사용하고 수정할 수 있습니다.) 프롬프트가 표시되었을 때 12000을 입력하면 보고서에 다음 결과가 표시됩니다.

	1	2
	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12008

11. HR 부서에서 관리자를 기준으로 보고서를 실행하려고 합니다. 사용자에게 관리자 ID 입력 프롬프트를 표시하고 해당 관리자에 속한 사원의 사원 ID, 성, 급여 및 부서를 생성하는 query를 작성합니다. HR 부서에서 선택한 열을 기준으로 보고서를 정렬하는 기능을 원합니다. 다음 값으로 데이터를 테스트할 수 있습니다.

manager_id = 103, last_name을 기준으로 정렬:

	1	2	3	4
	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	104	Ernst	6000	60
2	107	Lorentz	4200	60

manager_id = 201, salary를 기준으로 정렬:

	1	2	3	4
	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	202	Fay	6000	20

manager_id = 124, employee_id를 기준으로 정렬:

	1	2	3	4
	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	141	Rajs	3500	50
2	142	Davies	3100	50
3	143	Matos	2600	50
4	144	Vargas	2500	50

시간 여유가 있을 경우 다음 연습을 완료하십시오.

12. 이름의 세 번째 문자가 "a"인 모든 사원의 성을 표시합니다.

	LAST_NAME
1	Grant
2	Whalen

13. 성에 "a"와 "e"가 모두 포함된 모든 사원의 성을 표시합니다.

	LAST_NAME
1	Davies
2	De Haan
3	Hartstein
4	Whalen

다른 작업을 수행하려면 다음 연습을 완료합니다.

14. 직무가 판매 사원이나 자재 담당자이고 급여가 \$2,500, \$3,500 또는 \$7,000가 아닌 모든 사원의 성, 직무 및 급여를 표시합니다.

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000
2	Taylor	SA_REP	8600
3	Davies	ST_CLERK	3100
4	Matos	ST_CLERK	2600

15. 커미션이 20%인 모든 사원의 성, 급여 및 커미션을 표시하도록 lab_03_06.sql을 수정합니다. lab_03_06.sql을 lab_03_15.sql로 다시 저장합니다. lab_03_15.sql의 명령문을 다시 실행합니다.

	Employee	Monthly Salary	COMMISSION_PCT
1	Zlotkey	10500	0.2
2	Taylor	8600	0.2

해답 3-1: 데이터 제한 및 정렬

HR 부서에서 몇 가지 query 작성과 관련해 여러분의 도움을 요청합니다.

1. HR 부서에서 예산 문제로 인해 급여가 \$12,000가 넘는 사원의 성과 급여를 표시하는 보고서가 필요합니다. 작성한 SQL 문을 lab_03_01.sql이라는 파일로 저장합니다. query를 실행합니다.

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000;
```

2. 새 SQL Worksheet를 엽니다. 사원 번호 176의 성과 부서 ID를 표시하는 보고서를 작성합니다.

```
SELECT last_name, department_id
FROM employees
WHERE employee_id = 176;
```

3. HR 부서에서 급여가 높은 사원과 급여가 낮은 사원을 찾아야 합니다. 급여가 \$5,000 ~ \$12,000의 범위에 속하지 않는 모든 사원의 성 및 급여를 표시하도록 lab_03_01.sql을 수정합니다. 작성한 SQL 문을 lab_03_03.sql로 저장합니다.

```
SELECT last_name, salary
FROM employees
WHERE salary NOT BETWEEN 5000 AND 12000;
```

4. Matos 및 Taylor라는 성을 가진 사원의 성, 직무 ID, 채용 날짜를 표시하는 보고서를 작성합니다. 채용 날짜를 기준으로 오름차순으로 query를 정렬합니다.

```
SELECT last_name, job_id, hire_date
FROM employees
WHERE last_name IN ('Matos', 'Taylor')
ORDER BY hire_date;
```

5. 부서 20 또는 50에 속하는 모든 사원의 성과 부서 ID를 last_name별로 오름차순으로 정렬하여 표시합니다.

```
SELECT last_name, department_id
FROM employees
WHERE department_id IN (20, 50)
ORDER BY last_name ASC;
```

6. \$5,000 ~ \$12,000의 급여를 받고 부서 20 또는 50에 속하는 사원의 성과 급여를 표시하도록 lab_03_03.sql을 수정합니다. 열 레이블을 각각 Employee 및 Monthly Salary로 지정합니다. lab_03_03.sql을 lab_03_06.sql로 다시 저장합니다. lab_03_06.sql의 명령문을 실행합니다.

```
SELECT last_name "Employee", salary "Monthly Salary"
FROM employees
WHERE salary BETWEEN 5000 AND 12000
AND department_id IN (20, 50);
```

7. HR 부서에서 2006년에 채용된 모든 사원의 성과 채용 날짜를 표시하는 보고서를 요구합니다.

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date >= '01-JAN-06' AND hire_date < '01-JAN-07';
```

8. 담당 관리자가 없는 모든 사원의 성과 직책을 표시하는 보고서를 작성합니다.

```
SELECT last_name, job_id
FROM employees
WHERE manager_id IS NULL;
```

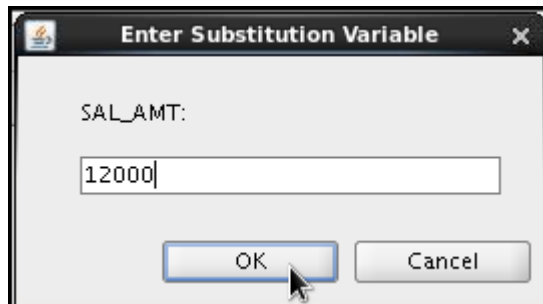
9. 커미션을 받는 모든 사원의 성, 급여 및 커미션을 표시하는 보고서를 작성합니다. 급여 및 커미션을 내림차순으로 데이터를 정렬합니다. ORDER BY 절에서 열의 숫자 위치를 사용합니다.

```
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 강화되기를 원합니다. 사용자가 프롬프트에 지정하는 액수보다 많은 급여를 받는 사원의 급여와 성을 표시하는 보고서를 기대합니다. (작업 1에서 생성한 query를 사용하고 수정할 수 있습니다.) 이 query를 lab_03_10.sql이라는 파일에 저장합니다.
프롬프트가 나타나면 12000을 입력합니다.

```
SELECT last_name, salary
FROM employees
WHERE salary > &sal_amt;
```

프롬프트가 나타나면 대화상자에 값으로 12000을 입력합니다. OK를 누릅니다.



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

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
FROM employees
WHERE manager_id = &mgr_num
ORDER BY &order_col;
```

시간 여유가 있을 경우 다음 연습을 완료하십시오.

12. 이름의 세 번째 문자가 "a"인 모든 사원의 성을 표시합니다.

```
SELECT last_name
FROM employees
WHERE last_name LIKE '___a%';
```

13. 성에 "a"와 "e"가 모두 포함된 모든 사원의 성을 표시합니다.

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%'
AND last_name LIKE '%e%';
```

다른 작업을 수행하려면 다음 연습을 완료합니다.

14. 직무가 판매 사원이나 자재 담당자이고 급여가 \$2,500, \$3,500 또는 \$7,000가 아닌 모든 사원의 성, 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 성, 급여 및 커미션을 표시하도록 lab_03_06.sql을 수정합니다. lab_03_06.sql을 lab_03_15.sql로 다시 저장합니다. lab_03_15.sql의 명령문을 다시 실행합니다.

```
SELECT last_name "Employee", salary "Monthly Salary",
       commission_pct
FROM employees
WHERE commission_pct = .20;
```


단원 4의 연습: 단일 행 함수를 사용하여 출력 커스터마이즈

4장

단원 4의 연습: 개요

연습 개요

이 연습에서는 다음 내용을 다룹니다.

- 현재 날짜를 표시하는 **query** 작성
- 숫자, 문자 및 날짜 함수를 사용해야 하는 **query** 작성
- 사원의 근무 연수 및 월수 계산

연습 4-1: 단일 행 함수를 사용하여 출력 커스터마이징

개요

이 연습에서는 문자, 숫자 및 날짜 데이터 유형에 사용할 수 있는 다양한 함수의 사용법을 익힙니다. 중첩 함수의 경우 결과는 가장 안쪽 함수에서 가장 바깥쪽 함수로 평가됩니다.

작업

1. 시스템 날짜를 표시하기 위한 **query**를 작성합니다. 열 레이블을 Date로 지정합니다.

주: 데이터베이스가 시간대가 다른 지역에 있는 경우 해당 데이터베이스가 상주하는 운영 체제의 날짜가 출력됩니다.

Date
1 30-AUG-12

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 지정합니다. 작성한 **SQL** 문을 lab_04_02.sql이라는 파일에 저장합니다.
3. lab_04_02.sql 파일의 **query**를 실행합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
1	100	King	24000	27720
2	101	Kochhar	17000	19635
3	102	De Haan	17000	19635
4	103	Hunold	9000	10395
5	104	Ernst	6000	6930
6	107	Lorentz	4200	4851
7	124	Mourgos	5800	6699
8	141	Rajs	3500	4043
9	142	Davies	3100	3581
10	143	Matos	2600	3003
11	144	Vargas	2500	2888
12	149	Zlotkey	10500	12128
13	174	Abel	11000	12705
14	176	Taylor	8600	9933
15	178	Grant	7000	8085
16	200	Whalen	4400	5082
17	201	Hartstein	13000	15015
18	202	Fay	6000	6930
19	205	Higgins	12008	13869
20	206	Gietz	8300	9587

4. 새 급여에서 이전 급여를 빼는 열을 추가하도록 lab_04_02.sql의 **query**를 수정합니다. 열 레이블을 Increase로 지정합니다. 파일 내용을 lab_04_04.sql로 저장합니다. 수정한 **query**를 실행합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
1	100	King	24000	27720	3720
2	101	Kochhar	17000	19635	2635
3	102	De Haan	17000	19635	2635
4	103	Hunold	9000	10395	1395
5	104	Ernst	6000	6930	930
6	107	Lorentz	4200	4851	651
7	124	Mourgos	5800	6699	899
8	141	Rajs	3500	4043	543
9	142	Davies	3100	3581	481
10	143	Matos	2600	3003	403
11	144	Vargas	2500	2888	388
12	149	Zlotkey	10500	12128	1628
13	174	Abel	11000	12705	1705
14	176	Taylor	8600	9933	1333
15	178	Grant	7000	8085	1085
16	200	Whalen	4400	5082	682
17	201	Hartstein	13000	15015	2015
18	202	Fay	6000	6930	930
19	205	Higgins	12008	13869	1861
20	206	Gietz	8300	9587	1287

5. 다음 작업을 수행합니다.
- a. "J", "A" 또는 "M"으로 시작하는 이름을 가진 모든 사원의 성(첫번째 문자는 대문자, 나머지는 모두 소문자)과 성의 길이를 표시하는 **query**를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 사원의 성을 기준으로 결과를 정렬합니다.

	Name	Length
1	Abel	4
2	Matos	5
3	Mourgos	7

- b. 유저에게 성의 첫 문자를 입력하는 프롬프트를 표시하도록 **query**를 재작성합니다. 예를 들어, 문자 입력 프롬프트가 표시되었을 때 유저가 "H"(대문자)를 입력하면 성이 "H"로 시작하는 모든 사원이 출력에 표시되어야 합니다.

	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

- c. 입력된 문자의 대소문자 여부에 따라 출력이 달라지지 않도록 **query**를 수정합니다. 입력된 문자는 **SELECT query**에서 처리되기 전에 대문자로 변경해야 합니다.

	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

시간 여유가 있을 경우 다음 연습을 완료하십시오.

6. **HR** 부서에서 각 사원의 근속 기간을 파악하려고 합니다. 각 사원에 대해 성을 표시하고 채용일부터 오늘까지 경과한 개월 수를 계산합니다. 열 레이블을 **MONTHS_WORKED**로 지정합니다. 재직 개월 수에 따라 결과를 정렬합니다. 개월 수는 가장 가까운 정수로 반올림해야 합니다.

주: 이 query는 실행된 날짜에 의존하므로 MONTHS_WORKED 열의 값이 이에 따라 달라집니다.

	LAST_NAME	MONTHS_WORKED
1	Zlotkey	55
2	Mourgos	57
3	Grant	63
4	Ernst	63
5	Lorentz	67
6	Vargas	74
7	Matos	77
8	Taylor	77
9	Hunold	80
10	Kochhar	83
11	Fay	84
12	Davies	91
13	Abel	100
14	Hartstein	102
15	Rajs	106
16	Whalen	107
17	King	110
18	Higgins	123
19	Gietz	123
20	De Haan	140

7. 모든 사원의 성과 급여를 표시하기 위한 **query**를 작성합니다. 급여가 15자 길이로 표시되고 왼쪽에 \$ 기호가 채워지도록 형식을 지정합니다. 열 레이블을 SALARY로 지정합니다.

R 2	LAST_NAME	R 2	SALARY
1	King		\$\$\$\$\$\$\$\$\$24000
2	Kochhar		\$\$\$\$\$\$\$\$\$17000
3	De Haan		\$\$\$\$\$\$\$\$\$17000
4	Hunold		\$\$\$\$\$\$\$\$\$9000
5	Ernst		\$\$\$\$\$\$\$\$\$6000
6	Lorentz		\$\$\$\$\$\$\$\$\$4200
7	Mourgos		\$\$\$\$\$\$\$\$\$5800
8	Rajs		\$\$\$\$\$\$\$\$\$3500
9	Davies		\$\$\$\$\$\$\$\$\$3100
10	Matos		\$\$\$\$\$\$\$\$\$2600
11	Vargas		\$\$\$\$\$\$\$\$\$2500
12	Zlotkey		\$\$\$\$\$\$\$\$\$10500
13	Abel		\$\$\$\$\$\$\$\$\$11000
14	Taylor		\$\$\$\$\$\$\$\$\$8600
15	Grant		\$\$\$\$\$\$\$\$\$7000
16	Whalen		\$\$\$\$\$\$\$\$\$4400
17	Hartstein		\$\$\$\$\$\$\$\$\$13000
18	Fay		\$\$\$\$\$\$\$\$\$6000
19	Higgins		\$\$\$\$\$\$\$\$\$12008
20	Gietz		\$\$\$\$\$\$\$\$\$8300

8. 사원의 성을 표시하고 급여 액수를 별표로 나타내는 **query**를 작성합니다. 각 별표는 **\$1,000**을 나타냅니다. 급여의 내림차순으로 데이터를 정렬합니다. 열 레이블을 **EMPLOYEES_AND_THEIR_SALARIES**로 지정합니다.

1	LAST_NAME	2	EMPLOYEES_AND_THEIR_SALARIES
1	King		*****
2	Kochhar		*****
3	De Haan		*****
4	Hartstein		*****
5	Higgins		*****
6	Abel		*****
7	Zlotkey		*****
8	Hunold		*****
9	Taylor		*****
10	Gietz		*****
11	Grant		*****
12	Ernst		*****
13	Fay		*****
14	Mourgos		*****
15	Whalen		*****
16	Lorentz		*****
17	Rajs		*****
18	Davies		*****
19	Matos		*****
20	Vargas		*****

9. 부서 90의 모든 사원에 대해 성 및 재직 기간(주 단위)을 표시하도록 **query**를 작성합니다. 주를 나타내는 숫자 열의 레이블을 **TENURE**로 지정합니다. 주를 나타내는 숫자 값을 소수점 왼쪽에서 **truncate**합니다. 직원 재직 기간의 내림차순으로 레코드를 표시합니다.

주: **TENURE** 값은 **query**를 실행한 날짜에 의존하므로 이에 따라 달라집니다.

1	LAST_NAME	2	TENURE
1	De Haan		606
2	King		480
3	Kochhar		362

해답 4-1: 단일 행 함수를 사용하여 출력 커스터마이징

1. 시스템 날짜를 표시하기 위한 **query**를 작성합니다. 열 레이블을 Date로 지정합니다.

주: 데이터베이스가 시간대가 다른 지역에 있는 경우 해당 데이터베이스가 상주하는 운영 체제의 날짜가 출력됩니다.

```
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 지정합니다. 작성한 SQL 문을 lab_04_02.sql이라는 파일에 저장합니다.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary"
FROM employees;
```

3. lab_04_02.sql 파일의 **query**를 실행합니다.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary"
FROM employees;
```

4. 새 급여에서 이전 급여를 뺀 열을 추가하도록 lab_04_02.sql의 **query**를 수정합니다. 열 레이블을 Increase로 지정합니다. 파일 내용을 lab_04_04.sql로 저장합니다. 수정한 **query**를 실행합니다.

```
SELECT employee_id, last_name, salary,
       ROUND(salary * 1.155, 0) "New Salary",
       ROUND(salary * 1.155, 0) - salary "Increase"
FROM employees;
```

5. 다음 작업을 수행합니다.

- a. "J", "A" 또는 "M"으로 시작하는 이름을 가진 모든 사원의 성(첫번째 문자는 대문자, 나머지는 모두 소문자)과 성의 길이를 표시하는 **query**를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 사원의 성을 기준으로 결과를 정렬합니다.

