

DATABASE HOMEWORK #3

제출일 : 2019.11.00 ~ 2019. 11.17. 23:00



과제 Index_

과제01. HW2_01

과제02. HW2_02

과제03. HW2_03

과제04. Load csv file to get started

과제05. HW2_2 JAVA JDBC DBM

과제06. 소감

과제0. Code (hwp 파일로 제출합니다!)

사용 버전

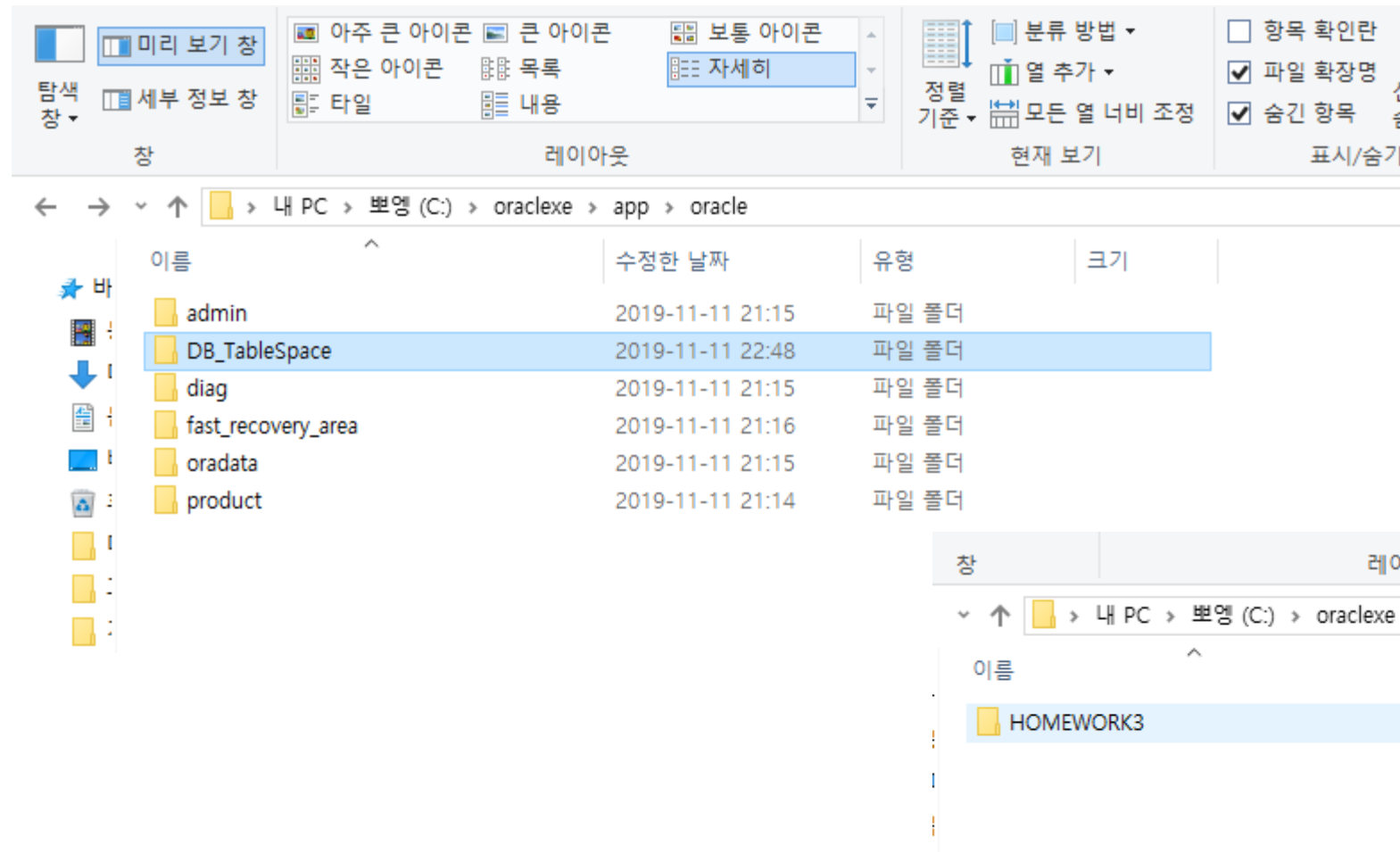
Erwin : Erwin Academic

Oracle : oracle database 11g express edition

이름 : 구명회
학과 : 컴퓨터공학과
학번 : 20154215
분반 : 01분반

과제01-1. CREATE TABLE SPACE

우선 TABLE SPACE를 생성하기위해 ORACLE이 설치된 폴더에서 TABLE SPACE 공간을 위한 디렉터리를 생성했습니다.



과제01-2. 관리자 계정 접속
과제01-3. 테이블 스페이스 생성

conn sys as sysdba

TABLE SPACE를 생성하기 위해서 관리자 계정에 접속하고 접속된것을 확인합니다.

```
SQL>
SQL> CONN SYS AS SYSDBA
Enter password:
Connected.
SQL>
SQL> SHOW USER
USER is "SYS"
SQL>
SQL>
```

```
SQL> Create Tablespace Homework3
2 Datafile 'C:\Woraclexe\Wapp\Woracle\WDB_TableSpace\WHomework3\WHomework03_data01.dbf' Size 10M;

Tablespace created.

SQL> Select tablespace_name From dba_data_files;

TABLESPACE_NAME
-----
USERS
SYSAUX
UNDOTBS1
SYSTEM
HOMEWORK3
```

Create Tablespace Homework3 Datafile 'C:\Woraclexe\Wapp\Woracle\WDB_TableSpace\WHomework3\WHomework03_data01.dbf' Size 10M;

Select tablespace_name From dba_data_files;

1에서 생성한 폴더를 지정해서 TABLE SPACE를 생성하고 TABLE SPACE가 생성된 것을 확인했습니다.

과제01-4. 사용자 계정 생성

```
SQL> Create user MHHW3  
2 Identified by 1234  
3 Default Tablespace Homework3  
4 Quota Unlimited on Homework3;
```

User created.

```
SQL> Grant Connect, Resource to MHHW3;
```

Grant succeeded.

```
SQL> CONNECT MHHW3/1234
```

Connected.

```
SQL>
```

```
SQL> SHOW USER
```

```
USER is "MHHW3"
```

해당 테이블 스페이스를 사용할 사용자 계정을 생성하고,

Create user MHHW3

Identified by 1234

Default Tablespace Homework3

Quota Unlimited on Homework3;

생성한 사용자 계정에 권한을 부여했습니다.

Grant Connect, Resource to MHHW3;

사용자 계정에 접속한 후 현재 사용자를 확인했습니다.

CONNECT MHHW3/1234

SHOW USER;

과제01-5. EMP 테이블 생성
과제01-6. DEPT 테이블 생성

※ 새로 만든 MHHW3 계정에서 EMP테이블과 DEPT테이블을 생성했습니다.
과제 01-5 ~ 과제01-8 RUN SQL 수행
과제 01-9 ~ 과제01-12 get started에서 수행

```
SQL> CREATE TABLE EMP(  
2 EMPNO NUMERIC(5),  
3 ENAME CHAR(10),  
4 EPHONE CHAR(15),  
5 JOB CHAR(6),  
6 MGR NUMERIC(5),  
7 HIREDATE DATE,  
8 SAL NUMERIC(5),  
9 DEPTNO NUMERIC(5),  
10 PRIMARY KEY(EMPNO),  
11 FOREIGN KEY(DEPTNO) REFERENCES DEPT  
12 );
```

Table created.

```
SQL>  
SQL> INSERT INTO EMP VALUES(1001, '강세아', '010-2033-1131', '사원', 1010, '20190713', 200, 10);  
1 row created.  
SQL> INSERT INTO EMP VALUES(1002, '강혜진', '010-3452-1132', '부장', 1020, '20051220', 500, 20);  
1 row created.
```

```
SQL> CREATE TABLE DEPT(  
2 DEPTNO NUMERIC(5),  
3 DNAME CHAR(10),  
4 LOC CHAR(10),  
5 PRIMARY KEY(DEPTNO)  
6 );
```

Table created.

```
SQL> INSERT INTO DEPT VALUES(10, '인사부', '광천동');  
1 row created.  
SQL> INSERT INTO DEPT VALUES(20, '영업부', '운남동');  
1 row created.
```

EMP 테이블과 DEPT 테이블을 생성하고 각각 데이터를 삽입했습니다.

과제01-7. DEPT, EMP 테이블 확인

```
SQL> SELECT * FROM DEPT;
```

DEPTNO	DNAME	LOC
10	인사부	관천
20	영업부	관천
30	경영부	관천
40	생산부	관천
50	개발부	관천
60	총무부	수원
70	기술부	수원

7 rows selected.

```
SQL>
```

```
SQL> SELECT * FROM EMP;
```

EMPNO	ENAME	EPHONE	JOB	MGR	HIREDATE	SAL	DEPTNO
1001	강세아	010-2033-1131	사원	1010	19/07/13	200	10
1002	강혜진	010-3452-1132	부장	1020	05/12/20	500	20
1003	구영희	010-5724-1133	대리	1004	16/10/13	250	20
1004	김영훈	010-8568-1134	사원	1003	19/05/11	200	10
1005	김영빈	010-1718-1135	과장	1008	08/09/18	350	70
1006	김향진	010-6794-1136	대리	1019	16/11/21	250	20
1007	박경재	010-7534-1137	과장	1017	08/01/03	350	30
1008	박준서	010-5479-1138	사원	1005	19/03/11	200	10
1009	박나인	010-6322-1139	사원	1018	18/12/04	200	20
1010	신나라	010-2954-1140	대리	1001	14/03/21	250	60
1011	송주영	010-1907-1141	사원	1016	19/10/27	200	20
1012	이동훈	010-1044-1142	사원	1013	19/09/02	200	30
1013	이지현	010-9673-1143	대리	1012	16/02/08	250	40
1014	이인호	010-8724-1144	사장		00/10/29	800	20
1015	정밀한	010-9951-1145	사원	1021	19/01/06	200	10
1016	조재희	010-5531-1146	과장	1011	09/09/10	350	60
1017	조건우	010-3865-1147	사원	1007	19/06/18	200	20
1018	지승재	010-1211-1148	부장	1009	03/01/16	500	70
1019	차대형	010-4632-1149	사원	1006	19/02/13	200	20
1020	하동기	010-5688-1150	사원	1002	18/11/05	200	10
1021	허재영	010-5688-1150	대리	1015	15/04/25	250	60

21 rows selected.

SELECT * FROM EMP;

SELECT * FROM DEPT;

EMP 테이블과 DEPT 테이블의

모든 레코드를 출력합니다.

과제01-8. 쿼리문 수행

```
SQL>  
SQL> SELECT DEPTNO, DNAME FROM DEPT;
```

DEPTNO	DNAME
10	인사부
20	영업부
30	경리부
40	생산부
50	개발부
60	총무부
70	기술부

7 rows selected.

SELECT DEPTNO, DNAME FROM DEPT;

부서 테이블에서 부서번호와 부서이름만 출력했습니다.

```
SQL> SELECT DEPTNO 부서번호, DNAME AS 부서명 FROM DEPT;
```

부서번호	부서명
10	인사부
20	영업부
30	경리부
40	생산부
50	개발부
60	총무부
70	기술부

7 rows selected.

SELECT DEPTNO 부서번호, DNAME AS 부서명 FROM DEPT;

부서 테이블에서 부서번호와 부서이름에 별명을 붙여 출력했습니다.

과제01-8. 쿼리문 수행

```
SQL> SELECT DISTINCT JOB FROM EMP;
```

```
JOB
```

```
-----  
과장  
대리  
사장  
부장  
사원
```

```
SELECT DISTINCT JOB FROM EMP;
```

EMP 테이블의 JOB 필드 중에서 중복된 필드를 제거하고 출력했습니다.

```
SQL> SELECT EMPNO, ENAME, SAL FROM EMP WHERE ENAME='구멍회';
```

```
EMPNO ENAME          SAL  
-----  
1003 구멍회          250
```

```
SELECT EMPNO, ENAME, SAL FROM EMP WHERE ENAME='구멍회';
```

EMP 테이블에서 '구멍회'의 사원번호, 사원명, 급여를 출력했습니다.

과제01-8. 쿼리문 수행

```
SQL> SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL=200 OR SAL=350 OR SAL=800;
```

EMPNO	SAL
1001	강세아
1004	김영훈
1005	김영빈
1007	박정재
1008	박준서
1009	박나인
1011	송주홍
1012	이웅호
1014	이인호
1015	정밀한
1016	조재혁
1017	조건우
1019	차대형
1020	하동기

14 rows selected.

```
SQL> SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL IN(200, 350, 800);
```

EMPNO	ENAME	SAL
1001	강세아	200
1004	김영훈	200
1005	김영빈	350
1007	박정재	350
1008	박준서	200
1009	박나인	200
1011	송주홍	200
1012	이웅호	200
1014	이인호	800
1015	정밀한	200
1016	조재혁	350
1017	조건우	200
1019	차대형	200
1020	하동기	200

14 rows selected.

```
SELECT EMPNO, ENAME, SAL  
FROM EMP  
WHERE SAL=200 OR SAL=350 OR SAL=800;
```

EMP 테이블에서 급여가 200이거나 350이거나 800인 직원들의
직원번호와 직원이름과 급여를 출력했습니다.

```
SELECT EMPNO, ENAME, SAL  
FROM EMP  
WHERE SAL IN(200, 350, 800);
```

IN 명령어를 사용해서

위의 쿼리문과 같은 결과를 출력할 수 있습니다.

과제01-8. 쿼리문 수행

```
SQL> SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL <> 250 AND SAL <>300 AND SAL <> 800;
```

EMPNO	ENAME	SAL
1001	강세아	200
1002	강혜진	500
1004	김영훈	200
1005	김영빈	350
1007	박경재	350
1008	박준서	200
1009	박나은	200
1011	송주훈	200
1012	이동훈	200
1015	정말한	200
1016	조재현	350
1017	조건우	200
1018	지승재	500
1019	차대령	200
1020	하동기	200

15 rows selected.

```
SELECT EMPNO, ENAME, SAL  
FROM EMP  
WHERE SAL <> 250 AND SAL <>300 AND SAL <> 800;
```

```
SELECT EMPNO, ENAME, SAL  
FROM EMP  
WHERE SAL NOT IN (250,300,800);
```

```
SQL> SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL NOT IN (250,300,800);
```

EMPNO	ENAME	SAL
1001	강세아	200
1002	강혜진	500
1004	김영훈	200
1005	김영빈	350
1007	박경재	350
1008	박준서	200
1009	박나은	200
1011	송주훈	200
1012	이동훈	200
1015	정말한	200
1016	조재현	350
1017	조건우	200
1018	지승재	500
1019	차대령	200
1020	하동기	200

15 rows selected.

EMP 테이블에서 급여가 200이거나 350이거나 800이 아닌

직원들의 직원번호와 직원이름과 급여를 출력했습니다.

마찬가지로 IN 명령어를 사용해서 위 쿼리문과 동일한 결과를

출력할 수 있습니다.

과제01-8. 쿼리문 수행

```
SQL> SELECT EMPNO, ENAME FROM EMP WHERE ENAME LIKE '김%' OR ENAME LIKE '%나%';
```

EMPNO	ENAME
1004	김영훈
1005	김영빈
1006	김향진
1009	박나은
1010	신나라

```
SELECT EMPNO, ENAME FROM EMP WHERE ENAME LIKE '김%' OR ENAME LIKE '%나%';
```

이름이 '김'으로 시작하거나 가운데 글자가 '나'인 사원의 사원번호와 이름을 출력했습니다.

```
SQL> SELECT * FROM EMP WHERE MGR IS NULL;
```

EMPNO	ENAME	EPHONE	JOB	MGR	HIREDATE	SAL	DEPTNO
1014	이인호	010-8724-1144	사장		00/10/29	800	20

```
SELECT * FROM EMP WHERE MGR IS NULL;
```

상관이 없는 사원의 정보를 출력했습니다.

과제01-8. 쿼리문 수행

```
SQL> SELECT EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY HIREDATE ASC;
```

EMPNO	ENAME	JOB	TO_CHAR(HIREDATE, 'YY
1014	이인호	사장	2000/10/29
1018	지승재	부장	2003/01/16
1002	강혜진	부장	2005/12/20
1007	박경재	과장	2008/01/03
1005	김영빈	과장	2008/09/18
1016	조재현	과장	2009/09/10
1010	신나라	대리	2014/03/21
1021	허재영	대리	2015/04/25
1013	이지은	대리	2016/02/08
1003	구명희	대리	2016/10/13
1006	김향진	대리	2016/11/21
1020	하동기	사원	2018/11/05
1009	박나은	사원	2018/12/04
1015	정밀한	사원	2019/01/06
1019	차대령	사원	2019/02/13
1008	박준서	사원	2019/03/11
1004	김영훈	사원	2019/05/11
1017	조건우	사원	2019/06/18
1001	강세아	사원	2019/07/13
1012	이동훈	사원	2019/09/02
1011	송주은	사원	2019/10/27

21 rows selected.

```
SELECT EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY HIREDATE DESC;
```

사원 테이블에서 최근 입사한 직원 순으로(내림차순) 사원번호, 사원이름, 직급, 입사일을 출력했습니다.

과제01-8. 쿼리문 수행

```
SQL> SELECT EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY HIREDATE DESC;
```

EMPNO	ENAME	JOB	TO_CHAR(HIREDATE, 'YY
1011	송주은	사원	2019/10/27
1012	이동훈	사원	2019/09/02
1001	강세아	사원	2019/07/13
1017	조거우	사원	2019/06/18
1004	김영훈	사원	2019/05/11
1008	박준서	사원	2019/03/11
1019	차대령	사원	2019/02/13
1015	정밀하	사원	2019/01/06
1009	박나은	사원	2018/12/04
1020	하동기	사원	2018/11/05
1006	김향진	대리	2016/11/21
1003	구명희	대리	2016/10/13
1013	이지은	대리	2016/02/08
1021	허재영	대리	2015/04/25
1010	신나라	대리	2014/03/21
1016	조재혁	과장	2009/09/10
1005	김영빈	과장	2008/09/18
1007	박경재	과장	2008/01/03
1002	강혜진	부장	2005/12/20
1018	지승재	부장	2003/01/16
1014	이인호	사장	2000/10/29

21 rows selected.

```
SELECT EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY HIREDATE ASC;
```

사원 테이블에서 입사한 기간이 오래된 직원 순으로(오름차순) 사원번호, 사원이름, 직급, 입사일을 출력했습니다.

과제01-8. 쿼리문 수행

```
SQL> SELECT DEPTNO, EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY DEPTNO ASC, HIREDATE ASC;
```

DEPTNO	EMPNO	ENAME	JOB	TO_CHAR(HIREDATE, 'YY
10	1020	하동기	사원	2018/11/05
10	1015	정밀한	사원	2019/01/06
10	1008	박주서	사원	2019/03/11
10	1004	김영훈	사원	2019/05/11
10	1001	강세아	사원	2019/07/13
20	1014	이인호	사장	2000/10/29
20	1002	강혜진	부장	2005/12/20
20	1003	구명회	대리	2016/10/13
20	1006	김향진	대리	2016/11/21
20	1009	박나은	사원	2018/12/04
20	1019	차대령	사원	2019/02/13
20	1017	조건우	사원	2019/06/18
20	1011	송주은	사원	2019/10/27
30	1007	박경재	과장	2008/01/03
30	1012	이동훈	사원	2019/09/02
40	1013	이지은	대리	2016/02/08
60	1016	조재혁	과장	2009/09/10
60	1010	신나라	대리	2014/03/21
60	1021	허재영	대리	2015/04/25
70	1018	지승재	부장	2003/01/16
70	1005	김영빈	과장	2008/09/18

21 rows selected.

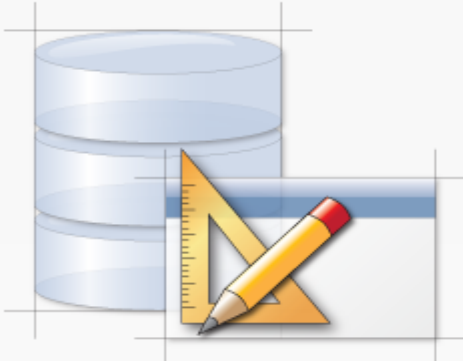
```
SELECT DEPTNO, EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY DEPTNO ASC, HIREDATE ASC;
```

사원 테이블에서 같은 부서중에서 오래된 직원 순으로 부서번호, 사원번호, 사원이름, 직급, 날짜를 출력했습니다.

(부서 번호를 오름차순으로 출력하고, 같은 부서 내에서 사원이 입사한 날짜를 오름차순으로 출력합니다.)

과제01-9. GET STARTED LOGIN

ORACLE[®] Application Express



Application Express 작업 공간 및 신임 정보를 입력하십시오.

작업 공간

사용자 이름

암호

[로그인](#)

시작하는 방법을 알아 보려면 여기를 클릭하십시오

Oracle Application Express는 데이터를 공유하고 사용자 정의 응용 프로그램을 생성 할 수있는 빠른 웹 응용 프로그램 개발 도구입니다. 웹 브라우저와 제한된 프로그래밍 경험 만 사용하면 빠르고 안전한 강력한 응용 프로그램을 개발하고 배포 할 수 있습니다.

언어 : 영어 , Português (브라질) , 中文 (简体) , 日本語

작업 공간

- 암호를 재설정
- 내 작업 공간 찾기
- 관리

시작하기

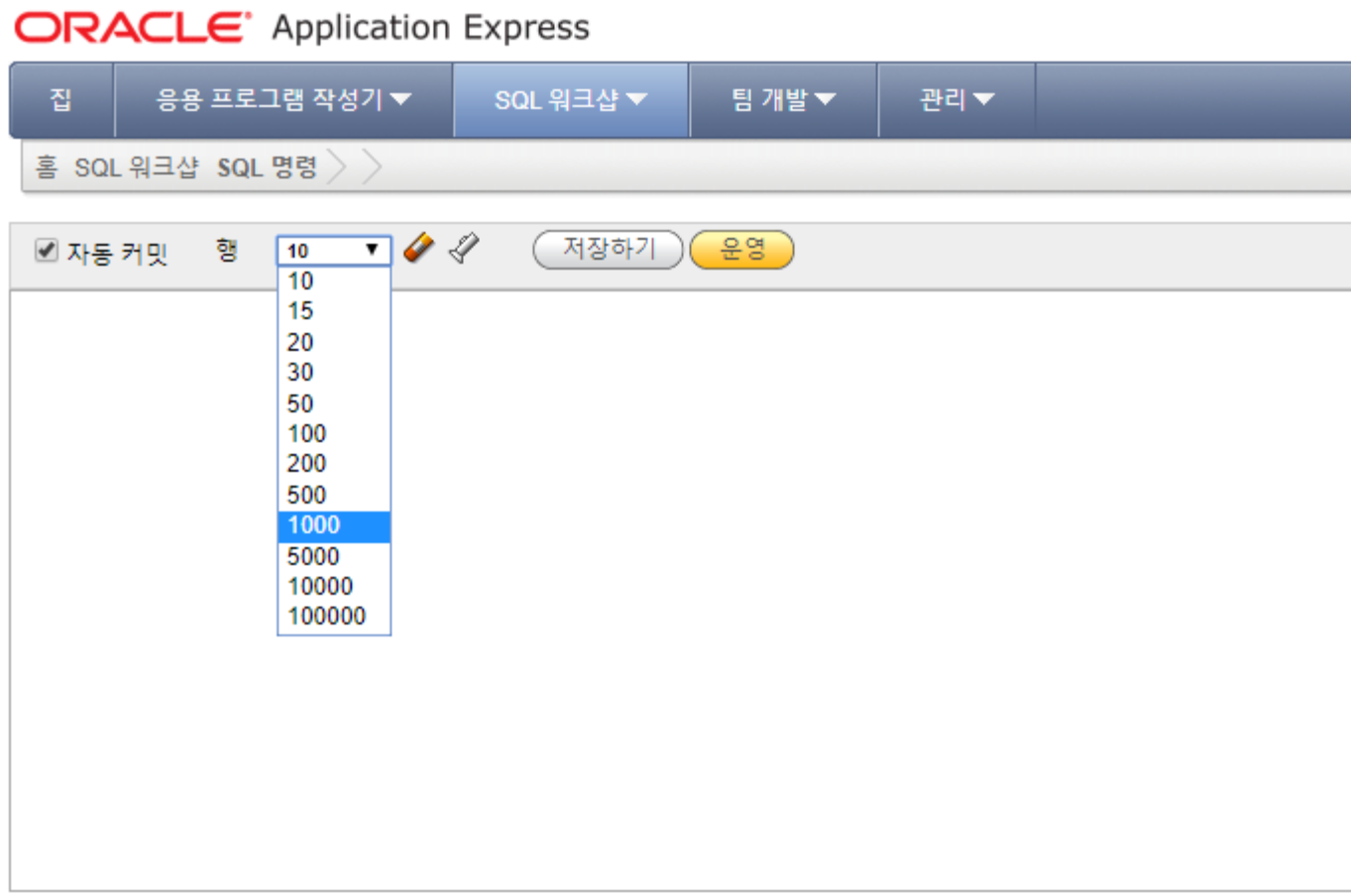
- 배우다 ...
- 오라클 기술 네트워크
- apex.oracle.com
- 예제 별 오라클

커뮤니티

- 토론 포럼
- 패키지 어플리케이션
- 파트너
- 블로그

생성한 작업 공간과 사용자 이름으로
GET STARTED에 로그인했습니다.

과제01-10. SET ROWS



퀴리문을 수행할 때

정상적으로 행이 출력되도록

출력될 수 있는 행의 개수를 여유롭게


1000으로 설정했습니다.


과제01-11. 모든 테이블 확인

☒ Autocommit

Rows

1000





Save

Run

select * from tabs;

Results

Explain

Describe

Saved SQL

History

TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS	PCT_FREE	PCT_USED	INI_TRANS	MAX_TRANS	INITIAL
DEPT	USERS	-	-	VALID	10	-	1	255	65536
EMP	USERS	-	-	VALID	10	-	1	255	65536
DEMO_USERS	USERS	-	-	VALID	10	-	1	255	65536
DEMO_CUSTOMERS	USERS	-	-	VALID	10	-	1	255	65536
DEMO_ORDERS	USERS	-	-	VALID	10	-	1	255	65536
DEMO_PRODUCT_INFO	USERS	-	-	VALID	10	-	1	255	65536
DEMO_ORDER_ITEMS	USERS	-	-	VALID	10	-	1	255	65536
DEMO_STATES	USERS	-	-	VALID	10	-	1	255	65536
APEX\$_ACL	USERS	-	-	VALID	10	-	1	255	65536
APEX\$_WS_WEBPG_SECTIONS	USERS	-	-	VALID	10	-	1	255	65536
APEX\$_WS_ROWS	USERS	-	-	VALID	10	-	1	255	65536
APEX\$_WS_HISTORY	USERS	-	-	VALID	10	-	1	255	65536
APEX\$_WS_NOTES	USERS	-	-	VALID	10	-	1	255	65536

select * from tabs;

현재 계정의 모든 테이블을 출력했습니다.

