USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science

CS311 Introduction to Database Systems

Lab#7

Objective:

- Data manipulation operations in SQL

Name of Student: **Muhammad Waleed**

Roll No: **20B-115-SE** **Sec.** **B**

Date of Experiment:

....

Marks Obtained/Remarks: ______________________________

Signature: ______________________________

THEORY

Data-Manipulation Language

Data manipulation language is a core part of SQL. When we want to add, update or delete data in the database, we execute a DML statement. A collection of DML statements that form a logical unit of work is called a *transaction*.

Consider a banking database. When a bank customer transfers money from a savings account to a checking account, the transaction might consist of three separate operations: decrease the savings account, increase the checking account, and record the transaction in the transaction journal. The Oracle server must guarantee that all three SQL statements are performed to maintain the accounts in the proper balance. When something prevents one of the statements in the transaction from executing, the other statements of the transaction must be undone.

The SQL DML includes statements to perform following operations:-

Statement	Description
INSERT	Enter new rows into tables
UPDATE	To change existing rows
DELETE	To delete existing rows

Table 6.1

Adding a new row to a table

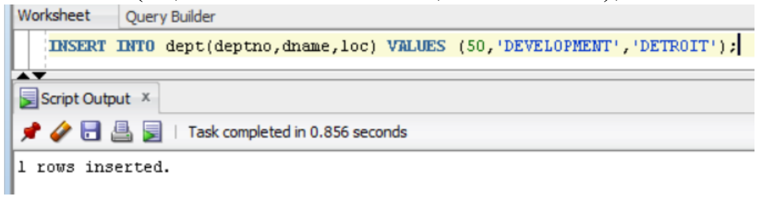
We can add new rows to a table by using the INSERT statement. The syntax is

INSERT INTO table [(column [, column ...])]
VALUES (value [, value ...]);

Examples

i. Inserting a new row in the dept table

INSERT INTO dept (deptno, dname, loc)
VALUES (50, 'DEVELOPMENT', 'DETROIT');



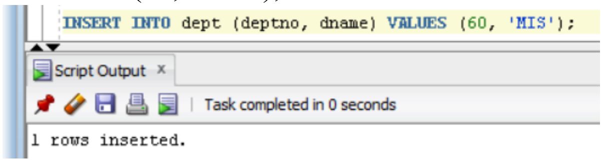
```
INSERT INTO dept(deptno,dname,loc) VALUES (50,'DEVELOPMENT','DETROIT');
```

```
1 rows inserted.
```

Note: If the column list is not included, the values must be listed according to the default order of the columns in the table. The order can be seen using the DESCRIBE command in SQL*PLUS (See lab session 1)

ii. Inserting rows with Null values ○ *Implicit Method*: Omit the column from the column list.

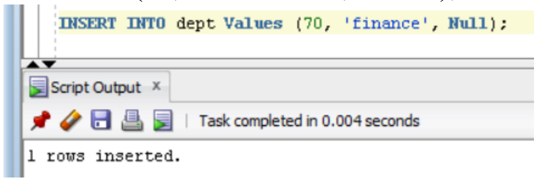
INSERT INTO dept (deptno, dname)
VALUES (60, 'MIS');



```
INSERT INTO dept (deptno, dname) VALUES (60, 'MIS');
```

```
1 rows inserted.
```

○ *Explicit Method:* Specify the NULL keyword
INSERT INTO dept
VALUES (70, 'FINANCE', NULL);



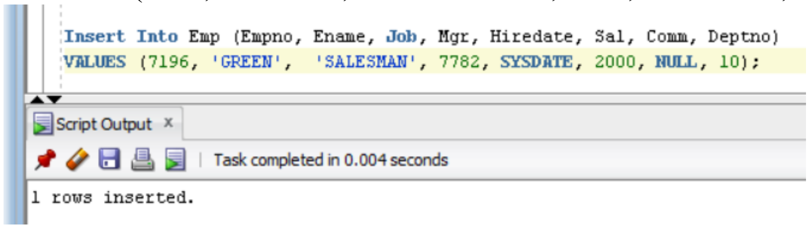
```
INSERT INTO dept Values (70, 'finance', Null);
```

```
1 rows inserted.
```