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
FROM employees
WHERE last_name LIKE 'J%'
OR      last_name LIKE 'M%'
OR      last_name LIKE 'A%'
ORDER BY last_name;
```

- b. 유저에게 성의 첫 문자를 입력하는 프롬프트를 표시하도록 **query**를 재작성합니다. 예를 들어, 문자 입력 프롬프트가 표시되었을 때 유저가 H(대문자)를 입력하면 성이 "H"로 시작하는 모든 사원이 표시되어야 합니다.

```
SELECT  INITCAP(last_name) "Name",
        LENGTH(last_name) "Length"
FROM    employees
WHERE    last_name LIKE '&start_letter%'
ORDER BY last_name;
```

- c. 입력된 문자의 대소문자 여부에 따라 출력이 달라지지 않도록 **query**를 수정합니다. 입력된 문자는 **SELECT query**에서 처리되기 전에 대문자로 변경해야 합니다.

```
SELECT  INITCAP(last_name) "Name",
        LENGTH(last_name) "Length"
FROM    employees
WHERE    last_name LIKE UPPER('&start_letter%' )
ORDER BY last_name;
```

시간 여유가 있을 경우 다음 연습을 완료하십시오.

6. HR 부서에서 각 사원의 근속 기간을 파악하려고 합니다. 각 사원에 대해 성을 표시하고 채용일부터 오늘까지 경과한 개월 수를 계산합니다. 열 레이블을 MONTHS_WORKED로 지정합니다. 재직 개월 수에 따라 결과를 정렬합니다. 개월 수는 가장 가까운 정수로 반올림해야 합니다.

주: 이 **query**는 실행된 날짜에 의존하므로 MONTHS_WORKED 열의 값이 이에 따라 달라집니다.

```
SELECT last_name, ROUND(MONTHS_BETWEEN(
        SYSDATE, hire_date)) MONTHS_WORKED
FROM    employees
ORDER BY months_worked;
```

7. 모든 사원의 성과 급여를 표시하기 위한 **query**를 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 성을 표시하고 급여 액수를 별표로 나타내는 **query**를 작성합니다. 각 별표는 **\$1,000**을 나타냅니다. 급여의 내림차순으로 데이터를 정렬합니다. 열 레이블을 **EMPLOYEES_AND_THEIR_SALARIES**로 지정합니다.

```
SELECT last_name,  
       rpad(' ', salary/1000, '*')  
         EMPLOYEES_AND_THEIR_SALARIES  
FROM   employees  
ORDER BY salary DESC;
```

9. 부서 90의 모든 사원에 대해 성 및 재직 기간(주 단위)을 표시하도록 **query**를 작성합니다. 주를 나타내는 숫자 열의 레이블을 **TENURE**로 지정합니다. 주를 나타내는 숫자 값을 소수점 왼쪽에서 **truncate**합니다. 직원 재직 기간의 내림차순으로 레코드를 표시합니다.
- 주: **TENURE** 값은 **query**를 실행한 날짜에 의존하므로 이에 따라 달라집니다.

```
SELECT last_name, trunc((SYSDATE-hire_date)/7) AS TENURE  
FROM   employees  
WHERE  department_id = 90  
ORDER BY TENURE DESC;
```


단원 5의 연습:
변환 함수 및 조건부 표현식 사용
5장

단원 5의 연습: 개요

연습 개요

이 연습에서는 다음 내용을 다룹니다.

- TO_CHAR 및 TO_DATE 함수를 사용하는 query를 작성
- CASE, 검색된 CASE 및 DECODE 등의 조건부 표현식을 사용하는 query 작성

연습 5-1: 변환 함수 및 조건부 표현식 사용

개요

이 연습에서는 TO_CHAR 및 TO_DATE 함수와 CASE, 검색된 CASE 및 DECODE와 같은 조건부 표현식을 사용하는 다양한 실습을 제공합니다.

작업

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.

	Dream Salaries
1	King earns \$24,000.00 monthly but wants \$72,000.00.
2	Kochhar earns \$17,000.00 monthly but wants \$51,000.00.
3	De Haan earns \$17,000.00 monthly but wants \$51,000.00.
4	Hunold earns \$12,008.00 monthly but wants \$36,024.00.
5	Ernst earns \$6,000.00 monthly but wants \$18,000.00.
6	Lorentz earns \$4,200.00 monthly but wants \$12,600.00.
7	Mourgos earns \$5,800.00 monthly but wants \$17,400.00.
8	Rajs earns \$3,500.00 monthly but wants \$10,500.00.
9	Davies earns \$3,100.00 monthly but wants \$9,300.00.
10	Matos earns \$2,600.00 monthly but wants \$7,800.00.
11	Vargas earns \$2,500.00 monthly but wants \$7,500.00.
12	Zlotkey earns \$10,500.00 monthly but wants \$31,500.00.
13	Abel earns \$11,000.00 monthly but wants \$33,000.00.
14	Taylor earns \$8,600.00 monthly but wants \$25,800.00.
15	Grant earns \$7,000.00 monthly but wants \$21,000.00.
16	Whalen earns \$4,400.00 monthly but wants \$13,200.00.
17	Hartstein earns \$13,000.00 monthly but wants \$39,000.00.
18	Fay earns \$6,000.00 monthly but wants \$18,000.00.
19	Higgins earns \$12,008.00 monthly but wants \$36,024.00.
20	Gietz earns \$8,300.00 monthly but wants \$24,900.00.

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

	A Z	LAST_NAME	A Z	HIRE_DATE	A Z	REVIEW
1		King		17-JUN-03		Monday, the Twenty-Second of December, 2003
2		Kochhar		21-SEP-05		Monday, the Twenty-Seventh of March, 2006
3		De Haan		13-JAN-01		Monday, the Sixteenth of July, 2001
4		Hunold		03-JAN-06		Monday, the Tenth of July, 2006
5		Ernst		21-MAY-07		Monday, the Twenty-Sixth of November, 2007
6		Lorentz		07-FEB-07		Monday, the Thirteenth of August, 2007
7		Mourgos		16-NOV-07		Monday, the Nineteenth of May, 2008
8		Rajs		17-OCT-03		Monday, the Nineteenth of April, 2004
9		Davies		29-JAN-05		Monday, the First of August, 2005
10		Matos		15-MAR-06		Monday, the Eighteenth of September, 2006
11		Vargas		09-JUL-06		Monday, the Fifteenth of January, 2007
12		Zlotkey		29-JAN-08		Monday, the Fourth of August, 2008
13		Abel		11-MAY-04		Monday, the Fifteenth of November, 2004
14		Taylor		24-MAR-06		Monday, the Twenty-Fifth of September, 2006
15		Grant		24-MAY-07		Monday, the Twenty-Sixth of November, 2007
16		Whalen		17-SEP-03		Monday, the Twenty-Second of March, 2004
17		Hartstein		17-FEB-04		Monday, the Twenty-Third of August, 2004
18		Fay		17-AUG-05		Monday, the Twentieth of February, 2006
19		Higgins		07-JUN-02		Monday, the Ninth of December, 2002
20		Gietz		07-JUN-02		Monday, the Ninth of December, 2002

3. 사원의 성 및 커미션 금액을 표시하는 **query**를 작성합니다. 사원이 커미션을 받지 않으면 "No Commission"을 표시합니다. 열 레이블을 COMM으로 지정합니다.

	LAST_NAME	COMM
1	King	No Commission
2	Kochhar	No Commission
3	De Haan	No Commission
4	Hunold	No Commission
5	Ernst	No Commission
6	Lorentz	No Commission
7	Mourgos	No Commission
8	Rajs	No Commission
9	Davies	No Commission
10	Matos	No Commission
11	Vargas	No Commission
12	Zlotkey	.2
13	Abel	.3
14	Taylor	.2
15	Grant	.15
16	Whalen	No Commission
17	Hartstein	No Commission
18	Fay	No Commission
19	Higgins	No Commission
20	Gietz	No Commission

4. 다음 데이터를 사용하여 CASE 함수를 통해 JOB_ID 열의 값을 기준으로 모든 사원의 등급을 표시하는 query를 작성합니다.

직무	등급
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
해당 사항 없음	0

	JOB_ID	GRADE
1	AC_ACCOUNT	0
2	AC_MGR	0
3	AD_ASST	0
4	AD_PRES	A
5	AD_VP	0
6	AD_VP	0
7	IT_PROG	C
8	IT_PROG	C
9	IT_PROG	C
10	MK_MAN	0
11	MK_REP	0
12	SA_MAN	0
13	SA_REP	D
14	SA_REP	D
15	SA_REP	D
16	ST_CLERK	E
17	ST_CLERK	E
18	ST_CLERK	E
19	ST_CLERK	E
20	ST_MAN	B

5. 검색된 CASE 구문을 사용하여 앞의 연습에 나오는 명령문을 재작성합니다.

	1/2	JOB_ID	1/2	GRADE
1		AC_ACCOUNT		O
2		AC_MGR		O
3		AD_ASST		O
4		AD_PRES		A
5		AD_VP		O
6		AD_VP		O
7		IT_PROG		C
8		IT_PROG		C
9		IT_PROG		C
10		MK_MAN		O
11		MK_REP		O
12		SA_MAN		O
13		SA_REP		D
14		SA_REP		D
15		SA_REP		D
16		ST_CLERK		E
17		ST_CLERK		E
18		ST_CLERK		E
19		ST_CLERK		E
20		ST_MAN		B

6. 검색된 DECODE 구문을 사용하여 앞의 연습에 나오는 명령문을 재작성합니다.

	<small>A Z</small> JOB_ID	<small>A Z</small> GRADE
1	AC_ACCOUNT	O
2	AC_MGR	O
3	AD_ASST	O
4	AD_PRES	A
5	AD_VP	O
6	AD_VP	O
7	IT_PROG	C
8	IT_PROG	C
9	IT_PROG	C
10	MK_MAN	O
11	MK_REP	O
12	SA_MAN	O
13	SA_REP	D
14	SA_REP	D
15	SA_REP	D
16	ST_CLERK	E
17	ST_CLERK	E
18	ST_CLERK	E
19	ST_CLERK	E
20	ST_MAN	B

해답 5-1: 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 첫번째 월요일에 해당하는 급여 심의 날짜를 표시합니다. 열 레이블을 REVIEW로 지정합니다. 날짜 형식을 "Monday, the Thirty-First of July, 2000"과 유사한 형식으로 지정합니다.

```
SELECT last_name, hire_date,
       TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),
               'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW
FROM   employees;
```

3. 사원의 성 및 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;
```

4. 다음 데이터를 사용하여 CASE 함수를 통해 JOB_ID 열의 값을 기준으로 모든 사원의 등급을 표시하는 query를 작성합니다.

직무	등급
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
해당 사항 없음	0

```

SELECT job_id, CASE job_id
                WHEN 'ST_CLERK' THEN 'E'
                WHEN 'SA_REP'   THEN 'D'
                WHEN 'IT_PROG'  THEN 'C'
                WHEN 'ST_MAN'   THEN 'B'
                WHEN 'AD_PRES'  THEN 'A'
                ELSE '0'   END  GRADE
FROM employees;

```

5. 검색된 CASE 구문을 사용하여 앞의 연습에 나오는 명령문을 재작성합니다.

```

SELECT job_id, CASE
                WHEN job_id = 'ST_CLERK' THEN 'E'
                WHEN job_id = 'SA_REP'   THEN 'D'
                WHEN job_id = 'IT_PROG'  THEN 'C'
                WHEN job_id = 'ST_MAN'   THEN 'B'
                WHEN job_id = 'AD_PRES'  THEN 'A'
                ELSE '0'   END  GRADE
FROM employees;

```

6. 검색된 DECODE 구문을 사용하여 앞의 연습에 나오는 명령문을 재작성합니다.

```

SELECT job_id, decode (job_id,
                        'ST_CLERK', 'E',
                        'SA_REP',   'D',
                        'IT_PROG',  'C',
                        'ST_MAN',   'B',
                        'AD_PRES',  'A',
                        '0') GRADE
FROM employees;

```

단원 6의 연습: 그룹 함수를 사용하고 집계된 데이터 보고

6장

단원 6의 연습: 개요

연습 개요

이 연습에서는 다음 내용을 다룹니다.

- 그룹 함수를 사용하는 **query** 작성
- 여러 결과를 볼 수 있도록 행별로 그룹화
- **HAVING** 절을 사용하여 그룹 제한

연습 6-1: 그룹 함수를 사용하여 집계 데이터 보고

개요

이 연습을 마친 후에는 그룹 함수 사용과 데이터 그룹 선택에 능숙해질 것입니다.

작업

다음 명령문의 유효성을 확인합니다. 맞음 또는 틀림에 **O**표 하십시오.

1. 그룹 함수는 다수 행에 대해 실행되어 그룹당 하나의 결과를 산출합니다.
맞음/틀림
2. 그룹 함수는 계산에 null을 포함합니다.
맞음/틀림
3. WHERE 절은 그룹 계산에 포함시키기 전에 행을 제한합니다.
맞음/틀림

HR 부서에서 다음 보고서를 요구합니다.

4. 모든 사원의 최고, 최저, 합계 및 평균 급여를 찾습니다. 열 레이블을 각각 Maximum, Minimum, Sum 및 Average로 지정합니다. 결과를 가장 가까운 정수로 반올림합니다. SQL 문을 lab_06_04.sql로 저장합니다. query를 실행합니다.

	Maximum	Minimum	Sum	Average
1	24000	2500	175508	8775

5. 각 직무 유형에 대해 최소, 최대, 합계 및 평균 급여를 표시하도록 lab_06_04.sql의 query를 수정합니다. lab_06_04.sql을 lab_06_05.sql로 다시 저장합니다. lab_06_05.sql의 명령문을 실행합니다.

	JOB_ID	Maximum	Minimum	Sum	Average
1	IT_PROG	9000	4200	19200	6400
2	AC_MGR	12008	12008	12008	12008
3	AC_ACCOUNT	8300	8300	8300	8300
4	ST_MAN	5800	5800	5800	5800
5	AD_ASST	4400	4400	4400	4400
6	AD_VP	17000	17000	34000	17000
7	SA_MAN	10500	10500	10500	10500
8	MK_MAN	13000	13000	13000	13000
9	AD_PRES	24000	24000	24000	24000
10	SA_REP	11000	7000	26600	8867
11	MK_REP	6000	6000	6000	6000
12	ST_CLERK	3500	2500	11700	2925

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

	JOB_ID	COUNT(*)
1	AC_ACCOUNT	1
2	AC_MGR	1
3	AD_ASST	1
4	AD_PRES	1
5	AD_VP	2
6	IT_PROG	3
7	MK_MAN	1
8	MK_REP	1
9	SA_MAN	1
10	SA_REP	3
11	ST_CLERK	4
12	ST_MAN	1

HR 부서의 유저에게 직무를 입력하는 프롬프트를 표시하도록 query를 일반화합니다. 이 스크립트를 lab_06_06.sql이라는 파일에 저장합니다. query를 실행합니다. 프롬프트가 표시되면 IT_PROG를 입력합니다.

	JOB_ID	COUNT(*)
1	IT_PROG	3

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

힌트: MANAGER_ID 열을 사용하여 관리자 수를 확인합니다.

	Number of Managers
1	8

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

	DIFFERENCE
1	21500






시간 여유가 있을 경우 다음 연습을 완료하십시오.

9. 관리자 번호 및 해당 관리자의 부하 사원 중 최저 급여를 받는 사원의 급여를 표시하는 보고서를 작성합니다. 관리자를 알 수 없는 모든 사원을 제외합니다. 최소 급여가 \$6,000 이하인 그룹을 제외합니다. 급여의 내림차순으로 출력을 정렬합니다.







	MANAGER_ID	MIN(SALARY)
1	102	9000
2	205	8300
3	149	7000

다른 작업을 수행하려면 다음 연습을 완료합니다.

10. 사원의 총 수와 그 총 수 중 2005년, 2006년, 2007년, 2008년에 채용된 사원의 수를 표시하는 **query**를 작성합니다. 적절한 열 머리글을 지정하십시오.

	 TOTAL	 2005	 2006	 2007	 2008
1	20	3	4	4	1

11. 부서 20, 50, 80 및 90에 대해 직무, 부서 번호별 해당 직무에 대한 급여 및 해당 직무에 대한 총 급여를 표시하고 각 열에 적절한 머리글을 지정하기 위한 **Matrix query**를 작성합니다.

	 Job	 Dept 20	 Dept 50	 Dept 80	 Dept 90	 Total
1	IT_PROG	(null)	(null)	(null)	(null)	19200
2	AC_MGR	(null)	(null)	(null)	(null)	12008
3	AC_ACCOUNT	(null)	(null)	(null)	(null)	8300
4	ST_MAN	(null)	5800	(null)	(null)	5800
5	AD_ASST	(null)	(null)	(null)	(null)	4400
6	AD_VP	(null)	(null)	(null)	34000	34000
7	SA_MAN	(null)	(null)	10500	(null)	10500
8	MK_MAN	13000	(null)	(null)	(null)	13000
9	AD_PRES	(null)	(null)	(null)	24000	24000
10	SA_REP	(null)	(null)	19600	(null)	26600
11	MK_REP	6000	(null)	(null)	(null)	6000
12	ST_CLERK	(null)	11700	(null)	(null)	11700

해답 6-1: 그룹 함수를 사용하여 집계 데이터 보고

다음 명령문의 유효성을 확인합니다. 맞음 또는 틀림에 O표 하십시오.

