Exercise 7 - Table Creation (DDL)

Create new tables by using the CREATE TABLE statement. Confirm that the new table

was added to the database. You also learn to set the status of a table as READ ONLY and

then revert to READ/WRITE.

**Note:** For all the DDL and DML statements, click the Run Script icon (or press [F5]) to

execute the query in SQL Developer. This way you get to see the feedback messages on

the Script Output tabbed page. For SELECT queries, continue to click the Execute

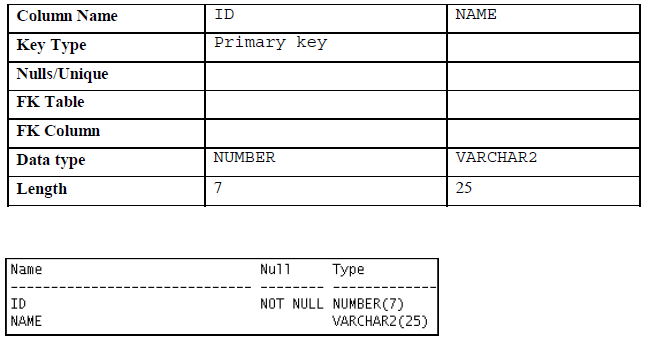
Statement icon or press [F9] to get the formatted output on the Results tabbed page.

***Using DDL Statements to Create and Manage Tables***

1) Create the **DEPT** table, based on the following table instance chart. Save the statement

in a script called lab\_10\_01.sql, and then execute the statement in the script to

create the table. Confirm that the table is created.



**create table DEPT(**

**ID NUMBER (7)NOT NULL PRIMARY KEY,**

**NAME VARCHAR2(25));**

2) Populate the DEPT table with data from the DEPARTMENTS table. Include only

columns that you need.

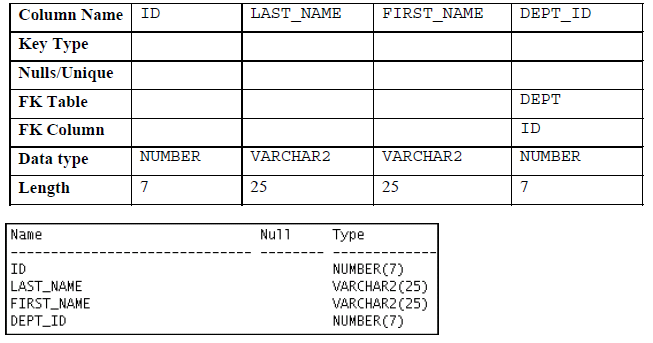
**insert into DEPT**

**(Select department\_id, department\_name from departments);**

3) Create the **EMP** **table** based on the following table instance chart. Save the statement

in a script called lab\_10\_03.sql, and then execute the statement in the script to

create the table. Confirm that the table is created.



**create table EMP(**

**ID NUMBER(7),**

**LAST\_NAME VARCHAR2(25),**

**FIRST\_NAME VARCHAR2(25),**

**DEPT\_ID NUMBER(7),**

**FOREIGN KEY (DEPT\_ID)**

**REFERENCES DEPT(ID)**

**);**

4) Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPARTMENT\_ID columns. Name the columns in your new table ID, FIRST\_NAME, LAST\_NAME, SALARY, and DEPT\_ID, respectively.

**CREATE TABLE EMPLOYEES2**

**(ID, FIRST\_NAME, LAST\_NAME, SALARY,DEPT\_ID)**

**AS (SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY,DEPARTMENT\_ID**

**FROM employees);**

5) Alter the EMPLOYEES2 table status to read-only.

**ALTER TABLE EMPLOYEES2 READ ONLY;**

6) Try to insert the following row in the EMPLOYEES2 table:

You get the following error message:

**INSERT INTO EMPLOYEES2 (SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY,DEPARTMENT\_ID**

**FROM employees);**

7) Revert the EMPLOYEES2 table to the read/write status. Now, try to insert the same

row again. You should get the following messages:

**ALTER TABLE EMPLOYEES2 READ WRITE;**

**INSERT INTO EMPLOYEES2 (SELECT EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, SALARY,DEPARTMENT\_ID**

**FROM employees);**

8) Drop the EMPLOYEES2 table.