과제01-11. DEPT, EMP 테이블 확인

Autocommit Rows 1000 Save Run

```
select * from emp;
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/1981	5000	-	10
7698	BLAKE	MANAGER	7839	05/01/1981	2850	-	30
7782	CLARK	MANAGER	7839	06/09/1981	2450	-	10
7566	JONES	MANAGER	7839	04/02/1981	2975	-	20
7788	SCOTT	ANALYST	7566	12/09/1982	3000	-	20
7902	FORD	ANALYST	7566	12/03/1981	3000	-	20
7369	SMITH	CLERK	7902	12/17/1980	800	-	20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100	-	20
7900	JAMES	CLERK	7698	12/03/1981	950	-	30
7934	MILLER	CLERK	7782	01/23/1982	1300	-	10

14 rows returned in 0.01 seconds [Download](#)

Autocommit Rows 1000 Save Run

```
select * from dept;
```

Results Explain Describe Saved SQL History



DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 rows returned in 0.01 seconds [Download](#)

SELECT * FROM EMP;
SELECT * FROM DEPT;

EMP 테이블과 DEPT 테이블의 모든 레코드를 출력합니다.

과제01-12. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
select empno, ename  
from emp;
```



Results Explain Describe Saved SQL History

EMPNO	ENAME
7839	KING
7698	BLAKE
7782	CLARK
7566	JONES
7788	SCOTT
7902	FORD
7369	SMITH
7499	ALLEN
7521	WARD
7654	MARTIN
7844	TURNER
7876	ADAMS
7900	JAMES
7934	MILLER

14 rows returned in 0.00 seconds [Download](#)

select empno, ename from emp;

emp테이블에서 empno 컬럼과 ename컬럼을 출력합니다.

☒ Autocommit Rows 1000   Save Run

```
select empno, ename  
from emp  
order by empno desc;
```

Results Explain Describe Saved SQL History



EMPNO	ENAME
7934	MILLER
7902	FORD
7900	JAMES
7876	ADAMS
7844	TURNER
7839	KING
7788	SCOTT
7782	CLARK
7698	BLAKE
7654	MARTIN
7566	JONES
7521	WARD
7499	ALLEN
7369	SMITH

14 rows returned in 0.01 seconds [Download](#)

select empno, ename from emp order by empno desc;

emp테이블에서 empno 컬럼과 ename컬럼을 출력하되, 레코드를 내림차순으로 출력합니다.

과제01-12. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run



```
select deptno, dname  
from dept;
```

Results Explain Describe Saved SQL History

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

4 rows returned in 0.00 seconds [Download](#)

SELECT DEPTNO, DNAME FROM DEPT;
부서 테이블에서 부서번호와 부서이름만 출력했습니다.

☒ Autocommit Rows 1000   Save Run

```
select deptno 부서번호, dname as 부서명  
from dept;
```



Results Explain Describe Saved SQL History

부서번호	부서명
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

4 rows returned in 0.01 seconds [Download](#)

SELECT DEPTNO 부서번호, DNAME AS 부서명 FROM DEPT;
부서 테이블에서 부서번호와 부서이름에 별명을 붙여 출력했습니다.

과제01-12. 쿼리문 수행

☒ Autocommit Rows: 1000   Save Run

```
select distinct job  
from emp;
```



Results Explain Describe Saved SQL History

JOB
CLERK
SALESMAN
PRESIDENT
MANAGER
ANALYST

5 rows returned in 0.00 seconds [Download](#)

SELECT DISTINCT JOB FROM EMP;

EMP 테이블의 JOB 필드 중에서 중복된 필드를 제거하고 출력했습니다.

☒ Autocommit Rows: 1000   Save Run

```
select empno, ename, sal  
from emp  
where sal<=3000;
```

Results Explain Describe Saved SQL History



EMPNO	ENAME	SAL
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7788	SCOTT	3000
7902	FORD	3000
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
7654	MARTIN	1250
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950
7934	MILLER	1300

13 rows returned in 0.00 seconds [Download](#)

SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<=3000;

EMP 테이블에서 급여가 3000 이하인 사원번호, 사원명, 급여를 출력했습니다.

과제01-12. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
select empno, ename, sal
from emp
where ename='KING'
```



Results Explain Describe Saved SQL History

EMPNO	ENAME	SAL
7839	KING	5000

1 rows returned in 0.01 seconds [Download](#)

select empno, ename, sal from emp where ename='KING'

EMP 테이블에서 이름이 'KING'인 사원의 사번번호, 사원이름, 급여를 출력했습니다.



☒ Autocommit Rows 1000   Save Run

```
select empno, ename, sal
from emp
where sal=2500 or sal=3000 or sal=5000;
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	SAL
7839	KING	5000
7788	SCOTT	3000
7902	FORD	3000

3 rows returned in 0.00 seconds [Download](#)

☒ Autocommit Rows 1000   Save Run

```
select empno, ename, sal
from emp
where sal IN (2500, 3000, 5000);
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	SAL
7839	KING	5000
7788	SCOTT	3000
7902	FORD	3000

3 rows returned in 0.00 seconds [Download](#)



SELECT EMPNO, ENAME SAL
FROM EMP
WHERE SAL=2500 OR SAL=3000 OR SAL=5000;

EMP 테이블에서 급여가 2500이거나 3000이거나 5000인
사원들의 사원번호와 사원이름과 급여를 출력했습니다.

SELECT EMPNO, ENAME, SAL
FROM EMP
WHERE SAL IN(2500, 3000, 5000);

IN 명령어를 사용해서
위의 쿼리문과 같은 결과를 출력할 수 있습니다.

과제01-12. 쿼리문 수행

☒ Autocommit Rows: 1000 ▼  



```
select empno, ename, sal
from emp
where sal not IN (2500, 3000, 5000);
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	SAL
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
7654	MARTIN	1250
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950
7934	MILLER	1300

11 rows returned in 0.00 seconds

[Download](#)

☒ Autocommit Rows: 1000 ▼  

```
select empno, ename, sal
from emp
where sal <> 2500 and sal <> 3000 and sal <> 5000;
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	SAL
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7369	SMITH	800
7499	ALLEN	1600
7521	WARD	1250
7654	MARTIN	1250
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950
7934	MILLER	1300

11 rows returned in 0.00 seconds

[Download](#)

SELECT EMPNO, ENAME, SAL FROM EMP
WHERE SAL <> 2500 AND SAL <> 3000 AND SAL <> 5000;

EMP 테이블에서 급여가 2500이거나 3000이거나 5000이 아닌
사원들의 사원번호와 사원이름과 급여를 출력했습니다.

SELECT EMPNO, ENAME, SAL FROM EMP
WHERE SAL NOT IN (2500,3000,5000);

마찬가지로 IN 명령어를 사용해서
위 쿼리문과 동일한 결과를 출력할 수 있습니다.

과제01-12. 쿼리문 수행

☒ Autocommit Rows: 1000 Save Run

```
select empno, ename
from emp
where ename like 'M%'
or ename like '%A%';
```

Results Explain Describe Saved SQL History

EMPNO	ENAME
7698	BLAKE
7782	CLARK
7499	ALLEN
7521	WARD
7654	MARTIN
7876	ADAMS
7900	JAMES
7934	MILLER

8 rows returned in 0.00 seconds [Download](#)

SELECT EMPNO, ENAME FROM EMP WHERE ENAME LIKE 'M%' OR ENAME LIKE '%A%';

이름이 'M'으로 시작하거나 가운데 글자가 'A'인 사원의 사원번호와 이름을 출력했습니다.

☒ Autocommit Rows: 1000 Save Run

```
select *
from emp
where mgr IS NULL;
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/1981	5000	-	10

1 rows returned in 0.00 seconds [Download](#)

SELECT * FROM EMP WHERE MGR IS NULL;

상관이 없는 사원의 정보를 출력했습니다.

과제01-12. 쿼리문 수행

```
Autocommit Rows 1000 Save Run
select empno, ename, job, to_char(hiredate, 'YYYY/MM/DD')
from emp
order by hiredate desc;
```

Results Explain Describe Saved SQL History				
EMPNO	ENAME	JOB	TO_CHAR(HIREDATE,'YYYY/MM/DD')	
7876	ADAMS	CLERK	1983/01/12	
7788	SCOTT	ANALYST	1982/12/09	
7934	MILLER	CLERK	1982/01/23	
7902	FORD	ANALYST	1981/12/03	
7900	JAMES	CLERK	1981/12/03	
7839	KING	PRESIDENT	1981/11/17	
7654	MARTIN	SALESMAN	1981/09/28	
7844	TURNER	SALESMAN	1981/09/08	
7782	CLARK	MANAGER	1981/06/09	
7698	BLAKE	MANAGER	1981/05/01	
7566	JONES	MANAGER	1981/04/02	
7521	WARD	SALESMAN	1981/02/22	
7499	ALLEN	SALESMAN	1981/02/20	
7369	SMITH	CLERK	1980/12/17	

14 rows returned in 0.00 seconds [Download](#)

SELECT EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY HIREDATE DESC;

사원 테이블에서 입사한 기간이 오래된 직원 순으로(내림차순) 사원번호, 사원이름, 직급, 입사일을 출력했습니다.

과제01-12. 쿼리문 수행

```
Autocommit Rows 1000 Save Run
select deptno, empno, ename, job, to_char(hiredate, 'YYYY/MM/DD')
from emp
order by deptno asc, hiredate asc
```

Results	Explain	Describe	Saved SQL	History
DEPTNO	EMPNO	ENAME	JOB	TO_CHAR(HIREDATE,'YYYY/MM/DD')
10	7782	CLARK	MANAGER	1981/06/09
10	7839	KING	PRESIDENT	1981/11/17
10	7934	MILLER	CLERK	1982/01/23
20	7369	SMITH	CLERK	1980/12/17
20	7566	JONES	MANAGER	1981/04/02
20	7902	FORD	ANALYST	1981/12/03
20	7788	SCOTT	ANALYST	1982/12/09
20	7876	ADAMS	CLERK	1983/01/12
30	7499	ALLEN	SALESMAN	1981/02/20
30	7521	WARD	SALESMAN	1981/02/22
30	7698	BLAKE	MANAGER	1981/05/01
30	7844	TURNER	SALESMAN	1981/09/08
30	7654	MARTIN	SALESMAN	1981/09/28
30	7900	JAMES	CLERK	1981/12/03

14 rows returned in 0.00 seconds [Download](#)

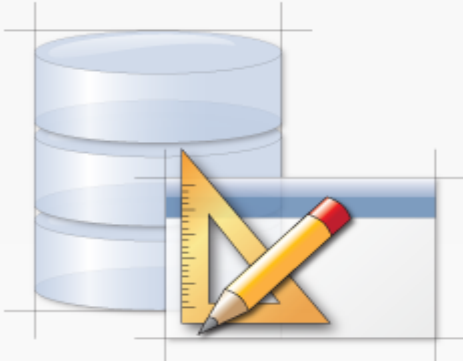
SELECT DEPTNO, EMPNO, ENAME, JOB, TO_CHAR(HIREDATE, 'YYYY/MM/DD') FROM EMP ORDER BY DEPTNO ASC, HIREDATE ASC;

사원 테이블에서 같은 부서중에서 오래된 직원 순으로 부서번호, 사원번호, 사원이름, 직급, 날짜를 출력했습니다.

(부서 번호를 오름차순으로 출력하고, 같은 부서 내에서 사원이 입사한 날짜를 오름차순으로 출력합니다.)

과제02-1. GETSTARTED LOGIN

ORACLE[®] Application Express



Application Express 작업 공간 및 신임 정보를 입력하십시오.

작업 공간

사용자 이름

암호

[로그인](#)

시작하는 방법을 알아 보려면 여기를 클릭하십시오

Oracle Application Express는 데이터를 공유하고 사용자 정의 응용 프로그램을 생성 할 수있는 빠른 웹 응용 프로그램 개발 도구입니다. 웹 브라우저와 제한된 프로그래밍 경험 만 사용하면 빠르고 안전한 강력한 응용 프로그램을 개발하고 배포 할 수 있습니다.

언어 : 영어 , Português (브라질) , 中文 (简体) , 日本語

작업 공간

- 암호를 재설정
- 내 작업 공간 찾기
- 관리

시작하기

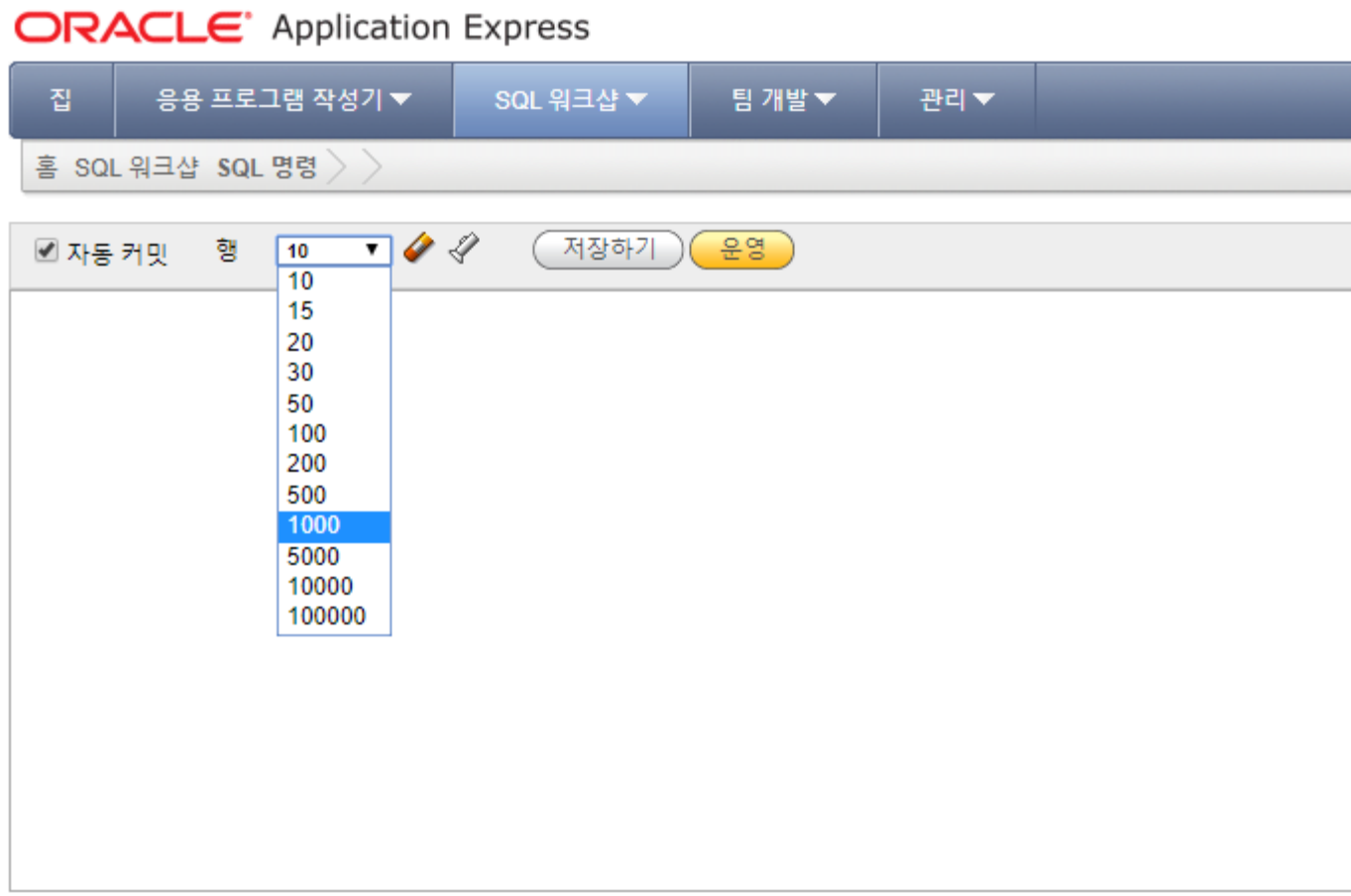
- 배우다 ...
- 오라클 기술 네트워크
- apex.oracle.com
- 예제 별 오라클

커뮤니티

- 토론 포럼
- 패키지 어플리케이션
- 파트너
- 블로그

생성한 작업 공간과 사용자 이름으로
GET STARTED에 로그인했습니다.

과제02-2. SET ROWS



퀴리문을 수행할 때

정상적으로 행이 출력되도록

출력될 수 있는 행의 개수를 여유롭게

1000으로 설정했습니다.

과제02-3. EMP01 테이블 생성

```
CREATE TABLE EMP01(  
  ENO NUMBER(4),  
  ENAME VARCHAR2(14),  
  SAL NUMBER(7, 3)  
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

```
DESC EMP01;
```

Results Explain Describe Saved SQL History

Object Type TABLE Object EMP01

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP01	ENO	NUMBER	-	4	0	-	✓	-	-
	ENAME	VARCHAR2	14	-	-	-	✓	-	-
	SAL	NUMBER	-	7	3	-	✓	-	-
1 - 3									

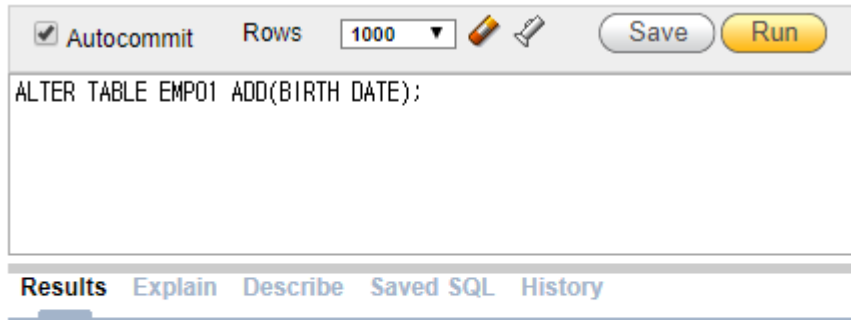
```
CREATE TABLE EMP01(  
  ENO NUMBER(4),  
  ENAME VARCHAR2(14),  
  SAL NUMBER(7, 3)  
);
```

// EMP01 테이블을 생성했습니다.
// ENO 크기 4인 NUMBER형
// ENAME 크기 14인 VARCHAR2형
// SAL 고정 소수점 숫자 NUMBER형
SAL은 최대 7자리,
소수점에서 최하위 유효자리수까지 자리수가 3입니다.

DESC EMP01;

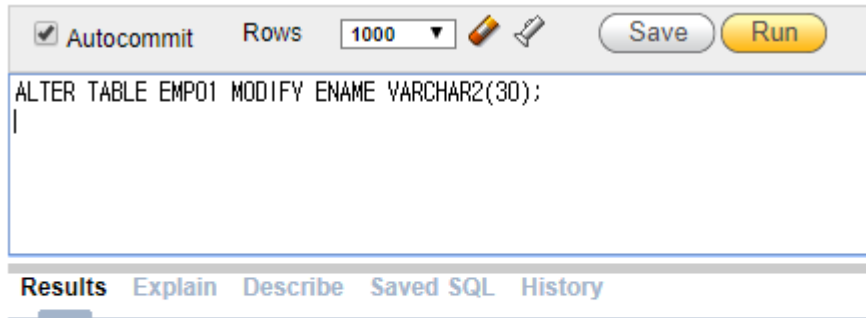
생성한 EMP01 테이블의 구조를 출력했습니다.

과제02-4. EMP01 테이블 구조 변경



`ALTER TABLE EMP01 ADD(BIRTH DATE);`

EMP01 테이블에 DATE 형식의 BIRTH 컬럼을 추가했습니다.



`ALTER TABLE EMP01 MODIFY ENAME VARCHAR2(30);`

EMP01 테이블의 ENAME 컬럼 타입을

크기 30의 VARCHAR2형으로 수정했습니다.

과제02-5. UNUSED COLUMNS

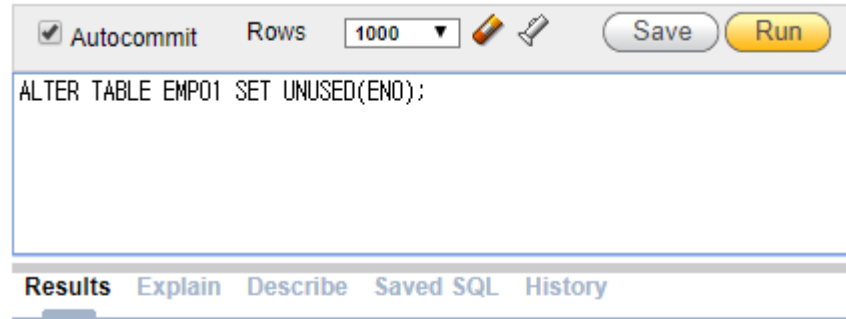


Table altered.

0.03 seconds

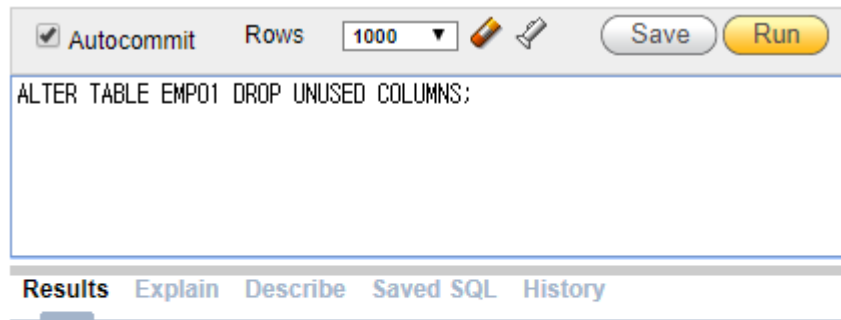


Table altered.

0.03 seconds

`ALTER TABLE EMP01 SET UNUSED(ENO);`



EMP01 테이블의 ENO 컬럼을 사용하지 않겠다고 마크했습니다.

실제로 삭제되지는 않지만 DROP된것처럼 취급합니다.

`ALTER TABLE EMP01 DROP UNUSED COLUMNS;`

EMP01 테이블에서 마킹해둔 컬럼을 실제로 삭제했습니다.

과제|02-6. RENAME TABLE



☒ Autocommit Rows: 1000   Save Run

```
RENAME EMP01 TO EMP02;
```

Results Explain Describe Saved SQL History

Statement processed.

0.00 seconds

☒ Autocommit Rows: 1000   Save Run

```
DESC EMP02;
```

Results Explain **Describe** Saved SQL History

Object Type: **TABLE** Object: **EMP02**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP02	ENAME	VARCHAR2	30	-	-	-	✓	-	-
	SAL	NUMBER	-	7	3	-	✓	-	-
	BIRTH	DATE	7	-	-	-	✓	-	-

1 - 3

RENAME EMP01 TO EMP02;

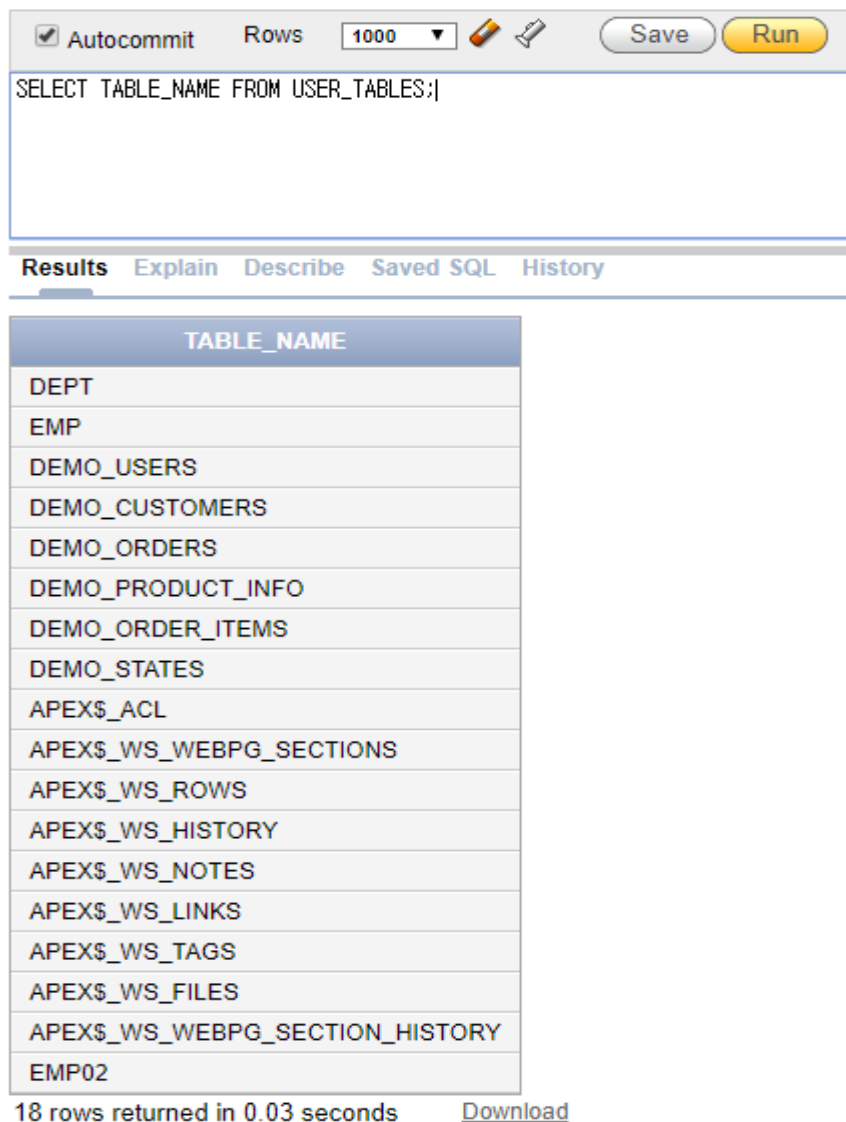
EMP01테이블의 이름을 EMP02로 변경했습니다.

DESC EMP02;

EMP02 테이블 구조를 출력했습니다

당연히 EMP01 테이블의 구조와 동일합니다.

과제02-7. 테이블 목록 출력



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with a checked 'Autocommit' checkbox, a 'Rows' dropdown set to '1000', and 'Save' and 'Run' buttons. Below the toolbar is a text area containing the SQL query: `SELECT TABLE_NAME FROM USER_TABLES;`. Underneath the text area is a tabbed interface with 'Results' selected. The 'Results' tab displays a table with one column, 'TABLE_NAME', and 18 rows of table names. At the bottom of the results, it says '18 rows returned in 0.03 seconds' and there is a 'Download' link.

TABLE_NAME
DEPT
EMP
DEMO_USERS
DEMO_CUSTOMERS
DEMO_ORDERS
DEMO_PRODUCT_INFO
DEMO_ORDER_ITEMS
DEMO_STATES
APEX\$_ACL
APEX\$_WS_WEBPG_SECTIONS
APEX\$_WS_ROWS
APEX\$_WS_HISTORY
APEX\$_WS_NOTES
APEX\$_WS_LINKS
APEX\$_WS_TAGS
APEX\$_WS_FILES
APEX\$_WS_WEBPG_SECTION_HISTORY
EMP02

18 rows returned in 0.03 seconds [Download](#)

`SELECT TABLE_NAME FROM USER_TABLES;`

현재 사용중인 사용자가 접근할 수 있는 모든 테이블 목록을 출력합니다.



SQL PLUS에서 단축키 만들기!

> `SELECT TABLE_NAME FROM USER_TABLES;`
또는 `SELECT TABLE_NAME FROM TABS;`

> save TN (TN은 자기가 편한 이름으로!)

> @TN

과제02-7. 테이블 목록 출력

☒ Autocommit Rows   Save Run

SELECT OWNER, TABLE_NAME FROM ALL_TABLES;

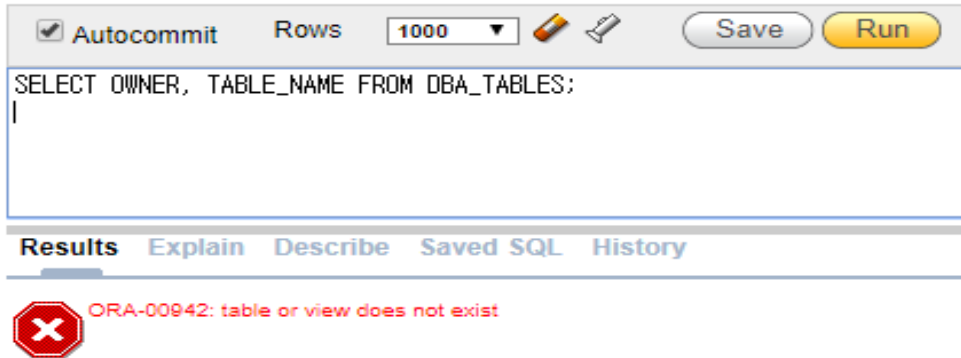
Results Explain Describe Saved SQL History

OWNER	TABLE_NAME
SYS	DUAL
SYS	SYSTEM_PRIVILEGE_MAP
SYS	TABLE_PRIVILEGE_MAP
SYS	STMT_AUDIT_OPTION_MAP
SYS	AUDIT_ACTIONS
SYS	WRR\$_REPLAY_CALL_FILTER
SYS	HS_BULKLOAD_VIEW_OBJ
SYS	HSS\$_PARALLEL_METADATA
SYS	HS_PARTITION_COL_NAME
SYS	HS_PARTITION_COL_TYPE
SYSTEM	HELP
CTXSYS	DR\$OBJECT_ATTRIBUTE
CTXSYS	DR\$POLICY_TAB
CTXSYS	DR\$THS
CTXSYS	DR\$THS_PHRASE

SELECT OWNER, TABLE_NAME FROM ALL_TABLES;

모든 사용자가 접근할 수 있는 모든 테이블을 출력합니다.

과제02-7. 테이블 목록 출력



```
SQL> set linesize 150 pagesize 150
SQL> SELECT OWNER, TABLE_NAME FROM DBA_TABLES;
```

OWNER	TABLE_NAME
SYS	ICOL\$
SYS	IND\$
SYS	COL\$
SYS	CLUS
SYS	TAB\$
SYS	LOB\$
SYS	COLTYPES

...

SYSTEM	LOGMNR_ERRORS
SYSTEM	LOGMNR_AGE_SPILL\$
SYSTEM	LOGMNR_SESSION_EVOLVE\$
SYS	UTL_RECOMP_SORTED
SYSTEM	LOGMNR_PROCESSED_LOG\$
SYSTEM	LOGMNR_GLOBAL\$
SYSTEM	LOGMNR_RESTART_CKPT_TXINFO\$
SYSTEM	LOGMNR_FILTER\$

1746 rows selected.