1. 그룹 함수는 다수 행에 대해 실행되어 그룹당 하나의 결과를 산출합니다.
맞음/틀림
2. 그룹 함수는 계산에 null을 포함합니다.
맞음/틀림
3. WHERE 절은 그룹 계산에 포함시키기 전에 행을 제한합니다.
맞음/틀림

HR 부서에서 다음 보고서를 요구합니다.

4. 모든 사원의 최고, 최저, 합계 및 평균 급여를 찾습니다. 열 레이블을 각각 Maximum, Minimum, Sum 및 Average로 지정합니다. 결과를 가장 가까운 정수로 반올림합니다. SQL 문을 lab_06_04.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;
```

5. 각 직무 유형에 대해 최소, 최대, 합계 및 평균 급여를 표시하도록 lab_06_04.sql의 query를 수정합니다. lab_06_04.sql을 lab_06_05.sql로 다시 저장합니다. lab_06_05.sql의 명령문을 실행합니다.

```
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;
```

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

```
SELECT job_id, COUNT(*)
FROM   employees
GROUP BY job_id;
```

HR 부서의 유저에게 직무를 입력하는 프롬프트를 표시하도록 query를 일반화합니다. 이 스크립트를 lab_06_06.sql이라는 파일에 저장합니다. query를 실행합니다. 프롬프트가 나타나면 IT_PROG를 입력하고 OK를 누릅니다.

```
SELECT job_id, COUNT(*)
FROM   employees
WHERE  job_id = '&job_title'
GROUP BY job_id;
```

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

힌트: MANAGER_ID 열을 사용하여 관리자 수를 확인합니다.

```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"
FROM   employees;
```

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

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE
FROM   employees;
```

시간 여유가 있을 경우 다음 연습을 완료하십시오.

9. 관리자 번호 및 해당 관리자의 부하 사원 중 최저 급여를 받는 사원의 급여를 표시하는 보고서를 작성합니다. 관리자를 알 수 없는 모든 사원을 제외합니다. 최소 급여가 \$6,000 이하인 그룹을 제외합니다. 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;
```

다른 작업을 수행하려면 다음 연습을 완료합니다.

10. 사원의 총 수와 2005년, 2006년, 2007년, 2008년에 채용된 사원의 수를 표시하는 query를 작성합니다. 적절한 열 머리를 지정하십시오.

```
SELECT COUNT(*) total,
       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",
       SUM(DECODE(TO_CHAR(hire_date, 'YYYY'), 2008, 1, 0)) "2008"
FROM   employees;
```

11. 부서 20, 50, 80 및 90에 대해 직무, 부서 번호별 해당 직무에 대한 급여 및 해당 직무에 대한 총 급여를 표시하고 각 열에 적절한 머리글을 지정하기 위한 **Matrix query**를 작성합니다.

```
SELECT    job_id "Job",
          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",
          SUM(salary) "Total"
FROM      employees
GROUP BY  job_id;
```

단원 7의 연습: 조인을 사용하여 다중 테이블의 데이터 표시

7장

단원 7의 연습: 개요

연습 개요

이 연습에서는 다음 내용을 다룹니다.

- **Equijoin**을 사용하여 테이블 조인
- **Outer join** 및 **self-join** 수행
- 조건 추가

연습 7-1: 조인을 사용하여 다중 테이블의 데이터 표시

개요

이 연습은 **SQL:1999** 호환 조인을 사용하여 다중 테이블에서 데이터를 추출하는 과정을 실습할 수 있도록 구성되었습니다.

작업

1. HR 부서를 위해 모든 부서의 주소를 생성하는 **query**를 작성합니다. LOCATIONS 및 COUNTRIES 테이블을 사용합니다. 출력에 위치 ID, 동/리, 구/군, 시/도 및 국가를 표시합니다. NATURAL JOIN을 사용하여 결과를 생성합니다.

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.	Toronto	Ontario	Canada
5	2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

2. HR 부서에서 해당 부서가 있는 모든 사원에 대해 보고서를 요구합니다. 이러한 사원의 성, 부서 번호 및 부서 이름을 표시하는 **query**를 작성합니다.

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Abel	80	Sales
2	Davies	50	Shipping
3	De Haan	90	Executive
4	Ernst	60	IT
5	Fay	20	Marketing
6	Gietz	110	Accounting
7	Hartstein	20	Marketing
8	Higgins	110	Accounting
9	Hunold	60	IT
10	King	90	Executive
11	Kochhar	90	Executive
12	Lorentz	60	IT
13	Matos	50	Shipping
14	Mourgos	50	Shipping
15	Rajs	50	Shipping
16	Taylor	80	Sales
17	Vargas	50	Shipping
18	Whalen	10	Administration
19	Zlotkey	80	Sales

3. HR 부서에서 **Toronto**에 근무하는 사원에 대한 보고서를 요구합니다. **Toronto**에서 근무하는 모든 사원의 성, 직무, 부서 번호 및 부서 이름을 표시합니다.

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

4. 사원의 성 및 사원 번호를 해당 관리자의 성 및 관리자 번호와 함께 표시하는 보고서를 작성합니다. 열 레이블을 각각 Employee, Emp#, Manager 및 Mgr#으로 지정합니다. SQL 문을 lab_07_04.sql로 저장합니다. query를 실행합니다.

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103
6	Hartstein	201	King	100
7	Zlotkey	149	King	100
8	Mourgos	124	King	100
9	De Haan	102	King	100
10	Kochhar	101	King	100
11	Higgins	205	Kochhar	101
12	Whalen	200	Kochhar	101
13	Vargas	144	Mourgos	124
14	Matos	143	Mourgos	124
15	Davies	142	Mourgos	124
16	Rajs	141	Mourgos	124
17	Grant	178	Zlotkey	149
18	Taylor	176	Zlotkey	149
19	Abel	174	Zlotkey	149

5. King을 비롯하여 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 lab_07_04.sql을 수정합니다. 사원 번호순으로 결과를 정렬합니다. SQL 문을 lab_07_05.sql로 저장합니다. lab_07_05.sql의 query를 실행합니다.

	Employee	EMP#	Manager	Mgr#
1	King	100	(null)	(null)
2	Kochhar	101	King	100
3	De Haan	102	King	100
4	Hunold	103	De Haan	102
5	Ernst	104	Hunold	103
6	Lorentz	107	Hunold	103

...

16 Whalen	200 Kochhar	101
17 Hartstein	201 King	100
18 Fay	202 Hartstein	201
19 Higgins	205 Kochhar	101
20 Gietz	206 Higgins	205

6. HR 부서용으로 사원의 성, 부서 번호 및 지정된 사원과 동일한 부서에서 근무하는 모든 사원을 표시하는 보고서를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 이 스크립트를 lab_07_06.sql이라는 파일에 저장합니다.

	DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20 Fay	Hartstein	
2	20 Hartstein	Fay	
3	50 Davies	Matos	
4	50 Davies	Mourgos	
5	50 Davies	Rajs	
6	50 Davies	Vargas	
7	50 Matos	Davies	

...

37	90 King	De Haan	
38	90 King	Kochhar	
39	90 Kochhar	De Haan	
40	90 Kochhar	King	
41	110 Gietz	Higgins	
42	110 Higgins	Gietz	

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.

DESC JOB_GRADES		
Name	Null	Type

GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER





	LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1	King	AD_PRES	Executive	24000	E
2	Kochhar	AD_VP	Executive	17000	E
3	De Haan	AD_VP	Executive	17000	E
4	Hartstein	MK_MAN	Marketing	13000	D
5	Higgins	AC_MGR	Accounting	12008	D
6	Abel	SA_REP	Sales	11000	D
7	Zlotkey	SA_MAN	Sales	10500	D
8	Hunold	IT_PROG	IT	9000	C
9	Taylor	SA_REP	Sales	8600	C
10	Gietz	AC_ACCOUNT	Accounting	8300	C
11	Ernst	IT_PROG	IT	6000	C
12	Fay	MK_REP	Marketing	6000	C
13	Mourgos	ST_MAN	Shipping	5800	B
14	Whalen	AD_ASST	Administration	4400	B
15	Lorentz	IT_PROG	IT	4200	B
16	Rajs	ST_CLERK	Shipping	3500	B
17	Davies	ST_CLERK	Shipping	3100	B
18	Matos	ST_CLERK	Shipping	2600	A
19	Vargas	ST_CLERK	Shipping	2500	A

다른 작업을 수행하려면 다음 연습을 완료합니다.

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

	LAST_NAME	HIRE_DATE
1	Kochhar	21-SEP-05
2	Hunold	03-JAN-06
3	Ernst	21-MAY-07
4	Lorentz	07-FEB-07
5	Mourgos	16-NOV-07
6	Matos	15-MAR-06
7	Vargas	09-JUL-06
8	Zlotkey	29-JAN-08
9	Taylor	24-MAR-06
10	Grant	24-MAY-07
11	Fay	17-AUG-05

9. HR 부서에서 관리자보다 먼저 채용된 모든 사원의 이름과 채용 날짜 및 해당 관리자의 이름과 채용 날짜를 찾으려고 합니다. 이 스크립트를 lab_07_09.sql이라는 파일에 저장합니다.

	 LAST_NAME	 HIRE_DATE	 LAST_NAME_1	 HIRE_DATE_1
1	De Haan	13-JAN-01	King	17-JUN-03
2	Higgins	07-JUN-02	Kochhar	21-SEP-05
3	Whalen	17-SEP-03	Kochhar	21-SEP-05
4	Vargas	09-JUL-06	Mourgos	16-NOV-07
5	Matos	15-MAR-06	Mourgos	16-NOV-07
6	Davies	29-JAN-05	Mourgos	16-NOV-07
7	Rajs	17-OCT-03	Mourgos	16-NOV-07
8	Grant	24-MAY-07	Zlotkey	29-JAN-08
9	Taylor	24-MAR-06	Zlotkey	29-JAN-08
10	Abel	11-MAY-04	Zlotkey	29-JAN-08

해답 7-1: 조인을 사용하여 다중 테이블의 데이터 표시

1. HR 부서용으로 모든 부서의 주소를 생성하는 **query**를 작성합니다. LOCATIONS 및 COUNTRIES 테이블을 사용합니다. 출력에 위치 ID, 동/리, 구/군, 시/도 및 국가를 표시합니다. NATURAL JOIN을 사용하여 결과를 생성합니다.

```
SELECT location_id, street_address, city, state_province,
       country_name
FROM   locations
NATURAL JOIN countries;
```

2. HR 부서에서 해당 부서가 있는 모든 사원에 대해 보고서를 요구합니다. 모든 사원의 성, 부서 번호 및 부서 이름을 표시하는 **query**를 작성합니다.

```
SELECT last_name, department_id, department_name
FROM   employees
JOIN   departments
USING (department_id);
```

3. HR 부서에서 Toronto에 근무하는 사원에 대한 보고서를 요구합니다. Toronto에서 근무하는 모든 사원의 성, 직무, 부서 번호 및 부서 이름을 표시합니다.

```
SELECT e.last_name, e.job_id, e.department_id, d.department_name
FROM   employees e JOIN departments d
ON      (e.department_id = d.department_id)
JOIN   locations l
USING   (location_id)
WHERE  LOWER(l.city) = 'toronto';
```

4. 사원의 성 및 사원 번호를 해당 관리자의 성 및 관리자 번호와 함께 표시하는 보고서를 작성합니다. 열 레이블을 각각 Employee, Emp#, Manager 및 Mgr#으로 지정합니다. SQL 문을 lab_07_04.sql로 저장합니다. **query**를 실행합니다.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w JOIN employees m
ON      (w.manager_id = m.employee_id);
```

5. King을 비롯하여 해당 관리자가 지정되지 않은 모든 사원을 표시하도록 lab_07_04.sql을 수정합니다. 사원 번호순으로 결과를 정렬합니다. SQL 문을 lab_07_05.sql로 저장합니다. lab_07_05.sql의 **query**를 실행합니다.

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id  "Mgr#"
FROM   employees w
LEFT   OUTER JOIN employees m
ON      (w.manager_id = m.employee_id)
ORDER BY 2;
```

6. HR 부서용으로 사원의 성과 부서 번호 및 해당 사원과 동일한 부서에 근무하는 모든 사원을 표시하는 보고서를 작성합니다. 각 열에 적절한 레이블을 지정합니다. 이 스크립트를 lab_07_06.sql이라는 파일에 저장합니다. **query**를 실행합니다.

```
SELECT e.department_id department, e.last_name employee,
       c.last_name colleague
FROM   employees e JOIN employees c
ON      (e.department_id = c.department_id)
WHERE  e.employee_id <> c.employee_id
ORDER BY e.department_id, e.last_name, c.last_name;
```

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 **query**를 작성합니다.

```
DESC JOB_GRADES
/
SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
FROM   employees e JOIN departments d
ON      (e.department_id = d.department_id)
JOIN    job_grades j
ON      (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

다른 작업을 수행하려면 다음 연습을 완료합니다.

8. HR 부서에서 **Davies** 이후에 채용된 모든 사원의 이름을 파악하려고 합니다. 사원 **Davies** 이후로 채용된 모든 사원의 이름과 채용 날짜를 표시하기 위한 **query**를 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_07_09.sql이라는 파일에 저장합니다.

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
FROM   employees w JOIN employees m
ON      (w.manager_id = m.employee_id)
WHERE  w.hire_date < m.hire_date;
```


단원 8의 연습:
Subquery를 사용하여
Query 해결

8장

단원 8의 연습: 개요

연습 개요

이 연습에서는 다음 내용을 다룹니다.

- 알 수 없는 조건을 기반으로 값을 **query**하는 **subquery** 작성
- **Subquery**를 사용하여 한 데이터 집합에 있고 다른 데이터 집합에는 없는 값 찾기

연습 8-1: Subquery를 사용하여 Query 해결

개요

이 연습에서는 중첩된 SELECT 문을 사용하여 복합 query를 작성합니다.

연습 문제의 경우 inner query를 먼저 작성할 수 있습니다. outer query를 코딩하기 전에 inner query를 실행하여 예상되는 데이터가 생성되는지 확인하십시오.

작업

1. HR 부서에서 유저에게 사원의 성을 입력하라는 프롬프트를 표시하는 query를 요구합니다. 그런 다음 이 query는 유저가 제공한 이름을 가진 사원과 동일한 부서에서 근무하는 사원의 성과 채용 날짜를 표시합니다(해당 사원은 제외). 예를 들어, 유저가 Zlotkey를 입력하면 Zlotkey와 함께 근무하는 모든 사원을 찾습니다(Zlotkey는 제외).

	LAST_NAME	HIRE_DATE
1	Abel	11-MAY-04
2	Taylor	24-MAR-06

2. 평균 급여 이상을 받는 모든 사원의 사원 번호, 성 및 급여를 표시하는 보고서를 작성합니다. 급여의 오름차순으로 결과를 정렬합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000
2	149	Zlotkey	10500
3	174	Abel	11000
4	205	Higgins	12008
5	201	Hartstein	13000
6	101	Kochhar	17000
7	102	De Haan	17000
8	100	King	24000

3. 성에 문자 "u"가 포함된 사원과 같은 부서에 근무하는 모든 사원의 사원 번호와 성을 표시하는 **query**를 작성합니다. 작성한 **SQL** 문을 lab_08_03.sql로 저장합니다. **query**를 실행합니다.

	EMPLOYEE_ID	LAST_NAME
1	124	Mourgos
2	141	Rajs
3	142	Davies
4	143	Matos
5	144	Vargas
6	103	Hunold
7	104	Ernst
8	107	Lorentz

4. HR 부서에서 부서 위치 ID가 1700인 모든 사원의 성, 부서 ID 및 작업 ID를 표시하는 보고서를 요구합니다.

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Whalen	10	AD_ASST
2	King	90	AD_PRES
3	Kochhar	90	AD_VP
4	De Haan	90	AD_VP
5	Higgins	110	AC_MGR
6	Gietz	110	AC_ACCOUNT

유저에게 위치 ID를 입력하는 프롬프트를 표시하도록 **query**를 수정합니다.
이 **query**를 lab_08_04.sql이라는 파일에 저장합니다.

5. HR을 위해 King에게 보고하는 모든 사원의 성과 급여를 표시하는 보고서를 작성합니다.

	LAST_NAME	SALARY
1	Kochhar	17000
2	De Haan	17000
3	Mourgos	5800
4	Zlotkey	10500
5	Hartstein	13000

6. HR을 위해 **Executive** 부서의 모든 사원에 대해 부서 ID, 성 및 직무 ID를 표시하는 보고서를 작성합니다.

	DEPARTMENT_ID	LAST_NAME	JOB_ID
1	90	King	AD_PRES
2	90	Kochhar	AD_VP
3	90	De Haan	AD_VP

7. 부서 60의 사원보다 급여가 많은 모든 사원 리스트를 표시하는 보고서를 작성합니다.

	LAST_NAME
1	King
2	Kochhar
3	De Haan
4	Hartstein
5	Hunold
6	Higgins
7	Abel
8	Zlotkey
9	Taylor
10	Gietz
11	Grant
12	Fay
13	Ernst
14	Mourgos
15	Whalen

시간 여유가 있을 경우 다음 연습을 완료합니다.

8. 평균보다 많은 급여를 받고 성에 문자 "u"가 포함된 사원이 속한 부서에서 근무하는 모든 사원의 사원 번호, 성 및 급여를 표시하도록 lab_08_03.sql의 **query**를 수정합니다. lab_08_03.sql을 lab_08_08.sql로 다시 저장합니다. lab_08_08.sql의 명령문을 실행합니다.

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000

해답 8-1: Subquery를 사용하여 Query 해결

1. HR 부서에서 유저에게 사원의 성을 입력하라는 프롬프트를 표시하는 query를 요구합니다. 그런 다음 이 query는 유저가 제공한 이름을 가진 사원과 동일한 부서에서 근무하는 사원의 성과 채용 날짜를 표시합니다(해당 사원은 제외). 예를 들어, 유저가 zlotkey를 입력하면 Zlotkey와 함께 근무하는 모든 사원을 찾습니다(Zlotkey는 제외).