Note: The oracle server automatically enforces all datatypes, data ranges and data integrity constraints. Any column that is not listed explicitly obtains a null value in the new row.

iii. Using special values, for example, SYSDATE function, to obtain data for a column when inserting a row in a table

INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
VALUES (7196, 'GREEN', 'SALESMAN', 7782, SYSDATE, 2000, NULL, 10);



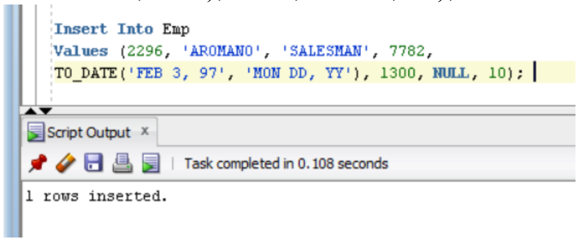
```
Insert Into Emp (Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno)
VALUES (7196, 'GREEN',  'SALESMAN', 7782, SYSDATE, 2000, NULL, 10);
```

```
1 rows inserted.
```

Similarly we can also use the USER function when inserting rows in a table. The USER function records the current username.

iv. Adding a new employee by inserting specific date values

INSERT INTO emp
VALUES (2296, 'AROMANO', 'SALESMAN', 7782, TO_DATE('FEB 3, 97', 'MON DD, YY'), 1300, NULL, 10);



```
Insert Into Emp
Values (2296, 'AROMANO', 'SALESMAN', 7782,
TO_DATE('FEB 3, 97', 'MON DD, YY'), 1300, NULL, 10);
```

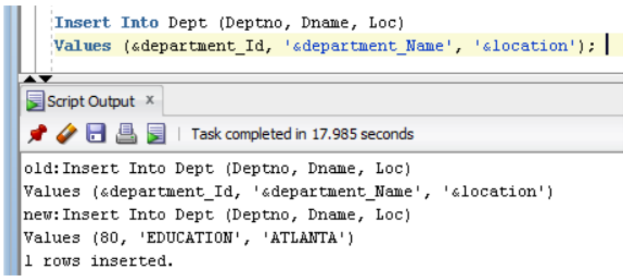
```
1 rows inserted.
```

v. We can produce an INSERT statement that allows the user to add values interactively by using SQL*Plus substitution variables.

INSERT INTO dept (deptno, dname, loc)
VALUES (&department_id, '&department_name', '&location');

```
Enter value for department_id: 80
Enter value for department_name: EDUCATION
Enter value for location: ATLANTA

1 row created
```



```
Insert Into Dept (Deptno, Dname, Loc)
Values (&department_Id, '&department_Name', '&location');
```

Script Output

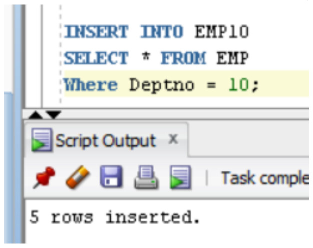
Task completed in 17.985 seconds

```
old:Insert Into Dept (Deptno, Dname, Loc)
Values (&department_Id, '&department_Name', '&location')
new:Insert Into Dept (Deptno, Dname, Loc)
Values (80, 'EDUCATION', 'ATLANTA')
1 rows inserted.
```

vi. Copying rows from another table

We can use the INSERT statement to add rows to a table where the values are derived from some other existing table. In place of the VALUES clause, we use a subquery. e.g. to insert rows from EMP table to EMP10 table,

INSERT INTO EMP10
SELECT * FROM EMP
WHERE DEPTNO = 10;



```
INSERT INTO EMP10
SELECT * FROM EMP
Where Deptno = 10;
```

Script Output

Task comple

```
5 rows inserted.
```

Changing data in a table

We can modify existing rows in a table with the UPDATE statement. The syntax is

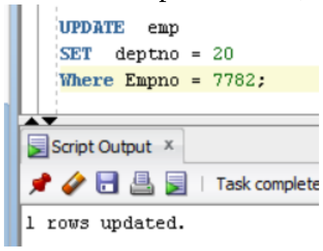
UPDATE table
SET column = value [, column = value , ...]
[WHERE condition];

As shown in the above syntax, we can update more than one row at a time depending on a condition.