```
SQL> show user
USER is "SYS"
```



`SELECT OWNER, TABLE_NAME FROM DBA_TABLES;`

SYS권한이 있을 때 접근할 수 있는 모든 테이블을 출력합니다.

현재 DBA권한이 없으므로(SYS계정으로 접속중이 아니므로) 접근할 수 있는 테이블이 존재하지 않습니다.

SQL RUN에서 SYS계정으로 접속하고 쿼리문을 수행하면 1746개의 행을 출력합니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
SELECT * FROM DEPT;
```



Results Explain Describe Saved SQL History

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 rows returned in 0.00 seconds [Download](#)

SELECT * FROM DEPT;

DEPT 테이블의 모든 레코드를 출력했습니다.



☒ Autocommit Rows 1000   Save Run

```
INSERT INTO DEPT VALUES(NULL, 'TEST', 'TEST');
```

INSERT INTO DEPT VALUES(NULL, 'TEST', 'TEST');

DEPT 테이블에 (null, 'TEST', 'TEST')값을 삽입했습니다.
DEPT 첫번째 컬럼은 기본키로 NULL값을 갖지 못해 ERROR

ORA-01400: cannot insert NULL into ("HOMEWORK3"."DEPT"."DEPTNO")

☒ Autocommit Rows 1000   Save Run

```
desc dept;
```

Results Explain Describe Saved SQL History

Object Type	TABLE	Object	DEPT				
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
DEPT	DEPTNO	NUMBER	-	2	0	1	-
DEPT	DNAME	VARCHAR2	14				
DEPT	LOC	VARCHAR2	13				

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

DESC USER_CONSTRAINTS;

Results Explain Describe Saved SQL History

Object Type VIEW Object USER_CONSTRAINTS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment	논평
USER_CONSTRAINTS	OWNER	VARCHAR2	120	-	-	-	✓	-	Owner of the table	테이블 소유자
	CONSTRAINT_NAME	VARCHAR2	30	-	-	-	-	-	Name associated with constraint definition	구속 조건 정의와 연관된 이름
	CONSTRAINT_TYPE	VARCHAR2	1	-	-	-	✓	-	Type of constraint definition	구속 조건 정의 유형
	TABLE_NAME	VARCHAR2	30	-	-	-	-	-	Name associated with table with constraint definition	제한 조건 정의가있는 테이블과 연관된 이름
	SEARCH_CONDITION	LONG	0	-	-	-	✓	-	Text of search condition for table check	테이블 확인을위한 검색 조건 텍스트
	R_OWNER	VARCHAR2	120	-	-	-	✓	-	Owner of table used in referential constraint	참조 제한 조건에 사용된 테이블의 소유자
	R_CONSTRAINT_NAME	VARCHAR2	30	-	-	-	✓	-	Name of unique constraint definition for referenced table	참조된 테이블의 고유 제한 조건 정의 이름
	DELETE_RULE	VARCHAR2	9	-	-	-	✓	-	The delete rule for a referential constraint	참조 제한 조건의 삭제 규칙
	STATUS	VARCHAR2	8	-	-	-	✓	-	Enforcement status of constraint - ENABLED or DISABLED	제약 조건의 시행 상태-ENABLED 또는 DISABLED
	DEFERRABLE	VARCHAR2	14	-	-	-	✓	-	Is the constraint deferrable - DEFERRABLE or NOT DEFERRABLE	제한 조건이 지연 가능합니까-DEFERRABLE 또는 NOT DEFERRABLE
	DEFERRED	VARCHAR2	9	-	-	-	✓	-	Is the constraint deferred by default - DEFERRED or IMMEDIATE	제약 조건이 기본적으로 연기됩니까? DEFERRED 또는 IMMEDIATE
	VALIDATED	VARCHAR2	13	-	-	-	✓	-	Was this constraint system validated? - VALIDATED or NOT VALIDATED	이 구속 시스템이 검증 되었습니까? -유효 또는 무효
	GENERATED	VARCHAR2	14	-	-	-	✓	-	Was the constraint name system generated? - GENERATED NAME or USER NAM	구속 조건 이름 시스템이 생성 되었습니까? -생성된 이름 또는 사용자 이름
	BAD	VARCHAR2	3	-	-	-	✓	-	Creating this constraint should give ORA-02436. Rewrite it before 2000 AD.	이 제한 조건을 작성하면 ORA-02436이 제공됩니다. AD 2000 년 전에 다시 작성하십시오.
	RELY	VARCHAR2	4	-	-	-	✓	-	If set, this flag will be used in optimizer	설정된 경우이 플래그는 옵티마이저에서 사용됩니다.
	LAST_CHANGE	DATE	7	-	-	-	✓	-	The date when this column was last enabled or disabled	이 열이 마지막으로 활성화 또는 비활성화 된 날짜
	INDEX_OWNER	VARCHAR2	30	-	-	-	✓	-	The owner of the index used by the constraint	제약 조건에 사용되는 인덱스의 소유자
	INDEX_NAME	VARCHAR2	30	-	-	-	✓	-	The index used by the constraint	제약 조건에 사용되는 인덱스
	INVALID	VARCHAR2	7	-	-	-	✓	-	-	-
	VIEW_RELATED	VARCHAR2	14	-	-	-	✓	-	-	-

1 - 20

1-20

DESC USER_CONSTRAINTS;

제약조건에 관한 정보를 출력합니다.

과제03-1. 쿼리문 수행

Autocommit Rows 1000 Save Run

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME FROM USER_CONSTRAINTS;

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SYS_C007258	C	APEX\$\$_WS_FILES
SYS_C007259	C	APEX\$\$_WS_FILES
SYS_C007260	C	APEX\$\$_WS_FILES
SYS_C007261	C	APEX\$\$_WS_FILES
SYS_C007262	C	APEX\$\$_WS_FILES
APEX\$\$_WS_FILES_CL_CK	C	APEX\$\$_WS_FILES
APEX\$\$_WS_FILES_PK	P	APEX\$\$_WS_FILES
APEX\$\$_WS_FILES_FK	R	APEX\$\$_WS_FILES
SYS_C007266	C	APEX\$\$_WS_WEBPG_SECTION_HISTORY
SYS_C007267	C	APEX\$\$_WS_WEBPG_SECTION_HISTORY
SYS_C007268	C	APEX\$\$_WS_WEBPG_SECTION_HISTORY
SYS_C007269	C	APEX\$\$_WS_WEBPG_SECTION_HISTORY
SYS_C007184	P	DEPT
SYS_C007185	C	EMP
SYS_C007186	P	EMP
SYS_C007187	R	EMP
SYS_C007188	R	EMP
DEMO_USERS_PK	P	DEMO_USERS
SYS_C007190	C	DEMO_CUSTOMERS
SYS_C007191	C	DEMO_CUSTOMERS
SYS_C007192	C	DEMO_CUSTOMERS
DEMO_CUST_CREDIT_LIMIT_MAX	C	DEMO_CUSTOMERS
DEMO_CUSTOMERS_PK	P	DEMO_CUSTOMERS
SYS_C007195	C	DEMO_ORDERS
SYS_C007196	C	DEMO_ORDERS
DEMO_ORDER_TOTAL_MIN	C	DEMO_ORDERS
DEMO_ORDER_PK	P	DEMO_ORDERS
DEMO_ORDERS_CUSTOMER_ID_FK	R	DEMO_ORDERS
DEMO_ORDERS_USER_ID_FK	R	DEMO_ORDERS
SYS_C007201	C	DEMO_PRODUCT_INFO
DEMO_PRODUCT_INFO_PK	P	DEMO_PRODUCT_INFO
SYS_C007203	C	DEMO_ORDER_ITEMS
SYS_C007204	C	DEMO_ORDER_ITEMS
SYS_C007205	C	DEMO_ORDER_ITEMS
SYS_C007206	C	DEMO_ORDER_ITEMS
SYS_C007207	C	DEMO_ORDER_ITEMS
DEMO_ORDER_ITEMS_PK	P	DEMO_ORDER_ITEMS
DEMO_ORDER_ITEMS_FK	R	DEMO_ORDER_ITEMS
DEMO_ORDER_ITEMS_PRODUCT_ID_FK	R	DEMO_ORDER_ITEMS
SYS_C007211	C	APEX\$\$_ACL
SYS_C007212	C	APEX\$\$_ACL
SYS_C007213	C	APEX\$\$_ACL
SYS_C007214	C	APEX\$\$_ACL

SYS_C007215	C	APEX\$\$_ACL
SYS_C007216	C	APEX\$\$_ACL
APEX\$\$_ACL_PRIV_CK	C	APEX\$\$_ACL
APEX\$\$_ACL_PK	P	APEX\$\$_ACL
SYS_C007219	C	APEX\$\$_WS_WEBPG_SECTIONS
SYS_C007220	C	APEX\$\$_WS_WEBPG_SECTIONS
SYS_C007221	C	APEX\$\$_WS_WEBPG_SECTIONS
SYS_C007222	C	APEX\$\$_WS_WEBPG_SECTIONS
SYS_C007223	C	APEX\$\$_WS_WEBPG_SECTIONS
APEX\$\$_WS_WEBPG_SECTION_TYPE_CK	C	APEX\$\$_WS_WEBPG_SECTIONS
APEX\$\$_WS_WEBPG_SECTION_LINK_CK	C	APEX\$\$_WS_WEBPG_SECTIONS
APEX\$\$_WS_WEBPG_SECTION_3D_CK	C	APEX\$\$_WS_WEBPG_SECTIONS
APEX\$\$_WS_WEBPG_SECTIONS_PK	P	APEX\$\$_WS_WEBPG_SECTIONS
SYS_C007228	C	APEX\$\$_WS_ROWS
SYS_C007229	C	APEX\$\$_WS_ROWS
SYS_C007230	C	APEX\$\$_WS_ROWS
SYS_C007231	C	APEX\$\$_WS_ROWS
SYS_C007232	C	APEX\$\$_WS_ROWS
APEX\$\$_WS_ROWS_PK	P	APEX\$\$_WS_ROWS
SYS_C007234	C	APEX\$\$_WS_HISTORY
SYS_C007235	C	APEX\$\$_WS_HISTORY
SYS_C007236	C	APEX\$\$_WS_HISTORY
SYS_C007237	C	APEX\$\$_WS_NOTES
SYS_C007238	C	APEX\$\$_WS_NOTES
SYS_C007239	C	APEX\$\$_WS_NOTES
APEX\$\$_WS_NOTES_CL_CK	C	APEX\$\$_WS_NOTES
APEX\$\$_WS_NOTES_PK	P	APEX\$\$_WS_NOTES
APEX\$\$_WS_NOTES_FK	R	APEX\$\$_WS_NOTES
SYS_C007243	C	APEX\$\$_WS_LINKS
SYS_C007244	C	APEX\$\$_WS_LINKS
SYS_C007245	C	APEX\$\$_WS_LINKS
SYS_C007246	C	APEX\$\$_WS_LINKS
SYS_C007247	C	APEX\$\$_WS_LINKS
APEX\$\$_WS_LINKS_CL_CK	C	APEX\$\$_WS_LINKS
APEX\$\$_WS_LINKS_SH_CK	C	APEX\$\$_WS_LINKS
APEX\$\$_WS_LINKS_PK	P	APEX\$\$_WS_LINKS
APEX\$\$_WS_LINKS_FK	R	APEX\$\$_WS_LINKS
SYS_C007252	C	APEX\$\$_WS_TAGS
SYS_C007253	C	APEX\$\$_WS_TAGS
SYS_C007254	C	APEX\$\$_WS_TAGS
APEX\$\$_WS_TAGS_CL_CK	C	APEX\$\$_WS_TAGS
APEX\$\$_WS_TAGS_PK	P	APEX\$\$_WS_TAGS
APEX\$\$_WS_TAGS_FK	R	APEX\$\$_WS_TAGS



87 rows returned in 0.13 seconds [Download](#)

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME FROM USER_CONSTRAINTS;

사용자의 제약조건중에서 제약조건 이름, 제약조건타입, 테이블 이름을 출력했습니다.

DEPT 테이블의 제약조건 PRIMARY KEY와 EMP 테이블의 제약조건 PRIMARY KEY, FOREIGN KEY이 있습니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000  



```
SELECT * FROM EMP02;
```

Results Explain Describe Saved SQL History

no data found

SELECT * FROM EMP02;

EMP02 테이블의 모든 레코드를 출력했습니다.
삽입한 레코드가 없으므로 아무것도 출력되지 않습니다.



☒ Autocommit Rows: 1000  

```
INSERT INTO EMP02 VALUES(NULL, NULL, '사원', '2000/01/02', 20);
```

INSERT INTO EMP02 VALUES(NULL, NULL, '사원', '2000/01/02', 20);

EMP02 테이블에 (NULL, NULL, '사원', '2000/01/02', 20)값을 삽입했습니다.
EMP02 테이블이 가지는 컬럼의 수보다 많은 값으로 ERROR

ORA-00913: too many values

☒ Autocommit Rows: 1000  

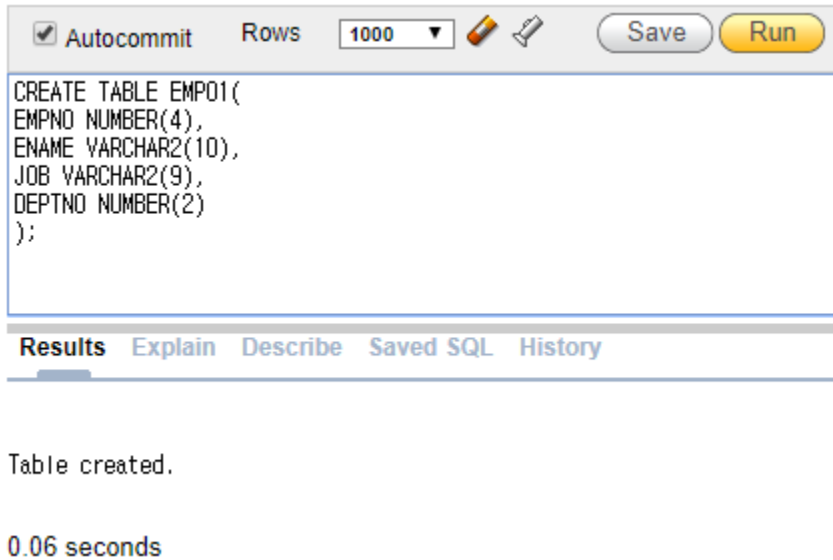
```
DROP TABLE EMP01;
```

DROP TABLE EMP01;

EMP01 테이블을 삭제했습니다.
EMP01 테이블은 존재하지 않으므로 ERROR

ORA-00942: table or view does not exist

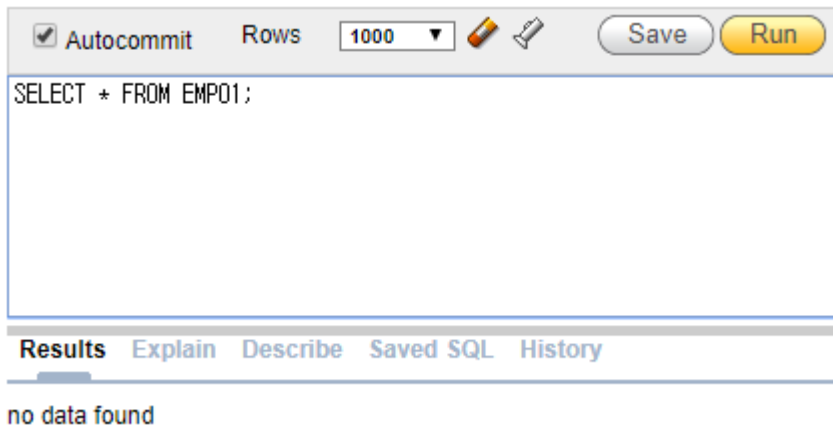
과제03-1. 쿼리문 수행



The screenshot shows the SQL Developer interface. At the top, there's a toolbar with 'Autocommit' checked, 'Rows' set to 1000, and 'Save' and 'Run' buttons. The SQL editor contains the following code:

```
CREATE TABLE EMP01(  
  EMPNO NUMBER(4),  
  ENAME VARCHAR2(10),  
  JOB VARCHAR2(9),  
  DEPTNO NUMBER(2)  
);
```

Below the editor, the 'Results' tab is selected, showing the message 'Table created.' and the execution time '0.06 seconds'.



The screenshot shows the SQL Developer interface. At the top, there's a toolbar with 'Autocommit' checked, 'Rows' set to 1000, and 'Save' and 'Run' buttons. The SQL editor contains the following code:

```
SELECT * FROM EMP01;
```

Below the editor, the 'Results' tab is selected, showing the message 'no data found'.

```
CREATE TABLE EMP01(  
  EMPNO NUMBER(4),  
  ENAME VARCHAR2(10),  
  JOB VARCHAR2(9),  
  DEPTNO NUMBER(2)  
);
```

EMP01 테이블을 생성했습니다.

EMPNO > 크기 4의 NUMBER형 사원 번호

ENAME > 크기 10의 VARCHAR2형 사원 이름

JOB > 크기 9의 VARCHAR2형 사원 직급



DEPTNO > 크기 2의 NUMBER형 부서 번호

```
SELECT * FROM EMP01;
```

EMP01 테이블의 모든 레코드를 출력했습니다.

삽입한 레코드가 없으므로 아무것도 출력되지 않습니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000   Save Run

```
INSERT INTO EMP01 VALUES(NULL, NULL, '사원', 30);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

☒ Autocommit Rows: 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME FROM USER_CONSTRAINTS WHERE TABLE_NAME='EMP01';
```

Results Explain Describe Saved SQL History

no data found

```
INSERT INTO EMP01 VALUES(NULL, NULL, '사원', 30);
```

EMP01 테이블에 (NULL, NULL, '사원', 30)값을 삽입했습니다.

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME='EMP01';
```

사용자의 제약조건중에서
EMP01테이블의 제약조건이름, 제약조건타입, 테이블이름을 출력했습니다.
EMP01 테이블은 제약조건이 없으므로 선택된 레코드가 없습니다.

과제03-1. 쿼리문 수행

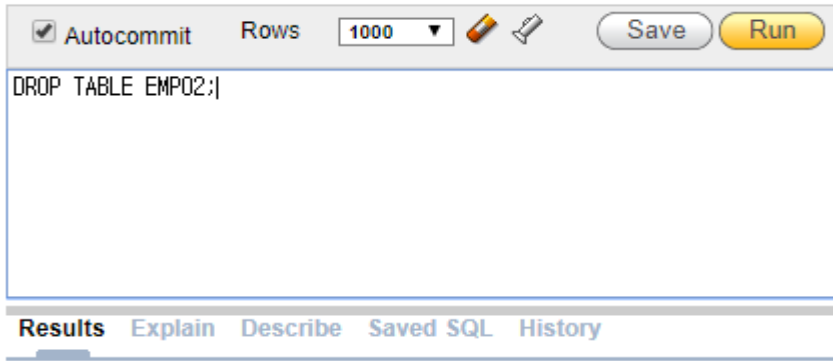


Table dropped.

0.04 seconds

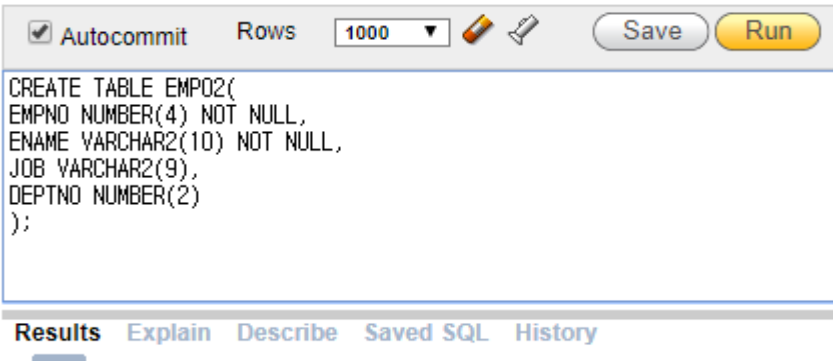


Table created.

0.00 seconds

DROP TABLE EMP02;

EMP02 테이블을 삭제했습니다.

```
CREATE TABLE EMP02(  
    EMPNO NUMBER(4) NOT NULL,  
    ENAME VARCHAR2(10) NOT NULL,  
    JOB VARCHAR2(9),  
    DEPTNO NUMBER(2)  
);
```

EMP02 테이블을 생성했습니다.



EMPNO NULL값을 가지지 못하는 크기 4의 NUMBER형

ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형

JOB 크기 9의 VARCHAR2형



DEPTNO 크기 2의 NUMBER형

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP02 VALUES(NULL, NULL, '사원', 30);
```

ORA-01400: cannot insert NULL into ("HOMEWORK3"."EMP02"."EMPNO")

☒ Autocommit Rows 1000   Save Run

```
DESC EMP02;
```

Results Explain Describe Saved SQL History

Object Type TABLE Object EMP02

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMP02	EMPNO	NUMBER	-	4	0	-	-	-	-
	ENAME	VARCHAR2	10	-	-	-	-	-	-
	JOB	VARCHAR2	9	-	-	-	✓	-	-
	DEPTNO	NUMBER	-	2	0	-	✓	-	-
1 - 4									



```
INSERT INTO EMP02 VALUES(NULL, NULL, '사원', 30);
```

EMP02 테이블에 (NULL, NULL, '사원', 30)값을 삽입했습니다.
EMPNO, ENAME 컬럼이 NULL값을 가지지 못하므로 ERROR

```
DESC EMP02;
```

EMP02 테이블의 구조를 출력합니다.

과제03-1. 쿼리문 수행


☒ Autocommit Rows 1000  

```
INSERT INTO EMP02 VALUES(1000, '허준', '사원', 30);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

☒ Autocommit Rows 1000  

```
COMMIT;|
```

Results Explain Describe Saved SQL History

Commit statement not applicable. All statements are automatically committed.

```
INSERT INTO EMP02 VALUES(1000, '허준', '사원', 30);
```



EMP02테이블에 (1000, '허준', '사원', 30)값을 삽입했습니다.

```
COMMIT;
```

지금까지의 결과를 저장합니다.

autocommit 옵션을 체크하면 쿼리문을 수행할때마다
COMMIT을 해주지 않아도 자동으로 결과를 저장합니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000   Save Run

```
SELECT * FROM EMP02;
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	JOB	DEPTNO
1000	허준	사원	30

1 rows returned in 0.01 seconds [Download](#)

☒ Autocommit Rows: 1000   Save Run

```
INSERT INTO EMP02 VALUES(1000, '홍길동', '과장', 20);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds



SELECT * FROM EMP02;

EMP02 테이블의 모든 레코드를 출력했습니다.

INSERT INTO EMP02 VALUES(1000, '홍길동', '과장', 20);

EMP02 테이블에 (1000, '홍길동', '과장', 20)값을 삽입했습니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000   Save Run

```
CREATE TABLE EMP03(  
EMPNO NUMBER(4) UNIQUE,  
ENAME VARCHAR2(10) NOT NULL,  
JOB VARCHAR2(9),  
DEPTNO NUMBER(2)  
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

☒ Autocommit Rows: 1000   Save Run

```
INSERT INTO EMP03 VALUES(1000, '허준', '사원', 30);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

```
CREATE TABLE EMP03(  
EMPNO NUMBER(4) UNIQUE,  
ENAME VARCHAR2(10) NOT NULL,  
JOB VARCHAR2(9),  
DEPTNO NUMBER(2)  
);
```

EMP03 테이블을 생성했습니다.
EMPNO UNIQUE한 값을 가지는 크기 4의 NUMBER형
ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형
JOB 크기 9의 VARCHAR2형
DEPTNO 크기 2의 NUMBER형

```
INSERT INTO EMP03 VALUES(1000, '허준', '사원', 30);
```

EMP03 테이블에 (1000, '허준', '사원', 30)값을 삽입했습니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP03 VALUES(1000, '홍길동', '과장', 20);
```

ORA-00001: unique constraint (HOMEWORK3.SYS_C007303) violated

INSERT INTO EMP03 VALUES(1000, '홍길동', '과장', 20);

EMP03 테이블에 (1000, '홍길동', '과장', 20)값을 삽입했습니다.
EMPNO 컬럼은 UNIQUE한 값을 가지므로 ERROR

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP03 VALUES(NULL, '안중근', '과장', 20);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

INSERT INTO EMP03 VALUES(NULL, '안중근', '과장', 20);

EMP03 테이블에 (NULL, '안중근', '과장', 20)값을 삽입했습니다.

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP03 VALUES(NULL, '이순신', '부장', 10);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

INSERT INTO EMP03 VALUES(NULL, '이순신', '부장', 10);

EMP03 테이블에 (NULL, '이순신', '부장', 10)값을 삽입했습니다.

과제03-1. 쿼리문 수행

```
CREATE TABLE EMP04(
EMPNO NUMBER(4) CONSTRAINT EMP04_EMPNO_UK UNIQUE,
ENAME VARCHAR2(10) CONSTRAINT EMP04_ENAME_NN NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);
```

Table created.

