UNIVERSITY OF CHITTAGONG

Department of Computer Science and Engineering

Session: 2021-2022
4th semester

Assignment No.	: 04
Course Title	: Database Systems
Course Code No.	: CSE-413

Submitted to:
Dr. Rudra Pratap Deb Nath
Associate Professor
Department of Computer Science and Engineering
University of Chittagong

Submitted by:
Sanzid Islam Mahi
ID: 22701065
Department of Computer Science and Engineering
University of Chittagong

Date: Jul 13, 2024

Oracle9i SQL-I

Chapter 9

Practice 9

1. Create the DEPT table based on the following table instance chart. Place the syntax in a script called `lab9_1.sql`, then execute the statement in the script to create the table. Confirm that the table is created

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

Solution:

```
CREATE TABLE dept(
    id NUMBER(7),
    name VARCHAR2(25)
);
DESCRIBE DEPT;
```

Output:

TABLE DEPT

Column	Null?	Type
ID	-	NUMBER(7,0)
NAME	-	VARCHAR2(25)

2. Populate the DEPT table with data from the DEPARTMENTS table. Include only columns that you need.

Solution:

```
INSERT INTO dept
SELECT department_id, department_name
FROM hr.departments;
```

3. Create the EMP table based on the following table instance chart. Place the syntax in a script called `lab9_3.sql`, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	Number	VARCHAR2	VARCHAR2	Number
Length	7	25	25	7

Solution:

```
CREATE TABLE emp(
    id number (7),
    last_name varchar2(25),
    first_name varchar2(25),
    dept_id number (7)
);

DESCRIBE emp;
```

Output:

TABLE EMP

Column	Null?	Type
ID	-	NUMBER(7,0)
LAST_NAME	-	VARCHAR2(25)
FIRST_NAME	-	VARCHAR2(25)
DEPT_ID	-	NUMBER(7,0)

4. Modify the EMP table to allow for longer employee last names. Confirm your modification.

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

Solution:

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

Output:

TABLE EMP

Column	Null?	Type
ID	-	NUMBER(7,0)
LAST_NAME	-	VARCHAR2(50)
FIRST_NAME	-	VARCHAR2(25)
DEPT_ID	-	NUMBER(7,0)

5. Confirm that both the DEPT and EMP tables are stored in the data dictionary. (Hint: `USER_TABLES`)

TABLE_NAME
DEPT
EMP

Solution:

```
select table_name
from user_tables
where table_name in ('DEPT','EMP');
```

Output:

TABLE_NAME
DEPT
EMP

6. 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.

Solution:

```
CREATE TABLE employees2 AS
SELECT employee_id id, first_name, last_name, salary,
    department_id dept_id
from hr.employees;
```

7. Drop the EMP table.

Solution:

```
DROP TABLE emp;
```

8. Rename the EMPLOYEES2 table as EMP.

   Solution:

```
ALTER TABLE employees2 RENAME TO emp;
```

9. Add a comment to the DEPT and EMP table definitions describing the tables. Confirm your additions in the data dictionary.

   Solution:

```
COMMENT ON TABLE emp IS
    'This table stores employee information';
COMMENT ON TABLE dept IS
    'This table stores dept information';
SELECT *
FROM user_tab_comments
WHERE table_name = 'DEPT'
        OR table_name = 'EMP';
```

   Output:

TABLE_NAME	TABLE_TYPE	COMMENTS
DEPT	TABLE	This table stores department information
EMP	TABLE	This table stores employee information

10. Drop the `FIRST_NAME` column from the EMP table. Confirm your modification by checking the description of the table.

    Solution:

```
ALTER TABLE emp
DROP COLUMN FIRST_NAME;
DESCRIBE emp;
```

    Output:

Column	Null?	Type
ID	-	NUMBER(6,0)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY	-	NUMBER(8,2)
DEPT_ID	-	NUMBER(4,0)

11. In the EMP table, mark the `DEPT_ID` column as UNUSED. Confirm your modification by checking the description of the table.

    Solution:

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

DESCRIBE emp;
```

    Output:

Column	Null?	Type
ID	-	NUMBER(6,0)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY	-	NUMBER(8,2)

12. Drop all the UNUSED columns from the EMP table. Confirm your modification by checking the description of the table.

    Solution:

```
ALTER TABLE emp
DROP UNUSED COLUMNS;

DESCRIBE emp;
```

    Output:

Column	Null?	Type
ID	-	NUMBER(6,0)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY	-	NUMBER(8,2)

Chapter 10

Practice 10

1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint `my_emp_id_pk`.
   Hint: The constraint is enabled as soon as the ALTER TABLE command executes successfully.

   Solution:

```
ALTER TABLE emp
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);
```

   Output:

2. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint `my_deptid_pk`.
   Hint: The constraint is enabled as soon as the ALTER TABLE command executes successfully.

   Solution:

```
ALTER TABLE dept
ADD CONSTRAINT my_deptid_pk PRIMARY KEY(id);
```

3. Add a column `DEPT_ID` to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to a nonexistent department. Name the constraint `my_emp_dept_id_fk`.

   Solution:

```
ALTER TABLE emp
ADD (dept_id NUMBER(7));
ALTER TABLE emp
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES dept(id);
```

4. Confirm that the constraints were added by querying the `USER_CONSTRAINTS` view. Note the types and names of the constraints. Save your statement text in a file called `lab10_4.sql`.

CONSTRAINT_NAME	C
MY_DEPT_ID_PK	P
SYS_C002541	C
MY_EMP_ID_PK	P
MY_EMP_DEPT_ID_FK	R

   Solution:

```
select constraint_name, constraint_type
from user_constraints
where table_name IN ('EMP', 'DEPT');
```

Output:

CONSTRAINT_NAME	CONSTRAINT_TYPE
MY_DEPTID_PK	P
MY_EMP_ID_PK	P
MY_EMP_DEPT_ID_FK	R

5. Display the object names and types from the `USER_OBJECTS` data dictionary view for the EMP and DEPT tables. Notice that the new tables and a new index were created.

   Solution:

```
SELECT OBJECT_NAME, OBJECT_TYPE
FROM USER_OBJECTS
WHERE OBJECT_NAME IN ('EMP', 'DEPT');
```

   Output:

TABLE_NAME
DEPT
EMP

6. Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero.

   Solution:

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

ALTER TABLE emp
ADD CONSTRAINT chk_commission_positive
    CHECK (COMMISSION > 0);
```

Chapter 8

Practice 8

Insert data into the `MY_EMPLOYEE` table.

1. Run the statement in the `lab8_1.sql` script to build the `MY_EMPLOYEE` table to be used for the lab.

   Solution:

```
create table my_employee(
    id number (4)
        constraint my_employee_id_not_null not null,
    last_name varchar2(25),
    first_name varchar2(25),
    userId varchar2(8),
    salary number (9,2)
);
```

2. Describe the structure of the `MY_EMPLOYEE` table to identify the column names.

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

   Solution:

```
describe my_employee;
```

3. Add the first row of data to the `MY_EMPLOYEE` table from the following sample data. Do not list the columns in the INSERT clause.

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

   Solution:

```
insert into my_employee
values (1, 'Patel', 'Ralph', 'rpatel', 895);
```

4. Populate the `MY_EMPLOYEE` table with the second row of sample data from the preceding list. This time, list the columns explicitly in the INSERT clause.

   Solution:

```
insert into my_employee(id,last_name,first_name,userId,salary)
values (2,'Dancs','Betty','bdancs',860);
```

5. Confirm your addition to the table.

   Solution:

```
select *
from my_employee;
```

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

6. Write an INSERT statement in a text file named loademp.sql to load rows into the `MY_EMPLOYEE` table. Concatenate the first letter of the first name and the first seven characters of the last name to produce the user ID.

   Solution: skipped(sql*plus required).

7. Populate the table with the next two rows of sample data by running the INSERT statement in the script that you created.

   Solution: skipped(sql*plus required).

8. Confirm your additions to the table. **Solution:**

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

```
select *
from my_employee;
```

9. Make the data additions permanent.

   Solution:

```
Commit;
```

Update and delete data in the `MY_EMPLOYEE` table.

10. Change the last name of employee 3 to Drexler.

    Solution:

```
-- adding the remaining rows to move forward
insert into my_employee
values (3, 'Biri', 'Ben', 'bbiri', 1100);
insert into my_employee
values (4, 'Newman', 'Chad', 'cnewman', 750);
insert into my_employee
values (5, 'Ropeburn', 'Audrey', 'aropebur', 1550);
```

```
update my_employee
set last_name = 'Drexler'
where id = 3;
```

11. Change the salary to 1000 for all employees with a salary less than 900.

    Solution:

```
update my_employee
set salary = 1000
where salary < 900;
```

12. Verify your changes to the table.

    Solution:

```
select id,last_name, salary
from my_employee;
```

    Output:

ID	LAST_NAME	SALARY
1	Patel	1000
2	Dancs	1000
3	Drexler	1100
4	Newman	1000
5	Ropeburn	1550

13. Delete Betty Dancs from the `MY_EMPLOYEE` table.

    Solution:

```
delete
from my_employee
where last_name = 'Dancs' and first_name = 'Betty';
```

14. Confirm your changes to the table.

    Solution:

```
select *
from my_employee
where last_name = 'Dancs' and first_name = 'Betty';

--No data found --
```

15. Commit all pending changes.

    Solution:

```
Commit;
```

    Control data transaction to the `MY_EMPLOYEE` table.

16. Populate the table with the last row of sample data by modifying the statements in the script that you created in step 6. Run the statements in the script.

    Solution: skipped(sql*plus required)

17. Confirm your addition to the table.

    Solution: skipped

18. Mark an intermediate point in the processing of the transaction.

    Solution:

```
SAVEPOINT before_deletion;
```

19. Empty the entire table.

    Solution:

```
delete from my_employee;
```

20. Confirm that the table is empty.

    Solution:

```
select *
from my_employee;
```

21. Discard the most recent DELETE operation without discarding the earlier INSERT operation.

    Solution:

```
ROLLBACK TO before_deletion;
```

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

22. Confirm that the new row is still intact.

    Solution:

```
select *
from my_employee;
```

23. Make the data addition permanent.

    Solution:

```
Commit;
```