```
-- UNDEFINE 명령을 실행하여 변수를 제거합니다.

UNDEFINE Enter_name

-- 아래 SELECT 문을 실행하여 employees 테이블의 값을 검색합니다.

SELECT last_name, hire_date
FROM   employees
WHERE  department_id = (SELECT department_id
                        FROM   employees
                        WHERE  last_name = '&&Enter_name')
AND    last_name <> '&Enter_name';
```

주: UNDEFINE 및 SELECT는 개별 query입니다. 차례로 실행하거나, Ctrl + A + F9를 눌러 함께 실행합니다.

2. 평균 급여 이상을 받는 모든 사원의 사원 번호, 성 및 급여를 표시하는 보고서를 작성합니다. 급여의 오름차순으로 결과를 정렬합니다.

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  salary > (SELECT AVG(salary)
                FROM   employees)
ORDER BY salary;
```

3. 성에 문자 "u"가 포함된 사원과 같은 부서에 근무하는 모든 사원의 사원 번호와 성을 표시하는 query를 작성합니다. 작성한 SQL 문을 lab_08_03.sql로 저장합니다. query를 실행합니다.

```
SELECT employee_id, last_name
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   employees
                        WHERE  last_name like '%u%');
```

4. HR 부서에서 부서 위치 ID가 1700인 모든 사원의 성, 부서 ID 및 작업 ID를 표시하는 보고서를 요구합니다.

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM departments
                        WHERE location_id = 1700);
```

유저에게 위치 ID를 입력하는 프롬프트를 표시하도록 query를 수정합니다.
이 query를 lab_08_04.sql이라는 파일에 저장합니다.

```
SELECT last_name, department_id, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM departments
                        WHERE location_id =
                        &Enter_location);
```

5. HR을 위해 King에게 보고하는 모든 사원의 성과 급여를 표시하는 보고서를 작성합니다.

```
SELECT last_name, salary
FROM   employees
WHERE  manager_id = (SELECT employee_id
                    FROM   employees
                    WHERE  last_name = 'King');
```

6. HR을 위해 Executive 부서의 모든 사원에 대해 부서 ID, 성 및 직무 ID를 표시하는 보고서를 작성합니다.

```
SELECT department_id, last_name, job_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   departments
                        WHERE  department_name =
                        'Executive');
```

7. 부서 60의 사원보다 급여가 많은 모든 사원 리스트를 표시하는 보고서를 작성합니다.

```
SELECT last_name FROM employees
WHERE salary > ANY (SELECT salary
                  FROM employees
                  WHERE department_id=60);
```

시간 여유가 있을 경우 다음 연습을 완료합니다.

8. 평균보다 많은 급여를 받고 성에 문자 "u"가 포함된 사원이 속한 부서에서 근무하는 모든 사원의 사원 번호, 성 및 급여를 표시하도록 lab_08_03.sql의 **query**를 수정합니다. lab_08_03.sql을 lab_08_08.sql로 다시 저장합니다. lab_08_08.sql의 명령문을 실행합니다.

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM   employees
                        WHERE  last_name like '%u%')
AND    salary > (SELECT AVG(salary)
                FROM employees);
```


단원 9의 연습: 집합 연산자 사용

9장

단원 9의 연습: 개요

연습 개요

이 연습에서는 다음을 사용하여 보고서를 작성합니다.

- UNION 연산자
- INTERSECT 연산자
- MINUS 연산자

연습 9-1: 집합 연산자 사용

개요

이 연습에서는 UNION, INTERSECT 및 MINUS와 같은 집합 연산자를 사용하여 query를 작성합니다.

작업

1. HR 부서에서 직무 ID ST_CLERK를 포함하지 않는 부서에 대한 부서 ID 리스트를 요구합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

	DEPARTMENT_ID
1	10
2	20
3	60
4	80
5	90
6	110
7	190

2. HR 부서에서 부서가 소재하지 않는 국가의 리스트를 요구합니다. 해당 국가의 국가 ID 및 이름을 표시합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

	COUNTRY_ID	COUNTRY_NAME
1	DE	Germany

3. 부서 50 및 80에 근무하는 모든 사원의 리스트를 생성합니다. 집합 연산자를 사용하여 사원 ID, 직무 ID 및 부서 ID를 표시합니다.

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	124	ST_MAN	50
2	141	ST_CLERK	50
3	142	ST_CLERK	50
4	143	ST_CLERK	50
5	144	ST_CLERK	50
6	149	SA_MAN	80
7	174	SA_REP	80
8	176	SA_REP	80

4. 영업 담당자이며 현재 영업 부서에서 근무 중인 모든 사원의 세부 정보를 나열하는 보고서를 생성합니다.

	EMPLOYEE_ID
1	174
2	176

5. HR 부서에서 다음과 같은 사양의 보고서를 요구합니다.

- 해당 사원이 부서에 소속되었는지 여부에 관계없이 EMPLOYEES 테이블에 있는 모든 사원의 성 및 부서 ID
- 해당 부서에 근무 중인 사원이 있는지 여부에 관계없이 DEPARTMENTS 테이블에 있는 모든 부서의 부서 ID 및 부서 이름

이 보고서를 생성하는 복합 query를 작성합니다.

R2	LAST_NAME	R2	DEPARTMENT_ID	R2	DEPT_NAME
1	Abel		80	(null)	
2	Davies		50	(null)	
3	De Haan		90	(null)	
4	Ernst		60	(null)	
5	Fay		20	(null)	
6	Gietz		110	(null)	
7	Grant		(null)	(null)	
8	Hartstein		20	(null)	
9	Higgins		110	(null)	
10	Hunold		60	(null)	
11	King		90	(null)	
12	Kochhar		90	(null)	
13	Lorentz		60	(null)	
14	Matos		50	(null)	
15	Mourgos		50	(null)	
16	Rajs		50	(null)	
17	Taylor		80	(null)	
18	Vargas		50	(null)	
19	Whalen		10	(null)	
20	Zlotkey		80	(null)	
21	(null)		10	Administration	
22	(null)		20	Marketing	
23	(null)		50	Shipping	
24	(null)		60	IT	
25	(null)		80	Sales	
26	(null)		90	Executive	
27	(null)		110	Accounting	
28	(null)		190	Contracting	

해답 9-1: 집합 연산자 사용

1. HR 부서에서 직무 ID ST_CLERK를 포함하지 않는 부서에 대한 부서 ID 리스트를 요구합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

```
SELECT department_id
FROM departments
MINUS
SELECT department_id
FROM employees
WHERE job_id = 'ST_CLERK';
```

2. HR 부서에서 부서가 소재하지 않는 국가의 리스트를 요구합니다. 해당 국가의 국가 ID 및 이름을 표시합니다. 집합 연산자를 사용하여 이 보고서를 작성합니다.

```
SELECT country_id, country_name
FROM countries
MINUS
SELECT l.country_id, c.country_name
FROM locations l JOIN countries c
ON (l.country_id = c.country_id)
JOIN departments d
ON d.location_id=l.location_id;
```

3. 부서 50 및 80에 근무하는 모든 사원의 리스트를 생성합니다. 집합 연산자를 사용하여 사원 ID, 직무 ID 및 부서 ID를 표시합니다.

```
SELECT employee_id, job_id, department_id
FROM EMPLOYEES
WHERE department_id=50
UNION ALL
SELECT employee_id, job_id, department_id
FROM EMPLOYEES
WHERE department_id=80;
```

4. 영업 담당자이며 현재 영업 부서에서 근무 중인 모든 사원의 세부 정보를 나열하는 보고서를 생성합니다.

```
SELECT EMPLOYEE_ID
FROM EMPLOYEES
WHERE JOB_ID='SA_REP'
INTERSECT
SELECT EMPLOYEE_ID
FROM EMPLOYEES
WHERE DEPARTMENT_ID=80;
```

5. HR 부서에서 다음과 같은 사양의 보고서를 요구합니다.

- 해당 사원이 부서에 소속되었는지 여부에 관계없이 EMPLOYEES 테이블에 있는 모든 사원의 성 및 부서 ID.
- 해당 부서에 근무 중인 사원이 있는지 여부에 관계없이 DEPARTMENTS 테이블에 있는 모든 부서의 부서 ID 및 부서 이름.

이 보고서를 생성하는 복합 **query**를 작성합니다.

```
SELECT last_name,department_id,TO_CHAR(null)dept_name
FROM   employees
UNION
SELECT TO_CHAR(null),department_id,department_name
FROM   departments;
```

단원 10의 연습:
DML 문을 사용하여 테이블 관리
10장

단원 10의 연습: 개요

단원 개요

이 연습에서는 다음 내용을 다룹니다.

- 테이블에 행 삽입
- 테이블에서 행 갱신 및 삭제
- 트랜잭션 제어

주: 이 연습을 시작하기 전에

`/home/oracle/labs/sql1/code_ex /cleanup_scripts/cleanup_10.sql`
스크립트를 실행합니다.

연습 10-1: DML 문을 사용하여 테이블 관리

개요

HR 부서에서 사원 데이터를 삽입, 갱신 및 삭제하는 SQL 문을 작성해 달라고 합니다.

HR 부서에 명령문을 제공하기에 앞서 프로토타입으로 MY_EMPLOYEE 테이블을 사용합니다.

주

- 모든 DML 문의 경우, query를 실행하려면 **Run Script** 아이콘을 사용하거나 **F5**를 누릅니다. 이렇게 하면 **Script Output** 탭 페이지에서 피드백 메시지를 볼 수 있습니다. **SELECT query**의 경우, 계속해서 **Execute Statement** 아이콘을 사용하거나 **F9**를 누르면 **Results** 탭 페이지에서 형식이 지정된 출력을 볼 수 있습니다.
- 다음 작업을 수행하기 전에 /home/oracle/labs/sql1/code_ex/cleanup_scripts/ 에서 cleanup_10.sql 스크립트를 실행합니다.

작업

- MY_EMPLOYEE라는 테이블을 생성합니다.
- 열 이름을 식별하도록 MY_EMPLOYEE 테이블의 구조를 기술합니다.

```
DESCRIBE my_employee
Name          Null          Type
-----
ID             NOT NULL      NUMBER(4)
LAST_NAME                        VARCHAR2(25)
FIRST_NAME     VARCHAR2(25)
USERID         VARCHAR2(8)
SALARY         NUMBER(9,2)
```

- 다음 예제 데이터에서 **첫번째 데이터 행**을 MY_EMPLOYEE 테이블에 추가하는 INSERT 문을 작성합니다. INSERT 절에 열을 나열하지 마십시오. **아직 모든 행을 입력하지 마십시오.**

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

- 위의 리스트에 있는 예제 데이터의 두번째 행으로 MY_EMPLOYEE 테이블을 채웁니다. 이번에는 INSERT 절에 명시적으로 열을 나열합니다.
- 테이블에 추가한 내용을 확인합니다.

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1 Patel	Ralph	rpatel		895
2	2 Dancs	Betty	bdancs		860

- INSERT 문을 재사용 가능한 동적 스크립트 파일에 작성하여 나머지 행을 MY_EMPLOYEE 테이블에 로드합니다. 스크립트는 모든 열(ID, LAST_NAME, FIRST_NAME, USERID 및 SALARY)에 대해 프롬프트를 표시해야 합니다. 이 스크립트를 lab_10_06.sql 파일에 저장합니다.
- 작성한 스크립트에서 INSERT 문을 실행하여 3단계에 나열된 예제 데이터의 다음 두 행으로 테이블을 채웁니다.
- 테이블에 추가한 내용을 확인합니다.






	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	1 Patel	Ralph	rpatel		895
2	2 Dancs	Betty	bdancs		860
3	3 Biri	Ben	bbiri		1100
4	4 Newman	Chad	cnewman		750

- 데이터 추가 내용이 영구적으로 적용되도록 합니다.






MY_EMPLOYEE 테이블에서 데이터를 갱신하고 삭제합니다.

- 사원 3의 성을 Drexler로 변경합니다.

11. 급여가 \$900 미만인 모든 사원에 대해 급여를 \$1000로 변경합니다.
12. 테이블에 대한 변경 사항을 확인합니다.

		ID		LAST_NAME		FIRST_NAME		USERID		SALARY
1		1		Patel		Ralph		rpate1		1000
2		2		Dancs		Betty		bdancs		1000
3		3		Drexler		Ben		bbiri		1100
4		4		Newman		Chad		cnewman		1000

13. MY_EMPLOYEE 테이블에서 **Betty Dancs**를 삭제합니다.
14. 테이블에 대한 변경 사항을 확인합니다.

	 ID	 LAST_NAME	 FIRST_NAME	 USERID	 SALARY
1	1 Patel	Ralph	rpate1	1000	
2	3 Drexler	Ben	bbiri	1100	
3	4 Newman	Chad	cnewman	1000	






15. 보류 중인 모든 변경 사항을 커밋합니다.

MY_EMPLOYEE 테이블에 대한 데이터 트랜잭션을 제어합니다.






16. 6단계에서 작성한 스크립트의 명령문을 사용하여 3단계에 나열된 예제 데이터의 마지막 행으로 테이블을 채웁니다. 스크립트의 명령문을 실행합니다.

주: 한 세션에서만 17-23 단계를 수행합니다.

17. 테이블에 추가한 내용을 확인합니다.

	 ID	 LAST_NAME	 FIRST_NAME	 USERID	 SALARY
1	1	Patel	Ralph	rpate1	1000
2	3	Drexler	Ben	bbiri	1100
3	4	Newman	Chad	cnewman	1000
4	5	Ropeburn	Audrey	aropebur	1550

18. 트랜잭션 처리의 중간 지점에 표시합니다.
19. 그런 다음 MY_EMPLOYEE 테이블에서 모든 행을 삭제합니다.
20. 테이블이 비어 있는지 확인합니다.
21. 이전의 INSERT 작업을 삭제하지 않은 채로 최근의 DELETE 작업을 삭제합니다.
22. 새 행이 여전히 원래 상태를 유지하는지 확인합니다.

	 ID	 LAST_NAME	 FIRST_NAME	 USERID	 SALARY
1	1 Patel	Ralph	rpate1		1000
2	3 Drexler	Ben	bbiri		1100
3	4 Newman	Chad	cnewman		1000
4	5 Ropeburn	Audrey	aropebur		1550

23. 데이터 추가 내용이 영구적으로 적용되도록 합니다.

시간 여유가 있을 경우 다음 연습을 완료합니다.

24. 이름의 첫번째 문자와 성의 앞부분 일곱 문자를 연결하여 USERID를 자동으로 생성하도록 lab_10_06.sql 스크립트를 수정합니다. 생성된 USERID는 소문자여야 합니다. 따라서 이 스크립트는 유저에게 USERID를 입력하도록 요구하지 않아야 합니다. 이 스크립트를 lab_10_24.sql이라는 파일에 저장합니다.

25. lab_10_24.sql 스크립트를 실행하여 다음 레코드를 삽입합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

26. 새 행이 올바른 USERID로 추가되었는지 확인합니다.

	ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	6	Anthony	Mark	manthony	1230

해답 10-1: DML 문을 사용하여 테이블 관리

MY_EMPLOYEE 테이블에 데이터를 삽입합니다.

1. MY_EMPLOYEE라는 테이블을 생성합니다.

```
CREATE TABLE my_employee
(id NUMBER(4) CONSTRAINT my_employee_id_pk PRIMARY Key,
last_name VARCHAR2(25),
first_name VARCHAR2(25),
userid VARCHAR2(8),
salary NUMBER(9,2));
```

2. 열 이름을 식별하도록 MY_EMPLOYEE 테이블의 구조를 기술합니다.

```
DESCRIBE my_employee
```

3. 다음 예제 데이터에서 첫번째 데이터 행을 MY_EMPLOYEE 테이블에 추가하는 INSERT 문을 작성합니다. INSERT 절에 열을 나열하지 마십시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

```
INSERT INTO my_employee
VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

4. 위의 리스트에 있는 예제 데이터의 두번째 행으로 MY_EMPLOYEE 테이블을 채웁니다. 이번에는 INSERT 절에 명시적으로 열을 나열합니다.