Examples

i. To transfer an employee with number 7782 to department 20.

UPDATE emp
SET deptno = 20
WHERE empno = 7782;



```
UPDATE  emp
SET  deptno = 20
Where Empno = 7782;
```

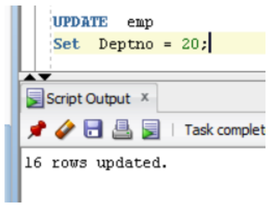
Script Output

Task complete

```
1 rows updated.
```

ii. All rows in the table are modified if the WHERE clause is omitted.

UPDATE emp
SET deptno = 20;



```
UPDATE  emp
Set  Deptno = 20;
```

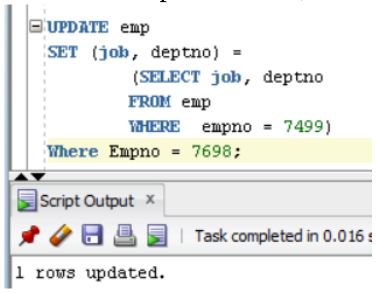
Script Output ×

Task complet

```
16 rows updated.
```

iii. Updating with multiple column subquery: Update employee 7698's job and department to match that of employee 7499.

```
UPDATE emp
SET (job, deptno) =
                    (SELECT job, deptno
                     FROM emp
                     WHERE  empno = 7499)
WHERE empno = 7698;
```



```
UPDATE emp
SET (job, deptno) =
        (SELECT job, deptno
        FROM emp
        WHERE  empno = 7499)
Where Empno = 7698;
```

Script Output ×

Task completed in 0.016 s

```
1 rows updated.
```

Removing a row from a table

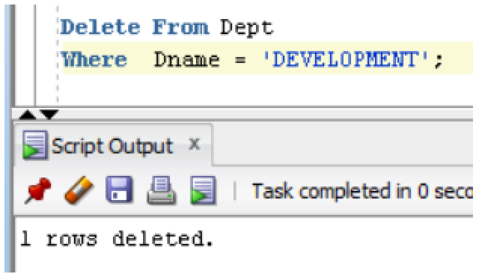
We can remove existing rows from a table by using the DELETE statement. The syntax is

```
DELETE [FROM] table
[WHERE      condition];
```

<u>Examples</u>

i. Specific rows are deleted from a table by specifying the WHERE clause.

```
DELETE FROM department
WHERE  dname = 'DEVELOPMENT';
```



```
Delete From Dept
Where  Dname = 'DEVELOPMENT';
```

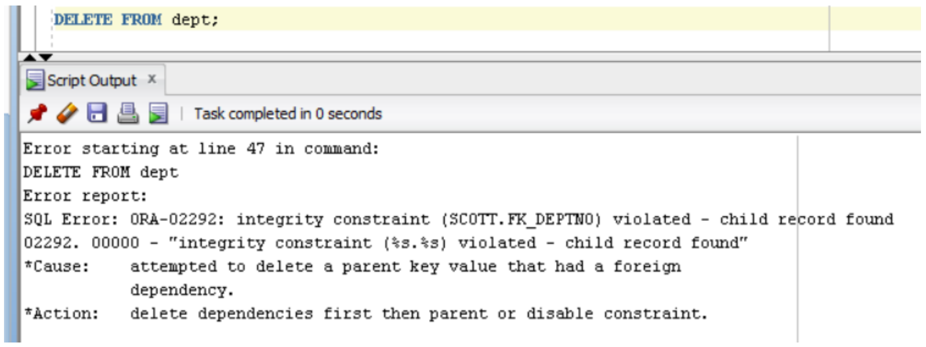
Script Output ×

Task completed in 0 seco

```
1 rows deleted.
```

ii. All rows in the table are deleted if we omit the WHERE clause.

```
DELETE FROM department;
```



```
DELETE FROM dept;
```

