**CH7. Subqueries를 사용하여 Query 해결**

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

select last\_name, hire\_date

from employees

where department\_id = (select department\_id

from employees

where last\_name = '&name')

and last\_name <> '&name'

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 "u"가 포함된 사원과 같은 부서에 근무하는 모든 사원의 사원 번호와 성을 표시하는 query를 작성합니다. 작성한 SQL 문을 lab\_07\_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 직무 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 location id를 입력하는 프롬프트를 표시하도록 query를 수정하고, 입력한 내용을 lab\_07\_04.sql이라는 파일에 저장합니다.

king 이라는 이름을 가진 직원이 2명이다

select employee\_id, job\_id

from employees

where last\_name = 'King';

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

2명 이기에 = 아닌 in 으로 복수형 받을 준비

select last\_name, salary

from employees

where manager\_id in (select employee\_id

from employees

where last\_name = 'King')

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

SELECT last\_name, salary

FROM employees

WHERE manager\_id in (SELECT employee\_id

FROM employees

WHERE last\_name = 'King')

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

select department\_id, last\_name, job\_id

from employees

where department\_id = (select department\_id

from departments

where department\_name = 'Executive')

7. 부서 60의 사원보다 급여가 많은 모든 사원의 last name 리스트를 표시하는 보고서를 작성합니다.

아래 결과(부서 60의 급여) 보다 많이 받아야한다

select salary

from employees

where department\_id = 60;

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

select last\_name, salary

from employees

where salary > any (select salary

from employees

where department\_id = 60)

8. 평균보다 많은 급여를 받고 성에 "u"가 포함된 사원이 있는 부서에서 근무하는 모든 사원의 사원 번호, 성 및 급여를 표시하도록 lab\_07\_03.sql의 query를 수정합니다. lab\_07\_03.sql을 lab\_07\_08.sql로 다시 저장합니다. lab\_07\_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)

rom employees)

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

select department\_id, min(salary), avg(salary)

from employees

group by department\_id

having avg(salary) = (select max(avg(salary))

from employees

group by department\_id)

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

select department\_id, department\_name, location\_id

from departments

where department\_id not in (select department\_id

from employees

where job\_id = 'SA\_REP'

and department\_id is not null)

1. HR 부서를 위해 통계 보고서를 작성합니다. 이 보고서에는 다음 조건의 부서에 대한 부서 번호, 부서 이름 및 근무하는 사원 수를 표시합니다.

* 사원 수가 3명 미만인 부서

select e.department\_id, d.department\_name, count(\*)

from employees e join departments d

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

group by e.department\_id, d.department\_name

having count(\*) < 3 (그룹핑에 대한 조건)

* 사원 수가 가장 많은 부서

select e.department\_id, d.department\_name, count(\*)

from employees e join departments d

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

group by e.department\_id, d.department\_name

having count(\*) = (select max(count(\*))

from employees

group by department\_id)

* 사원 수가 가장 적은 부서

select e.department\_id, d.department\_name, count(\*)

from employees e join departments d

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

group by e.department\_id, d.department\_name

having count(\*) = (select min(count(\*))

from employees

group by department\_id)

1. 사원 채용 수가 가장 많은 요일에 채용된 사원을 모두 표시(성, 요일)합니다.

select sysdate from dual (오늘 날짜 관찰 가능)

select to\_char(sysdate, ‘day’) from dual (오늘 요알 관찰 가능)

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

먼저 가장 많이 채용된 날짜를 본다

select max(count(\*))

from employees

group by to\_char(hire\_date, 'day')

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

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

from employees

where to\_char(hire\_date, 'Day') in (select to\_char(hire\_date, 'Day')

from employees

group by to\_char(hire\_date, 'Day')

having count(\*) = (select max(count(\*))

from employees

group by to\_char(hire\_date, 'day')))

**CH8. Set 연산자 사용**

1. HR 부서에서 직무 ID ST\_CLERK을 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.

국가 몇개인지 보자

select count(\*) from countries

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

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 10, 50 및 20에 대한 직무 목록을 이 순서대로 생성합니다. 집합 연산자를 사용하여 직무 ID 및 부서 ID를 표시합니다.