0.00 seconds

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP04');
```

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
EMP04_ENAME_NN	C	EMP04
EMP04_EMPNO_UK	U	EMP04

2 rows returned in 0.02 seconds

[Download](#)

```
CREATE TABLE EMP04(
EMPNO NUMBER(4) CONSTRAINT EMP04_EMPNO_UK UNIQUE,
ENAME VARCHAR2(10) CONSTRAINT EMP04_ENAME_NN NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2)
);
```

EMP04 테이블을 생성했습니다.

EMPNO UNIQUE한 값을 가지는 크기 4의 NUMBER형

설정한 UNIQUE 제약조건의 이름을 EMP04_EMPNO_UK로 설정

ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형

설정한 NOT NULL 제약조건의 이름을 EMP04_ENAME_NN으로 설정

JOB 크기 9의 VARCHAR2형

DEPTNO 크기 2의 NUMBER형



```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP04');
```

사용자의 제약조건중에서

EMP04테이블의 제약조건이름, 제약조건타입, 테이블이름을 출력했습니다.

EMP04 테이블의 제약조건 UNIQUE과 EMP04 테이블의 제약조건 NOT NULL를 확인합니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP04 VALUES(1000, '허준', '사원', 30);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP04 VALUES(1000, '홍길동', '과장', 20);
```

ORA-00001: unique constraint (HOMEWORK3.EMP04_EMPNO_UK) violated

```
INSERT INTO EMP04 VALUES(1000, '허준', '사원', 30);
```

EMP04 테이블에 (1000, '허준', '사원', 30)값을 삽입했습니다.

```
INSERT INTO EMP04 VALUES(1000, '홍길동', '과장', 20);
```

EMP04 테이블에 (1000, '홍길동', '과장', 20)값을 삽입했습니다.
EMPNO는 UNIQUE한 값을 가지므로 ERROR

과제03-1. 쿼리문 수행

```
CREATE TABLE EMP05(  
  EMPNO NUMBER(4) CONSTRAINT EMP05_EMPNO_PK PRIMARY KEY ,  
  ENAME VARCHAR2(10) CONSTRAINT EMP05_ENAME_NN NOT NULL,  
  JOB VARCHAR2(9),  
  DEPTNO NUMBER(2)  
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

```
INSERT INTO EMP05 VALUES(1000, '허준', '사원', 30);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

```
CREATE TABLE EMP05(  
  EMPNO NUMBER(4) CONSTRAINT EMP05_EMPNO_PK PRIMARY KEY ,  
  ENAME VARCHAR2(10) CONSTRAINT EMP05_ENAME_NN NOT NULL,  
  JOB VARCHAR2(9),  
  DEPTNO NUMBER(2)  
);
```

EMP05 테이블을 생성했습니다.

EMPNO PRIMARY KEY값을 가지는 크기 4의 NUMBER형

설정한 UNIQUE 제약조건의 이름을 EMP05_EMPNO_PK로 설정

ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형

설정한 NOT NULL 제약조건의 이름을 EMP05_ENAME_NN으로 설정

JOB 크기 9의 VARCHAR2형

DEPTNO 크기 2의 NUMBER형

```
INSERT INTO EMP05 VALUES(1000, '허준', '사원', 30);
```

EMP05 테이블에 (1000, '허준', '사원', 30)값을 삽입했습니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000   Save Run

```
INSERT INTO EMP05 VALUES(1000, '홍길동', '과장', 20);
```

ORA-00001: unique constraint (HOMEWORK3.EMP05_EMPNO_PK) violated

INSERT INTO EMP05 VALUES(1000, '홍길동', '과장', 20);

EMP05 테이블에 (1000, '홍길동', '과장', 20)값을 삽입했습니다.
EMPNO 컬럼은 PK속성이므로 UNIQUE한 값을 가지므로 ERROR


☒ Autocommit Rows: 1000   Save Run

```
INSERT INTO EMP05 VALUES(NULL, '이순신', '부장', 10);
```

ORA-01400: cannot insert NULL into ("HOMEWORK3"."EMP05"."EMPNO")

INSERT INTO EMP05 VALUES(NULL, '이순신', '부장', 10);

EMP05 테이블에 (NULL, '이순신', '부장', 10)값을 삽입했습니다.
EMPNO 컬럼은 PK속성이므로 NULL값을 가질 수 없으므로 ERROR

☒ Autocommit Rows: 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('DEPT');
```

Results Explain Describe Saved SQL History



CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
SYS_C007184	P	DEPT

1 rows returned in 0.02 seconds [Download](#)

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('DEPT');

사용자의 제약조건중에서
DEPT 테이블의 제약조건이름, 제약조건타입, 테이블이름을 출력했습니다.
DEPT 테이블의 제약조건 PRIMARY KEY를 확인합니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000   Save Run

```
SELECT CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME
FROM USER_CONS_COLUMNS
WHERE TABLE_NAME IN('DEPT');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	TABLE_NAME	COLUMN_NAME
SYS_C007184	DEPT	DEPTNO

1 rows returned in 0.02 seconds [Download](#)

☒ Autocommit Rows: 1000   Save Run

```
SELECT * FROM DEPT;
```

Results Explain Describe Saved SQL History

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 rows returned in 0.00 seconds [Download](#)

```
SELECT CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME
FROM USER_CONS_COLUMNS
WHERE TABLE_NAME IN('DEPT');
```

사용자의 제약조건 중에서
DEPT 테이블의 제약조건이름, 테이블이름, 속성명을 출력했습니다.
DEPT 테이블의 제약조건 PRIMARY KEY를 확인했습니다.

```
SELECT * FROM DEPT;
```

DEPT 테이블의 모든 레코드 출력했습니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP(EMPNO, ENAME, JOB, DEPTNO) VALUES(1010, '홍길동', '사원', 50);
```

```
INSERT INTO EMP(EMPNO, ENAME, JOB, DEPTNO) VALUES(1010, '홍길동', '사원', 50);
```

EMP 테이블의 (EMPNO, ENAME, JOB, DEPTNO) 컬럼에 (1010, '홍길동', '사원', 50) 값을 삽입했습니다.
DEPT 테이블의 DEPT 컬럼에 50이란 값을 가지는 레코드가 없으므로 ERROR

ORA-02291: integrity constraint (HOMEWORK3.SYS_C007188) violated - parent key not found

☒ Autocommit Rows 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('DEPT', 'EMP');
```

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('DEPT', 'EMP');
```

사용자의 제약조건중에서
EMP 테이블과 DEPT 테이블의 제약조건이름, 제약조건타입,
외래키인경우 어떤 기본키를 참조했는지, 그리고 테이블이름을 출력합니다.
DEPT 테이블의 제약조건 PRIMARY KEY
EMP 테이블의 제약조건 PRIMARY KEY
EMP 테이블의 제약조건 NOT NULL



Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
SYS_C007184	P	-	DEPT
SYS_C007185	C	-	EMP
SYS_C007186	P	-	EMP
SYS_C007187	R	SYS_C007186	EMP
SYS_C007188	R	SYS_C007184	EMP

5 rows returned in 0.03 seconds

[Download](#)

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000   Save Run

```
SELECT * FROM EMP05;
```

Results Explain Describe Saved SQL History

EMPNO	ENAME	JOB	DEPTNO
1000	허준	사원	30

1 rows returned in 0.00 seconds [Download](#)

☒ Autocommit Rows: 1000   Save Run

```
DELETE EMP05 WHERE EMPNO=1010;
```

Results Explain Describe Saved SQL History

0 row(s) deleted.

0.00 seconds

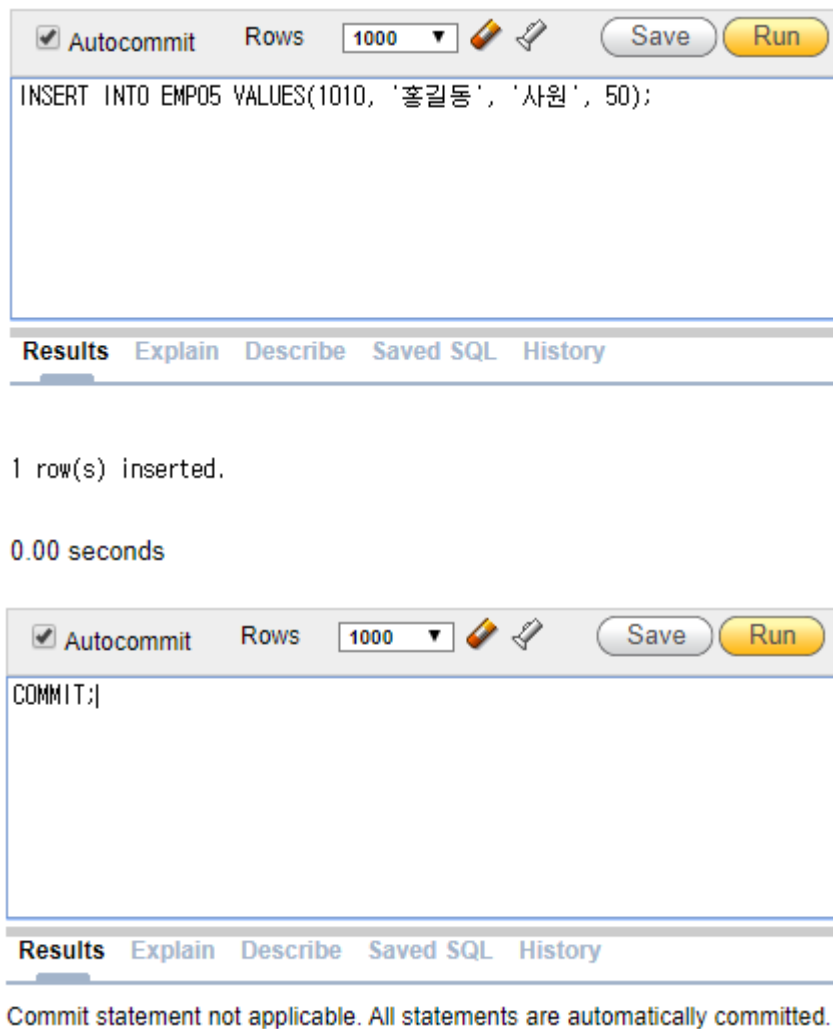
SELECT * FROM EMP05;

EMP05 테이블의 모든 레코드를 출력합니다.

DELETE EMP05 WHERE EMPNO=1010;

EMP05 테이블에서 EMPNO가 1010인 레코드를 삭제했습니다.

과제03-1. 쿼리문 수행



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with a checked 'Autocommit' checkbox, a 'Rows' dropdown set to '1000', and 'Save' and 'Run' buttons. Below the toolbar is a text area containing the SQL statement: `INSERT INTO EMP05 VALUES(1010, '홍길동', '사원', 50);`. Below the text area is a tabbed interface with 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History' tabs. The 'Results' tab is active and displays the text: '1 row(s) inserted.' and '0.00 seconds'. Below this is another instance of the same IDE interface, but with the text area containing `COMMIT;|`. The 'Results' tab in this instance displays the message: 'Commit statement not applicable. All statements are automatically committed.'



```
INSERT INTO EMP05 VALUES(1010, '홍길동', '사원', 50);
```

EMP05 테이블에 (1010, '홍길동', '사원', 50)값을 삽입했습니다.

```
COMMIT;
```

지금까지의 결과를 저장합니다.
autocommit 옵션을 체크하면 쿼리문을 수행할때마다
COMMIT을 해주지 않아도 자동으로 결과를 저장합니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
CREATE TABLE EMP06(  
EMPNO NUMBER(4) CONSTRAINT EMP06_EMPNO_PK PRIMARY KEY ,  
ENAME VARCHAR2(10) CONSTRAINT EMP06_ENAME_NN NOT NULL,  
JOB VARCHAR2(9),  
DEPTNO NUMBER(2) CONSTRAINT EMP06_DEPTNO_FK REFERENCES DEPT(DEPTNO)  
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP06 VALUES(1010, '홍길동', '사원', 50);
```

ORA-02291: integrity constraint (HOMEWORK3.EMP06_DEPTNO_FK) violated - parent key not found

```
CREATE TABLE EMP06(  
EMPNO NUMBER(4) CONSTRAINT EMP06_EMPNO_PK PRIMARY KEY ,  
ENAME VARCHAR2(10) CONSTRAINT EMP06_ENAME_NN NOT NULL,  
JOB VARCHAR2(9),  
DEPTNO NUMBER(2) CONSTRAINT EMP06_DEPTNO_FK REFERENCES DEPT(DEPTNO)  
);
```

EMP06 테이블을 생성했습니다.

EMPNO PRIMARY KEY값을 가지는 크기 4의 NUMBER형

설정한 PRIMARY KEY 제약조건의 이름을 EMP06_EMPNO_PK로 설정

ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형

설정한 NOT NULL 제약조건의 이름을 EMP06_ENAME_NN으로 설정

JOB 크기 9의 VARCHAR2형

DEPTNO FOREIGN KEY값을 가지는 크기 2의 NUMBER형

설정한 FOREIGN KEY값을 가지는 제약조건의 이름을 EMP06_DEPTNO_FK로 설정

```
INSERT INTO EMP06 VALUES(1010, '홍길동', '사원', 50);
```

EMP06 테이블에 (1010, '홍길동', '사원', 50)값을 삽입했습니다.

외래키인 DEPT 테이블의 DEPTNO 컬럼의 레코드에 존재하는 값이 아니라 ERROR

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP06 VALUES(1010, '홍길동', '사원', 30);
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

☒ Autocommit Rows 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('EMP06');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
EMP06_ENAME_NN	C	-	EMP06
EMP06_EMPNO_PK	P	-	EMP06
EMP06_DEPTNO_FK	R	SYS_C007184	EMP06

3 rows returned in 0.03 seconds [Download](#)

```
INSERT INTO EMP06 VALUES(1010, '홍길동', '사원', 30);
```

EMP06 테이블에 (1010, '홍길동', '사원', 30)값을 삽입했습니다.

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('EMP06');
```

사용자의 제약조건중에서
EMP06테이블의 제약조건이름, 제약조건타입,
외래키인경우 어떤 기본키를 참조했는지, 그리고 테이블이름을 출력합니다.
EMP06 테이블의 제약조건 PRIMARY KEY
EMP06 테이블의 제약조건 NOT NULL
EMP06 테이블의 제약조건 FOREIGN KEY

과제03-1. 쿼리문 수행

```
Autocommit Rows 1000 Save Run

CREATE TABLE EMP07(
EMPNO NUMBER(4) CONSTRAINT EMP07_EMPNO_PK PRIMARY KEY ,
ENAME VARCHAR2(10) CONSTRAINT EMP07_ENAME_NN NOT NULL,
SAL NUMBER(7, 2) CONSTRAINT EMP07_SAL_CK CHECK(SAL BETWEEN 500 AND 5000),
GENDER VARCHAR2(1) CONSTRAINT EMP07_GENDER_CK CHECK(GENDER IN('M', 'F'))
);

Results Explain Describe Saved SQL History
```

Table created.

0.01 seconds

```
Autocommit Rows 1000 Save Run

INSERT INTO EMP07 VALUES(1000, '허준', 200, 'M');
```

ORA-02290: check constraint (HOMEWORK3.EMP07_SAL_CK) violated

```
CREATE TABLE EMP07(
EMPNO NUMBER(4) CONSTRAINT EMP07_EMPNO_PK PRIMARY KEY ,
ENAME VARCHAR2(10) CONSTRAINT EMP07_ENAME_NN NOT NULL,
SAL NUMBER(7, 2) CONSTRAINT EMP07_SAL_CK CHECK(SAL BETWEEN 500 AND 5000),
GENDER VARCHAR2(1) CONSTRAINT EMP07_GENDER_CK CHECK(GENDER IN('M', 'F'))
);
```

EMP07 테이블을 생성합니다.

EMPNO PRIMARY KEY값을 가지는 크기 4의 NUMBER형

설정한 PRIMARY KEY 제약조건의 이름을 EMP07_EMPNO_PK로 설정

ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형

설정한 NOT NULL 제약조건의 이름을 EMP07_ENAME_NN으로 설정

SAL 고정 소수점 숫자 NUMBER형

최대 7자리이며, 소수점에서 최하위 유효자리수까지의 자리수가 2입니다

SAL 컬럼은 500부터 5000사이의 값만 삽입할 수 있다.

설정한 CHECK 제약조건의 이름을 EMP07_SAL_CK로 설정

GENDER 크기 1의 VARCHAR2형

GENDER 컬럼은 M, F인 값만 삽입할 수 있다.



설정한 CHECK 제약조건의 이름을 EMP07_GENDER_CK로 설정

```
INSERT INTO EMP07 VALUES(1000, '허준', 200, 'M');
```

EMP07 테이블에 (1000, '허준', 200, 'M') 값을 삽입합니다.

SAL 컬럼은 500부터 5000사이의 값만 가지므로 ERROR

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP07 VALUES(1000, '허준', 600, 'A');
```

ORA-02290: check constraint (HOMEWORK3.EMP07_GENDER_CK) violated

INSERT INTO EMP07 VALUES(1000, '허준', 600, 'A');

EMP07 테이블에 (1000, '허준', 600, 'A')값을 삽입했습니다.
GENDER 컬럼은 M과 F값만 가지므로 ERROR

☒ Autocommit Rows 1000   Save Run

```
SELECT * FROM DEPT01;
```



SELECT * FROM DEPT01;

DEPT01 테이블의 모든 레코드를 출력했습니다.
DEPT01 테이블이 생성되어 있지 않기 때문에 ERROR

Results Explain Describe Saved SQL History



ORA-00942: table or view does not exist

☒ Autocommit Rows 1000   Save Run

```
DROP TABLE DEPT01;
```

DROP TABLE DEPT01;

DEPT01 테이블을 제거했습니다.
DEPT01 테이블이 생성되어 있지 않기 때문에 ERROR

ORA-00942: table or view does not exist

과제03-1. 쿼리문 수행

```
Autocommit Rows 1000 Save Run
CREATE TABLE DEPT01(
  DEPTNO NUMBER(2) PRIMARY KEY,
  DNAME VARCHAR2(14),
  LOC VARCHAR2(13) DEFAULT '서울'
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

```
Autocommit Rows 1000 Save Run
INSERT INTO DEPT01(DEPTNO, DNAME) VALUES(10, '경리부');
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

```
CREATE TABLE DEPT01(
  DEPTNO NUMBER(2) PRIMARY KEY,
  DNAME VARCHAR2(14),
  LOC VARCHAR2(13) DEFAULT '서울'
);
```

DEPT01 테이블 생성합니다.
DEPTNO PRIMARY KEY 값을 가지는 크기 2의 NUMBER형
DNAME 크기 14의 VARCHAR2형
LOC DEFAULT값으로 '서울' 을 가지는 크기 13의 VARCHAR2형

```
INSERT INTO DEPT01(DEPTNO, DNAME) VALUES(10, '경리부');
```

DEPT01 테이블의 DEPTNO, DNAME 컬럼에 (10, '경리부')값을 삽입했습니다.

과제03-1. 쿼리문 수행

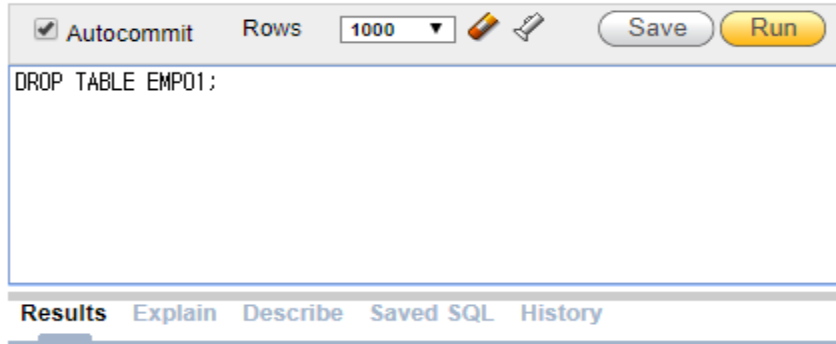


Table dropped.

0.00 seconds

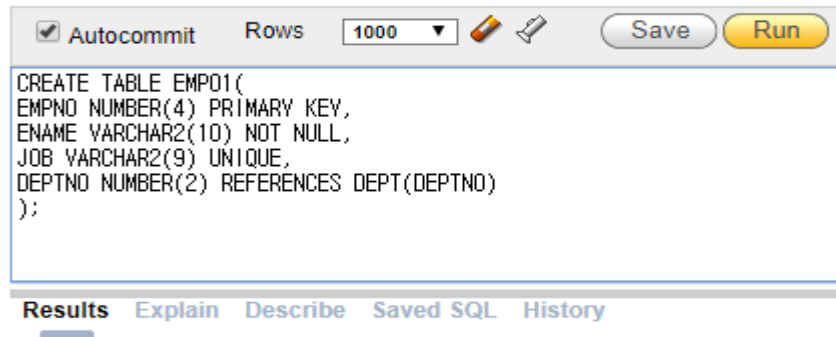


Table created.

0.01 seconds

```
DROP TABLE EMP01;
```

EMP01 테이블을 삭제했습니다.

```
CREATE TABLE EMP01(  
  EMPNO NUMBER(4) PRIMARY KEY,  
  ENAME VARCHAR2(10) NOT NULL,  
  JOB VARCHAR2(9) UNIQUE,  
  DEPTNO NUMBER(2) REFERENCES DEPT(DEPTNO)  
);
```

EMP01 테이블을 생성했습니다.
EMPNO PRIMARY KEY 값을 갖는 크기 4의 NUMBER형
ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형
JOB UNIQUE 값을 갖는 크기 9의 VARCHAR2형
DEPTNO FOREIGN KEY 값을 가지는 크기2의 NUMBER형

과제03-1. 쿼리문 수행

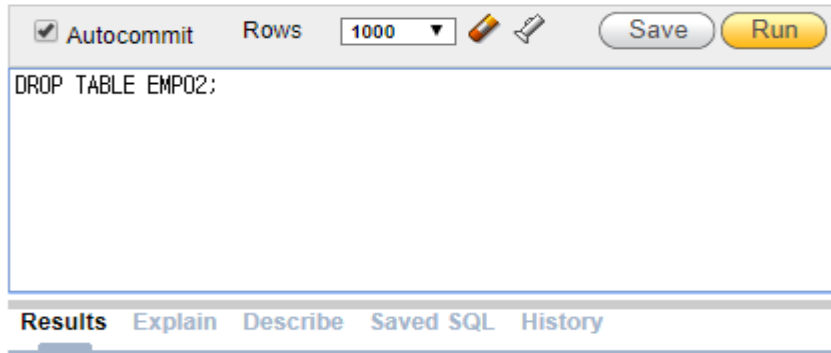


Table dropped.

0.01 seconds

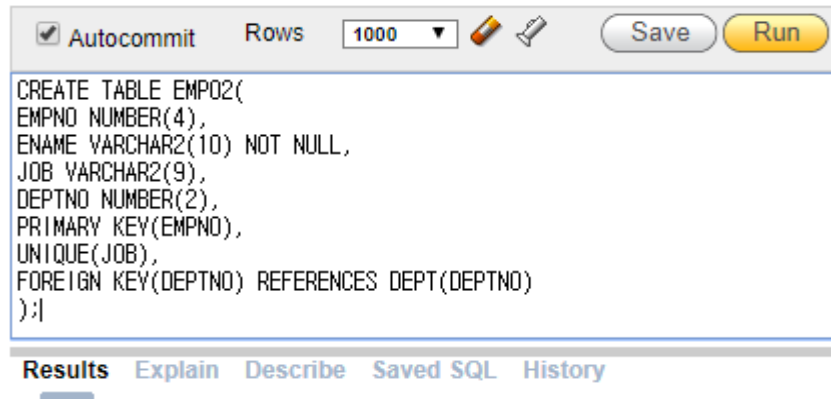


Table created.

0.00 seconds

DROP TABLE EMP02;



EMP02 테이블을 제거했습니다.

```
CREATE TABLE EMP02(  
    EMPNO NUMBER(4),  
    ENAME VARCHAR2(10) NOT NULL,  
    JOB VARCHAR2(9),  
    DEPTNO NUMBER(2),  
    PRIMARY KEY(EMPNO),  
    UNIQUE(JOB),  
    FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO)  
);
```

EMP02 테이블을 생성했습니다.

EMPNO PRIMARY KEY값을 갖는 크기 4의 NUMBER형
ENAME NULL값을 가지지 못하는 크기 10의 VARCHAR2형
JOB UNIQUE 값을 갖는 크기 9의 VARCHAR2형
DEPTNO FOREIGN KEY 값을 갖는 크기 2의 NUMBER형

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP02');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
SYS_C007320	C	-	EMP02
SYS_C007321	P	-	EMP02
SYS_C007322	U	-	EMP02
SYS_C007323	R	SYS_C007184	EMP02

4 rows returned in 0.03 seconds [Download](#)

☒ Autocommit Rows 1000   Save Run

```
DROP TABLE EMP03;
```

Results Explain Describe Saved SQL History

Table dropped.

0.00 seconds

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP02');
```

사용자의 제약조건중에서
EMP02 테이블의 제약조건이름, 제약조건타입,
외래키인 경우 어떤 기본키를 참조했는지, 그리고 테이블이름을 출력합니다.

EMP02 테이블의 제약조건 PRIMARY KEY
EMP02 테이블의 제약조건 NOT NULL
EMP02 테이블의 제약조건 UNIQUE
EMP02 테이블의 제약조건 FOREIGN KEY

```
DROP TABLE EMP03;
```

EMP03 테이블을 제거했습니다.

과제03-1. 쿼리문 수행

```
CREATE TABLE EMP03(
EMPNO NUMBER(4),
ENAME VARCHAR2(10) CONSTRAINT EMP03_ENAME_NN NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2),
CONSTRAINT EMP03_EMPNO_PK PRIMARY KEY(EMPNO),
CONSTRAINT EMP03_JOB_UK UNIQUE(JOB),
CONSTRAINT EMP03_DEPTNO_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO)
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP03');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
EMP03_ENAME_NN	C	-	EMP03
EMP03_EMPNO_PK	P	-	EMP03
EMP03_JOB_UK	U	-	EMP03
EMP03_DEPTNO_FK	R	SYS_C007184	EMP03

4 rows returned in 0.03 seconds

[Download](#)

```
CREATE TABLE EMP03(
EMPNO NUMBER(4),
ENAME VARCHAR2(10) CONSTRAINT EMP03_ENAME_NN NOT NULL,
JOB VARCHAR2(9),
DEPTNO NUMBER(2),
CONSTRAINT EMP03_EMPNO_PK PRIMARY KEY(EMPNO),
CONSTRAINT EMP03_JOB_UK UNIQUE(JOB),
CONSTRAINT EMP03_DEPTNO_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO)
);
```

EMP03 테이블을 생성했습니다.