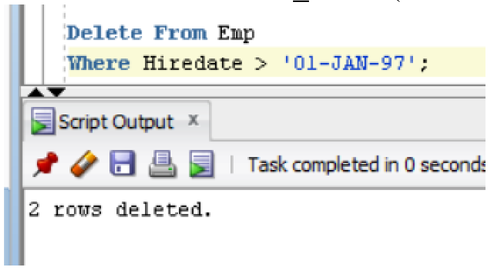
Script Output ×

Task completed in 0 seconds

```
Error starting at line 47 in command:
DELETE FROM dept
Error report:
SQL Error: ORA-02292: integrity constraint (SCOTT.FK_DEPTNO) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
           dependency.
*Action:   delete dependencies first then parent or disable constraint.
```

iii. Remove all employees who started after January 1, 1997.

DELETE FROM employee
WHERE hiredate > TO_DATE('01.01.97', 'DD.MM.YY');



```
Delete From Emp
Where Hiredate > '01-JAN-97';
```

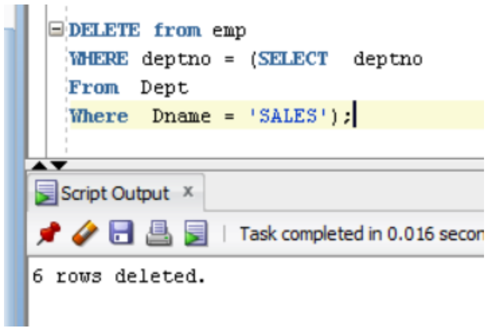
Script Output ×

Task completed in 0 seconds

```
2 rows deleted.
```

iv. Deleting rows based on another table by using subqueries in DELETE statements.

```
DELETE from employee
WHERE       deptno =
                    (SELECT  deptno
                     FROM  dept
                     WHERE  dname = 'SALES');
```



```
DELETE from emp
WHERE deptno = (SELECT   deptno
From  Dept
Where  Dname = 'SALES');
```

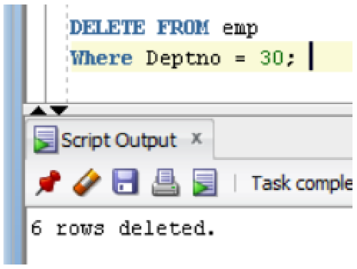
Script Output ×

Task completed in 0.016 secor

```
6 rows deleted.
```

Delete record of employees in department 30

DELETE FROM employee
WHERE DEPTNO = 30;



```
DELETE FROM emp
Where Deptno = 30;
```

Script Output ×

Task comple

```
6 rows deleted.
```

Database Transactions

The oracle server ensures data consistency based on transactions. Transactions consist of DML statements that makeup one consistent change to the data. For example, a transfer of funds between two accounts should include the debit to one account and a credit to another account in the same amount. Both actions should either fail or succeed together. The credit should not be committed without the debit.

Transaction Types

Type	Description
Data Manipulation language (DML)	Consists of any number of DML statements that the Oracle Server treats as a single entity or a logical unit of work
Data Definition language (DDL)	Consists of only one DDL statement
Data Control language (DCL)	Consists of only one DCL statement

Table 6.2

A transaction begins when the first executable SQL statement is encountered and terminates when one of the following occurs:

v. A COMMIT or ROLLBACK statement is issued vi. A DDL statement, such as CREATE, is issued vii. A DCL statement is issued

viii. The user exits SQL*Plus

ix. A machine fails or the system crashes

After one transaction ends, the next executable SQL statement automatically starts the next transaction. A DDL or DCL statement is automatically committed and therefore implicitly ends a transaction.