```
INSERT INTO my_employee (id, last_name, first_name,
userid, salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

5. 테이블에 추가한 내용을 확인합니다.

```
SELECT *
FROM my_employee;
```

6. INSERT 문을 재사용 가능한 동적 스크립트 파일에 작성하여 나머지 행을 MY_EMPLOYEE 테이블에 로드합니다. 스크립트는 모든 열(ID, LAST_NAME, FIRST_NAME, USERID 및 SALARY)에 대해 프롬프트를 표시해야 합니다. 이 스크립트를 lab_10_06.sql이라는 파일에 저장합니다.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
        '&p_userid', &p_salary);
```

7. 작성한 스크립트에서 INSERT 문을 실행하여 3단계에 나열된 예제 데이터의 다음 두 행으로 테이블을 채웁니다.

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
        '&p_userid', &p_salary);
```

8. 테이블에 추가한 내용을 확인합니다.

```
SELECT *
FROM my_employee;
```

9. 데이터 추가 내용이 영구적으로 적용되도록 합니다.

```
COMMIT;
```

MY_EMPLOYEE 테이블에서 데이터를 갱신하고 삭제합니다.

10. 사원 3의 성을 Drexler로 변경합니다.

```
UPDATE my_employee
SET last_name = 'Drexler'
WHERE id = 3;
```

11. 급여가 \$900 미만인 모든 사원에 대해 급여를 \$1000로 변경합니다.

```
UPDATE my_employee
SET salary = 1000
WHERE salary < 900;
```

12. 테이블에 대한 변경 사항을 확인합니다.

```
SELECT *  
FROM my_employee;
```

13. MY_EMPLOYEE 테이블에서 **Betty Dancs**를 삭제합니다.

```
DELETE  
FROM my_employee  
WHERE last_name = 'Dancs';
```

14. 테이블에 대한 변경 사항을 확인합니다.

```
SELECT *  
FROM my_employee;
```

15. 보류 중인 모든 변경 사항을 커밋합니다.

```
COMMIT;
```

MY_EMPLOYEE 테이블에 대한 데이터 트랜잭션을 제어합니다.

16. 6단계에서 작성한 스크립트의 명령문을 사용하여 3단계에 나열된 예제 데이터의 마지막 행으로 테이블을 채웁니다. 스크립트의 명령문을 실행합니다.

```
INSERT INTO my_employee  
VALUES (&p_id, '&p_last_name', '&p_first_name',  
        '&p_userid', &p_salary);
```

주: 한 세션에서만 17-23 단계를 수행합니다.

17. 테이블에 추가한 내용을 확인합니다.

```
SELECT *  
FROM my_employee;
```

18. 트랜잭션 처리의 중간 지점에 표시합니다.

```
SAVEPOINT step_17;
```

19. 그런 다음 MY_EMPLOYEE 테이블에서 모든 행을 삭제합니다.

```
DELETE  
FROM my_employee;
```

20. 테이블이 비어 있는지 확인합니다.

```
SELECT *
FROM   my_employee;
```

21. 이전의 INSERT 작업을 삭제하지 않은 채로 최근의 DELETE 작업을 삭제합니다.

```
ROLLBACK TO step_17;
```

22. 새 행이 여전히 원래 상태를 유지하는지 확인합니다.

```
SELECT *
FROM   my_employee;
```

23. 데이터 추가 내용이 영구적으로 적용되도록 합니다.

```
COMMIT;
```

시간 여유가 있을 경우 다음 연습을 완료합니다.

24. 이름의 첫번째 문자와 성의 앞부분 일곱 문자를 연결하여 USERID를 자동으로 생성하도록 lab_10_06.sql 스크립트를 수정합니다. 생성된 USERID는 소문자여야 합니다. 따라서 이 스크립트는 유저에게 USERID를 입력하도록 요구하지 않아야 합니다. 이 스크립트를 lab_10_24.sql이라는 파일에 저장합니다.

```
SET ECHO OFF
SET VERIFY OFF
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       lower(substr('&p_first_name', 1, 1) ||
       substr('&p_last_name', 1, 7)), &p_salary);

SET VERIFY ON
SET ECHO ON
UNDEFINE p_first_name
UNDEFINE p_last_name
```

25. lab_10_24.sql 스크립트를 실행하여 다음 레코드를 삽입합니다.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

26. 새 행이 올바른 USERID로 추가되었는지 확인합니다.

```
SELECT *  
FROM my_employee  
WHERE ID='6';
```


단원 11의 연습: 데이터 정의어 소개

11장

단원 11의 연습: 개요

단원 개요

이 연습에서는 다음 내용을 다룹니다.

- 새 테이블 생성
- CREATE TABLE AS 구문을 사용하여 새 테이블 생성
- 테이블이 있는지 확인
- 테이블 변경
- 열 추가
- 열 삭제
- 읽기 전용 상태로 테이블 설정
- 테이블 삭제

주: 이 연습을 시작하기 전에
`/home/oracle/labs/sql1/code_ex/cleanup_scripts/cleanup_11.sql`
스크립트를 실행합니다.

연습 11-1: 데이터 정의어 소개

개요

이 연습에서는 CREATE TABLE 문을 사용하여 새 테이블을 생성합니다. 데이터베이스에 새 테이블이 추가되었는지 확인합니다. 또한 테이블의 상태를 READ ONLY로 설정한 다음 READ/WRITE로 되돌리는 방법을 배웁니다. ALTER TABLE 명령을 사용하여 테이블 열을 수정합니다.

주

- 모든 DDL 및 DML 문의 경우, SQL Developer에서 query를 실행하려면 Run Script 아이콘 또는 F5를 누릅니다. 이렇게 하면 Script Output 탭 페이지에서 피드백 메시지를 볼 수 있습니다. SELECT query의 경우 계속해서 Execute Statement 아이콘을 누르거나 F9를 누르면 Results 탭 페이지에서 형식이 지정된 출력을 볼 수 있습니다.
- 다음 작업을 수행하기 전에
/home/oracle/labs/sql1/code_ex/cleanup_scripts/cleanup_11.sql에서 cleanup_11.sql 스크립트를 실행합니다.

작업

- 다음 테이블 **instance** 차트를 기준으로 DEPT 테이블을 생성합니다. lab_11_01.sql 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	NAME
키 유형	Primary key	
Nulls/Unique		
FK 테이블		
FK 열		
데이터 유형	NUMBER	VARCHAR2
길이	7	25

```

DESCRIBE dept
Name Null      Type
-----
ID      NOT NULL  NUMBER(7)
NAME                   VARCHAR2(25)
  
```

2. 다음 테이블 **instance** 차트를 기준으로 EMP 테이블을 생성합니다. lab_11_02.sql 스크립트에 명령문을 저장하고 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	LAST_NAME	FIRST_NAME	DEPT_ID
키 유형				
Nulls/Unique				
FK 테이블				DEPT
FK 열				ID
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	NUMBER
길이	7	25	25	7

```
DESCRIBE emp
Name          Null Type
-----
ID            NUMBER(7)
LAST_NAME     VARCHAR2(25)
FIRST_NAME    VARCHAR2(25)
DEPT_ID       NUMBER(7)
```

3. EMP 테이블을 수정합니다. 전체 자릿수가 2이고 소수점 이하 자릿수가 2인 NUMBER 데이터 유형의 COMMISSION 열을 추가합니다. 수정을 확인합니다.

```
table EMP altered.
DESCRIBE emp
Name          Null Type
-----
ID            NUMBER(7)
LAST_NAME     VARCHAR2(25)
FIRST_NAME    VARCHAR2(25)
DEPT_ID       NUMBER(7)
COMMISSION    NUMBER(2,2)
```

4. 긴 사원 성을 허용하도록 EMP 테이블을 수정합니다. 수정한 내용을 확인합니다.

```
table EMP altered.
DESCRIBE emp
Name          Null Type
-----
ID            NUMBER(7)
LAST_NAME     VARCHAR2(50)
FIRST_NAME    VARCHAR2(25)
DEPT_ID       NUMBER(7)
COMMISSION    NUMBER(2,2)
```

5. EMP 테이블에서 FIRST_NAME 열을 삭제합니다. 테이블 설명을 검사하여 수정 사항을 확인합니다.

```
table EMP altered.
DESCRIBE emp
Name          Null Type
-----
ID            NUMBER(7)
LAST_NAME     VARCHAR2(50)
DEPT_ID       NUMBER(7)
COMMISSION    NUMBER(2,2)
```

6. EMP 테이블에서 DEPT_ID 열을 UNUSED로 표시합니다. 테이블 설명을 검사하여 수정 사항을 확인합니다.

```
table EMP altered.
DESCRIBE emp
Name          Null Type
-----
ID            NUMBER(7)
LAST_NAME     VARCHAR2(50)
COMMISSION    NUMBER(2,2)
```

7. EMP 테이블에서 UNUSED 열을 모두 삭제합니다.
8. EMPLOYEES 테이블의 구조를 기준으로 EMPLOYEES2 테이블을 생성합니다. EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPARTMENT_ID 열만 포함합니다. 새 테이블에서 열 이름을 각각 ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPT_ID로 지정합니다.

```
describe employees2
Name          Null    Type
-----
ID            NUMBER(6)
FIRST_NAME    VARCHAR2(20)
LAST_NAME     NOT NULL VARCHAR2(25)
SALARY        NUMBER(8,2)
DEPT_ID       NUMBER(4)
```

9. EMPLOYEES2 테이블 상태를 읽기 전용으로 변경합니다.

10. EMPLOYEES2 테이블에서 JOB_ID 열을 추가해 보십시오.

주: “Update operation not allowed on table” 오류 메시지가 나타납니다. 테이블의 상태가 읽기 전용으로 지정되어 있으므로 열을 추가할 수 없습니다.

```
Error starting at line 4 in command:
ALTER TABLE EMPLOYEES2
ADD job_id VARCHAR2(9)
Error report:
SQL Error: ORA-12081: update operation not allowed on table "ORA1"."EMPLOYEES2"
12081. 00000 - "update operation not allowed on table \"%s\".\"%s\""
*Cause:      An attempt was made to update a read-only materialized view.
*Action:     No action required. Only Oracle is allowed to update a
              read-only materialized view.
```

11. EMPLOYEES2 테이블을 읽기/쓰기 상태로 되돌립니다. 이제 동일한 열을 다시 추가해 봅니다. 테이블의 상태가 READ WRITE로 지정되어 있으므로 열을 테이블에 추가할 수 있습니다. 다음 메시지가 표시되어야 합니다.

```
table EMPLOYEES2 altered.
table EMPLOYEES2 altered.
DESCRIBE employees2
Name          Null    Type
-----
ID             NUMBER(6)
FIRST_NAME     VARCHAR2(20)
LAST_NAME      NOT NULL VARCHAR2(25)
SALARY         NUMBER(8,2)
DEPT_ID        NUMBER(4)
JOB_ID         VARCHAR2(9)
```

12. EMP, DEPT 및 EMPLOYEES2 테이블을 삭제합니다.

해답 11-1: 데이터 정의어 소개

1. 다음 테이블 **instance** 차트를 기준으로 DEPT 테이블을 생성합니다. lab_11_01.sql이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	NAME
키 유형	Primary key	
Nulls/Unique		
FK 테이블		
FK 열		
데이터 유형	NUMBER	VARCHAR2
길이	7	25

```
CREATE TABLE dept
(id NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
name VARCHAR2(25));
```

테이블이 생성되었는지 확인하고 테이블 구조를 보려면 다음 명령을 실행합니다.

```
DESCRIBE dept;
```

2. 다음 테이블 **instance** 차트를 기준으로 EMP 테이블을 생성합니다. lab_11_02.sql이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

열 이름	ID	LAST_NAME	FIRST_NAME	DEPT_ID
키 유형				
Nulls/Unique				
FK 테이블				DEPT
FK 열				ID
데이터 유형	NUMBER	VARCHAR2	VARCHAR2	NUMBER
길이	7	25	25	7

```
CREATE TABLE emp
(id NUMBER(7),
last_name VARCHAR2(25),
first_name VARCHAR2(25),
dept_id NUMBER(7)
CONSTRAINT emp_dept_id_FK REFERENCES dept (id)
);
```

테이블이 생성되었는지 확인하고 테이블 구조를 보려면 다음 명령을 실행합니다.

```
DESCRIBE emp
```

3. EMP 테이블을 수정합니다. 전체 자릿수가 2이고 소수점 이하 자릿수가 2인 NUMBER 데이터 유형의 COMMISSION 열을 추가합니다. 수정을 확인합니다.

```
ALTER TABLE emp
    ADD commission NUMBER(2,2);

DESCRIBE emp
```

4. 긴 사원 성을 허용하도록 EMP 테이블을 수정합니다. 수정한 내용을 확인합니다.

```
ALTER TABLE emp
MODIFY (last_name VARCHAR2(50));

DESCRIBE emp
```

5. EMP 테이블에서 FIRST_NAME 열을 삭제합니다. 테이블 설명을 검사하여 수정 사항을 확인합니다.

```
ALTER TABLE emp
    DROP COLUMN first_name;

DESCRIBE emp
```

6. EMP 테이블에서 DEPT_ID 열을 UNUSED로 표시합니다. 테이블 설명을 검사하여 수정 사항을 확인합니다.

```
ALTER TABLE emp
SET    UNUSED (dept_id);

DESCRIBE emp
```

7. EMP 테이블에서 UNUSED 열을 모두 삭제합니다.

```
ALTER TABLE emp
    DROP UNUSED COLUMNS;
```

8. EMPLOYEES 테이블의 구조를 기준으로 EMPLOYEES2 테이블을 생성합니다. EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPARTMENT_ID 열만 포함합니다. 새 테이블에서 열 이름을 각각 ID, FIRST_NAME, LAST_NAME, SALARY 및 DEPT_ID로 지정합니다. 테이블이 생성되었는지 확인합니다.

```
CREATE TABLE employees2 AS
  SELECT  employee_id id, first_name, last_name, salary,
          department_id dept_id
  FROM    employees;
DESCRIBE employees2
```

9. EMPLOYEES2 테이블 상태를 읽기 전용으로 변경합니다.

```
ALTER TABLE employees2 READ ONLY;
```

10. EMPLOYEES2 테이블에서 JOB_ID 열을 추가해 보십시오.

주: “Update operation not allowed on table” 오류 메시지가 나타납니다. 테이블의 상태가 읽기 전용으로 지정되어 있으므로 열을 추가할 수 없습니다.

```
ALTER TABLE employees2
ADD job_id VARCHAR2(9);
```

11. EMPLOYEES2 테이블을 읽기/쓰기 상태로 복귀시킵니다. 이제 동일한 열을 다시 추가해 봅니다.

테이블의 상태가 READ WRITE로 지정되어 있으므로 열을 테이블에 추가할 수 있습니다.

```
ALTER TABLE employees2 READ WRITE;

ALTER TABLE employees2
ADD job_id VARCHAR2(9);
DESCRIBE employees2
```

12. EMP, DEPT 및 EMPLOYEES2 테이블을 삭제합니다.

주: READ ONLY 모드인 테이블도 삭제할 수 있습니다. 이를 테스트하려면 테이블을 다시 READ ONLY 상태로 변경한 다음 DROP TABLE 명령을 실행합니다. 테이블이 삭제됩니다.

```
DROP TABLE emp;
DROP TABLE dept;
DROP TABLE employees2;
```


추가 연습과 해답

12장

단원 1의 연습

연습 개요

이러한 연습에서는 다음 항목을 기반으로 하는 추가 연습을 수행합니다.

- 기본 SQL SELECT 문
- 기본 SQL Developer 명령
- SQL 함수

연습 1-1: 추가 연습

개요

이 연습에서는 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수 등의 항목을 살펴 본 후에 수행하도록 디자인되었습니다.