EMPNO PRIMARY KEY 값을 갖는 크기 4의 NUMBER형

설정한 PRIMARY KEY 제약조건의 이름을 EMP03_ENAME_PK로 설정

ENAME NULL 값을 가지지 못하는 크기 10의 VARCHAR2형

설정한 NOT NULL 제약조건의 이름을 EMP03_ENAME_NN으로 설정

JOB UNIQUE 값을 갖는 크기 9의 VARCHAR2형

설정한 UNIQUE 제약조건의 이름을 EMP03_JOB_UK로 설정

DEPTNO FOREIGN KEY 값을 갖는 크기 2의 NUMBER형

설정한 FOREIGN KEY 제약조건의 이름을 EMP03_DEPTNO_FK로 설정

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP03');
```

사용자의 제약조건중에서

EMP03 테이블의 제약조건 이름, 제약조건 타입,

외래키인 경우 어떤 기본키를 참조했는지, 그리고 테이블 이름을 출력합니다.

EMP03 테이블의 제약조건 PRIMARY KEY

EMP03 테이블의 제약조건 NOT NULL

EMP03 테이블의 제약조건 UNIQUE

EMP03 테이블의 제약조건 FOREIGN KEY

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000 Save Run

```
CREATE TABLE MEMBER01(  
NAME VARCHAR2(10),  
ADDRESS VARCHAR2(30),  
HPHONE VARCHAR2(16),  
CONSTRAINT MEMBER01_COMBO_PK PRIMARY KEY(NAME, HPHONE)  
);
```

Results Explain Describe Saved SQL History

Table created.

0.01 seconds

☒ Autocommit Rows: 1000 Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('MEMBER01');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
MEMBER01_COMBO_PK	P	-	MEMBER01



1 rows returned in 0.03 seconds [Download](#)

```
CREATE TABLE MEMBER01(  
NAME VARCHAR2(10),  
ADDRESS VARCHAR2(30),  
HPHONE VARCHAR2(16),  
CONSTRAINT MEMBER01_COMBO_PK PRIMARY KEY(NAME, HPHONE)  
);
```

MEMBER01 테이블을 생성했습니다.
NAME 크기 10의 VARCHAR2형
ADDRESS 크기 30의 VARCHAR2형
HPHONE 크기 16의 VARCHAR2형
NAME과 HPHONE의 컬럼을 묶어 기본키로 설정
설정한 PRIMARY KEY 제약조건의 이름을 MEMBER01_COMBO_PK로 설정

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('MEMBER01');
```

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000  

```
DROP TABLE EMP01;
```

Results Explain Describe Saved SQL History

Table dropped.

0.01 seconds

☒ Autocommit Rows: 1000  

```
CREATE TABLE EMP01(  
EMPNO NUMBER(4),  
ENAME VARCHAR2(10),  
JOB VARCHAR2(9),  
DEPTNO NUMBER(2)  
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

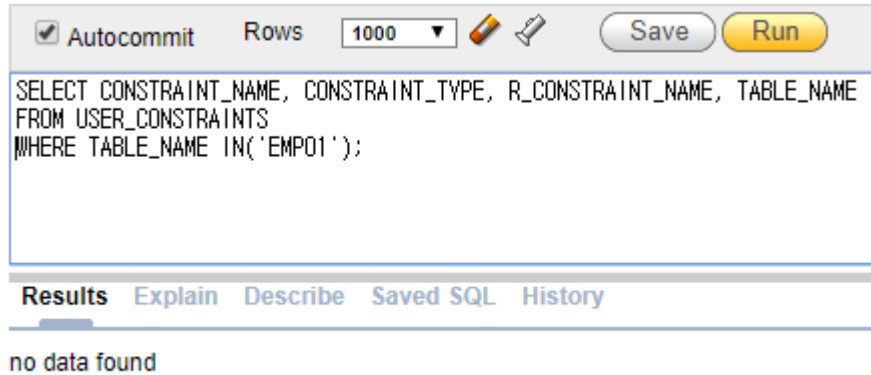
```
DROP TABLE EMP01;
```

EMP01 테이블을 삭제했습니다.

```
CREATE TABLE EMP01(  
    EMPNO NUMBER(4),  
    ENAME VARCHAR2(10),  
    JOB VARCHAR2(9),  
    DEPTNO NUMBER(2)  
);
```

EMP01 테이블을 생성했습니다.
EMPNO 크기 4의 NUMBER형
ENAME 크기 10의 VARCHAR2형
JOB 크기 9의 VARCHAR2형
DEPTNO 크기 2의 NUMBER형

과제03-1. 쿼리문 수행

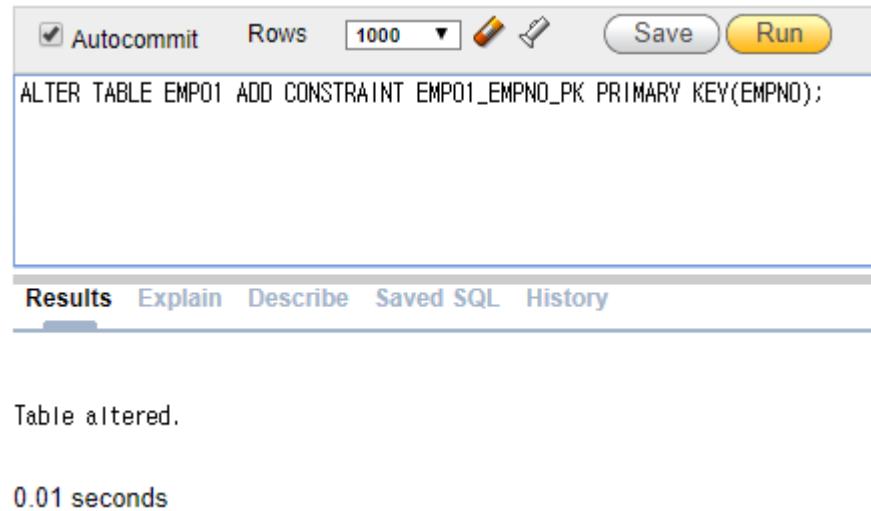


The screenshot shows a SQL query execution window. At the top, there are buttons for 'Autocommit' (checked), 'Rows' (set to 1000), and 'Save' and 'Run' buttons. The SQL query entered is: `SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME FROM USER_CONSTRAINTS WHERE TABLE_NAME IN('EMP01');`. Below the query, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, and it displays the message 'no data found'.

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP01');
```

사용자의 제약조건중에서
EMP01테이블의 제약조건이름, 제약조건타입,
외래키인 경우 무슨 기본키를 참조하는지, 그리고 테이블이름을 출력합니다.

설정된 제약조건이 없으므로 출력되지 않습니다.



The screenshot shows a SQL query execution window. At the top, there are buttons for 'Autocommit' (checked), 'Rows' (set to 1000), and 'Save' and 'Run' buttons. The SQL query entered is: `ALTER TABLE EMP01 ADD CONSTRAINT EMP01_EMPNO_PK PRIMARY KEY(EMPNO);`. Below the query, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is selected, and it displays the message 'Table altered.' and the execution time '0.01 seconds'.

```
ALTER TABLE EMP01 ADD CONSTRAINT EMP01_EMPNO_PK PRIMARY KEY(EMPNO);
```

EMP01 테이블의 EMPNO 속성이 PRIMARY KEY 값을 갖도록 하고
설정된 제약조건의 이름을 EMP01_EMPNO_PK로 설정했습니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000 Save Run

```
ALTER TABLE EMP01  
ADD CONSTRAINT EMP01_DEPTNO_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO);
```

Results Explain Describe Saved SQL History

Table altered.

0.01 seconds

☒ Autocommit Rows: 1000 Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('EMP01');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
EMP01_EMPNO_PK	P	-	EMP01
EMP01_DEPTNO_FK	R	SYS_C007184	EMP01

2 rows returned in 0.03 seconds

[Download](#)

```
ALTER TABLE EMP01  
ADD CONSTRAINT EMP01_DEPTNO_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO);
```

EMP01 테이블의 DEPTNO 속성이 FOREIGN KEY 값을 갖도록 하고
설정한 제약조건의 이름을 EMP01_DEPTNO_FK로 설정했습니다.

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('EMP01');
```

사용자의 제약조건중에서
EMP01 테이블의 제약조건이름, 제약조건타입,
외래키의 경우 무슨 기본키를 참조하는지, 그리고 테이블이름을 출력합니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000 Save Run

ALTER TABLE EMP01 MODIFY ENAME CONSTRAINT EMP01_ENAME_NN NOT NULL;

Results Explain Describe Saved SQL History

Table altered.

0.01 seconds

☒ Autocommit Rows: 1000 Save Run

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP05');

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
EMP05_ENAME_NN	C	-	EMP05
EMP05_EMPNO_PK	P	-	EMP05

2 rows returned in 0.03 seconds [Download](#)



ALTER TABLE EMP01 MODIFY ENAME CONSTRAINT EMP01_ENAME_NN NOT NULL;

EMP01 테이블의 ENAME 속성을 NOT NULL을 갖도록 하고
설정한 제약조건의 이름을 EMP01_ENAME_NN으로 설정했습니다.

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP05');

사용자의 제약조건중에서
EMP05 테이블의 제약조건이름, 제약조건타입
외래키의 경우 무슨 기본키를 참조하는지, 그리고 테이블이름을 출력합니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000   Save Run

```
SELECT * FROM EMP05;
```



Results Explain Describe Saved SQL History

EMPNO	ENAME	JOB	DEPTNO
1000	허준	사원	30
1010	홍길동	사원	50

2 rows returned in 0.00 seconds [Download](#)

SELECT * FROM EMP05;

EMP05 테이블의 모든 레코드를 출력합니다.

☒ Autocommit Rows: 1000   Save Run

```
SELECT * FROM DEPT;|
```

Results Explain Describe Saved SQL History



DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 rows returned in 0.00 seconds [Download](#)

SELECT * FROM DEPT;

DEPT 테이블의 모든 레코드를 출력합니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

SELECT * FROM EMP;

Results Explain Describe Saved SQL History

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	-	11/17/1981	5000	-	10
7698	BLAKE	MANAGER	7839	05/01/1981	2850	-	30
7782	CLARK	MANAGER	7839	06/09/1981	2450	-	10
7566	JONES	MANAGER	7839	04/02/1981	2975	-	20
7788	SCOTT	ANALYST	7566	12/09/1982	3000	-	20
7902	FORD	ANALYST	7566	12/03/1981	3000	-	20
7369	SMITH	CLERK	7902	12/17/1980	800	-	20
7499	ALLEN	SALESMAN	7698	02/20/1981	1600	300	30
7521	WARD	SALESMAN	7698	02/22/1981	1250	500	30
7654	MARTIN	SALESMAN	7698	09/28/1981	1250	1400	30
7844	TURNER	SALESMAN	7698	09/08/1981	1500	0	30
7876	ADAMS	CLERK	7788	01/12/1983	1100	-	20
7900	JAMES	CLERK	7698	12/03/1981	950	-	30
7934	MILLER	CLERK	7782	01/23/1982	1300	-	10

14 rows returned in 0.00 seconds [Download](#)

SELECT * FROM EMP;

EMP 테이블의 모든 레코드를 출력합니다.

과제03-1. 쿼리문 수행

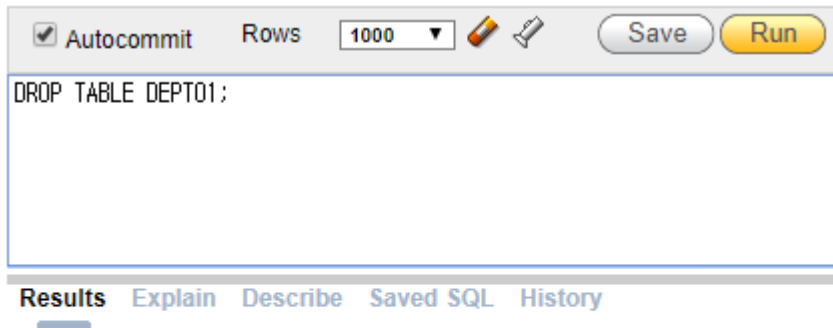


Table dropped.

0.01 seconds

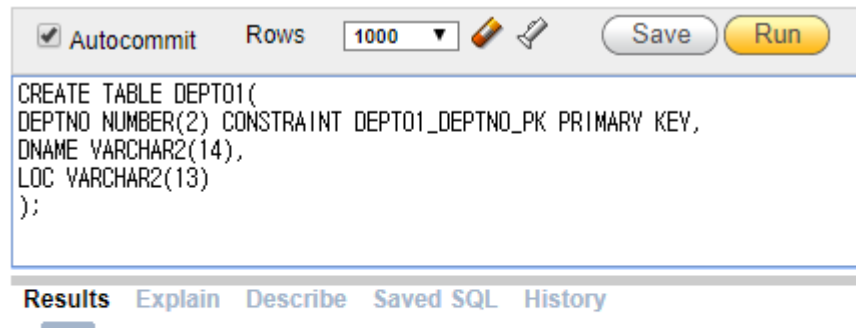


Table created.

0.01 seconds



```
DROP TABLE DEPT01;
```

DEPT01 테이블을 삭제했습니다.

```
CREATE TABLE DEPT01(  
    DEPTNO NUMBER(2) CONSTRAINT DEPT01_DEPTNO_PK PRIMARY KEY,  
    DNAME VARCHAR2(14),  
    LOC VARCHAR2(13)  
);
```

DEPT01 테이블을 생성했습니다.
DEPTNO PRIMARY KEY 값을 갖는 크기 2의 NUMBER형
설정한 제약조건의 이름을 DEPT01_DEPTNO_PK로 설정
DNAME 크기 14의 VARCHAR2형
LOC 크기 13의 VARCHAR2형

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('DEPT01');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
DEPT01_DEPTNO_PK	P	-	DEPT01

1 rows returned in 0.03 seconds [Download](#)



☒ Autocommit Rows 1000   Save Run

```
INSERT INTO DEPT01 VALUES(10, '경리부', '서울');
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO DEPT01 VALUES(20, '인사부', '인천');
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('DEPT01');
```

사용자의 제약조건중에서
DEPT01 테이블의 제약조건이름, 제약조건타입
외래키의 경우 무슨 기본키를 참조하는지, 그리고 테이블이름을 출력합니다.



```
INSERT INTO DEPT01 VALUES(10, '경리부', '서울');
```

DEPT01 테이블에 (10, '경리부', '서울')값을 삽입했습니다.

```
INSERT INTO DEPT01 VALUES(20, '인사부', '인천');
```

DEPT01 테이블에 (20, '인사부', '인천')값을 삽입했습니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000   Save Run

```
SELECT * FROM DEPT01;
```



Results Explain Describe Saved SQL History

DEPTNO	DNAME	LOC
10	경리부	서울
20	인사부	인천

2 rows returned in 0.00 seconds [Download](#)

SELECT * FROM DEPT01;

DEPT01 테이블의 모든 레코드를 출력합니다.

☒ Autocommit Rows: 1000   Save Run

```
COMMIT;
```

Results Explain Describe Saved SQL History

Commit statement not applicable. All statements are automatically committed.

COMMIT;

지금까지의 결과를 저장합니다.

autocommit 옵션을 체크하면 쿼리문을 수행할때마다 COMMIT을 해주지 않아도 자동으로 결과를 저장합니다.

과제03-1. 쿼리문 수행

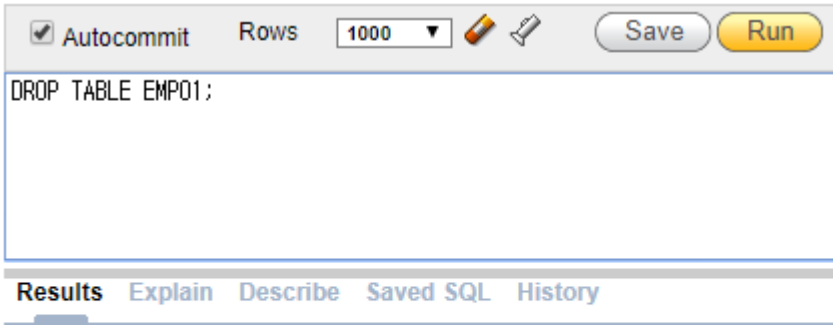


Table dropped.

0.01 seconds

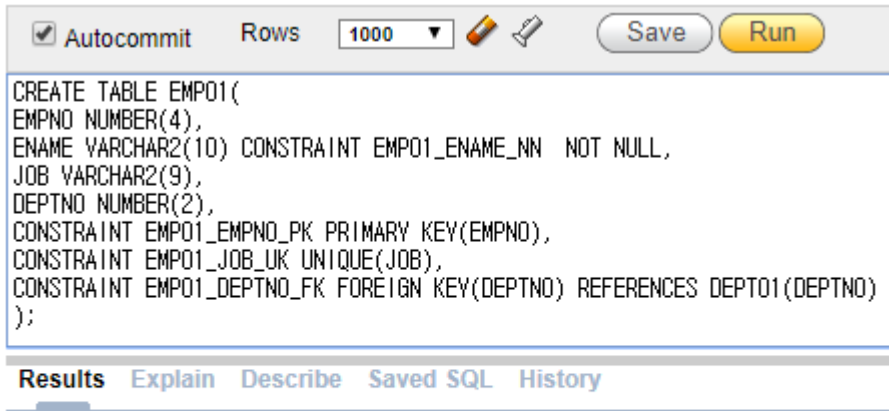


Table created.

0.01 seconds

DROP TABLE EMP01;

EMP01 테이블을 삭제했습니다.

```
CREATE TABLE EMP01(  
  EMPNO NUMBER(4),  
  ENAME VARCHAR2(10) CONSTRAINT EMP01_ENAME_NN NOT NULL,  
  JOB VARCHAR2(9),  
  DEPTNO NUMBER(2),  
  CONSTRAINT EMP01_EMPNO_PK PRIMARY KEY(EMPNO),  
  CONSTRAINT EMP01_JOB_UK UNIQUE(JOB),  
  CONSTRAINT EMP01_DEPTNO_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT01(DEPTNO)  
);
```

EMP01 테이블을 생성합니다.

EMPNO PRIMARY KEY 값을 가지는 크기 4의 NUMBER형

설정한 제약조건의 이름을 EMP01_EMPNO_PK로 설정

ENAME 크기 10의 VARCHAR2형



JOB UNIQUE 값을 갖는 크기9의 VARCHAR2형

설정한 제약조건의 이름을 EMP01_JOB_UK로 설정

DEPTNO FOREIGN KEY 값을 가지는 크기2의 NUMBER형

설정한 제약조건의 이름을 EMP01_DEPTNO_FK로 설정

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP01');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
EMP01_ENAME_NN	C	-	EMP01
EMP01_EMPNO_PK	P	-	EMP01
EMP01_JOB_UK	U	-	EMP01
EMP01_DEPTNO_FK	R	DEPT01_DEPTNO_PK	EMP01

4 rows returned in 0.00 seconds [Download](#)

☒ Autocommit Rows 1000   Save Run

```
SELECT * FROM DEPT01;
```

Results Explain Describe Saved SQL History

DEPTNO	DNAME	LOC
10	경리부	서울
20	인사부	인천

2 rows returned in 0.00 seconds [Download](#)



```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP01');
```

사용자의 제약조건중에서
EMP01 테이블의 제약조건이름, 제약조건타입
외래키의 경우 무슨 기본키를 참조하는지, 그리고 테이블이름을 출력합니다.

```
SELECT * FROM DEPT01;
```

DEPT01 테이블의 모든 레코드를 출력했습니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP01');
```



Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	R_CONSTRAINT_NAME	TABLE_NAME
EMP01_ENAME_NN	C	-	EMP01
EMP01_EMPNO_PK	P	-	EMP01
EMP01_JOB_UK	U	-	EMP01
EMP01_DEPTNO_FK	R	DEPT01_DEPTNO_PK	EMP01

4 rows returned in 0.00 seconds [Download](#)

SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, R_CONSTRAINT_NAME, TABLE_NAME
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP01');

사용자의 제약조건중에서
EMP01 테이블의 제약조건이름, 제약조건타입
외래키의 경우 무슨 기본키를 참조하는지, 그리고 테이블이름을 출력합니다.

☒ Autocommit Rows 1000   Save Run

```
INSERT INTO EMP01 VALUES(1000, '허준', '사원', 10);
```



Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

INSERT INTO EMP01 VALUES(1000, '허준', '사원', 10);

EMP01 테이블에 (1000, '허준', '사원', 10)값을 삽입했습니다.

☒ Autocommit Rows 1000   Save Run



```
INSERT INTO EMP01 VALUES(1010, '홍길동', '사원', 50);
```

ORA-00001: unique constraint (HOMEWORK3.EMP01_JOB_UK) violated

INSERT INTO EMP01 VALUES(1010, '홍길동', '사원', 50);

EMP01 테이블에 (1010, '홍길동', '사원', 50)값을 삽입했습니다.

과제03-1. 쿼리문 수행

☒ Autocommit Rows 1000   Save Run

```
SELECT * FROM EMP01;
```



Results Explain Describe Saved SQL History

EMPNO	ENAME	JOB	DEPTNO
1000	허준	사원	10

1 rows returned in 0.00 seconds [Download](#)

SELECT * FROM EMP01;

EMP01 테이블의 모든 레코드를 출력했습니다.



☒ Autocommit Rows 1000   Save Run

```
DELETE FROM DEPT01 WHERE DEPTNO=10;
```

DELETE FROM DEPT01 WHERE DEPTNO=10;

DEPT01 테이블에서 DEPTNO컬럼의 값이 10인 레코드를 삭제합니다.

ORA-02292: integrity constraint (HOMEWORK3.EMP01_DEPTNO_FK) violated - child record found

☒ Autocommit Rows 1000   Save Run

```
ALTER TABLE EMP01 DISABLE CONSTRAINT EMP01_DEPTNO_FK;
```

Results Explain Describe Saved SQL History

ALTER TABLE EMP01 DISABLE CONSTRAINT EMP01_DEPTNO_FK;

EMP01테이블의 EMP01_DEPTNO_FK 속성을 제거합니다.

Table altered.

0.01 seconds

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000 Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_CONSTRAINT_NAME, STATUS
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP01');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	R_CONSTRAINT_NAME	STATUS
EMP01_ENAME_NN	C	EMP01	-	ENABLED
EMP01_EMPNO_PK	P	EMP01	-	ENABLED
EMP01_JOB_UK	U	EMP01	-	ENABLED
EMP01_DEPTNO_FK	R	EMP01	DEPT01_DEPTNO_PK	DISABLED

4 rows returned in 0.03 seconds [Download](#)

☒ Autocommit Rows: 1000 Save Run

```
SELECT * FROM DEPT01;
```

Results Explain Describe Saved SQL History

DEPTNO	DNAME	LOC
10	경리부	서울
20	인사부	인천

2 rows returned in 0.00 seconds [Download](#)

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_CONSTRAINT_NAME, STATUS
FROM USER_CONSTRAINTS
WHERE TABLE_NAME IN('EMP01');
```



사용자의 제약조건중에서
EMP01 테이블의 제약조건이름, 제약조건타입, 테이블이름,
외래키의 경우 무슨 기본키를 참조하는지, 그리고 제약조건의 상태를 출력합니다.

DISABLED 시킨 DEPT01_DEPTNO_PK를 확인합니다.

```
SELECT * FROM DEPT01;
```

DEPT01 테이블의 모든 레코드를 출력했습니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows: 1000   Save Run

```
ALTER TABLE EMP01 ENABLE CONSTRAINT EMP01_DEPTNO_FK;
```

Results Explain Describe Saved SQL History

Table altered.

0.01 seconds

☒ Autocommit Rows: 1000   Save Run

```
INSERT INTO DEPT01 VALUES(10, '경리부', '서울');
```

ALTER TABLE EMP01 ENABLE CONSTRAINT EMP01_DEPTNO_FK;

EMP01 테이블에서 DISABLED 시켰던
EMP01_DEPTNO_FK 제약조건의 상태를 ENABLE로 변경합니다.

INSERT INTO DEPT01 VALUES(10, '경리부', '서울');

DEPT01 테이블에 (10, '경리부', '서울')값을 삽입했습니다.

ORA-00001: unique constraint (HOMEWORK3.DEPT01_DEPTNO_PK) violated

과제03-1. 쿼리문 수행

☒ Autocommit Rows: 1000

```
CREATE TABLE TEST(  
  ID NUMBER(4) DEFAULT 1 PRIMARY KEY,  
  NAME VARCHAR2(20) NULL UNIQUE  
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

☒ Autocommit Rows: 1000

```
CREATE TABLE EMPLOYEE(  
  EMP_NO NUMBER(4),  
  EMP_NAME VARCHAR2(20),  
  SALARY NUMBER(6),  
  BIRTHDAY DATE  
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

```
CREATE TABLE TEST(  
  ID NUMBER(4) DEFAULT 1 PRIMARY KEY,  
  NAME VARCHAR2(20) NULL UNIQUE  
);
```

TEST 테이블을 생성합니다.
ID DEFAULT값으로 1을 갖고 PRIMARY KEY 속성인 크기 4의 NUMBER형
NAME NULL 값을 가지지 못하는 UNIQUE 속성인 크기 20의 VARCHAR2형

```
CREATE TABLE EMPLOYEE(  
  EMP_NO NUMBER(4),  
  EMP_NAME VARCHAR2(20),  
  SALARY NUMBER(6),  
  BIRTHDAY DATE  
);
```

EMPLOYEE 테이블을 생성합니다.
EMP_NO 크기 4의 NUMBER형
EMP_NAME 크기 20의 VARCHAR2형
SALARY 크기 6의 NUMBER형
BIRTHDAY DATE형

과제03-1. 쿼리문 수행

```
Autocommit Rows 1000 Save Run
CREATE TABLE PROJECT(
  PRO_NO NUMBER(4),
  PRO_CONTENT VARCHAR2(100),
  START_DATE DATE,
  FINISH_DATE DATE
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

```
Autocommit Rows 1000 Save Run
CREATE TABLE SPECIALTY(
  EMP_NO NUMBER(4),
  SPECIALTY VARCHAR2(20)
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

```
Autocommit Rows 1000 Save Run
CREATE TABLE ASSIGN(
  EMP_NO NUMBER(4),
  PRO_NO NUMBER(4)
);
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

```
CREATE TABLE PROJECT(
  PRO_NO NUMBER(4),
  PRO_CONTENT VARCHAR2(100),
  START_DATE DATE,
  FINISH_DATE DATE
);
```

```
// PROJECT 테이블을 생성했습니다.
// PRO_CONTENT 크기 100인 VARCHAR2형
// START_DATE DATE형
// FINISH_DATE DATE형
```

```
CREATE TABLE SPECIALTY(
  EMP_NO NUMBER(4),
  SPECIALTY VARCHAR2(20)
);
```

```
// SPECIALTY 테이블 생성
// EMP_NO 크기 4의 NUMBER형
// SPECIALTY 크기 20의 VARCHAR2형
```

```
CREATE TABLE ASSIGN(
  EMP_NO NUMBER(4),
  PRO_NO NUMBER(4)
);
```

```
// ASSIGN 테이블 생성
// EMP_NO 크기 4의 NUMBER형
// PRO_NO 크기 4의 NUMBER형
```

과제03-1. 쿼리문 수행

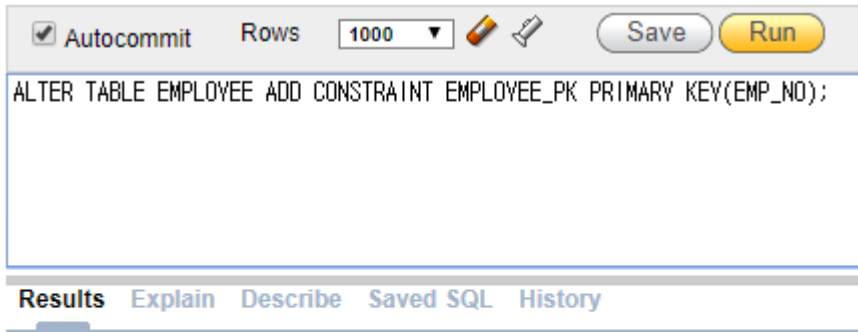


Table altered.

0.01 seconds

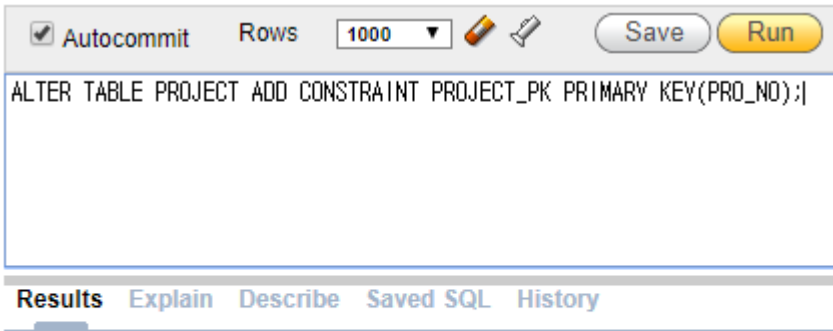


Table altered.