Transaction Control

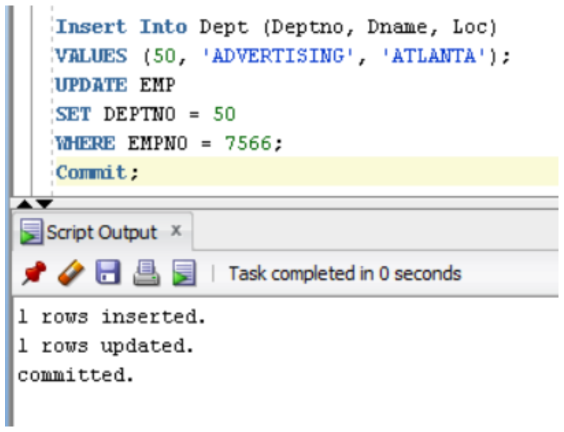
COMMIT: Ends the current transaction by making all pending data changes permanent.

ROLLBACK: Ends the current transaction by discarding all pending data changes. SAVEPOINT: Marks a savepoint within the current transaction.

Example

To create a new advertising department with at least one employee and make the data changes permanent.

```
INSERT INTO dept (deptno, dname, loc)
VALUES (50, ‘ADVERTISING’, ‘ATLANTA’);
UPDATE EMP
SET DEPTNO = 50
WHERE EMPNO = 7566;
COMMIT;
```



```
Insert Into Dept (Deptno, Dname, Loc)
VALUES (50, 'ADVERTISING', 'ATLANTA');
UPDATE EMP
SET DEPTNO = 50
WHERE EMPNO = 7566;
Commit;
```

```
1 rows inserted.
1 rows updated.
committed.
```

EXERCISES

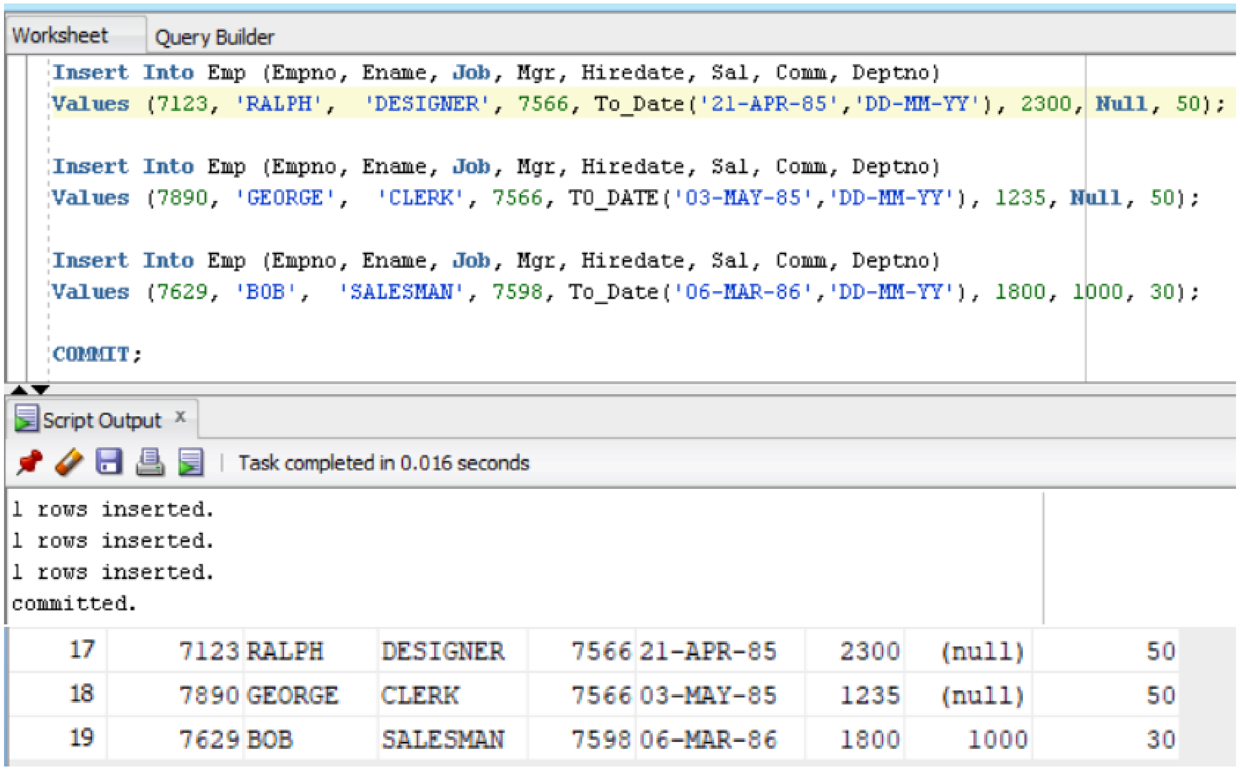
1. Define Transaction. How it is terminated? Describe the different operations included in a transaction.