작업

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

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-03	ST_CLERK	3500
2	142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-05	ST_CLERK	3100
3	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-06	ST_CLERK	2600
4	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-06	ST_CLERK	2500

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

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Abel	SA_REP	11000	0.3
2	Zlotkey	SA_MAN	10500	0.2
3	Taylor	SA_REP	8600	0.2
4	Grant	SA_REP	7000	0.15

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

	New salary
1	The salary of Whalen after a 10% raise is 4840
2	The salary of Hartstein after a 10% raise is 14300
3	The salary of Fay after a 10% raise is 6600
4	The salary of Higgins after a 10% raise is 13209
5	The salary of Gietz after a 10% raise is 9130
6	The salary of King after a 10% raise is 26400
7	The salary of Kochhar after a 10% raise is 18700
8	The salary of De Haan after a 10% raise is 18700
9	The salary of Hunold after a 10% raise is 9900
10	The salary of Ernst after a 10% raise is 6600
11	The salary of Lorentz after a 10% raise is 4620
12	The salary of Mourgous after a 10% raise is 6380
13	The salary of Rajs after a 10% raise is 3850
14	The salary of Davies after a 10% raise is 3410
15	The salary of Matos after a 10% raise is 2860
16	The salary of Vargas after a 10% raise is 2750

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

	LAST_NAME	YEARS	MONTHS
3	Higgins	11	11
4	King	10	11
5	Whalen	10	8
6	Rajs	10	7
7	Hartstein	10	3
8	Abel	10	0
9	Davies	9	4
10	Fay	8	9
11	Kochhar	8	8
12	Hunold	8	5
13	Taylor	8	2
14	Matos	8	2
15	Vargas	7	10
16	Lorentz	7	3
17	Grant	7	0
18	Ernst	7	0
19	Mourgos	6	6
20	Zlotkey	6	4

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

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos

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

	LAST_NAME	SALARY	COMMISSION
1	Whalen	4400	No
2	Hartstein	13000	No
3	Fay	6000	No
4	Higgins	12008	No
5	Gietz	8300	No
6	King	24000	No
7	Kochhar	17000	No
8	De Haan	17000	No
9	Hunold	9000	No
10	Ernst	6000	No
11	Lorentz	4200	No
12	Mourgos	5800	No
13	Rajs	3500	No
14	Davies	3100	No
15	Matos	2600	No
16	Vargas	2500	No
17	Zlotkey	10500	Yes
18	Abel	11000	Yes
19	Taylor	8600	Yes
20	Grant	7000	Yes

다음 연습은 기본 **SQL SELECT** 문, 기본 **SQL Developer** 명령, **SQL** 함수, 조인, 그룹 함수 등의 항목을 살펴본 후에 추가 연습용으로 사용할 수 있습니다.

7. 특정 위치에서 근무하는 사원의 부서 이름, 위치 ID, 성, 직책 및 급여를 표시하는 보고서를 작성합니다. 유저에게 위치를 입력하라는 프롬프트를 표시합니다. 예를 들어, 유저가 1800을 입력하면 결과는 다음과 같습니다.

	DEPARTMENT_NAME	LOCATION_ID	LAST_NAME	JOB_ID	SALARY
1	Marketing	1800	Hartstein	MK_MAN	13000
2	Marketing	1800	Fay	MK_REP	6000

8. 성이 "n"으로 끝나는 사원의 수를 알아냅니다. 가능한 두 가지 해결책을 작성합니다.

	COUNT(*)
1	3

9. 각 부서에 대한 이름, 위치 및 사원 수를 보여주는 보고서를 작성합니다. 보고서에 사원이 없는 department_ID도 포함되어 있는지 확인합니다.

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	COUNT(E.EMPLOYEE_ID)
1	80	Sales	2500	3
2	110	Accounting	1700	2
3	60	IT	1400	3
4	10	Administration	1700	1
5	90	Executive	1700	3
6	20	Marketing	1800	2
7	50	Shipping	1500	5
8	190	Contracting	1700	0

10. HR 부서에서 부서 번호 10 및 20에 있는 직책을 찾으려고 합니다. 해당 부서의 직무 ID를 표시하는 보고서를 작성합니다.

	JOB_ID
1	AD_ASST
2	MK_MAN
3	MK_REP

11. Administration 및 Executive 부서에서 찾은 직무를 표시하는 보고서를 작성합니다. 또한 해당 직무에 대한 사원 수도 표시합니다. 사원 수가 가장 많은 직무를 가장 먼저 표시합니다.

	JOB_ID	FREQUENCY
1	AD_VP	2
2	AD_PRES	1
3	AD_ASST	1

다음 연습은 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수, 조인, 그룹함수, subquery 등의 항목을 살펴본 후에 추가 연습용으로 사용할 수 있습니다.

12. 연도에 관계없이 각 월의 16일 이전에 채용된 사원을 모두 표시합니다.

	LAST_NAME	HIRE_DATE
1	De Haan	13-JAN-01
2	Hunold	03-JAN-06
3	Lorentz	07-FEB-07
4	Matos	15-MAR-06
5	Vargas	09-JUL-06
6	Abel	11-MAY-04
7	Higgins	07-JUN-02
8	Gietz	07-JUN-02

13. 모든 사원에 대해 성, 급여 및 \$1000 단위로 표현된 급여를 표시하는 보고서를 작성합니다.

	LAST_NAME	SALARY	THOUSANDS
1	King	24000	24
2	Kochhar	17000	17
3	De Haan	17000	17
4	Hunold	9000	9
5	Ernst	6000	6
6	Lorentz	4200	4
7	Mourgos	5800	5
8	Rajs	3500	3
9	Davies	3100	3
10	Matos	2600	2
11	Vargas	2500	2
12	Zlotkey	10500	10
13	Abel	11000	11
14	Taylor	8600	8
15	Grant	7000	7
16	Whalen	4400	4
17	Hartstein	13000	13
18	Fay	6000	6
19	Higgins	12008	12
20	Gietz	8300	8

14. 급여가 \$15,000 이상인 관리자 휘하의 모든 사원을 표시합니다. 사원 이름, 관리자 이름, 관리자 급여 및 관리자의 급여 등급을 표시합니다.

	LAST_NAME	MANAGER	SALARY	GRADE_LEVEL
1	Kochhar	King	24000	E
2	De Haan	King	24000	E
3	Mourgos	King	24000	E
4	Zlotkey	King	24000	E
5	Hartstein	King	24000	E
6	Whalen	Kochhar	17000	E
7	Higgins	Kochhar	17000	E
8	Hunold	De Haan	17000	E

15. 모든 부서의 부서 번호, 이름, 사원 수, 평균 급여와 각 부서에서 근무하는 사원의 이름, 급여 및 직무를 표시합니다.

	DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEES	AVG_SAL	LAST_NAME	SALARY	JOB_ID
1	10	Administration	1	4400.00	Whalen	4400	AD_ASST
2	20	Marketing	2	9500.00	Hartstein	13000	MK_MAN
3	20	Marketing	2	9500.00	Fay	6000	MK_REP
4	50	Shipping	5	3500.00	Davies	3100	ST_CLERK
5	50	Shipping	5	3500.00	Matos	2600	ST_CLERK
6	50	Shipping	5	3500.00	Rajs	3500	ST_CLERK
7	50	Shipping	5	3500.00	Mourgos	5800	ST_MAN
8	50	Shipping	5	3500.00	Vargas	2500	ST_CLERK
9	60	IT	3	6400.00	Hunold	9000	IT_PROG
10	60	IT	3	6400.00	Lorentz	4200	IT_PROG
11	60	IT	3	6400.00	Ernst	6000	IT_PROG
12	80	Sales	3	10033.33	Zlotkey	10500	SA_MAN
13	80	Sales	3	10033.33	Abel	11000	SA_REP
14	80	Sales	3	10033.33	Taylor	8600	SA_REP
15	90	Executive	3	19333.33	Kochhar	17000	AD_VP
16	90	Executive	3	19333.33	King	24000	AD_PRES
17	90	Executive	3	19333.33	De Haan	17000	AD_VP
18	110	Accounting	2	10154.00	Gietz	8300	AC_ACCOUNT
19	110	Accounting	2	10154.00	Higgins	12008	AC_MGR
20	(null)	(null)	0	No average	Grant	7000	SA_REP

16. 평균 급여가 가장 높은 부서의 부서 번호와 최저 급여를 표시하는 보고서를 작성합니다.

	DEPARTMENT_ID	MIN(SALARY)
1	90	17000

17. 영업 사원이 근무하지 않는 부서를 표시하는 보고서를 작성합니다. 출력에 부서 번호, 부서 이름, 관리자 ID 및 위치를 포함합니다.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	50	Shipping	124	1500
2	60	IT	103	1400
3	110	Accounting	205	1700
4	20	Marketing	201	1800
5	10	Administration	200	1700
6	190	Contracting	(null)	1700
7	90	Executive	100	1700

20. 사원의 채용일에 준하여 기념일 개요를 생성합니다. 기념일을 오름차순으로 정렬하십시오.

	A Z	LAST_NAME	A Z	BIRTHDAY
1		Hunold		January 03
2		De Haan		January 13
3		Davies		January 29
4		Zlotkey		January 29
5		Lorentz		February 07
6		Hartstein		February 17
7		Matos		March 15
8		Taylor		March 24
9		Abel		May 11
10		Ernst		May 21
11		Grant		May 24
12		Higgins		June 07
13		Gietz		June 07
14		King		June 17
15		Vargas		July 09
16		Fay		August 17
17		Whalen		September 17
18		Kochhar		September 21
19		Rajs		October 17
20		Mourgos		November 16

해답 1-1: 추가 연습

개요

추가 연습 1-1의 해답은 다음과 같습니다.

작업

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

```
SELECT *
FROM   employees
WHERE  job_id = 'ST_CLERK'
AND    hire_date > '31-DEC-1997';
```

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

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees
WHERE  commission_pct IS NOT NULL
ORDER BY salary DESC;
```

3. 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;
```

4. 사원 및 근속 기간에 대한 보고서를 작성합니다. 모든 사원들의 성 및 근무 기간(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) YEARS,
       TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, hire_date), 12))
       MONTHS
FROM   employees
ORDER BY years DESC, MONTHS desc;
```

5. 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');
```

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

```
SELECT last_name, salary,
       decode(commission_pct, NULL, 'No', 'Yes') commission
FROM employees;
```

다음 연습은 기본 **SQL SELECT** 문, 기본 **SQL Developer** 명령, **SQL** 함수, 조인, 그룹 함수 등의 항목을 살펴본 후에 추가 연습용으로 사용할 수 있습니다.

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

프롬프트가 표시되면 location_id에 1800을 입력합니다.

```
SELECT d.department_name, d.location_id, e.last_name, e.job_id,
       e.salary
FROM   employees e JOIN departments d
ON     e.department_id = d.department_id
AND    d.location_id = &location_id;
```

8. 성이 "n"으로 끝나는 사원의 수를 알아냅니다. 가능한 두 가지 해결책을 작성합니다.

```
SELECT COUNT(*)
FROM   employees
WHERE  last_name LIKE '%n';
--or
SELECT COUNT(*)
FROM   employees
WHERE  SUBSTR(last_name, -1) = 'n';
```

9. 각 부서에 대한 이름, 위치 및 사원 수를 보여주는 보고서를 작성합니다. 보고서에 사원이 없는 department_ID도 포함되어 있는지 확인합니다.

```
SELECT d.department_id, d.department_name,
       d.location_id, COUNT(e.employee_id)
FROM   employees e RIGHT OUTER JOIN departments d
ON     e.department_id = d.department_id
GROUP BY d.department_id, d.department_name, d.location_id;
```


10. HR 부서에서 부서 번호 10 및 20에 있는 직책을 찾으려고 합니다. 해당 부서의 직무 ID를 표시하는 보고서를 작성합니다.

```
SELECT DISTINCT job_id
FROM   employees
WHERE  department_id IN (10, 20);
```

11. Administration 및 Executive 부서에서 찾은 직무를 표시하는 보고서를 작성합니다. 또한 해당 직무에 대한 사원 수도 표시합니다. 사원 수가 가장 많은 직무를 가장 먼저 표시합니다.

```
SELECT e.job_id, count(e.job_id) FREQUENCY
FROM   employees e JOIN departments d
ON     e.department_id = d.department_id
WHERE  d.department_name IN ('Administration', 'Executive')
GROUP BY e.job_id
ORDER BY FREQUENCY DESC;
```

다음 연습은 기본 SQL SELECT 문, 기본 SQL Developer 명령, SQL 함수, 조인, 그룹 함수, subquery 등의 항목을 살펴본 후에 추가 연습용으로 사용할 수 있습니다.

12. 연도에 관계없이 각 월의 16일 이전에 채용된 사원을 모두 표시합니다.

```
SELECT last_name, hire_date
FROM   employees
WHERE  TO_CHAR(hire_date, 'DD') < 16;
```

13. 모든 사원에 대해 성, 급여 및 \$1000 단위로 표현된 급여를 표시하는 보고서를 작성합니다.

```
SELECT last_name, salary, TRUNC(salary, -3)/1000 Thousands
FROM   employees;
```

14. 급여가 \$15,000 이상인 관리자 휘하의 모든 사원을 표시합니다. 사원 이름, 관리자 이름, 관리자 급여 및 관리자의 급여 등급을 표시합니다.

```
SELECT e.last_name, m.last_name manager, m.salary,
       j.grade_level
FROM   employees e JOIN employees m
ON     e.manager_id = m.employee_id
JOIN   job_grades j
ON     m.salary BETWEEN j.lowest_sal AND j.highest_sal
AND    m.salary > 15000;
```

15. 모든 부서의 부서 번호, 이름, 사원 수, 평균 급여와 각 부서에서 근무하는 사원의 이름, 급여 및 직무를 표시합니다.

```
SELECT  d.department_id, d.department_name,
        count(e1.employee_id) employees,
        NVL(TO_CHAR(AVG(e1.salary), '99999.99'), 'No average' )
        avg_sal,
        e2.last_name, e2.salary, e2.job_id
FROM    departments d RIGHT OUTER JOIN employees e1
ON      d.department_id = e1.department_id
RIGHT OUTER JOIN  employees e2
ON      d.department_id = e2.department_id
GROUP BY d.department_id, d.department_name, e2.last_name,
        e2.salary,
        e2.job_id
ORDER BY d.department_id, employees;
```

16. 평균 급여가 가장 높은 부서의 부서 번호와 최저 급여를 표시하는 보고서를 작성합니다.

```
SELECT department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING AVG(salary) = (SELECT MAX(AVG(salary))
                     FROM    employees
                     GROUP BY department_id);
```

17. 영업 사원이 근무하지 않는 부서를 표시하는 보고서를 작성합니다. 출력에 부서 번호, 부서 이름, 관리자 ID 및 위치를 포함합니다.

```
SELECT *
FROM    departments
WHERE   department_id NOT IN(SELECT department_id
                             FROM employees
                             WHERE job_id = 'SA_REP'
                             AND department_id IS NOT NULL);
```

18. HR 부서용으로 다음과 같은 통계 보고서를 작성합니다. 다음 조건의 부서에 대한 부서 번호, 부서 이름 및 근무하는 사원 수를 포함합니다.

- a. 사원 수가 3명 미만인 부서:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON      d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) < 3;
```

- b. 사원 수가 가장 많은 부서:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON      d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                   FROM   employees
                   GROUP BY department_id);
```

- c. 사원 수가 가장 적은 부서:

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON      d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                   FROM   employees
                   GROUP BY department_id);
```

19. 모든 사원에 대해 사원 번호, 성, 급여, 부서 번호 및 해당 부서의 평균 급여를 표시하는 보고서를 작성합니다.