(union 쓰면 정렬해버림 순서대로가 안됨 그래서 union all 쓰자)

(동일한 값들이 출력되기에 중복되는 값을 안보여주기 위해 distinct)

select distinct job\_id, department\_id

from employees

where department\_id = 10

union all (답은 맞지만 union 은 정렬을 하기에 문제 에서 원하는 순서대로 뿌리지 않음)

select distinct job\_id, department\_id

from employees

where department\_id = 50

union all

select distinct job\_id, department\_id

from employees

where department\_id = 20

4. 현재 직책이 회사에 처음 입사할 때의 직책과 동일한 사원(즉, 직무가 변경된 적은 있지만 현재 원래 직무로 복귀한 사원)의 사원 ID와 직무 ID를 나열하는 보고서를 작성합니다.

5. HR 부서에서 다음과 같은 사양의 보고서를 요구합니다.

• 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

이를 수행하는 집합 연산자를 사용하는 복합 query를 작성합니다.

CH9. 데이터 조작

MY\_EMPLOYEE 테이블에 데이터를 삽입합니다.

1. lab\_09\_01.sql 스크립트의 명령문을 실행하여 이 연습에서 사용되는 MY\_EMPLOYEE 테이블을 생성합니다.

lab\_09\_01.sql

=> 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 파악합니다.

DESCRIBE my\_employee

3. 다음 예제 데이터의 첫 번째 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 |

4. 앞의 리스트에서 가져온 예제 데이터의 두 번째 행으로 MY\_EMPLOYEE 테이블을 채웁니다. 이번에는 INSERT 절에 명시적으로 열을 나열합니다.

5. 테이블에 추가한 내용을 확인합니다.

SELECT \*

FROM my\_employee;

6. 나머지 행을 MY\_EMPLOYEE 테이블에 로드하는 INSERT 문을 재사용 가능한 동적 스크립트 파일에 작성합니다. 스크립트는 모든 열(ID, LAST\_NAME, FIRST\_NAME, USERID, SALARY)에 대해 프롬프트를 표시해야 합니다.

lab\_09\_06.sql이라는 파일에 이 스크립트를 저장합니다.

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

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.

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

12. 테이블에 대한 변경 사항을 확인합니다.

SELECT \*

FROM my\_employee;

13. MY\_EMPLOYEE 테이블에서 Betty Dancs를 삭제합니다.

14. 테이블에 대한 변경 사항을 확인합니다.

SELECT \*

FROM my\_employee;

15. 보류 중인 모든 변경 사항을 커밋합니다.

COMMIT;

**MY\_EMPLOYEE테이블에 대한 데이터 트랜잭션을 제어합니다.**

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

17. 테이블에 추가한 내용을 확인합니다.

SELECT \*

FROM my\_employee;

18. 트랜잭션 처리의 중간 지점에 step\_17이라는 save point를 표시합니다.

19. MY\_EMPLOYEE 테이블에서 모들 행을 삭제합니다.

20. 테이블이 비어 있는지 확인합니다.

SELECT \*

FROM my\_employee;

21. 이전의 INSERT 작업을 삭제하지 않은 채로 가장 최근의 DELETE 작업을 취소합니다.

22. 새 행이 여전히 원래 상태를 유지하는지 확인합니다.

SELECT \*

FROM my\_employee;

23. 데이터 추가 내용이 영구적으로 적용되도록 합니다.

COMMIT;

24. 이름의 첫번째 문자와 성의 앞부분 일곱 문자를 연결하여 USERID를 자동으로 생성하도록 lab\_09\_06.sql 스크립트를 수정합니다. 생성된 USERID는 소문자여야 합니다. 따라서 이 스크립트는 유저에게 USERID를 입력하도록 요구하지 않아야 합니다. lab\_09\_24.sql이라는 파일에 이 스크립트를 저장합니다.