Transactions consist of DML statements that makeup one consistent change to the data.
A transaction ends when it is committed or rolled back, either explicitly with a COMMIT or ROLLBACK statement or implicitly when a DDL statement is issued
There are two types of transaction.

1) COMMIT: Ends the current transaction by making all pending data changes permanent.
2) ROLLBACK: Ends the current transaction by discarding all pending data changes. SAVEPOINT:

2. Write a transaction to insert following rows in EMP table.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7123	RALPH	DESIGNER	7566	21-APR-85	2300		50
7890	GEORGE	CLERK	7566	03-MAY-85	1235		50
7629	BOB	SALESMAN	7698	06-MAR-86	1800	1000	30



Worksheet Query Builder

```
Insert Into Emp (Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno)
Values (7123, 'RALPH',  'DESIGNER', 7566, To_Date('21-APR-85','DD-MM-YY'), 2300, Null, 50);

Insert Into Emp (Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno)
Values (7890, 'GEORGE',  'CLERK', 7566, TO_DATE('03-MAY-85','DD-MM-YY'), 1235, Null, 50);

Insert Into Emp (Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno)
Values (7629, 'BOB',  'SALESMAN', 7598, To_Date('06-MAR-86','DD-MM-YY'), 1800, 1000, 30);

COMMIT;
```

Script Output

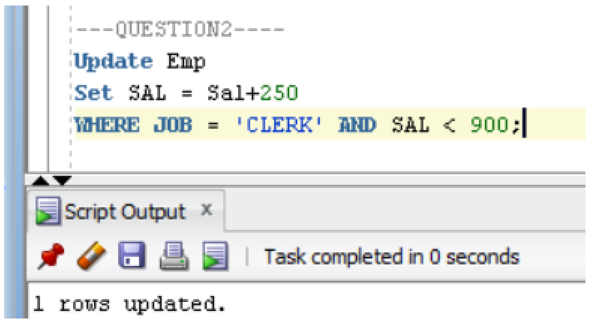
Task completed in 0.016 seconds

```
1 rows inserted.
1 rows inserted.
1 rows inserted.
committed.
```

17	7123	RALPH	DESIGNER	7566	21-APR-85	2300	(null)	50
18	7890	GEORGE	CLERK	7566	03-MAY-85	1235	(null)	50
19	7629	BOB	SALESMAN	7598	06-MAR-86	1800	1000	30

3. Write down SQL statements to perform following functions:-

i. Increase the salary by 250 of all clerks with a salary less than 900



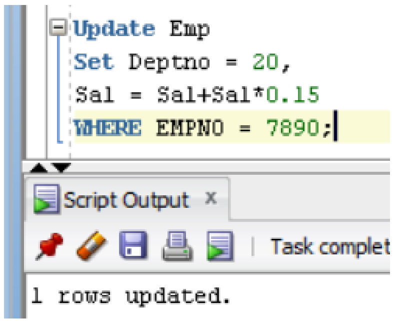
```
---QUESTION2----
Update Emp
Set SAL = Sal+250
WHERE JOB = 'CLERK' AND SAL < 900;
```

Script Output

Task completed in 0 seconds

```
1 rows updated.
```

ii. Transfer the employee with number 7890 to department 20 and increase his salary by 15%.