0.00 seconds

```
ALTER TABLE EMPLOYEE ADD CONSTRAINT EMPLOYEE_PK PRIMARY KEY(EMP_NO);
```

EMPLOYEE 테이블의 EMP_NO 속성이 PRIMARY KEY 값을 갖도록 하고
설정한 제약조건의 이름을 EMPLOYEE_PK로 설정했습니다.

```
ALTER TABLE PROJECT ADD CONSTRAINT PROJECT_PK PRIMARY KEY(PRO_NO);
```

PROJECT 테이블의 PRO_NO 속성이 PRIMARY KEY 값을 갖도록 하고
설정한 제약조건의 이름을 PROJECT_PK로 설정했습니다.

과제03-1. 쿼리문 수행

```
ALTER TABLE SPECIALTY ADD CONSTRAINT SPECIALTY_PK PRIMARY KEY(EMP_NO, SPECIALTY);
```

Results Explain Describe Saved SQL History

Table altered.

0.00 seconds

```
ALTER TABLE ASSIGN ADD CONSTRAINT ASSIGN_PK PRIMARY KEY(EMP_NO, PRO_NO);
```

Results Explain Describe Saved SQL History

Table altered.

0.01 seconds



ALTER TABLE SPECIALTY
ADD CONSTRAINT SPECIALTY_PK PRIMARY KEY(EMP_NO, SPECIALTY);

SPECIALTY 테이블의 EMP_NO 컬럼과 SPECIALTY 컬럼 두 개가
동시에 PRIMARY KEY 값을 갖도록 하고
설정된 제약조건의 이름을 SPECIALTY_PK로 설정했습니다.

ALTER TABLE ASSIGN
ADD CONSTRAINT ASSIGN_PK PRIMARY KEY(EMP_NO, PRO_NO);

ASSIGN 테이블의 EMP_NO 컬럼과 PRO_NO 컬럼 두 개가
동시에 PRIMARY KEY 값을 갖도록 하고
설정된 제약조건의 이름을 SPECIALTY_PK로 설정했습니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
ALTER TABLE SPECIALTY ADD CONSTRAINT SPECIALTY_FK FOREIGN KEY(EMP_NO) REFERENCES EMPLOYEE(EMP_NO);
```

Results Explain Describe Saved SQL History

Table altered.

0.01 seconds

☒ Autocommit Rows 1000   Save Run

```
ALTER TABLE ASSIGN ADD CONSTRAINT ASSIGN_PROJECT_FK FOREIGN KEY(PRO_NO) REFERENCES PROJECT;
```

Results Explain Describe Saved SQL History

Table altered.

0.00 seconds



```
ALTER TABLE SPECIALTY  
ADD CONSTRAINT SPECIALTY_FK FOREIGN KEY(EMP_NO) REFERENCES EMPLOYEE(EMP_NO);
```

SPECIALTY 테이블에서 EMP_NO 컬럼이 FOREIGN KEY 값을 갖도록 하고
설정된 제약조건의 이름을 SPECIALTY_FK로 설정했습니다.

```
ALTER TABLE ASSIGN  
ADD CONSTRAINT ASSIGN_PROJECT_FK FOREIGN KEY(PRO_NO) REFERENCES PROJECT;
```

ASSIGN 테이블에서 PRO_NO 컬럼이 FOREIGN KEY 값을 갖도록 하고
설정된 제약조건의 이름을 ASSIGN_PROJECT_FK로 설정했습니다.

과제03-1. 쿼리문 수행



☒ Autocommit Rows 1000   Save Run

```
ALTER TABLE ASSIGN ADD CONSTRAINT ASSIGN_EMPLOYEE_FK FOREIGN KEY(EMP_NO) REFERENCES EMPLOYEE;
```

Results Explain Describe Saved SQL History

Table altered.

0.00 seconds

☒ Autocommit Rows 1000   Save Run

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_CONSTRAINT_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('EMPLOYEE', 'PROJECT', 'SPECIALTY', 'ASSIGN');
```

Results Explain Describe Saved SQL History

CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	R_CONSTRAINT_NAME
SPECIALTY_FK	R	SPECIALTY	EMPLOYEE_PK
ASSIGN_EMPLOYEE_FK	R	ASSIGN	EMPLOYEE_PK
ASSIGN_PROJECT_FK	R	ASSIGN	PROJECT_PK
SPECIALTY_PK	P	SPECIALTY	-
PROJECT_PK	P	PROJECT	-
EMPLOYEE_PK	P	EMPLOYEE	-
ASSIGN_PK	P	ASSIGN	-

7 rows returned in 0.04 seconds

[Download](#)

ALTER TABLE ASSIGN

ADD CONSTRAINT ASSIGN_EMPLOYEE_FK FOREIGN KEY(EMP_NO) REFERENCES EMPLOYEE;

ASSIGN 테이블의 EMP_NO 컬럼이 FOREIGN KEY 값을 갖도록 하고
설정한 제약조건의 이름을 ASSIGN_EMPLOYEE로 설정했습니다.

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_CONSTRAINT_NAME  
FROM USER_CONSTRAINTS  
WHERE TABLE_NAME IN('EMPLOYEE', 'PROJECT', 'SPECIALTY', 'ASSIGN');
```

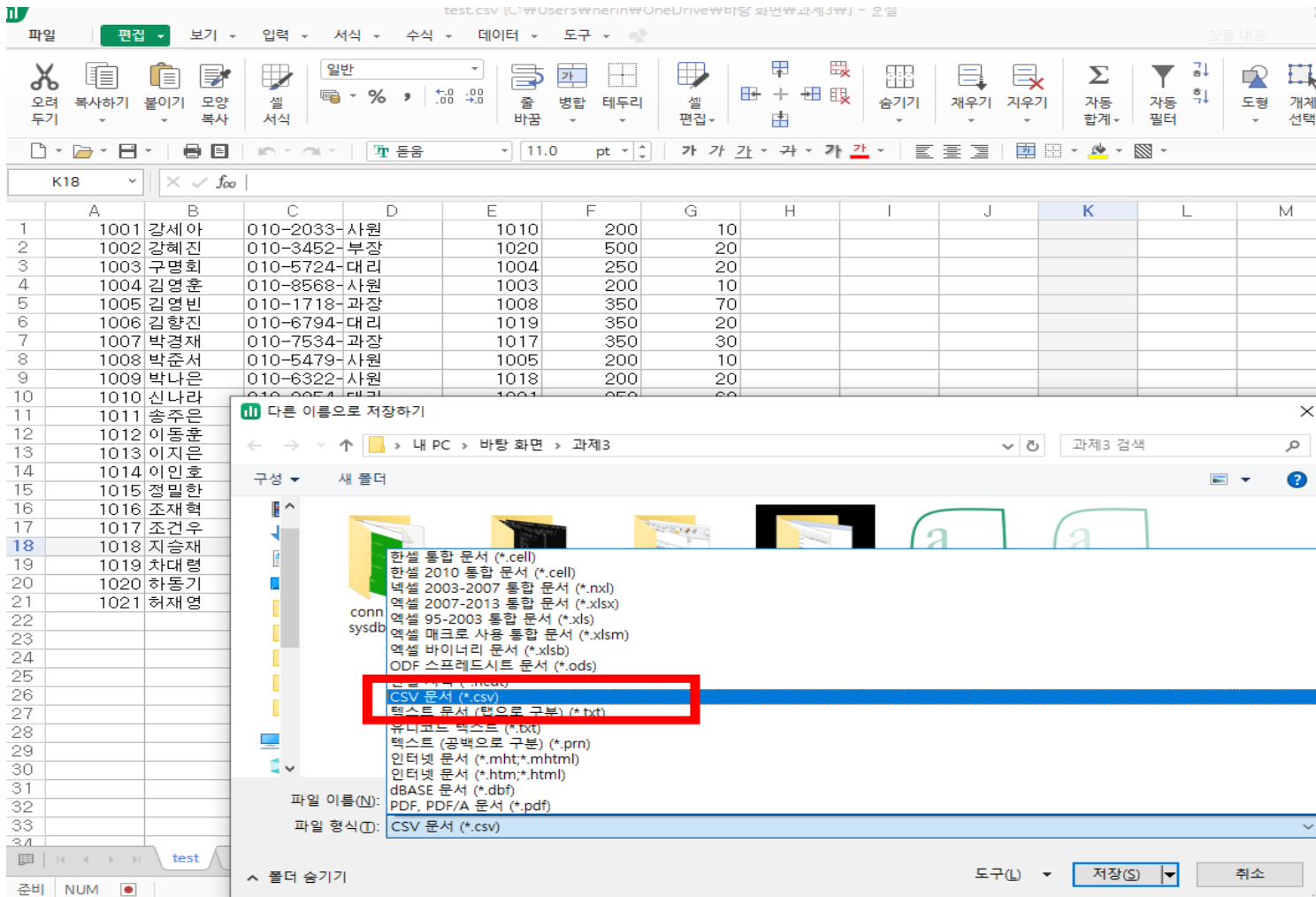
사용자의 제약조건중에서

EMPLOYEE, PROJECT, SPECIALTY, ASSIGN 테이블의

제약조건이름, 제약조건타입, 테이블이름, 외래키인 경우 참조하는 기본키를 출력했습니다.

과제04. GET STARTED >> LOAD CSV file

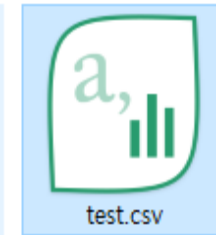
0. excel file > csv파일 생성, csv파일 확인



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1001	강세아	010-2033-사원		1010	200	10						
2	1002	강혜진	010-3452-부장		1020	500	20						
3	1003	구명희	010-5724-대리		1004	250	20						
4	1004	김영훈	010-8568-사원		1003	200	10						
5	1005	김영빈	010-1718-과장		1008	350	70						
6	1006	김향진	010-6794-대리		1019	350	20						
7	1007	박경재	010-7534-과장		1017	350	30						
8	1008	박준서	010-5479-사원		1005	200	10						
9	1009	박나은	010-6322-사원		1018	200	20						
10	1010	신나라	010-2954-대리		1001	250	60						
11	1011	송주은	010-1907-사원		1016	200	20						
12	1012	이동훈	010-1044-사원		1013	200	30						
13	1013	이지은	010-9673-대리		1012	250	40						
14	1014	이인호	010-8724-사장		800	20							
15	1015	정밀한	010-9951-사원		1012	200	10						
16	1016	조재혁	010-5531-과장		1011	350	60						
17	1017	조건우	010-3865-사원		1007	200	20						
18	1018	지승재	010-1211-부장		1009	500	70						
19	1019	차대령	010-4632-사원		1006	200	20						
20	1020	하동기	010-5688-사원		1002	200	10						
21	1021	허재영	010-5688-대리		1015	250	60						

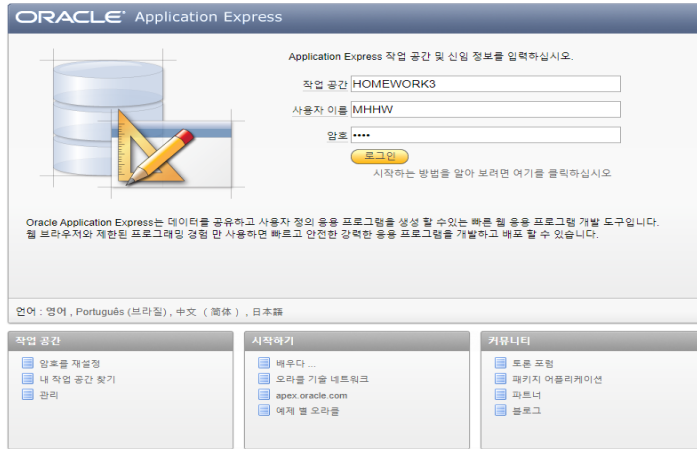
The 'Save As' dialog box shows the file type set to 'CSV file (*.csv)' and the file name 'test.csv'.



1001,강세아,010-2033-1131,사원,1010,200,10
1002,강혜진,010-3452-1132,부장,1020,500,20
1003,구명희,010-5724-1133,대리,1004,250,20
1004,김영훈,010-8568-1134,사원,1003,200,10
1005,김영빈,010-1718-1135,과장,1008,350,70
1006,김향진,010-6794-1136,대리,1019,350,20
1007,박경재,010-7534-1137,과장,1017,350,30
1008,박준서,010-5479-1138,사원,1005,200,10
1009,박나은,010-6322-1139,사원,1018,200,20
1010,신나라,010-2954-1140,대리,1001,250,60
1011,송주은,010-1907-1141,사원,1016,200,20
1012,이동훈,010-1044-1142,사원,1013,200,30
1013,이지은,010-9673-1143,대리,1012,250,40
1014,이인호,010-8724-1144,사장,,800,20
1015,정밀한,010-9951-1145,사원,1012,200,10
1016,조재혁,010-5531-1146,과장,1011,350,60
1017,조건우,010-3865-1147,사원,1007,200,20
1018,지승재,010-1211-1148,부장,1009,500,70
1019,차대령,010-4632-1149,사원,1006,200,20
1020,하동기,010-5688-1150,사원,1002,200,10
1021,허재영,010-5688-1150,대리,1015,250,60

과제04. GET STARTED >> LOAD CSV file

1. GET STARTED 로그인



2. 로그인한 계정에서 테이블 생성

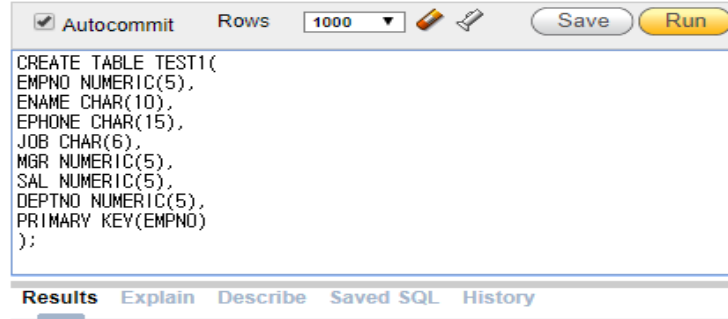
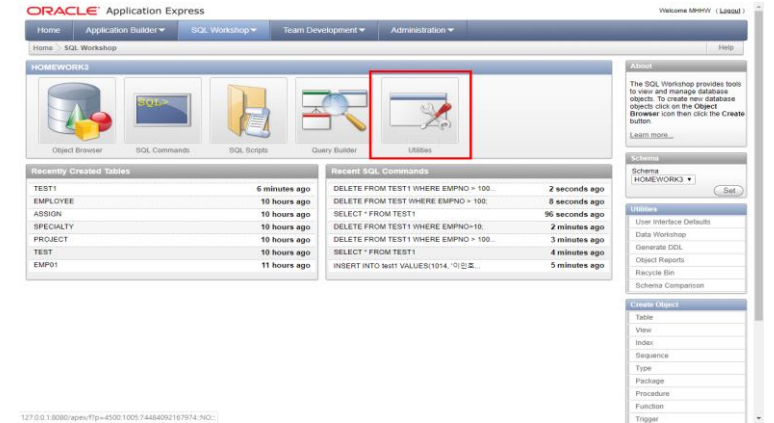


Table created.

0.01 seconds

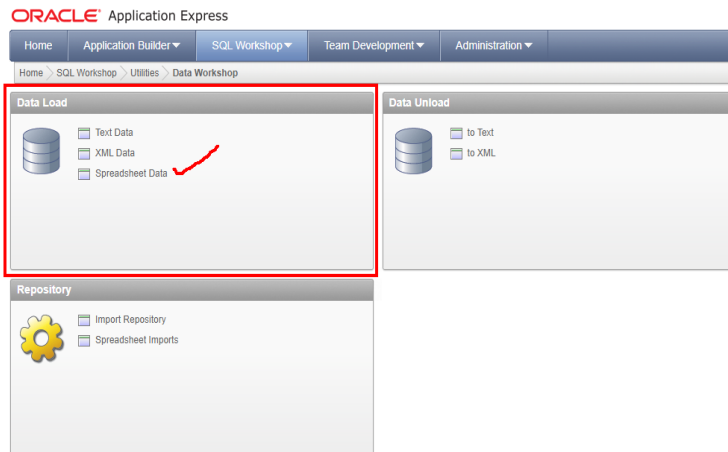
3. 유틸리티



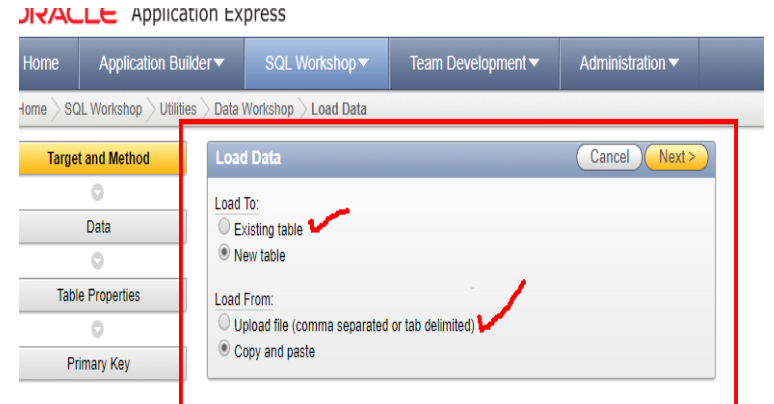
4. 데이터 워크샵



5. DATA LOAD > SpreadSheet Data

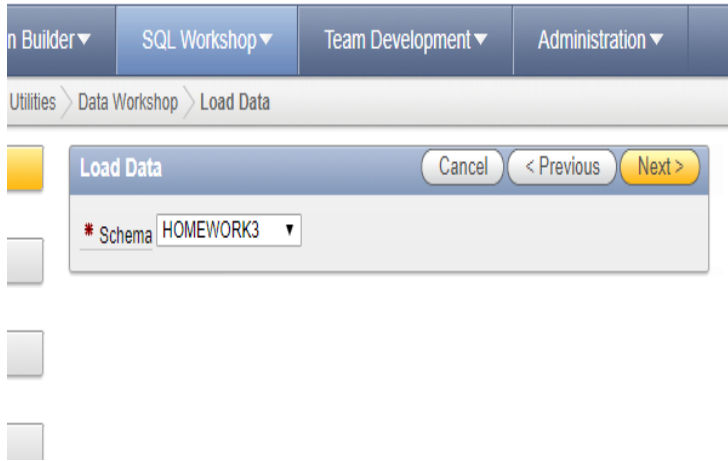


6. Load To : Existing table Load Form : Upload file

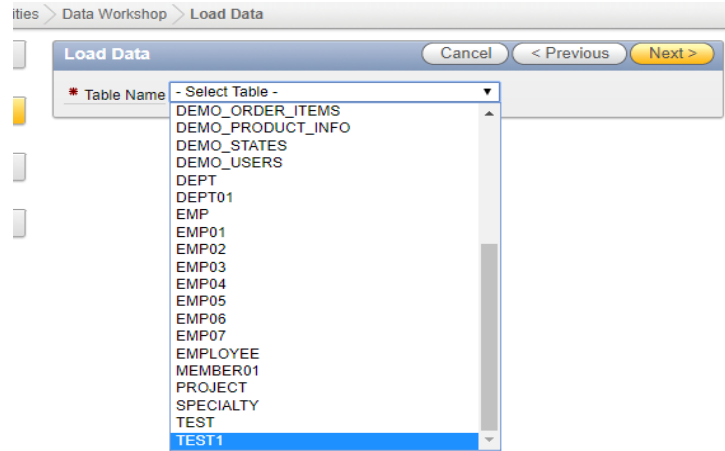


과제04. GET STARTED >> LOAD CSV file

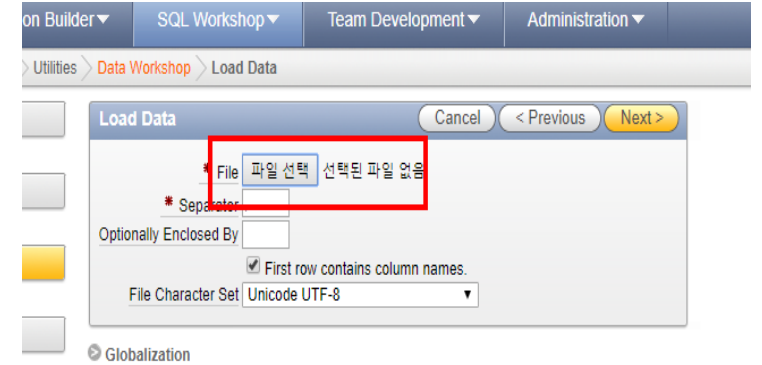
7. 현재 작업공간 선택



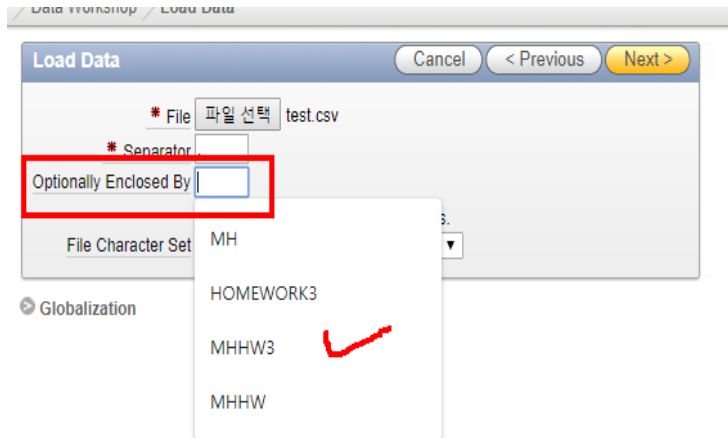
8. 2에서 생성한 테이블 선택



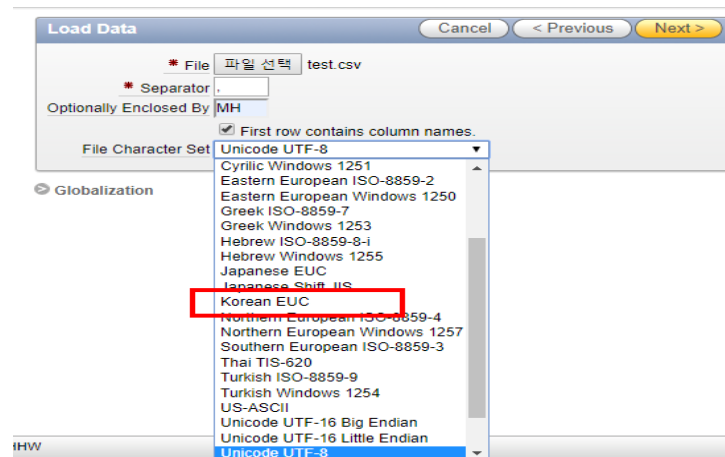
9. file 선택 > csv file load



10. 사용자 계정 선택



11. 한글설정 > Korean EUC



과제04. GET STARTED >> LOAD CSV file

12. 데이터 속성 설정

Define Column Mapping

Column Names	%	%	%	%	%	%	%
Format	%						
Upload	EMPNO - number *	Yes	Yes	Yes	Yes	Yes	Yes
Row 1	ENAME - char	강혜진	010-3452-1132	부장	1020	500	20
Row 2	EPHONE - char	구명회	010-5724-1133	대리	1004	250	20
Row 3	JOB - char	김영훈	010-8568-1134	사원	1003	200	10
Row 4	MGR - number	김영빈	010-1718-1135	과장	1008	350	70
Row 5	SAL - number	김향진	010-6794-1136	대리	1019	350	20
Row 6	DEPTNO - number	박경재	010-7534-1137	과장	1017	350	30
Row 7		박준서	010-5479-1138	사원	1005	200	10
Row 8		박나은	010-6322-1139	사원	1018	200	20
Row 9		신나라	010-2954-1140	대리	1001	250	60
Row 10		송주은	010-1907-1141	사원	1016	200	20

12. 데이터 속성 설정 완료


Define Column Mapping

Column Names	EMPNO - number *	ENAME - char	EPHONE - char	JOB - char	MGR - number	SAL - number	DEPTNO - number
Format							
Upload	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Row 1	1002	강혜진	010-3452-1132	부장	1020	500	20
Row 2	1003	구명회	010-5724-1133	대리	1004	250	20
Row 3	1004	김영훈	010-8568-1134	사원	1003	200	10
Row 4	1005	김영빈	010-1718-1135	과장	1008	350	70
Row 5	1006	김향진	010-6794-1136	대리	1019	350	20
Row 6	1007	박경재	010-7534-1137	과장	1017	350	30
Row 7	1008	박준서	010-5479-1138	사원	1005	200	10
Row 8	1009	박나은	010-6322-1139	사원	1018	200	20
Row 9	1010	신나라	010-2954-1140	대리	1001	250	60
Row 10	1011	송주은	010-1907-1141	사원	1016	200	20

과제04. GET STARTED >> LOAD CSV file

13. Data Load 완료

Show	All Import Files ▾	Set
------	--------------------	-----

Details	File	Imported By	Imported On ▾	Type	Schema	Table	Bytes	Succeeded	Failed
<input type="checkbox"/> 	test.csv	MHHW	11 seconds ago	Text Import	HOMEWORK3	TEST1	920	20	0

1-1

14. Data 확인

☒ Autocommit
 Rows

SELECT * FROM TEST1

Results

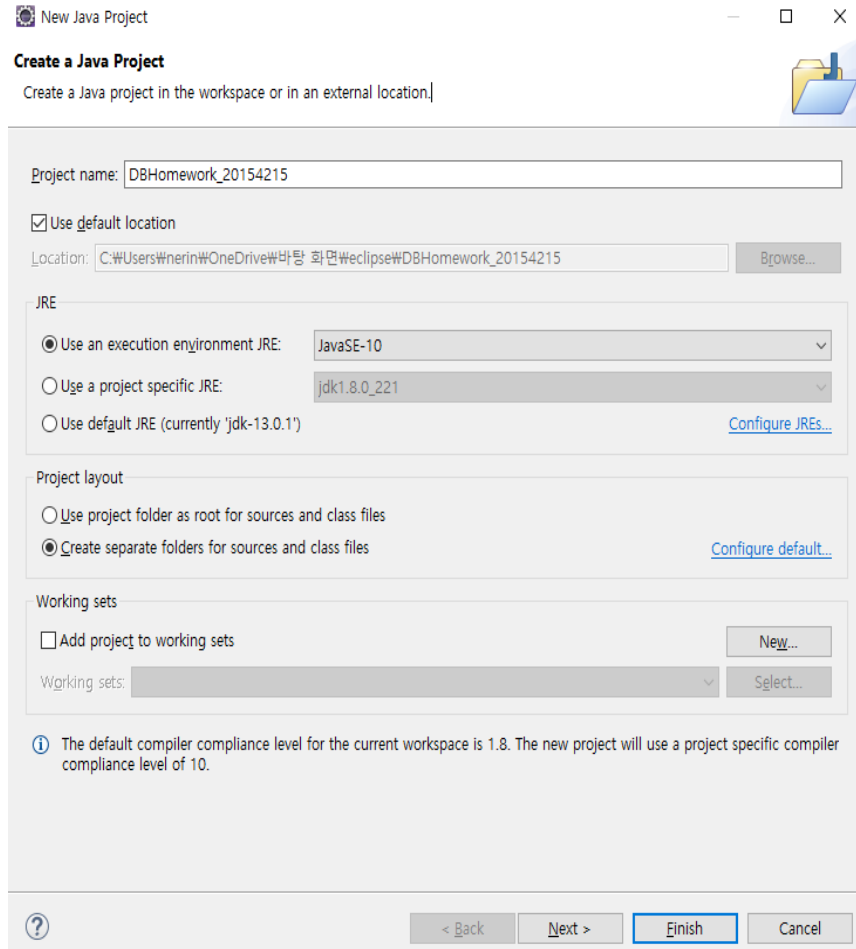
[Explain](#)
[Describe](#)
[Saved SQL](#)
[History](#)

EMPNO	ENAME	EPHONE	JOB	MGR	SAL	DEPTNO
1002	강혜진	010-3452-1132	부장	1020	500	20
1003	구명회	010-5724-1133	대리	1004	250	20
1004	김영훈	010-8568-1134	사원	1003	200	10
1005	김영빈	010-1718-1135	과장	1008	350	70
1006	김향진	010-6794-1136	대리	1019	350	20
1007	박경재	010-7534-1137	과장	1017	350	30
1008	박준서	010-5479-1138	사원	1005	200	10
1009	박나은	010-6322-1139	사원	1018	200	20
1010	신나라	010-2954-1140	대리	1001	250	60
1011	송주은	010-1907-1141	사원	1016	200	20
1012	이동훈	010-1044-1142	사원	1013	200	30
1013	이지은	010-9673-1143	대리	1012	250	40
1014	이인호	010-8724-1144	사장	-	800	20
1015	정밀한	010-9951-1145	사원	1012	200	10
1016	조재혁	010-5531-1146	과장	1011	350	60
1017	조건우	010-3865-1147	사원	1007	200	20
1018	지승재	010-1211-1148	부장	1009	500	70
1019	차대형	010-4632-1149	사원	1006	200	20
1020	하동기	010-5688-1150	사원	1002	200	10
1021	허재영	010-5688-1150	대리	1015	250	60

20 rows returned in 0.00 seconds
 [Download](#)

과제05-1. JAVA JDBC 연동

1. 프로젝트 생성 > Next



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: DBHomework_20154215

☒ Use default location

Location: C:\Users\nerin\OneDrive\바탕 화면\Eclipse\DBHomework_20154215 [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-10

☐ Use a project specific JRE: jdk1.8.0_221

☐ Use default JRE (currently 'jdk-13.0.1') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

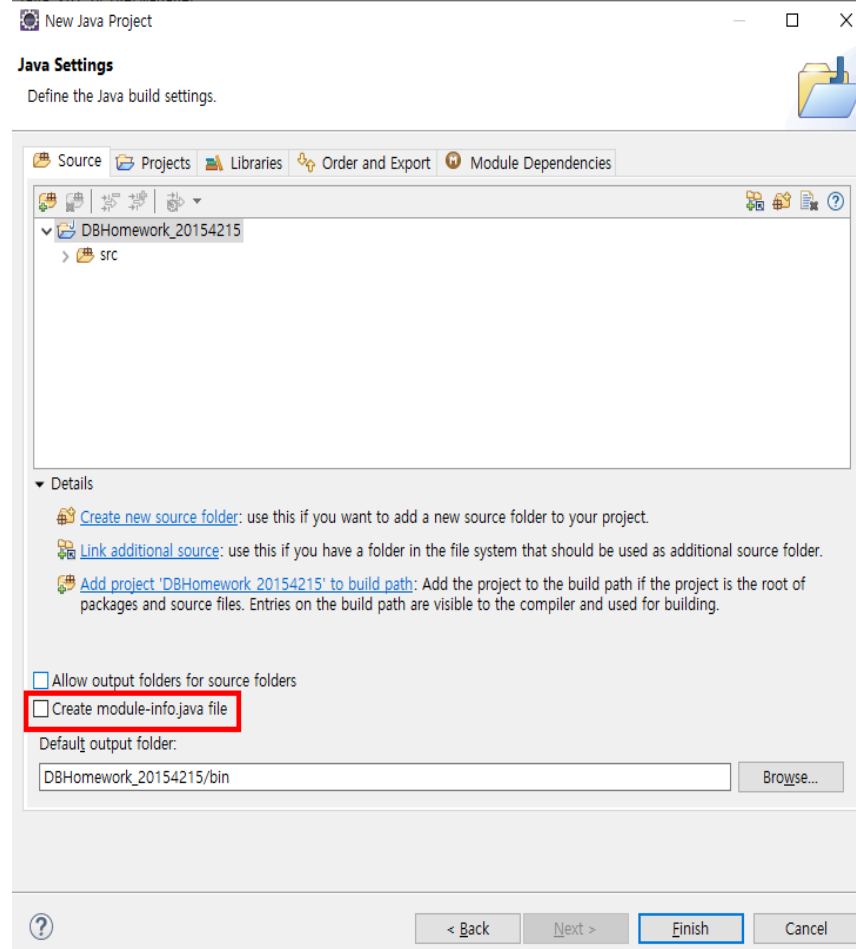
☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

① The default compiler compliance level for the current workspace is 1.8. The new project will use a project specific compiler compliance level of 10.

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

2. 현재 jdk 10 이상의 버전을 사용중이기 때문에 module을 사용안함으로 설정했습니다.



New Java Project

Java Settings

Define the Java build settings.

Source Projects Libraries Order and Export Module Dependencies

DBHomework_20154215

src

Details

[Create new source folder](#): use this if you want to add a new source folder to your project.

[Link additional source](#): use this if you have a folder in the file system that should be used as additional source folder.

[Add project 'DBHomework_20154215' to build path](#): Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

☐ Allow output folders for source folders

☐ Create module-info.java file

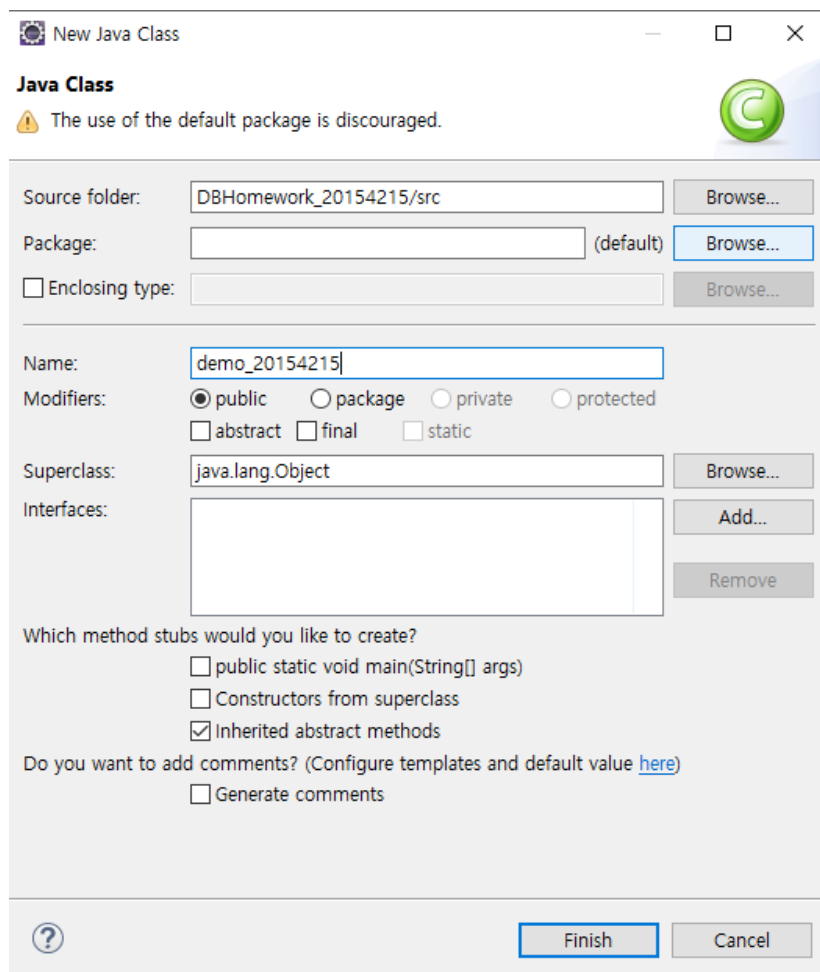
Default output folder:

DBHomework_20154215/bin [Browse...](#)

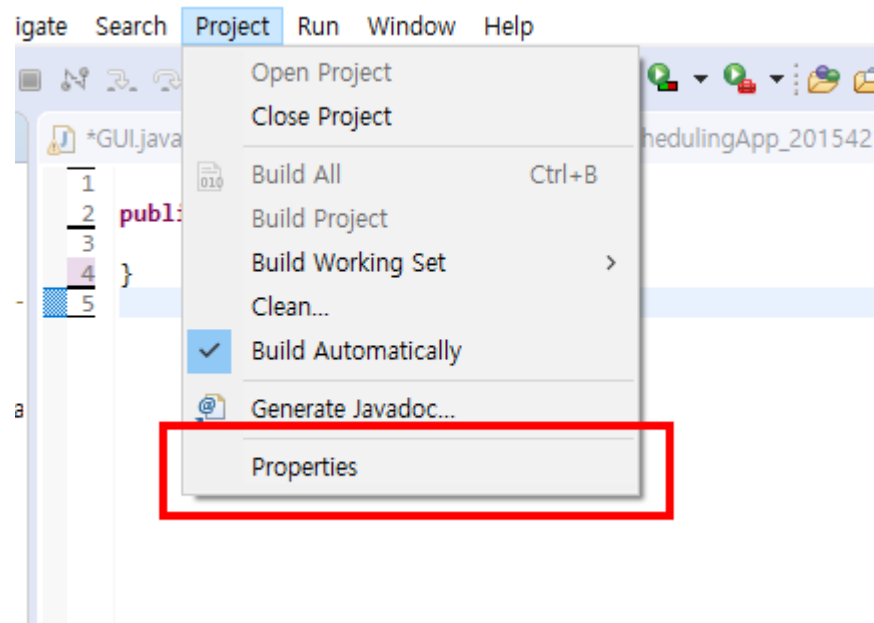
[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

과제05-1. JAVA JDBC 연동

3. 클래스 생성



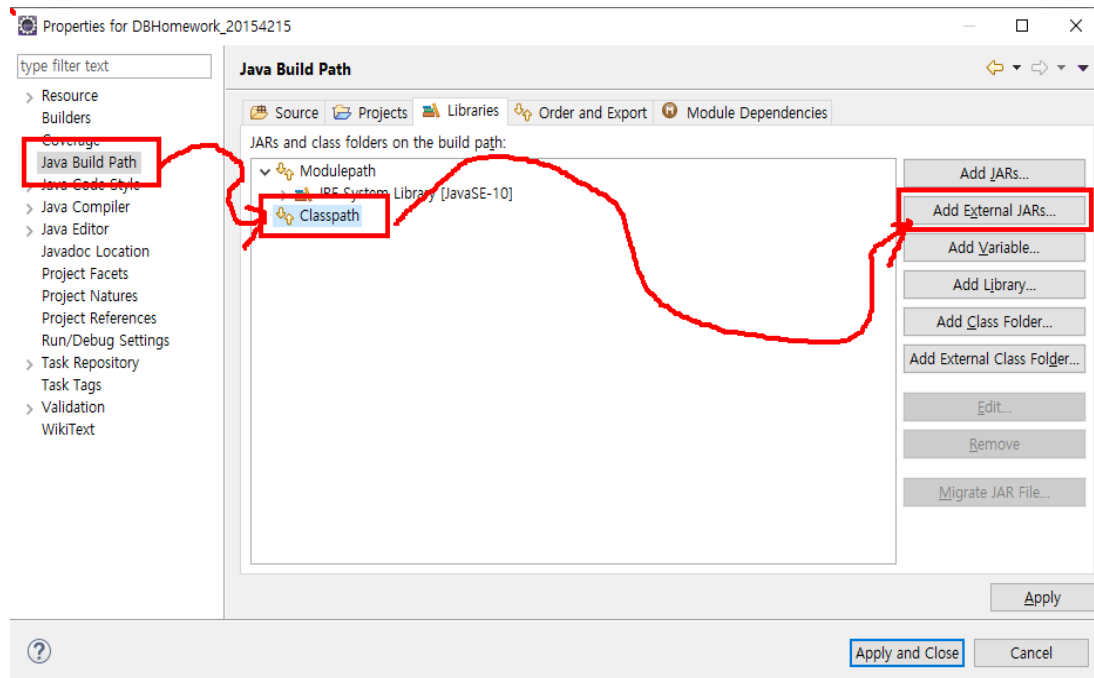
4. SQL과 JAVA를 연동시키기 위한 3가지 방법이 있는데, 저는 그 중에서 가장 간편하게 적용할 수 있는 현재 생성한 프로젝트에만 직접 jdbc 파일을 추가하려고 합니다.



과제05-1. JAVA JDBC 연동

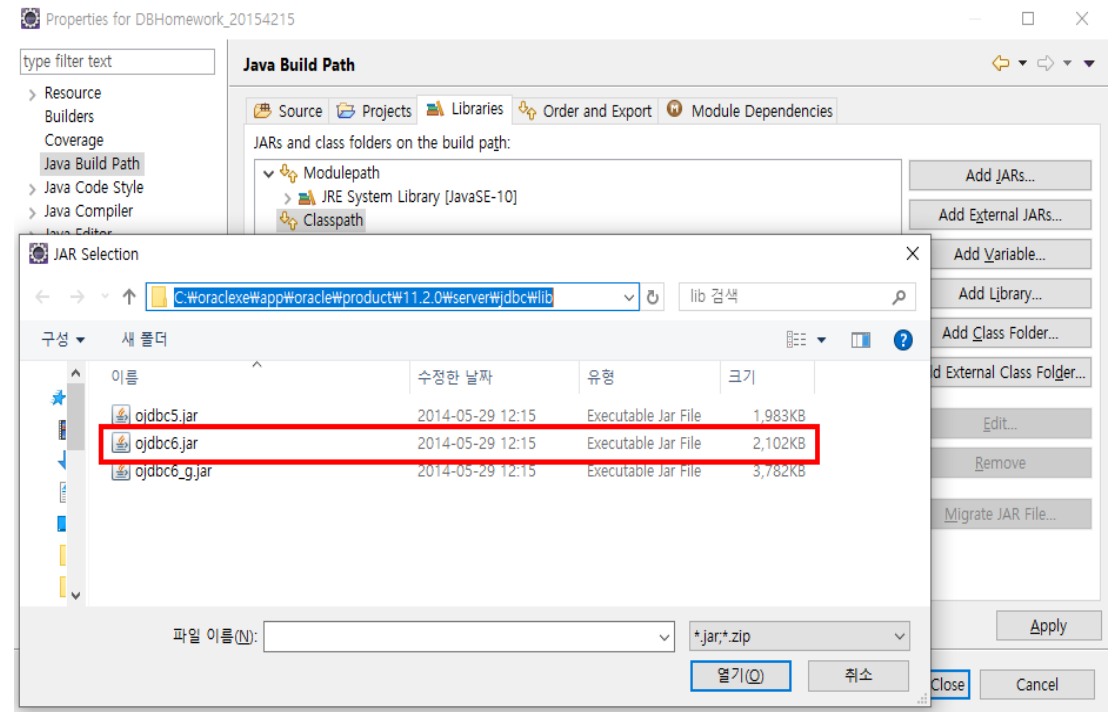
5. JDBC JAR file 추가

JAVA Build Path > Classpath에서 Add External JARs



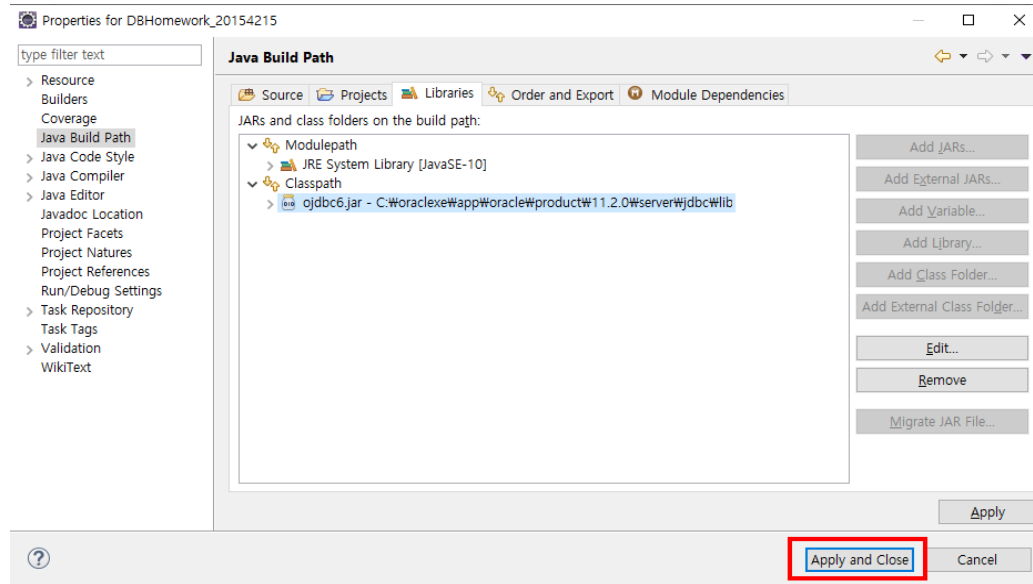
6. oracle database 11g express edition에서 jar 파일의 경로는 수정하지 않았다면 대부분 다음 경로에 있습니다.

c:\oracle\app\oracle\product\11.2.0\server\jdbc\lib

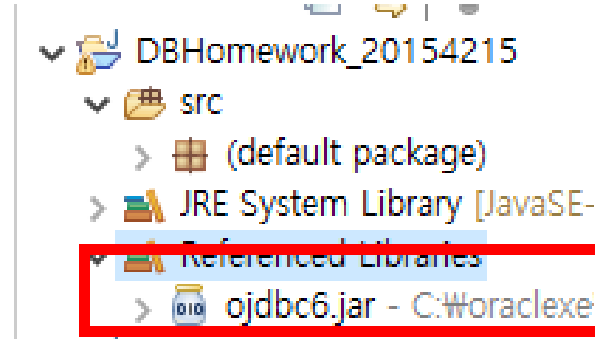


과제05-1. JAVA JDBC 연동

7. jar file 추가된것을 확인하고 Apply and Close



8. 현재 프로젝트에서 Referenced Libraries에 ojdbc6.jar 확인



과제05-2. JAVA JDBC demo_20154215