```
SELECT e.employee_id, e.last_name, e.department_id, e.salary,
       AVG(s.salary)
FROM   employees e JOIN employees s
ON      e.department_id = s.department_id
GROUP BY e.employee_id, e.last_name, e.department_id,
         e.salary;
```

20. 사원의 채용일을 기준으로 기념일 개요를 생성합니다. 기념일을 오름차순으로 정렬하십시오.

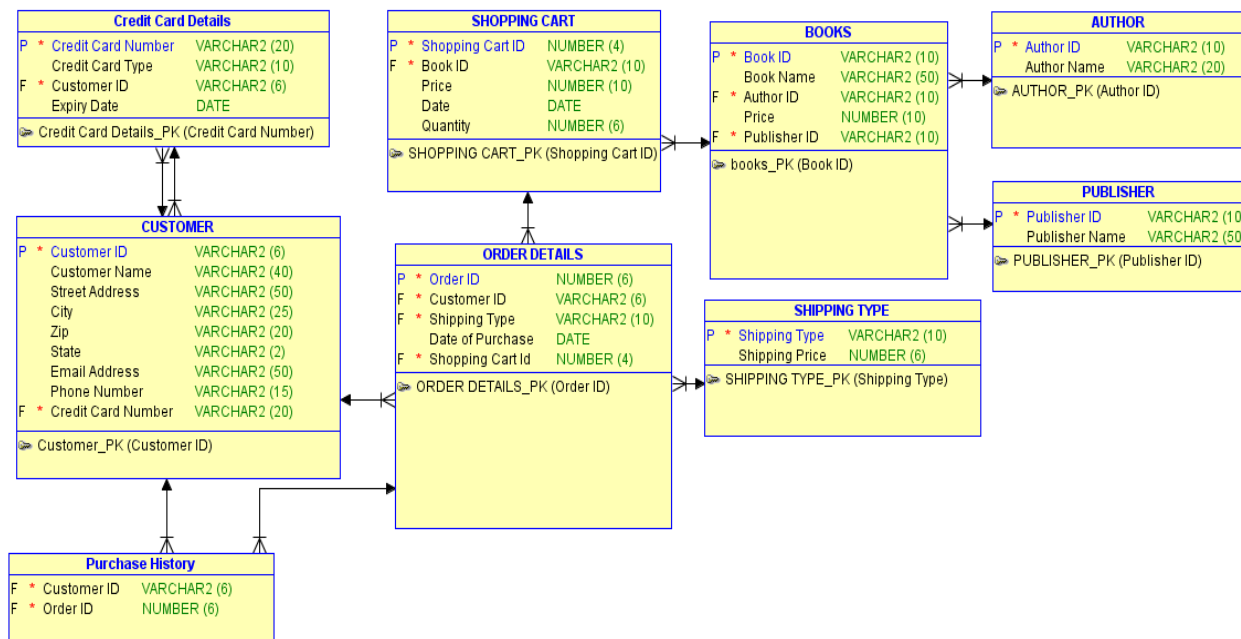
```
SELECT last_name, TO_CHAR(hire_date, 'Month DD') BIRTHDAY  
FROM employees  
ORDER BY TO_CHAR(hire_date, 'DDD');
```

사례 연구: 온라인 서점

개요

본 사례 연구에서는 온라인 서점에 대한 일련의 데이터베이스 테이블(E-Commerce 쇼핑 카트)을 작성합니다. 테이블을 생성한 후 서점 데이터베이스의 레코드를 삽입, 갱신 및 삭제하고 보고서를 생성합니다. 데이터베이스에는 필수 테이블만 포함됩니다.

다음은 온라인 서점 응용 프로그램용 테이블과 열을 나타낸 다이어그램입니다.



주: 테이블을 작성하려면 **SQL Developer**에서 `Online_Book_Store_Create_Table.sql` 스크립트의 명령을 실행합니다. 참고: 테이블을 삭제하려면 **SQL Developer**에서 `Online_Book_Store_Drop_Tables.sql` 스크립트의 명령을 실행합니다. 그런 다음 **SQL Developer**에서 `<<Online_Book_Store_Populate.sql>>` 스크립트의 명령을 실행하여 테이블을 생성하고 채울 수 있습니다.

세 가지 **SQL** 스크립트는 모두 `/home/oracle/labs/sql1/labs` 폴더에 있습니다.

- `Online_Book_Store_Create_Table.sql` 스크립트를 사용하여 테이블을 작성하는 경우 2단계부터 시작합니다.
- `Online_Book_Store_Drop_Tables.sql` 스크립트를 사용하여 테이블을 제거하는 경우 1단계부터 시작합니다.
- `Online_Book_Store_Populate.sql` 스크립트를 사용하여 테이블을 작성하고 채우는 경우 6단계부터 시작합니다.

연습 1-2

개요

이 연습에서는 다음 테이블 **instance** 차트를 기준으로 테이블을 생성합니다. 해당 데이터 유형을 선택하고 무결성 제약 조건을 추가합니다.

작업

1. 테이블 세부 정보

a. 테이블 이름: AUTHOR

열	데이터 유형	키	테이블 종속 유형
Author_ID	VARCHAR2	PK	
Author_Name	VARCHAR2		

b. 테이블 이름: BOOKS

열	데이터 유형	키	테이블 종속 On
Book_ID	VARCHAR2	PK	
Book_Name	VARCHAR2		
Author_ID	VARCHAR2	FK	AUTHORS
Price	NUMBER		
Publisher_ID	VARCHAR2	FK	PUBLISHER

c. 테이블 이름: CUSTOMER

열 이름	데이터 유형	키	테이블 종속 On
Customer_ID	VARCHAR2	PK	
Customer_Name	VARCHAR2		
Street_Address	VARCHAR2		
City	VARCHAR2		
Phone_Number	VARCHAR2		
Credit_Card_Number	VARCHAR2	FK	Credit_Card_Details

d. CREDIT_CARD_DETAILS

열 이름	데이터 유형	키	테이블 종속 On
Credit_Card_Number	VARCHAR2	PK	
Credit_Card_Type	VARCHAR2		
Expiry_Date	DATE		

e. 테이블 이름: ORDER_DETAILS

열	데이터 유형	키	테이블 종속 On
Order_ID	NUMBER	PK	
Customer_ID	VARCHAR2	FK	CUSTOMER
Shipping_Type	VARCHAR2	FK	SHIPPING_TYPE
Date_of_Purchase	DATE		
Shopping_Cart_ID	NUMBER	FK	SHOPPING_CART

f. 테이블 이름: PUBLISHER

열	데이터 유형	키	테이블 종속 유형
Publisher_ID	VARCHAR2	PK	
Publisher_Name	VARCHAR2		

g. 테이블 이름: PURCHASE_HISTORY

열	데이터 유형	키	테이블 종속 유형
Customer_ID	VARCHAR2	FK	CUSTOMER
Order_ID	NUMBER	FK	ORDER_DETAILS

h. 테이블 이름: SHIPPING_TYPE

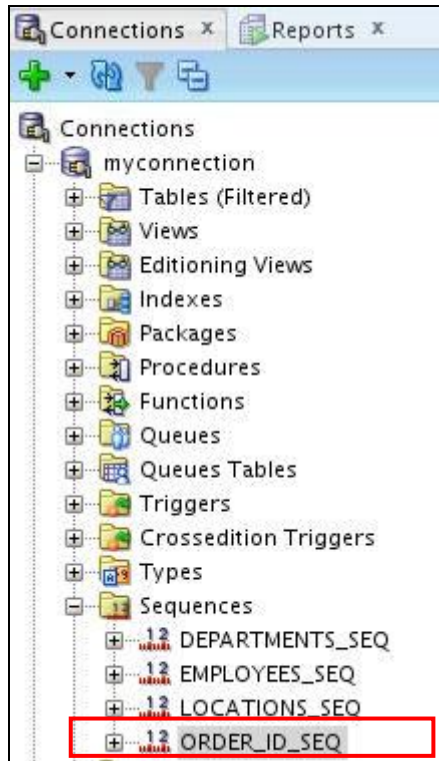
열	데이터 유형	키	테이블 종속 유형
Shipping_Type	VARCHAR2	PK	
Shipping_Price	NUMBER		

i. 테이블 이름: SHOPPING_CART

열	데이터 유형	키	테이블 종속 On
Shopping_Cart_ID	NUMBER	PK	
Book_ID	VARCHAR2	FK	BOOKS
Price	NUMBER		
날짜	DATE		
Quantity	NUMBER		

2. 생성된 테이블에 다른 참조 무결성 제약 조건을 추가합니다.
3. **SQL Developer**의 **Connections Navigator**에서 검사를 진행하여 해당 테이블이 제대로 생성되었는지 확인합니다.
4. ORDER_DETAILS 테이블에서 각 행을 고유하게 식별하기 위한 시퀀스를 생성합니다.
 - a. 100으로 시작합니다. 값 캐시를 허용하지 마십시오. 시퀀스 이름을 ORDER_ID_SEQ로 지정하십시오.

- b. SQL Developer의 Connections Navigator에 시퀀스가 존재하는지 확인합니다.



5. 테이블에 데이터를 추가합니다. 추가할 각 데이터 집합의 스크립트를 작성합니다.
다음 테이블에 데이터를 추가합니다.

- a. AUTHOR
- b. PUBLISHER
- c. SHIPPING_TYPE
- d. CUSTOMER
- e. CREDIT_CARD_DETAILS
- f. BOOKS
- g. SHOPPING_CART
- h. ORDER_DETAILS
- i. PURCHASE_HISTORY

주: 작업 번호를 사용하여 스크립트를 저장합니다. 예를 들어, BOOKS 테이블에 대해 생성된 스크립트를 저장하려면 해당 스크립트를 labs_apcs_5a_1.sql로 저장합니다. 반드시 /home/oracle/labs 폴더에 스크립트를 저장해야 합니다.

6. CUSTOMER_DETAILS라는 뷰를 생성하여 고객 이름, 고객 주소 및 고객의 세부 주문 정보를 표시합니다. 고객 ID순으로 결과를 정렬합니다.

	CUSTOMER_NAME	STREET_ADDRESS	ORDER_ID	CUSTOMER_ID	SHIPPING_TYPE	DATE_OF_PURCHASE	SHOPPING_CART_ID
1	VelasquezCarmen	283 King Street	OD0001	CN0001	USPS	12-JUN-01	SC0002
2	Ngao LaDoris	5 Modrany	OD0002	CN0002	USPS	28-JUN-05	SC0005
3	Nagayama Midori	68 Via Centrale	OD0003	CN0003	FedEx	31-JUL-05	SC0007
4	Quick-To-See Mark	6921 King Way	OD0004	CN0004	FedEx	14-AUG-06	SC0004
5	Ropeburn Audry	86 Chu Street	OD0005	CN0005	FedEx	21-SEP-06	SC0003
6	Urguhart Molly	3035 Laurier Blvd.	OD0006	CN0006	DHL	28-OCT-07	SC0001
7	Menchu Roberta	Boulevard de Waterloo 41	OD0007	CN0007	DHL	11-AUG-07	SC0006
8	Biri Ben	398 High St.	OD0008	CN0008	DHL	18-SEP-09	SC0008
9	Catchpole Antoinette	88 Alfred St.	OD0009	CN0009	USPS	25-NOV-09	SC0009

7. 테이블의 데이터를 변경합니다.
- a. 새 도서 세부 정보를 추가합니다. 도서의 저자 세부 정보가 AUTHOR 테이블에서 제공되는지 확인합니다. 저자 정보가 없으면 AUTHOR 테이블에 입력합니다.






	BOOK_ID	BOOK_NAME	AUTHOR_ID	PRICE	PUBLISHER_ID
1	BN0001	Florentine Tragedy	AN0002	150	PN0002
2	BN0002	A Vision	AN0002	100	PN0003
3	BN0003	Citizen of the World	AN0001	100	PN0001
4	BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
5	BN0005	Androcles and the Lion	AN0003	90	PN0004
6	BN0006	An Unsocial Socialist	AN0003	80	PN0004
7	BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002
8	BN0008	Beyond the Pale	AN0008	75	PN0005
9	BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
10	BN0010	Bride of Frankenstein	AN0006	200	PN0001
11	BN0011	Shelley Poetry and Prose	AN0005	150	PN0003
12	BN0012	War and Peace	AN0004	150	PN0002
13	BN0013	Two States	AN0009	150	PN0005

- b. 7(a)에 입력한 도서 세부 정보에 대한 쇼핑 카트 세부 정보를 입력합니다.

	SHOPPING_CART_ID	BOOK_ID	PRICE	SHOPPING_CART_DATE	QUANTITY
1	SC0001	BN0002	200	12-JUN-01	10
2	SC0002	BN0003	90	31-JUL-05	8
3	SC0003	BN0003	175	28-JUN-05	7
4	SC0004	BN0001	80	14-AUG-06	9
5	SC0005	BN0001	175	21-SEP-06	4
6	SC0006	BN0004	100	11-AUG-07	6
7	SC0007	BN0005	200	28-OCT-07	5
8	SC0008	BN0006	100	25-NOV-09	7
9	SC0009	BN0006	150	18-SEP-09	8
10	SC0010	BN0013	200	12-JUN-06	12

8. 각 고객의 도서 구매 내역을 포함하는 보고서를 작성합니다. 고객 이름, 고객 ID, 도서 ID, 구매 날짜 및 쇼핑 카트 ID를 포함해야 합니다. lab_apcs_8.sql이라는 스크립트 파일에서 보고서를 생성하는 명령을 저장합니다.

주: 결과는 다를 수 있습니다.

	 CUSTOMER	 CUSTOMER_ID	 SHOPPING_CART_ID	 BOOK_ID	 DATE_OF_PURCHASE
1	VelasquezCarmen	CN0001	SC0002	BN0003	12-JUN-01
2	Ngao LaDoris	CN0002	SC0005	BN0001	28-JUN-05
3	Nagayama Midori	CN0003	SC0007	BN0005	31-JUL-05
4	Quick-To-See Mark	CN0004	SC0004	BN0001	14-AUG-06
5	Ropeburn Audry	CN0005	SC0003	BN0003	21-SEP-06
6	Urguhart Molly	CN0006	SC0001	BN0002	28-OCT-07
7	Menchu Roberta	CN0007	SC0006	BN0004	11-AUG-07
8	Biri Ben	CN0008	SC0008	BN0006	18-SEP-09
9	Catchpole Antoinette	CN0009	SC0009	BN0006	25-NOV-09

해답 1-2

개요

연습 1-2의 해답은 다음과 같습니다.

작업

1. 테이블 세부 정보

a. AUTHOR

```
CREATE TABLE AUTHOR
(
    Author_ID VARCHAR2 (10) NOT NULL ,
    Author_Name VARCHAR2 (20)
)
;

COMMENT ON TABLE AUTHOR IS 'Author'
;

ALTER TABLE AUTHOR
    ADD CONSTRAINT AUTHOR_PK PRIMARY KEY (Author_ID);
```

b. BOOKS

```
CREATE TABLE BOOKS
(
    Book_ID VARCHAR2 (10) NOT NULL ,
    Book_Name VARCHAR2 (50) ,
    Author_ID VARCHAR2 (10) NOT NULL ,
    Price NUMBER (10) ,
    Publisher_ID VARCHAR2 (10) NOT NULL
)
;

COMMENT ON TABLE BOOKS IS 'Books'
;

ALTER TABLE BOOKS
    ADD CONSTRAINT books_PK PRIMARY KEY ( Book_ID );
```

c. CUSTOMER

```
CREATE TABLE CUSTOMER
(
    Customer_ID VARCHAR2 (6)  NOT NULL ,
    Customer_Name VARCHAR2 (40) ,
    Street_Address VARCHAR2 (50) ,
    City VARCHAR2 (25) ,
    Phone_Number VARCHAR2 (15) ,
    Credit_Card_Number VARCHAR2 (20)  NOT NULL
)
;

COMMENT ON TABLE CUSTOMER IS 'Customer'
;

ALTER TABLE CUSTOMER
    ADD CONSTRAINT Customer_PK PRIMARY KEY ( Customer_ID ) ;
```

d. CREDIT_CARD_DETAILS

```
CREATE TABLE CREDIT_CARD_DETAILS
(
    Credit_Card_Number VARCHAR2 (20)  NOT NULL ,
    Credit_Card_Type VARCHAR2 (10) ,
    Expiry_Date DATE
)
;

COMMENT ON TABLE CREDIT_CARD_DETAILS IS 'Credit Card Details'
;

ALTER TABLE CREDIT_CARD_DETAILS
    ADD CONSTRAINT Credit_Card_Details_PK PRIMARY KEY
( Credit_Card_Number) ;
```

e. ORDER_DETAILS

```
CREATE TABLE ORDER_DETAILS
(
  Order_ID VARCHAR2 (6)  NOT NULL ,
  Customer_ID VARCHAR2 (6)  NOT NULL ,
  Shipping_Type VARCHAR2 (10)  NOT NULL ,
  Date_of_Purchase DATE ,
  Shopping_Cart_ID varchar2(6)  NOT NULL
)
;

COMMENT ON TABLE ORDER_DETAILS IS 'Order Details'
;

ALTER TABLE ORDER_DETAILS
  ADD CONSTRAINT ORDER_DETAILS_PK PRIMARY KEY (Order_ID ) ;
```

f. PUBLISHER

```
CREATE TABLE PUBLISHER
(
  Publisher_ID VARCHAR2 (10)  NOT NULL ,
  Publisher_Name VARCHAR2 (50)
)
;

COMMENT ON TABLE PUBLISHER IS 'Publisher'
;

ALTER TABLE PUBLISHER
  ADD CONSTRAINT PUBLISHER_PK PRIMARY KEY ( Publisher_ID) ;
```

g. PURCHASE_HISTORY

```
CREATE TABLE PURCHASE_HISTORY
(
    Customer_ID VARCHAR2 (6)  NOT NULL ,
    Order_ID VARCHAR2 (6)  NOT NULL
)
;

COMMENT ON TABLE PURCHASE_HISTORY IS 'Purchase History'
;
```

h. SHIPPING_TYPE

```
CREATE TABLE SHIPPING_TYPE
(
    Shipping_Type VARCHAR2 (10)  NOT NULL ,
    Shipping_Price NUMBER (6)
)
;

COMMENT ON TABLE SHIPPING_TYPE IS 'Shipping Type'
;

ALTER TABLE SHIPPING_TYPE
    ADD CONSTRAINT SHIPPING_TYPE_PK PRIMARY KEY ( Shipping_Type
) ;
```

i. SHOPPING_CART

```

CREATE TABLE SHOPPING_CART
(
    Shopping_Cart_ID VARCHAR2 (6) NOT NULL ,
    Book_ID VARCHAR2 (10) NOT NULL ,
    Price NUMBER (10) ,
    Shopping_cart_Date DATE ,
    Quantity NUMBER (6)
)
;

COMMENT ON TABLE SHOPPING_CART IS 'Shopping Cart'
;

ALTER TABLE SHOPPING_CART
ADD CONSTRAINT SHOPPING_CART_PK PRIMARY KEY (SHOPPING_CART_ID)
;

```

2. 생성된 테이블에 다른 참조 무결성 제약 조건 추가

a. BOOKS 테이블에 **Foreign Key** 제약 조건을 포함합니다.