25. lab\_09\_24.sql 스크립트를 실행하여 다음 레코드를 삽입합니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | LAST\_NAME | FIRST\_NAME | USERID | SALARY |
| 6 | Anthony | Mark | Manthony | 1230 |

26. 새 행에 올바른 USERID가 추가되었는지 확인합니다.

SELECT \*

FROM my\_employee;

**CH10. DDL문을 사용하여 테이블 생성 및 관리**

1. 다음 테이블 instance 차트에 준하여 DEPTS 테이블을 생성합니다.

lab\_10\_01.sql이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

|  |  |  |
| --- | --- | --- |
| Column Name | ID | NAME |
| Key Type | Primary Key |  |
| Nulls/Unique |  |  |
| FK Table |  |  |
| FK Column |  |  |
| Data Type | NUMBER | VARCHAR2 |
| Length | 7 | 25 |

테이블이 생성되었는지 확인하고 테이블 구조를 보려면 다음 명령을 실행합니다.

DESCRIBE depts;

2. DEPARTMENTS 테이블의 데이터로 DEPTS 테이블을 채웁니다. 필요한 열만 포함시키시오.

3. 다음 테이블 instance 차트에 준하여 EMPS 테이블을 생성합니다.

lab\_10\_03.sql이라는 스크립트에 명령문을 저장한 다음 해당 스크립트의 명령문을 실행하여 테이블을 생성합니다. 테이블이 생성되었는지 확인합니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | ID | LAST\_NAME | FIRST\_NAME | DEPT\_ID |
| Key Type | Primary Key |  |  |  |
| Nulls/Unique |  |  |  |  |
| FK Table |  |  |  | DEPTS |
| FK Column |  |  |  | ID |
| Data Type | NUMBER | VARCHAR2 | VARCHAR2 | NUMBER |
| Length | 7 | 25 | 25 | 7 |

테이블이 생성되었는지 확인하고 테이블 구조를 보려면 다음 명령을 실행합니다.

DESCRIBE emps

4. EMPLOYEES 테이블의 구조에 준하여 EMPLOYEES2 테이블을 생성합니다.

EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY 및 DEPARTMENT\_ID 열만 포함시킵니다. 새 테이블의 열 이름을 각각 ID, FIRST\_NAME, LAST\_NAME, SALARY 및 DEPT\_ID로 지정합니다.

5. EMPLOYEES2 테이블 상태를 읽기 전용으로 변경합니다.

6. EMPLOYEES2 테이블에 다음 행을 삽입해 봅니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | FIRST\_NAME | LAST\_NAME | SALARY | DEPT\_ID |
| 34 | Grant | Marcie | 5678 | 10 |

"Update operation not allowed on table"이라는 오류 메시지가 나타납니다. 테이블의 상태가 읽기 전용으로 지정되어 있으므로 행을 추가할 수 없습니다.

7. EMPLOYEES2 테이블을 읽기/쓰기 상태로 복귀시키고, 동일한 행을 다시 삽입해 봅니다. 테이블의 상태가 READ WRITE로 지정되어 있으므로 행을 삽입할 수 있습니다.

8. EMPLOYEES2 테이블을 삭제합니다.

DROP TABLE employees2;

**CH11. 기타 스키마 객체 생성**

1. HR 부서의 임원이 EMPLOYEES 테이블의 일부 데이터를 숨길 것을 요구합니다. EMPLOYEES 테이블의 사원 번호, 사원 성 및 부서 번호를 기반으로 하는 EMPLOYEES\_VU라는 뷰를 생성합니다. 사원 이름의 머리글은 EMPLOYEE여야 합니다.

2. 뷰가 작동하는지 확인합니다. EMPLOYEES\_VU 뷰의 내용을 표시합니다.

SELECT \*

FROM employees\_vu;

3. EMPLOYEES\_VU 뷰를 사용하여 HR 부서를 위해 모든 사원 이름과 부서 번호를

표시하는 query를 작성합니다.