```
SQL>
SQL> CONN MHHW3/1234
Connected.
SQL>
SQL> create table test
2  ( num number,
3    name varchar(50),
4    address varchar(50),
5    phone varchar(50)
6  );

Table created.

SQL> insert into test values (1, '홍길정', '광주 서구', '010-1234-5678');

1 row created.

SQL> insert into test values (2, '최정성', '광주 북구', '010-1111-7148');

1 row created.

SQL> insert into test values (3, '김소민', '광주 남구', '010-2222-1248')
2  ;

1 row created.

SQL>
SQL>
```

0. SQL RUN에서

직접 테이블을 생성하고 데이터를 삽입합니다.

과제05-2. JAVA JDBC demo_20154215

1. import문

```
2
3- /*****
4  * import 설명 *
5  * import java.sql.Connection;
6  * Connection 객체를 자동완성으로 import 하기 위한 java 표준 java.sql.Connection 클래스
7  *
8  * import java.sql.DriverManager;
9  * java.sql의 인터페이스들을 상속해서 메소드의 문제를 구현한 JDBC 드라이버 클래스
10 *
11 * import java.sql.ResultSet;
12 * select 쿼리 실행 시 executeQuery() 메서드를 사용하며
13 * 실행 결과를 java.sql.ResultSet형으로 리턴하는데, 이 결과 집합을 사용하기 위한 클래스
14 *
15 * import java.sql.SQLException;
16 * DB 액세스 에러 또는 그 외의 에러에 관한 정보를 제공합니다.
17 *
18 * import java.sql.Statement;
19 * 정적인 쿼리문을 처리할 수 있는 Statement를 사용하기 위한 클래스
20 *****/
21- import java.sql.Connection;
22 import java.sql.DriverManager;
23 import java.sql.ResultSet;
24 import java.sql.SQLException;
25 import java.sql.Statement;
26
```

2. DB 연결, 쿼리객체 생성

```
27- /*****
28  * DB에서 Select 쿼리문 읽어오는 과정 설명 *
29  * >> Select 쿼리문을 사용하기 전에 먼저 Connection 객체를 얻어야 합니다.
30  * >> Connection 객체의 createStatement() 메서드를 호출해서 쿼리를 실행할 수 있는 Statement 객체를 얻습니다.
31  * >> 작성한 쿼리문을 executeQuery() 메서드의 인자로 전달해서 호출하면 쿼리가 실행됩니다.
32  * >> 쿼리를 실행시키면 ResultSet 객체가 반환되며, ResultSet 객체에 Select 쿼리 결과 레코드가 저장됩니다.
33 *****/
34 public class demo_20154215 {
35     // >> DB와 연결된 상태를 담은 객체 conn을 선언했습니다.
36     private Connection conn = null;
37
38-     public void connect() {
39         try {
40             /*****
41              * DriverManager 클래스에 등록 *
42              * >> 드라이버 로딩
43              * >> 드라이버 인터페이스를 구현한 클래스를 로딩합니다.
44 *****/
45             Class.forName( "oracle.jdbc.driver.OracleDriver");
46
47             /*****
48              * DriverManager.getConnection() : DB에 연결(접속)하기 위한 메소드 *
49              * >> 드라이버 매니저에게 Connection 객체를 요청합니다.
50              * >> Connection을 얻기 위해 getConnection메소드를 사용하며 인자는 ("접속url", "사용자 계정", "계정 비밀번호")입니다.
51              * >> 이 객체를 사용해서 statement로 쿼리문을 수행할 수 있습니다.
52 *****/
53             conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "MHHW3", "1234");
54             System.out.println( "성공적으로 로딩되었음" );
55
56             /*****
57              * conn.createStatement() : Statement 생성 *
58              * >> 쿼리 수행을 위한 Statement 객체를 생성합니다.
59 *****/
60             Statement stmt = conn.createStatement();
61         }
62     }
63 }
```

과제05-2. JAVA JDBC demo_20154215

3. 쿼리문 수행

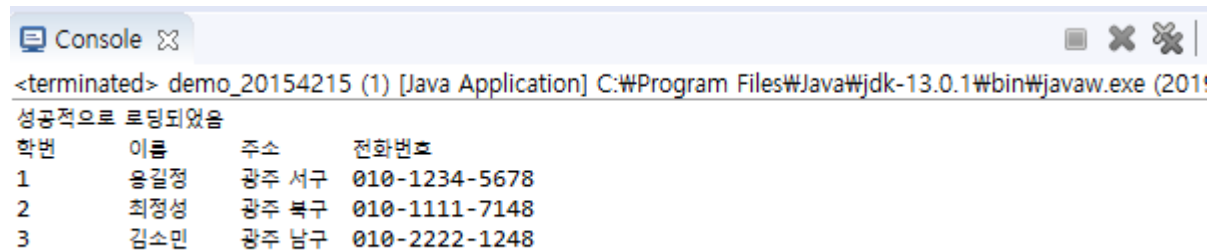
```
61
62  /*****
63   * 쿼리문 수행 *
64   * >> JDBC에서 쿼리를 작성할 때 세미콜론 (;)을 빼고 작성합니다.
65   * >> *로 모든 칼럼을 읽어오는 것보다 가져와야 할 칼럼을 직접 명시해주는게 좋다고 합니다.
66   * >> 쿼리를 한 줄로 쓰기 어려운 경우 들여쓰기를 사용해도 상관 없습니다.
67   * >> 레코드들은 ResultSet 객체에 추가됩니다.
68   *****/
69  ResultSet rset = stmt.executeQuery("select * from test");
70
71  System.out.println("학번\t이름\t주소\t전화번호");
72
73  /*****
74   * while(rset.next()) : 쿼리문 결과 출력 *
75   * >> test table
76   * >> num      name      address      phone
77   * >> 1         용길정     광주 서구     010-1234-5678
78   * >> 2         최정성     광주 북구     010-1111-7148
79   * >> 3         김소민     광주 남구     010-2222-1248
80   * >> .... 이런식으로 저장되어 있고
81   * >> next() 메소드는 현재 행에서 한 행 다음으로 이동하므로
82   * >> test의 레코드 첫행부터 출력하게 됩니다.
83   *****/
```

4. 쿼리문 출력

```
84
85  while (rset.next ())
86  {
87      for(int i = 1 ; i < 5 ; i++)
88          System.out.print(rset.getString(i)+ "\t");
89      System.out.println();
90
91      /*****
92       * 위의 반복문과 같은 역할을 수행합니다. *
93       * >> DB에서 가져오는 데이터의 타입에 맞게 직접 하나하나 호출할 수도 있습니다.
94       * >> System.out.print(rset.getString("num") + "\t");
95       * >> System.out.print(rset.getString("name") + "\t");
96       * >> System.out.print(rset.getString("address") + "\t");
97       * >> System.out.println(rset.getString("phone"));
98       *****/
99  }
100
101  catch( ClassNotFoundException e ) {
102      System.out.println( "해당 드라이버를 찾을 수 없습니다.\n" + e);
103  }
104  catch( SQLException e) {
105      System.out.println( "오라클 연결에 실패하였습니다.\n" + e);
106  }
107  }
108
109  public static void main( String[] args ){
110      new demo_20154215().connect();
111  }
112 }
```

과제05-2. JAVA JDBC demo_20154215

5. 결과



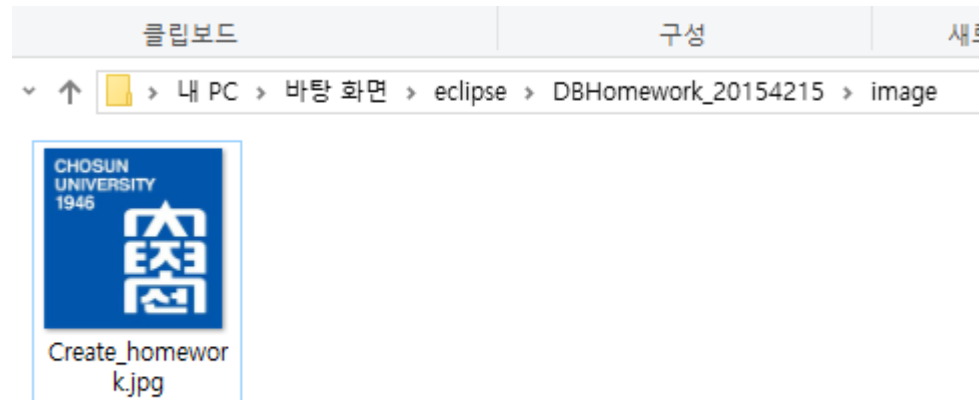
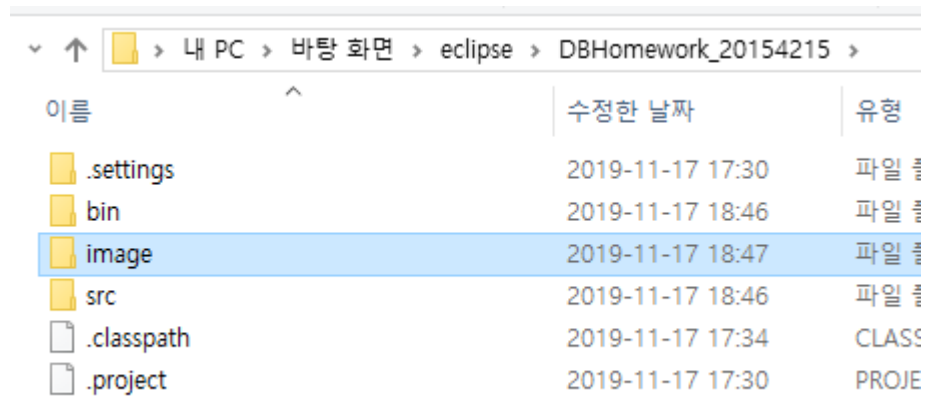
```
<terminated> demo_20154215 (1) [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (2019)
성공적으로 로딩되었음
학번      이름      주소      전화번호
1         용길정    광주 서구 010-1234-5678
2         최정성    광주 북구 010-1111-7148
3         김소민    광주 남구 010-2222-1248
```

정상적으로 출력됩니다.

과제05-3. JAVA JDBC GUI

0. 우선 과제 품을 맞추기 위해 사진을 현재 작업중인 프로젝트의 폴더에 저장합니다.

image 폴더를 생성하고 그 안에 Create_homework.jpg 파일을 저장했습니다.



과제05-3. JAVA JDBC GUI

1. import