```

ALTER TABLE BOOKS
ADD CONSTRAINT BOOKS_AUTHOR_FK FOREIGN KEY
(
    Author_ID
)
REFERENCES AUTHOR
(
    Author_ID
)
;

ALTER TABLE BOOKS
ADD CONSTRAINT BOOKS_PUBLISHER_FK FOREIGN KEY
(
    Publisher_ID
)
REFERENCES PUBLISHER
(
    Publisher_ID
)
;

```

- b. ORDER_DETAILS 테이블에 **Foreign Key** 제약 조건을 포함합니다.

```
ALTER TABLE ORDER_DETAILS
  ADD CONSTRAINT Order_ID_FK FOREIGN KEY
  (
    Customer_ID
  )
  REFERENCES CUSTOMER
  (
    Customer_ID
  )
;

ALTER TABLE ORDER_DETAILS
  ADD CONSTRAINT FK_Order_details FOREIGN KEY
  (
    Shipping_Type
  )
  REFERENCES SHIPPING_TYPE
  (
    Shipping_Type
  )
;

ALTER TABLE ORDER_DETAILS
  ADD CONSTRAINT Order_Details_fk FOREIGN KEY
  (
    Shopping_Cart_ID
  )
  REFERENCES SHOPPING_CART
  (
    Shopping_Cart_ID
  )
;
```


- c. PURCHASE_HISTORY 테이블에 Foreign Key 제약 조건을 포함합니다.

```
ALTER TABLE PURCHASE_HISTORY
    ADD CONSTRAINT Pur_Hist_ORDER_DETAILS_FK FOREIGN KEY
    (
        Order_ID
    )
    REFERENCES ORDER_DETAILS
    (
        Order_ID
    )
;
ALTER TABLE PURCHASE_HISTORY
    ADD CONSTRAINT Purchase_History_CUSTOMER_FK FOREIGN KEY
    (
        Customer_ID
    )
    REFERENCES CUSTOMER
    (
        Customer_ID
    )
;
```

- d. SHOPPING_CART 테이블에 Foreign Key 제약 조건을 포함합니다.

```
ALTER TABLE SHOPPING_CART
    ADD CONSTRAINT SHOPPING_CART_BOOKS_FK FOREIGN KEY
    (
        Book_ID
    )
    REFERENCES BOOKS
    (
        Book_ID
    )
;
```

3. SQL Developer의 Connections Navigator에서 검사를 진행하여 해당 테이블이 제대로 생성되었는지 확인합니다. Connections Navigator에서 Connections > myconnection > Tables를 확장합니다.
4. ORDER DETAILS 테이블에서 각 행을 고유하게 식별하기 위한 시퀀스를 생성합니다.
 - a. 100으로 시작합니다. 값 캐시를 허용하지 마십시오. 시퀀스 이름을 ORDER_ID_SEQ로 지정하십시오.

```
CREATE SEQUENCE order_id_seq
START WITH 100
NOCACHE;
```

- b. SQL Developer의 Connections Navigator에 시퀀스가 존재하는지 확인합니다. Connections Navigator에서 myconnection 노드가 확장되어 있으면 Sequences를 확장합니다.

또는 다음과 같이 user_sequences 데이터 디렉터리 뷰도 query할 수 있습니다.

```
SELECT * FROM user_sequences;
```

5. 테이블에 데이터를 추가합니다.
 - a. AUTHOR 테이블

Author_ID	Author_Name
AN0001	Oliver Goldsmith
AN0002	Oscar Wilde
AN0003	George Bernard Shaw
AN0004	Leo Tolstoy
AN0005	Percy Shelley
AN0006	Lord Byron
AN0007	John Keats
AN0008	Rudyard Kipling
AN0009	P. G. Wodehouse

	AUTHOR_ID	AUTHOR_NAME
1	AN0001	Oliver Goldsmith
2	AN0002	Oscar Wilde
3	AN0003	George Bernard Shaw
4	AN0004	Leo Tolstoy
5	AN0005	Percy Shelley
6	AN0006	Lord Byron
7	AN0007	John Keats
8	AN0008	Rudyard Kipling
9	AN0009	P. G. Wodehouse

b. PUBLISHER 테이블

Publisher_ID	Publisher_Name
PN0001	Elsevier
PN0002	Penguin Group
PN0003	Pearson Education
PN0004	Cambridge University Press
PN0005	Dorling Kindersley

PUBLISHER_ID	PUBLISHER_NAME
1 PN0001	Elsevier
2 PN0002	Penguin Group
3 PN0003	Pearson Education
4 PN0004	Cambridge University Press
5 PN0005	Dorling Kindersley

c. SHIPPING _TYPE

Shipping_Type	Shipping_Price
USPS	200
FedEx	250
DHL	150

SHIPPING_TYPE	SHIPPING_PRICE
1 USPS	200
2 FedEx	250
3 DHL	150

d. CUSTOMER

Customer _ ID	Customer _Name	Street _Address	City	Phone _number	Credit _Card _Number
CN0001	VelasquezCarmen	283 King Street	Seattle	587-99-6666	000-111-222-333
CN0002	Ngao LaDoris	5 Modrany	Bratislava	586-355-8882	000-111-222-444
CN0003	Nagayama Midori	68 Via Centrale	Sao Paolo	254-852-5764	000-111-222-555
CN0004	Quick-To-See Mark	6921 King Way	Lagos	63-559-777	000-111-222-666
CN0005	Ropeburn Audry	86 Chu Street	Hong Kong	41-559-87	000-111-222-777
CN0006	Urguhart Molly	3035 Laurier Blvd.	Quebec	418-542-9988	000-111-222-888
CN0007	Menchu Roberta	Boulevard de Waterloo 41	Brussels	322-504-2228	000-111-222-999
CN0008	Biri Ben	398 High St.	Columbus	614-455-9863	000-111-222-222
CN0009	Catchpole Antoinette	88 Alfred St.	Brisbane	616-399-1411	000-111-222-111

	CUSTOMER_ID	CUSTOMER_NAME	STREET_ADDRESS	CITY	PHONE_NUMBER	CREDIT_CARD_NUMBER
1	CN0001	VelasquezCarmen	283 King Street	Seattle	587-99-6666	000-111-222-333
2	CN0002	Ngao LaDoris	5 Modrany	Bratislava	586-355-8882	000-111-222-444
3	CN0003	Nagayama Midori	68 Via Centrale	Sao Paolo	254-852-5764	000-111-222-555
4	CN0004	Quick-To-See Mark	6921 King Way	Lagos	63-559-777	000-111-222-666
5	CN0005	Ropeburn Audry	86 Chu Street	Hong Kong	41-559-87	000-111-222-777
6	CN0006	Urguhart Molly	3035 Laurier Blvd.	Quebec	418-542-9988	000-111-222-888
7	CN0007	Menchu Roberta	Boulevard de Waterloo 41	Brussels	322-504-2228	000-111-222-999
8	CN0008	Biri Ben	398 High St.	Columbus	614-455-9863	000-111-222-222
9	CN0009	Catchpole Antoinette	88 Alfred St.	Brisbane	616-399-1411	000-111-222-111

e. CREDIT_CARD_DETAILS

Credit_Card_Number	Credit_Card_Type	Expiry_Date
000-111-222-333	VISA	17-JUN-2009
000-111-222-444	MasterCard	24-SEP-2005
000-111-222-555	AMEX	11-JUL-2006
000-111-222-666	VISA	22-OCT-2008
000-111-222-777	AMEX	26-AUG-2000
000-111-222-888	MasterCard	15-MAR-2008
000-111-222-999	VISA	4-AUG-2009
000-111-222-111	Maestro	27-SEP-2001
000-111-222-222	AMEX	9-AUG-2004

R2	CREDIT_CARD_NUMBER	R2	CREDIT_CARD_TYPE	R2	EXPIRY_DATE
1	000-111-222-333		VISA		17-JUN-09
2	000-111-222-444		MasterCard		24-SEP-05
3	000-111-222-555		AMEX		11-JUL-06
4	000-111-222-666		VISA		22-OCT-08
5	000-111-222-777		AMEX		26-AUG-00
6	000-111-222-888		MasterCard		15-MAR-08
7	000-111-222-999		VISA		04-AUG-09
8	000-111-222-111		Maestro		27-SEP-01
9	000-111-222-222		AMEX		09-AUG-04

f. BOOKS

Book_ID	Book_Name	Author_ID	Price	Publisher_ID
BN0001	Florentine Tragedy	AN0002	150	PN0002
BN0002	A Vision	AN0002	100	PN0003
BN0003	Citizen of the World	AN0001	100	PN0001
BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
BN0005	Androcles and the Lion	AN0003	90	PN0004
BN0006	An Unsocial Socialist	AN0003	80	PN0004






BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002
BN0008	Beyond the Pale	AN0008	75	PN0005
BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
BN00010	Bride of Frankenstein	AN0006	200	PN0001
BN00011	Shelley's Poetry and Prose	AN0005	150	PN0003
BN00012	War and Peace	AN0004	150	PN0002

	BOOK_ID	BOOK_NAME	AUTHOR_ID	PRICE	PUBLISHER_ID
1	BN0001	Florentine Tragedy	AN0002	150	PN0002
2	BN0002	A Vision	AN0002	100	PN0003
3	BN0003	Citizen of the World	AN0001	100	PN0001
4	BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
5	BN0005	Androcles and the Lion	AN0003	90	PN0004
6	BN0006	An Unsocial Socialist	AN0003	80	PN0004
7	BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002
8	BN0008	Beyond the Pale	AN0008	75	PN0005
9	BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
10	BN0010	Bride of Frankenstein	AN0006	200	PN0001
11	BN0011	Shelley Poetry and Prose	AN0005	150	PN0003
12	BN0012	War and Peace	AN0004	150	PN0002

g. SHOPPING_CART






Shopping_Cart_ID	Book_ID	Price	Shopping_Cart_Date	Quantity
SC0001	BN0002	200	12-JUN-2001	10
SC0002	BN0003	90	31-JUL-2004	8
SC0003	BN0003	175	28-JUN-2005	7
SC0004	BN0001	80	14-AUG-2006	9
SC0005	BN0001	175	21-SEP-2006	4
SC0006	BN0004	100	11-AUG-2007	6
SC0007	BN0005	200	28-OCT-2007	5

SC0008	BN0006	100	25-NOV-2009	7
SC0009	BN0006	150	18-SPET-2009	8

	 SHOPPING_CART_ID	 BOOK_ID	 PRICE	 SHOPPING_CART_DATE	 QUANTITY
1	SC0001	BN0002	200	12-JUN-01	10
2	SC0002	BN0003	90	31-JUL-05	8
3	SC0003	BN0003	175	28-JUN-05	7
4	SC0004	BN0001	80	14-AUG-06	9
5	SC0005	BN0001	175	21-SEP-06	4
6	SC0006	BN0004	100	11-AUG-07	6
7	SC0007	BN0005	200	28-OCT-07	5
8	SC0008	BN0006	100	25-NOV-09	7
9	SC0009	BN0006	150	18-SEP-09	8

h. ORDER _DETAILS

Order _ID	Customer _ID	Shipping_ Type	Date _of _Purchase	Shopping _Cart _ID
OD0001	CN0001	USPS	12-JUN-2001	SC0002
OD0002	CN0002	USPS	28-JUN-2005	SC0005
OD0003	CN0003	FedEx	31-JUL-2004	SC0007
OD0004	CN0004	FedEx	14-AUG-2006	SC0004
OD0005	CN0005	FedEx	21-SEP-2006	SC0003
OD0006	CN0006	DHL	28-OCT-2007	SC0001
OD0007	CN0007	DHL	11-AUG-2007	SC0006
OD0008	CN0008	DHL	18-SEP-2009	SC0008
OD0009	CN0009	USPS	25-NOV-2009	SC0009

	 ORDER_ID	 CUSTOMER_ID	 SHIPPING_TYPE	 DATE_OF_PURCHASE	 SHOPPING_CART_ID
1	OD0001	CN0001	USPS	12-JUN-01	SC0002
2	OD0002	CN0002	USPS	28-JUN-05	SC0005
3	OD0003	CN0003	FedEx	31-JUL-05	SC0007
4	OD0004	CN0004	FedEx	14-AUG-06	SC0004
5	OD0005	CN0005	FedEx	21-SEP-06	SC0003
6	OD0006	CN0006	DHL	28-OCT-07	SC0001
7	OD0007	CN0007	DHL	11-AUG-07	SC0006
8	OD0008	CN0008	DHL	18-SEP-09	SC0008
9	OD0009	CN0009	USPS	25-NOV-09	SC0009

i. PURCHASE_HISTORY

Customer_ID	Order_ID
CN0001	OD0001
CN0003	OD0002
CN0004	OD0005
CN0009	OD0007

	CUSTOMER_ID	ORDER_ID
1	CN0001	OD0001
2	CN0003	OD0002
3	CN0004	OD0005
4	CN0009	OD0007

6. CUSTOMER_DETAILS라는 뷰를 생성하여 고객 이름, 고객 주소 및 고객의 세부 주문 정보를 표시합니다. 고객 ID순으로 결과를 정렬합니다.

```
CREATE VIEW customer_details AS
    SELECT  c.customer_name, c.street_address, o.order_id,
    o.customer_id, o.shipping_type, o.date_of_purchase,
    o.shopping_cart_id
    FROM    customer c JOIN order_details o
    ON      c.customer_id = o.customer_id;

SELECT  *
FROM    customer_details
ORDER BY customer_id;
```

	CUSTOMER_NAME	STREET_ADDRESS	ORDER_ID	CUSTOMER_ID	SHIPPING_TYPE	DATE_OF_PURCHASE	SHOPPING_CART_ID
1	VelasquezCarmen	283 King Street	OD0001	CN0001	USPS	12-JUN-01	SC0002
2	Ngao LaDoris	5 Modrany	OD0002	CN0002	USPS	28-JUN-05	SC0005
3	Nagayama Midori	68 Via Centrale	OD0003	CN0003	FedEx	31-JUL-05	SC0007
4	Quick-To-See Mark	6921 King Way	OD0004	CN0004	FedEx	14-AUG-06	SC0004
5	Ropeburn Audry	86 Chu Street	OD0005	CN0005	FedEx	21-SEP-06	SC0003
6	Urguhart Molly	3035 Laurier Blvd.	OD0006	CN0006	DHL	28-OCT-07	SC0001
7	Menchu Roberta	Boulevard de Waterloo 41	OD0007	CN0007	DHL	11-AUG-07	SC0006
8	Biri Ben	398 High St.	OD0008	CN0008	DHL	18-SEP-09	SC0008
9	Catchpole Antoinette	88 Alfred St.	OD0009	CN0009	USPS	25-NOV-09	SC0009

7. 테이블의 데이터를 변경합니다.

- a. 새 도서 세부 정보를 추가합니다. 도서의 저자 세부 정보가 AUTHOR 테이블에서 제공되는지 확인합니다. 저자 정보가 없으면 AUTHOR 테이블에 입력합니다.

```
INSERT INTO books(book_id, book_name, author_id, price,
publisher_id)
VALUES ('BN0013','Two States','AN0009','150','PN0005');
```

BOOK_ID	BOOK_NAME	AUTHOR_ID	PRICE	PUBLISHER_ID
1 BN0001	Florentine Tragedy	AN0002	150	PN0002
2 BN0002	A Vision	AN0002	100	PN0003
3 BN0003	Citizen of the World	AN0001	100	PN0001
4 BN0004	The Complete Poetical Works of Oliver Goldsmith	AN0001	300	PN0001
5 BN0005	Androcles and the Lion	AN0003	90	PN0004
6 BN0006	An Unsocial Socialist	AN0003	80	PN0004
7 BN0007	A Thing of Beauty is a Joy Forever	AN0007	100	PN0002
8 BN0008	Beyond the Pale	AN0008	75	PN0005
9 BN0009	The Clicking of Cuthbert	AN0009	175	PN0005
10 BN0010	Bride of Frankenstein	AN0006	200	PN0001
11 BN0011	Shelley Poetry and Prose	AN0005	150	PN0003
12 BN0012	War and Peace	AN0004	150	PN0002
13 BN0013	Two States	AN0009	150	PN0005

- b. 7(a)에 입력한 도서 세부 정보에 대한 쇼핑 카트 세부 정보를 입력합니다.

```
INSERT INTO shopping_cart(shopping_cart_id, book_id, price,
Shopping_cart_date,quantity)
VALUES ('SC0010','BN0013','200',TO_DATE('12-JUN-2006','DD-MON-
YYYY'),'12');
```

8. 각 고객의 도서 구매 내역을 포함하는 보고서를 작성합니다. 고객 이름, 고객 ID, 도서 ID, 구매 날짜 및 쇼핑 카트 ID를 포함해야 합니다. lab_apcs_8.sql이라는 스크립트 파일에서 보고서를 생성하는 명령을 저장합니다.

주: 결과는 다를 수 있습니다.

```
SELECT c.customer_name CUSTOMER, c.customer_id,
s.shopping_cart_id, s.book_id,o.date_of_purchase
FROM customer c
JOIN order_details o
ON o.customer_id=c.customer_id
JOIN shopping_cart s
ON o.shopping_cart_id=s.shopping_cart_id;
```