4. 부서 50의 사원 데이터에 액세스하려고 합니다. 부서 50의 모든 사원에 대해 사원 번호, 사원 last name 및 부서 번호를 포함하는 DEPT50이라는 뷰를 생성합니다. 뷰 의 레이블을 각각 EMPNO, EMPLOYEE 및 DEPTNO로 지정하라는 요청을 받았습니다. 보안상의 이유로 사원이 뷰를 통해 다른 부서에 재배치되는 것을 허용하지 않습니다.

5. DEPT50 뷰의 구조 및 내용을 표시합니다.

DESCRIBE dept50

SELECT \*

FROM dept50;

6. 뷰를 테스트합니다. Matos를 부서 80에 재배치해 보시오.

DEPT50 뷰가 WITH CHECK OPTION 제약 조건을 사용하여 생성되었으므로 오류가 발생합니다. 이 제약 조건은 뷰의 DEPTNO 열이 변경되지 않도록 합니다.

7. DEPT 테이블의 primary key 열로 사용할 수 있는 시퀀스가 필요합니다. 시퀀스는 200부터 시작하고 최대값은 1,000이어야 합니다. 시퀀스 증가분 값을 10으로 설정하고 시퀀스 이름을 DEPT\_ID\_SEQ로 지정합니다.

8. 시퀀스를 테스트하기 위해 DEPTS 테이블의 두 행을 삽입하는 스크립트를 작성합니다. 스크립트 이름을 lab\_11\_08.sql로 지정합니다. ID 열에 대해 생성한 시퀀스를 사용해야 합니다. Education 및 Administration이라는 두 개의 부서를 추가합니다. 추가한 내용을 확인합니다. 스크립트의 명령을 실행합니다.

INSERT INTO depts

VALUES (depts\_id\_seq.nextval, 'Education');

INSERT INTO depts

VALUES (depts\_id\_seq.nextval, 'Administration');

9. DEPTS 테이블의 NAME 열에 non-unique 인덱스를 생성합니다.

10. EMPLOYEES 테이블의 동의어를 생성하고 이름을 EMP로 지정합니다.

1. 다음 테이블 instance 차트에 준하여 테이블을 생성합니다. 해당 데이터 유형을 선택하고 무결성 제약 조건을 추가합니다.

* MEMBER

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Column name | MEMBER\_ID | LAST\_NAME | FIRST\_NAME | ADDRESS | CITY | PHONE | JOIN\_DATE |
| 키 유형 | PK |  |  |  |  |  |  |
| Null/Unique | NN, U | NN |  |  |  |  | NN |
| 기본 값 |  |  |  |  |  |  | 시스템날짜 |
| 데이터 유형 | NUMBER | VARCHAR2 | VARCHAR2 | VARCHAR2 | VARCHAR2 | VARCHAR2 | DATE |
| 길이 | 10 | 25 | 25 | 100 | 30 | 15 |  |

* TITLE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column name | TITLE\_ID | TITLE | DESCRIPTION | RATING | CATEGORY | RELEASE\_DATE |
| 키 유형 | PK |  |  |  |  |  |
| Null/Unique | NN, U | NN | NN |  |  |  |
| 확인(Check) |  |  |  | G, PG, R, NC17, NR | DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY |  |
| 데이터 유형 | NUMBER | VARCHAR2 | VARCHAR2 | VARCHAR2 | VARCHAR2 | DATE |
| 길이 | 10 | 60 | 400 | 4 | 20 |  |

* TITLE\_COPY

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | COPY\_ID | TITLE\_ID | STATUS |
| 키 유형 | PK | PK, FK |  |
| Null/Unique | NN, U | NN, U | NN |
| 확인(check) |  |  | AVAILABLE, DESTROYED, RENTED, RESERVED |
| FK Ref 테이블 |  | TITLE |  |
| FK Ref Col |  | TITLE\_ID |  |
| 데이터 유형 | NUMBER | NUMBER | VARCHAR@ |
| 길이 | 10 | 10 | 15 |