**DROP TABLE EMPLOYEES2;**

**Oracle Database 11*g*: SQL Fundamentals I A - 76**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***Practice 1-1: Creating Other Schema Objects***

**Part 1**

1) The staff in the HR department wants to hide some of the data in the EMPLOYEES

table. Create a view called EMPLOYEES\_VU based on the employee numbers,

employee last names, and department numbers from the EMPLOYEES table. The

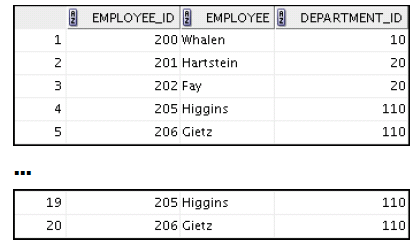
heading for the employee name should be EMPLOYEE.

**CREATE view EMPLOYEES\_VU**

**AS SELECT EMPLOYEE\_ID, LAST\_NAME as "Employee\_name", DEPARTMENT\_ID**

**FROM EMPLOYEES;**

2) Confirm that the view works. Display the contents of the EMPLOYEES\_VU view.

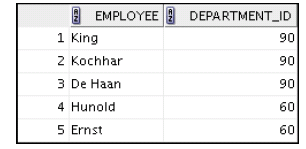
****

**select \* from employees\_vu;**

3) Using your EMPLOYEES\_VU view, write a query for the HR department to display all

employee names and department numbers.

**SELECT Employee, DEPARTMENT\_ID FROM employees\_vu;**

****

4) Department 50 needs access to its employee data. Create a view named DEPT50 that

contains the employee numbers, employee last names, and department numbers for

all employees in department 50. You have been asked to label the view columns

EMPNO, EMPLOYEE, and DEPTNO. For security purposes, do not allow an employee to

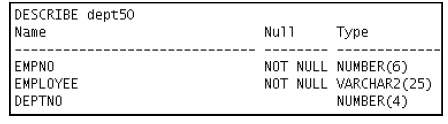
be reassigned to another department through the view.

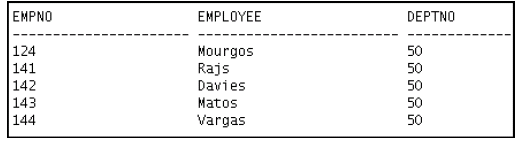
**create view DEPT50**

**as select employee\_id as EMPNO,LAST\_NAME AS EMPLOYEE, DEPARTMENT\_ID AS DEPTNO**

**FROM EMPLOYEES WHERE department\_id=50;**

5) Display the structure and contents of the DEPT50 view.

****

****

**DESC DEPT50;**

**SELECT \* FROM DEPT50;**

6) Test your view. Attempt to reassign Matos to department 80.

**UPDATE DEPT50**

**SET DEPTNO=80 WHERE EMPLOYEE='Matos';**

**Part 2**

7) You need a sequence that can be used with the PRIMARY KEY column of the DEPT

table. The sequence should start at 200 and have a maximum value of 1,000. Have

your sequence increment by 10. Name the sequence DEPT\_ID\_SEQ.

**create sequence DEPT\_ID\_SEQ**

**START WITH 200**

**INCREMENT BY 10**

**MAXVALUE 1000**

**NOCYCLE**

**NOCACHE;**

8) To test your sequence, write a script to insert two rows in the DEPT table. Name your

script lab\_11\_08.sql. Be sure to use the sequence that you created for the ID

column. Add two departments: Education and Administration. Confirm your

additions. Run the commands in your script.

**INSERT INTO DEPT(ID,NAME)VALUES**

**(DEPT\_ID\_SEQ.NEXTVAL, 'EDUCATION');**

**INSERT INTO DEPT(ID,NAME)VALUES**

**(DEPT\_ID\_SEQ.NEXTVAL, 'ADMINISTRATION');**

9) Create a nonunique index on the NAME column in the DEPT table.

**CREATE INDEX NAME\_IDX ON DEPT(NAME);**

**SELECT \* FROM DEPT;**

10) Create a synonym for your EMPLOYEES table. Call it EMP.

**CREATE SYNONYM EMP2 FOR EMPLOYEES;**

**SELECT \* FROM EMP2;**