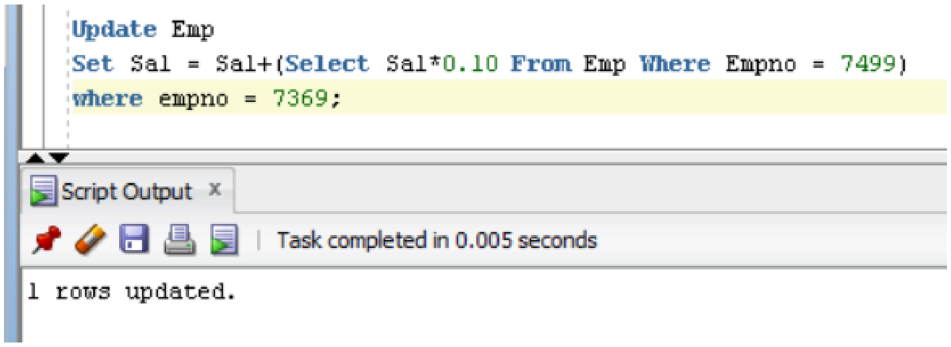
```
Update Emp
Set Deptno = 20,
Sal = Sal+Sal*0.15
WHERE EMPNO = 7890;
```

Script Output

Task complet

```
1 rows updated.
```

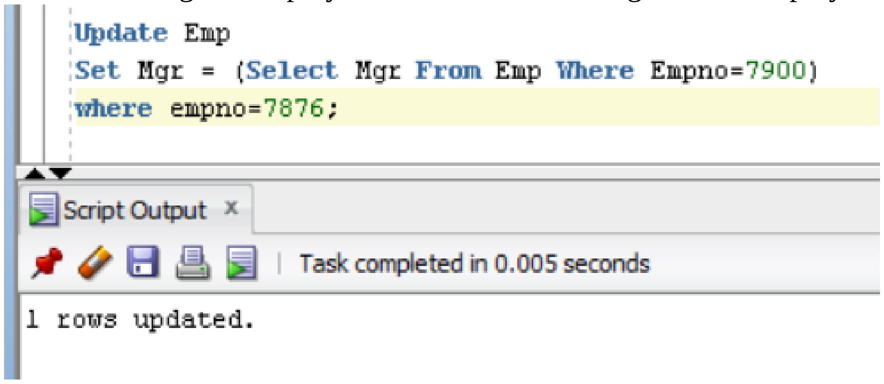
iii. Increase the salary of employee with number 7369 by 10% of the salary of employee with number 7499.



```
Update Emp
Set Sal = Sal+(Select Sal*0.10 From Emp Where Empno = 7499)
where empno = 7369;
```

```
1 rows updated.
```

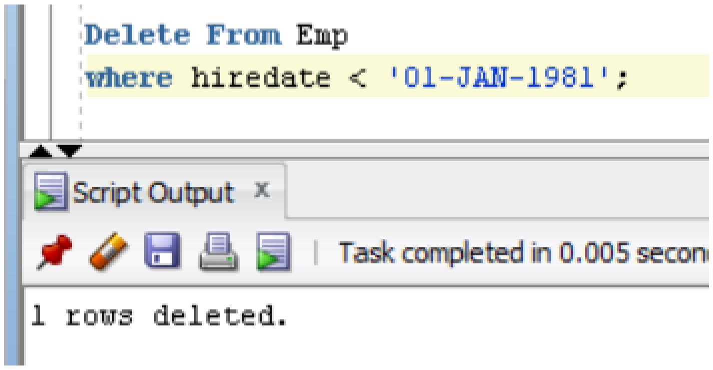
iv. Assign to employee 7876 the same manager as the employee 7900.



```
Update Emp
Set Mgr = (Select Mgr From Emp Where Empno=7900)
where empno=7876;
```

```
1 rows updated.
```

v. Remove all employees who were hired before 1981.



```
Delete From Emp
where hiredate < '01-JAN-1981';
```

```
1 rows deleted.
```