* RENTAL

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column name | BOOK\_DATE | MEMBER\_ID | COPY\_ID | ACT\_RET\_DATE | EXP\_RET\_DATE | TITLE\_ID |
| 키 유형 | PK | PK, FK1 | PK, FK2 |  |  | PK, FK2 |
| Null/Unique | 시스템날짜 |  |  |  | 시스템날짜 + 2일 |  |
| FK Ref 테이블 |  | MEMBER | TITLE\_COPY |  |  | TITLE\_COPY |
| FK Ref Col |  | MEMBER\_ID | COPY\_ID |  |  | TITLE\_ID |
| 데이터 유형 | DATE | NUMBER | NUMBER | DATE | DATE | NUMBER |
| 길이 |  | 10 | 10 |  |  | 10 |

;

* RESERVATION

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | RES\_DATE | MEMBER\_ID | TITLE\_ID |
| 키 유형 | PK | PK, FK1 | PK, FK2 |
| Null/Unique | NN, U | NN, U | NN |
| FK Ref 테이블 |  | MEMBER | TITLE |
| FK Ref Col |  | MEMBER\_ID | TITLE\_ID |
| 데이터 유형 | DATE | NUMBER | NUMBER |
| 길이 |  | 10 | 10 |

1. MEMBER 테이블 및 TITLE 테이블의 각 행을 고유하게 식별하기 위한 시퀀스를 생성합니다.

* MEMBER 테이블의 멤버 번호: 101부터 시작하고 값을 캐시에 저장할 수 없습니다. 파일 이름을 시퀀스 MEMBER\_ID\_SEQ로 지정하십시오.

CREATE SEQUENCE member\_id\_seq

START WITH 101

NOCACHE;

* TITLE 테이블의 제목 번호: 92부터 시작하고 값을 캐시에 저장할 수 없습니다. 파일 이름을 시퀀스 TITLE\_ID\_SEQ로 지정하십시오.

CREATE SEQUENCE title\_id\_seq

START WITH 92

NOCACHE;

1. 테이블에 데이터를 추가합니다. 추가할 각 데이터 집합에 대해 스크립트를 작성합니다.

* TITLE 테이블에 영화 제목을 추가합니다. 영화 정보를 입력하기 위한 스크립트를 작성합니다. lab\_11\_13a.sql이라는 스크립트에 해당 명령문을 저장합니다. 시퀀스를 사용하여 각 영화 제목을 고유하게 식별합니다. 출시 날짜를 DD-MM-YYYY 형식으로 입력합니다. 문자 필드의 작은 따옴표는 특수하게 처리해야 한다는 점을 기억하십시오. 추가한 내용을 확인합니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Description | Rating | Category | Release date |
| Willie and Christmas Too | All of Willie’s friends make a Christmas list for Santa, but Willie has yet to add his own wish list. | G | CHILD | 05-oct-1995 |
| Alien Again | Yet another installment of science fiction history. Can the heroine save the planet from the alien life form? | R | SCIFI | 19-MAY-1995 |
| The Glob | A meteor crashes near a small American town and unleashes carnivorous goo in this classic. | NR | SCIFI | 12-AUG-1995 |
| My Day Off | With a little luck and a lot ingenuity, a teenager skips school for a day in New York. | PG | COMEDY | 12-JUL-1995 |
| Miracles on Ice | A six-year-old has doubts about Santa Claus, but she discovers that miracles really do exist. | PG | DRAMA | 12-SEP-1995 |
| Soda Gang | After discovering a cache of drugs, a young couple find themselves pitted against a vicious gang. | NR | ACTION | 01-JUN-1995 |

COMMIT

/

SELECT title

FROM title;

* MEMBER 테이블에 데이터를 추가합니다. lab\_11\_13b.sql이라는 스크립트에 insert 문을 저장합니다. 스크립트의 명령을 실행합니다. 시퀀스를 사용하여 멤버 번호를 추가하십시오. (MON => MM으로)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| First\_Name | Last\_Name | Address | City | Phone | Join\_Date |
| Carmen | Velasquez | 283 King Street | Seattle | 206-899-6666 | 08-MAR-1990 |
| LaDoris | Ngao | 5 Modrany | Bratislava | 586-355-8882 | 08-MAR-1990 |
| Midori | Nagayama | 68 King Way | Sao Paolo | 254-852-5764 | 17-JUN-1991 |
| Mark | Quick-to-See | 6921 King Way | Lagos | 63-559-7777 | 07-APR-1990 |
| Audry | Ropeburn | 86 Chu Street | Hong Kong | 41-559-87 | 18-JAN-1991 |
| Molly | Urguhart | 3035 Laurier | Quebec | 418-542-9988 | 18-JAN-1991 |

COMMIT

* 다음 영화 제목을 TITLE\_COPY 테이블에 추가합니다.