```
38 /*****
39 * import 설명 *
40 * import java.awt.BorderLayout;
41 * >> BorderLayout은 컨테이너를 North, South, East, West, Center 모두 5개의 영역으로 나누고
42 * >> 각 영역에 하나의 컴포넌트만을 배치할 수 있도록 합니다.
43 *
44 * import java.awt.Image;
45 * >> 이미지를 만들고 수정하기 위한 클래스
46 *
47 * import java.awt.event.ActionEvent;
48 * >> 버튼이 클릭되거나 리스트, 메뉴 등이 선택되었을 때 발생하는 이벤트 클래스
49 * >> ActionListener 인터페이스의 actionPerformed() 메서드를 이용해서 처리합니다.
50 *
51 * import java.awt.event.ActionListener;
52 * >> 이벤트 인터페이스
53 *
54 * import java.sql.Connection;
55 * >> Connection 객체를 자동완성으로 import 하기 위한 java 표준 java.sql.Connection 클래스
56 *
57 * import java.sql.DriverManager;
58 * >> java.sql의 인터페이스들을 상속해서 메소드의 몸체를 구현한 JDBC 드라이버 클래스
59 *
60 * import java.sql.ResultSet;
61 * >> select 쿼리 실행 시 executeQuery() 메서드를 사용하며
62 * >> 실행 결과를 java.sql.ResultSet형으로 리턴하는데, 이 결과 집합을 사용하기 위한 클래스
63 *
64 * import java.sql.SQLException;
65 * >> DB 액세스 에러 또는 그 외의 에러에 관한 정보를 제공합니다.
66 *
67 * import java.sql.Statement;
68 * >> 정적인 쿼리문을 처리할 수 있는 Statement를 사용하기 위한 클래스
69 *
70 * import javax.swing.ImageIcon;
71 * >> 이미지에서 아이콘을 그리는 Icon 인터페이스를 구현한 클래스
72 */
```

```
38 * import javax.swing.JButton;
39 * >> 버튼을 구성하기 위해 사용하는 클래스
40 *
41 * import javax.swing.JFrame;
42 * >> 프레임을 생성시키기 위해 사용하는 클래스
43 *
44 * import javax.swing.JLabel;
45 * >> 컴포넌트에 텍스트와 이미지를 넣을 때 사용하는 클래스
46 * >> getText()와 setText(String str)을 사용할 수 있습니다.
47 *
48 * import javax.swing.JPanel;
49 * >> 컴포넌트들을 그룹별로 묶어서 처리할 때 사용하는 컨테이너 클래스
50 * >> 일반적으로 Frame에 컴포넌트들을 직접 붙이지 않고 Panel을 사용합니다.
51 *
52 * import javax.swing.JScrollPane;
53 * >> 스크롤을 이용해서 컴포넌트들을 보여주는 컴포넌트 클래스
54 * >> 스크롤을 이용해서 보여주는 화면을 상하좌우로 이동하여 포함된 컴포넌트의 원래 크기를 유지합니다.
55 *
56 * import javax.swing.JTable;
57 * >> 데이터를 행과 열로 구성되어 있는 테이블 형식으로 보여주는 컴포넌트 클래스
58 * >> JTable을 사용하기 위해서는 먼저 데이터를 저장할 모델을 만들고 JTable에 연결합니다.
59 *
60 * import javax.swing.SwingConstants;
61 * >> 일반적으로 화면에서 구성 요소를 배치하고 방향을 지정할 때 사용하는 클래스
62 *
63 * import javax.swing.table.DefaultTableModel;
64 * >> 데이터를 행과 열로 구성되어 있는 테이블 형식으로 보여주는 컴포넌트 클래스
65 * >> 데이터를 Vector로 생성하기 때문에 JTable에서 데이터의 추가, 삭제를 할 수 있습니다.
66 *
67 * *****/
68
69 import java.awt.BorderLayout;
70 import java.awt.Image;
71 import java.awt.event.ActionEvent;
72 import java.awt.event.ActionListener;
73
74 import java.sql.Connection;
75 import java.sql.DriverManager;
76 import java.sql.ResultSet;
77 import java.sql.SQLException;
78 import java.sql.Statement;
79
80 import javax.swing.ImageIcon;
81 import javax.swing.JButton;
82 import javax.swing.JFrame;
83 import javax.swing.JLabel;
84 import javax.swing.JPanel;
85 import javax.swing.JScrollPane;
86 import javax.swing.JTable;
87 import javax.swing.table.DefaultTableModel;
```

과제05-3. JAVA JDBC GUI

2. JFrame

```
-- /*
90 * Title : Oracle DB 연동 Test * Date : 2019.11.17
91 * Purpose : Oracle DB 연동 및 JFrame 사용
92 */
93
94
95 public class GUI extends JFrame {
96     // DB와 연결된 상태를 담은 객체 conn을 선언했습니다.
97     private Connection conn = null;
98     private JLabel state;
99     // Table Header Name
100     private String colName[] = { "학번", "이름", "주소", "전화번호" };
101     // Table Data list (Header data, adding row)
102     private DefaultTableModel model = new DefaultTableModel(colName, 0);
103
104     // Create Table
105     private JTable table = new JTable(model);
106     private String row[] = new String[4];
107
108     // JFrame 생성자
109     public GUI() {
110         /*****
111          * >> setTitle() 메소드를 사용해서 프레임의 타이틀을 설정합니다. DataBase Test 20154215 구명회
112          * >> setDefaultCloseOperation(EXIT_ON_CLOSE) : 프레임 윈도우를 닫으면 프로그램이 종료됩니다.
113          *****/
114         setTitle("DataBase Test 20154215 구명회");
115         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
116
117         /*****
118          * 사진 부착하기 *
119          * >> ImageIcon() 메소드를 사용해서 사진을 읽어왔습니다.
120          * >> 사진의 경로는 현재 프로젝트 폴더에서 생성한 image디렉터리입니다.
121          * >> 그런데 읽은 사진의 크기가 커서 크기를 조절했습니다.
122          *****/
123         ImageIcon image = new ImageIcon("image/Create_homework.jpg");
124         Image resizeImage = image.getImage();
125         Image FinresizeImage = resizeImage.getScaledInstance(400, 400, java.awt.Image.SCALE_SMOOTH);
126         ImageIcon reimage = new ImageIcon(FinresizeImage);
127         JLabel imageLabel = new JLabel(reimage);
128     }
129 }
```

```
128
129
130 /*****
131  * 컴포넌트용 패널 *
132  * >> 하단부 연결 및 출력 버튼 컴포넌트용 패널
133  * >> 이미지용 패널
134  *****/
135 JPanel Btn_panel = new JPanel();
136 JPanel image_panel = new JPanel();
137
138 /*****
139  * 컴포넌트 설정 *
140  * >> 버튼의 이름을 각각 Connect와 Show로 설정합니다.
141  * >> add() 메소드를 사용해서 패널에 버튼과 이미지를 부착합니다.
142  *****/
143 JButton con = new JButton("Connect");
144 JButton show = new JButton("Show");
145 Btn_panel.add(con);
146 Btn_panel.add(show);
147 image_panel.add(imageLabel);
148
149 /*****
150  * Connect 버튼 이벤트 리스너 (액션 리스너) *
151  * >> Connect 버튼을 클릭했을 때 DB와 연결하는 Connect() 메소드를 호출하는 부분입니다.
152  *****/
153 con.addActionListener(new ActionListener() {
154     @Override
155     public void actionPerformed(ActionEvent e) {
156         connect();
157     }
158 });
159
160 /*****
161  * Show 버튼 이벤트 리스너 (액션 리스너) *
162  * >> Show 버튼을 클릭했을 때 DB를 읽어오는 show() 메소드를 호출하는 부분입니다.
163  * >> 버튼을 클릭하면 쿼리문을 수행한 결과를 보여줍니다.
164  *****/
165 show.addActionListener(new ActionListener() {
166     @Override
167     public void actionPerformed(ActionEvent e) {
168         show_db();
169     }
170 });
```

과제05-3. JAVA JDBC GUI

2. JFrame

```
171
172  /*****
173   * 배치 *
174   * >> DB 상태 출력용 라벨에 "Oracle DB 연동 테스트"를 출력합니다.
175   * >> DB 상태 출력용 라벨의 위치를 BorderLayout을 사용해서 배치합니다.
176   * >> JScrollPane() : 화면에서 넘어갈 경우 스크롤바가 생깁니다.
177   * >> Btn_panel과 image_panel의 위치를 폼에 맞춰 조절했습니다.
178   * >> setSize() 메소드를 사용해서 화면에 900x500 크기의 프레임을 보여줍니다.
179   * >> setResizable(false) : 프레임의 크기는 수정할 수 없습니다.
180   *****/
181  state = new JLabel();
182  state.setText("Oracle DB 연동 테스트");
183  add(state, BorderLayout.NORTH);
184
185  add(new JScrollPane(table), BorderLayout.CENTER);
186  add(Btn_panel, BorderLayout.SOUTH);
187  add(image_panel, BorderLayout.WEST);
188
189  setSize(900, 500);
190  setVisible(true);
191  setResizable(false);
192 }
```

3. Connect()

```
193
194  /*****
195   * DB 상태 출력용 라벨을 변경합니다.
196   * 초기 상태는 "Oracle DB 연동 테스트"를 출력하며
197   * Connect 버튼을 눌렀을 때 DB와 연결이 성공하면 "성공적 연결"문장으로, 실패하면 "DB 연결 실패" 문장으로 바꾸어 출력합니다.
198   * Show 버튼을 눌렀을 때 쿼리는 수행에 성공하면 "DB 읽기 성공"문장으로, 실패하면 "DB 읽기 실패" 문장으로 바꾸어 출력합니다.
199   *****/
200  public void connect() {
201      try {
202          /*****
203           * DriverManager 클래스에 등록 *
204           * >> 드라이버 로딩
205           * >> 드라이버 인터페이스를 구현한 클래스를 로딩합니다.
206           *
207           * DriverManager.getConnection() : DB에 연결(접속)하기 위한 메소드 *
208           * >> 드라이버 매니저에게 Connection 객체를 요청합니다.
209           * >> Connection을 얻기위해 getConnection메소드를 사용하며 인자는 ("접속url", "사용자 계정", "계정 비밀번호")입니다.
210           * >> 이 객체를 사용해서 statement로 쿼리문을 수행할 수 있습니다.
211           *****/
212          Class.forName("oracle.jdbc.driver.OracleDriver");
213          conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "MHHW3", "1234");
214          System.out.println("성공적 로딩");
215          state.setText("성공적 연결");
216
217      } catch (ClassNotFoundException e) {
218          e.printStackTrace();
219          state.setText("DB 연결 실패 " + e.toString());
220
221      } catch (SQLException e) {
222          e.printStackTrace();
223          state.setText("DB 연결 실패 " + e.toString());
224      }
225  }
```


과제05-3. JAVA JDBC GUI

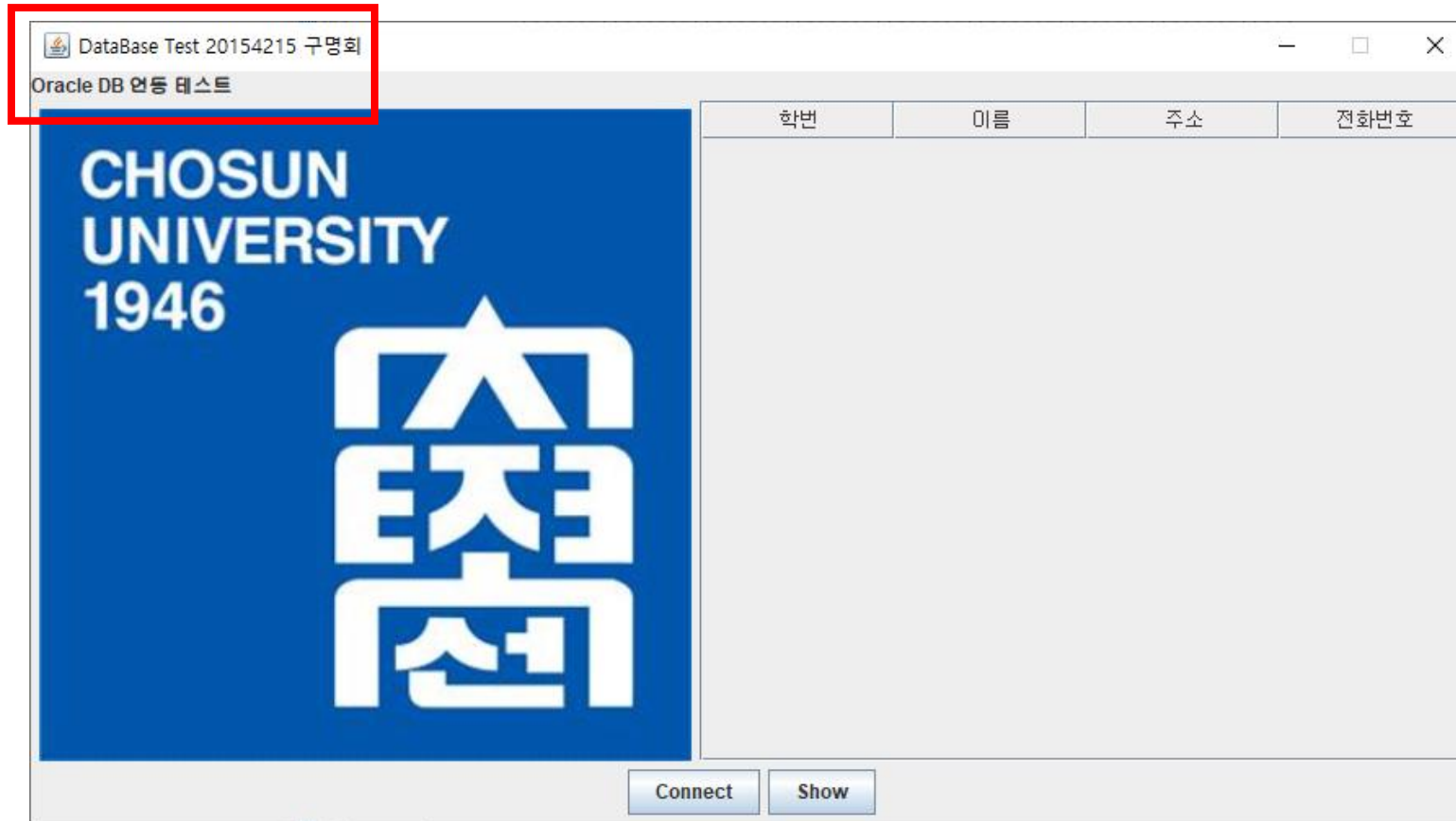
4. show()

```
227- /*****
228  * 쿼리문을 수행하는 메소드입니다.
229  * >> Show 버튼의 이벤트 리스너에 의해 호출되며, 버튼이 클릭되면 try ~ catch문을 수행해서 결과를 보여줍니다.
230  *****/
231- public void show_db() {
232     try {
233         /*****
234          * conn.createStatement() : Statement 생성 *
235          * >> 쿼리 수행을 위한 Statement 객체를 생성합니다.
236          * 쿼리문 수행 *
237          * >> JDBC에서 쿼리를 작성할 때 세미콜론 (;)을 빼고 작성합니다.
238          * >> *로 모든 칼럼을 읽어오는 것보다 가져와야 할 칼럼을 직접 명시해주는게 좋다고 합니다.
239          * >> 쿼리를 한 줄로 쓰기 어려운 경우 들여쓰기를 사용해도 상관 없습니다.
240          * >> 레코드들은 ResultSet 객체에 추가됩니다.
241          *****/
242         Statement stmt = conn.createStatement();
243         ResultSet rset = stmt.executeQuery("select * from test order by NAME");
244         System.out.println("학번\t이름\t주소\t전화번호");
245
246         /*****
247          * while(rset.next()) : 쿼리문 결과 출력 *
248          * >> test table
249          * >> num      name      address    phone
250          * >> 1         홍길정     광주 서구   010-1234-5678
251          * >> 2         최정성     광주 북구   010-1111-7148
252          * >> 3         김소민     광주 남구   010-2222-1248
253          * >> .... 이런식으로 저장되어 있고
254          * >> next() 메소드는 현재 행에서 한 행 다음으로 이동하므로
```

```
255          * >> test의 레코드 첫행부터 출력하게 됩니다.
256          * jTable에 출력 *
257          * addRow() 메소드를 사용해서 DB에서 읽어온 레코드들을 보여줍니다.
258          *****/
259         while (rset.next()) {
260             for (int i = 1; i < 5; i++) {
261                 System.out.print(rset.getString(i) + "\t");
262                 row[i - 1] = rset.getString(i);
263             }
264
265             System.out.println();
266             model.addRow(row); // jTable에 출력
267         }
268
269         state.setText("DB 읽기 성공");
270     } catch (SQLException e) {
271         e.printStackTrace();
272         state.setText("DB 읽기 실패 " + e.toString());
273     }
274 }
275
276- public static void main(String[] args)
277 {
278     new GUI();
279 }
280 }
```

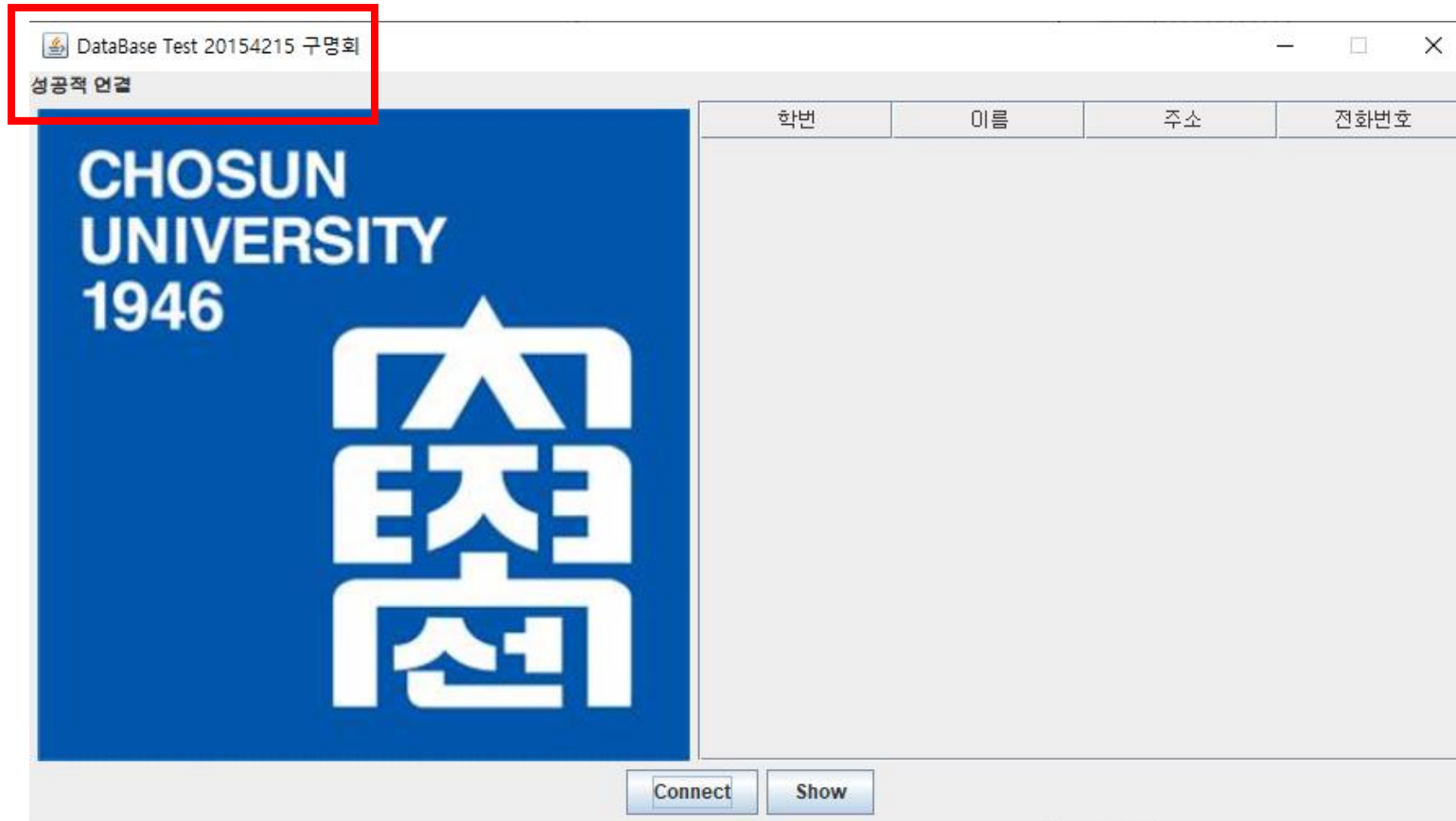

과제05-3. JAVA JDBC GUI

5. 실행결과 (프로그램 실행)



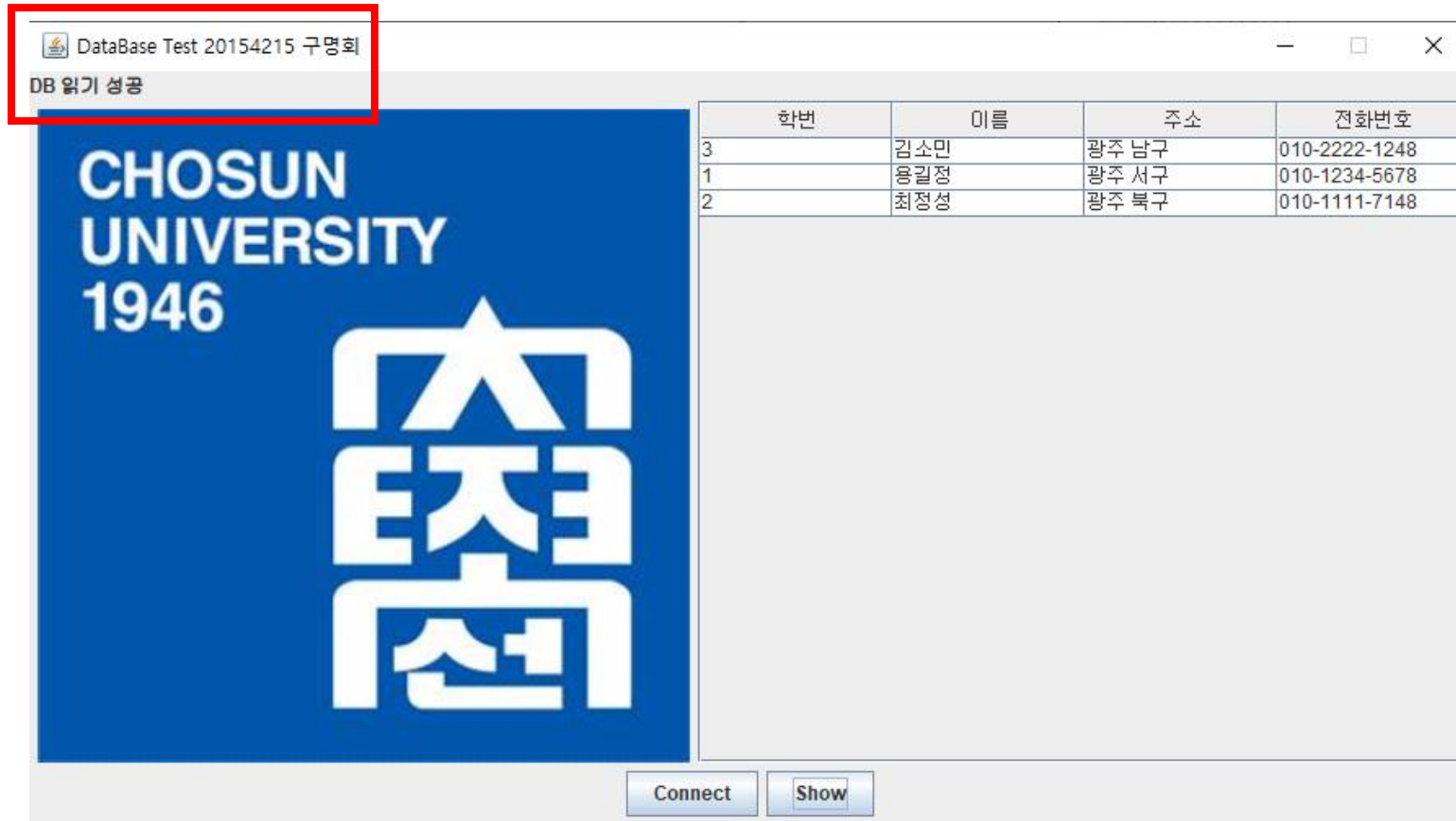
과제05-3. JAVA JDBC GUI

5. 실행결과 (Connect)



과제05-3. JAVA JDBC GUI

5. 실행결과 (Show)



과제06. 소감

저는 데이터베이스 수업듣기전에 SQL문에 대해 접할 기회가 있었습니다.

물론 직접 써보라고 하면 기억이 잘 안나서 찾아보면서 써야하지만.. 일단 쿼리문 이해하는것에 대해서는 어려운 부분은 없었습니다.

걱정되는점은 쿼리문이 너무 많아서 사진 추가할때 중복되거나 잘못 넣은 부분이 있을것 같은점? 확인 몇번씩 하긴 했는데.. 음

그런데 제가 이번 과제에서 어려웠던 점..? 어려웠던점은 아니고 뭔가 문제가 하나 생겨서 해결하느라 시간이

한시간정도 걸렸었는데 SQL RUN에서 테이블에 데이터를 삽입하고나서 COMMIT을 안한것이 문제였습니다.

JAVA에서 DB를 SELECT문으로 읽어올때 분명히 DB에 연결은 되는데 레코드를 읽어오질 못해서

뭔가 잘못됐건지 몰라서 이클립스 JDK 설치랑 JAR파일도 몇번이고 옮겨다녔습니다. 그러다 혹시 COMMIT 안해서 그러나 하

고 SQL에서 COMMIT 실행시키고 보니까 정상적으로 읽어올 수 있었습니다.

저는 SQL DB에서 테이블을 생성하고 데이터를 삽입한 다음 원래 SQL에선 DML을 실행하면 자동으로 COMMIT 된다고

알고 있어서 지금까지 계속 COMMIT을 사용하지 않았었는데, (이번에 소감 쓰려고 다시 한번 저장되는걸 확인도 했습니다!)

왠지 모르겠는데 SQL에서 COMMIT으로 저장하고 나서야 JAVA에서 불러올 수 있었습니다!

그래서 이번 과제를 계기로 COMMIT의 중요성을 알게 되었습니다..