참고**:** 이 연습에 사용할 수 있는 TITLE\_ID 번호가 있어야 합니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Copy\_Id | Status | Title | Copy\_Id |
| Willie and Christmas Too | 1 | Available | Willie and Christmas Too | 1 |
| Alien Again | 1 | Available |  | 1 |
|  | 2 | Rented | Alien Again | 2 |
| The Glob | 1 | Available |  | 1 |
| My Day Off | 1 | Available | The Glob | 1 |
|  | 2 | Available | My Day Off | 2 |
|  | 3 | Rented |  | 3 |
| Miracles on Ice | 1 | Available | Miracles on Ice | 1 |
| Soda Gang | 1 | Available | Soda Gang | 1 |

* RENTAL 테이블에 다음 대여 비디오를 추가합니다.

참고**:** 제목 번호는 시퀀스에 따라 다를 수 있습니다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title\_Id | Copy\_Id | Member\_Id | Book\_Date | Exp\_Ret\_Date |
| 92 | 1 | 101 | 3 days ago | 1 day ago |
| 93 | 2 | 101 | 1 day ago | 1 day from now |
| 95 | 3 | 102 | 2 days ago | Today |
| 97 | 1 | 106 | 4 days ago | 2 days ago |

COMMIT

/

1. 영화 제목, 각 복사본의 대여 가능성 및 대여된 경우 예상 반납 날짜를 표시하는 TITLE\_AVAIL이라는 뷰를 생성합니다. 뷰에서 모든 행을 질의합니다. 제목에 따라 결과를 정렬합니다.

SELECT \*

FROM title\_avail

ORDER BY title, copy\_id;

1. 테이블의 데이터를 변경합니다

* 새 제목을 추가합니다. 영화는 공상 과학 영화로 분류된 PG 등급의 "Interstellar Wars"입니다. 출시 날짜는 07-JUL-77입니다. 설명은 "Futuristic interstellar action movie. Can the rebels save the humans from the evil empire?"입니다. 두 사본에 대해 영화 사본 기록을 추가해야 합니다.
* 두 건의 예약을 입력합니다. 하나는 "Interstellar Wars"를 대여하려는 Carmen Velasquez 고객이고 다른 하나는 "Soda Gang"을 대여하려는 Mark Quick-to-See 고객입니다.

1. 테이블 중 하나를 수정합니다.

* TITLE 테이블에 비디오 구입 가격을 기록하는 PRICE 열을 추가합니다. 수정 사항을 확인합니다.

ALTER TABLE title

ADD (price NUMBER(8,2));

DESCRIBE title

* 리스트에 따른 가격으로 각 비디오를 갱신하는 update 문이 포함된 스크립트를 작성합니다. 스크립트에서 명령을 실행합니다.

참고**:** 이 연습에 사용할 수 있는 TITLE\_ID 번호가 있어야 합니다.

|  |  |
| --- | --- |
| Title | Price |
| Willie and Christmas Too | 25 |
| Alien Again | 35 |
| The Glob | 35 |
| My Day Off | 35 |
| Miracles on Ice | 30 |
| Soda Gang | 35 |
| Interstellar Wars | 29 |

UPDATE title

SET price = &price

WHERE title\_id = &title\_id;

1. 각 고객의 비디오 대여 내역을 포함하는 보고서를 작성합니다. 고객 이름, 대여한 영화, 대여 날짜 및 대여 기간을 포함해야 합니다. Reporting 기간 동안 모든 고객이 대여한 비디오 수를 합산합니다. lab\_11\_17.sql이라는 스크립트 파일에 보고서를 생성하는 명령을 저장합니다.