Experiment no: - 5

Experiment Name: - DML and Transaction control statements.

Aim: - Performing practical by using DML statements and control transaction.

Resource required: - Oracle 9i - iSQLplus

Theory: -

- **DATA MANIPULATION LANGUAGE:**
- **DML statements:**

A DML statement is executed when:
- Add new rows to a table
- Modify existing rows in a table
- Remove existing rows from a table

A transaction consists of a collection of DML statements that form a logical unit of work.

INSERT Statement: add new rows to a table

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

1. Insert one – row at a time
   e.g.: INSERT INTO departments (department_id, department_name, manager_id, location-id)
   VALUES (70, „Public Relations", 100, 1700);
2. Inserting rows with Null values
   e.g.: INSERT INTO departments
   VALUES (100, „Finance", NULL, NULL);
3. Inserting specific Date values
   e.g.: INSERT INTO employees
   VALUES (114, „Den", „Raphealy", „DRAPHEAL", „515.127.4561",TO_DATE („FEB 3, 1999", „MON DD, YYYY")"
   „AC_ACCOUNT", 11000, NULL, 100, 30);

UPDATE Statement: modify existing rows

Syntax: UPDATE table
SET column = value [, column = value …]
[WHERE condition];

1. Updating rows in a table
   e.g.: UPDATE employees
   SET department_id = 70
   WHERE employee_id = 113;
2. Updating rows based on another table: use subqueries

e.g.: UPDATE copy_emp
SET department_id = (SELECT department_id
FROM employees
WHERE employee_id = 100)
WHERE job_id = (SELECT job_id
FROM employees
WHERE employee_id = 200);

DELETE Statement: remove existing rows from a table

Syntax: DELETE [FROM] table
[WHERE condition];

1. Deleting rows from a table

e.g.: DELETE FROM departments
WHERE department_name = „Finance“;

2. Deleting rows based on another table

e.g.: DELETE FROM employees
WHERE department_id =
(SELECT department-id
FROM departments
WHERE department_name LIKE „%Public %“);

MERGE Statement: ability to insert and update rows in a table

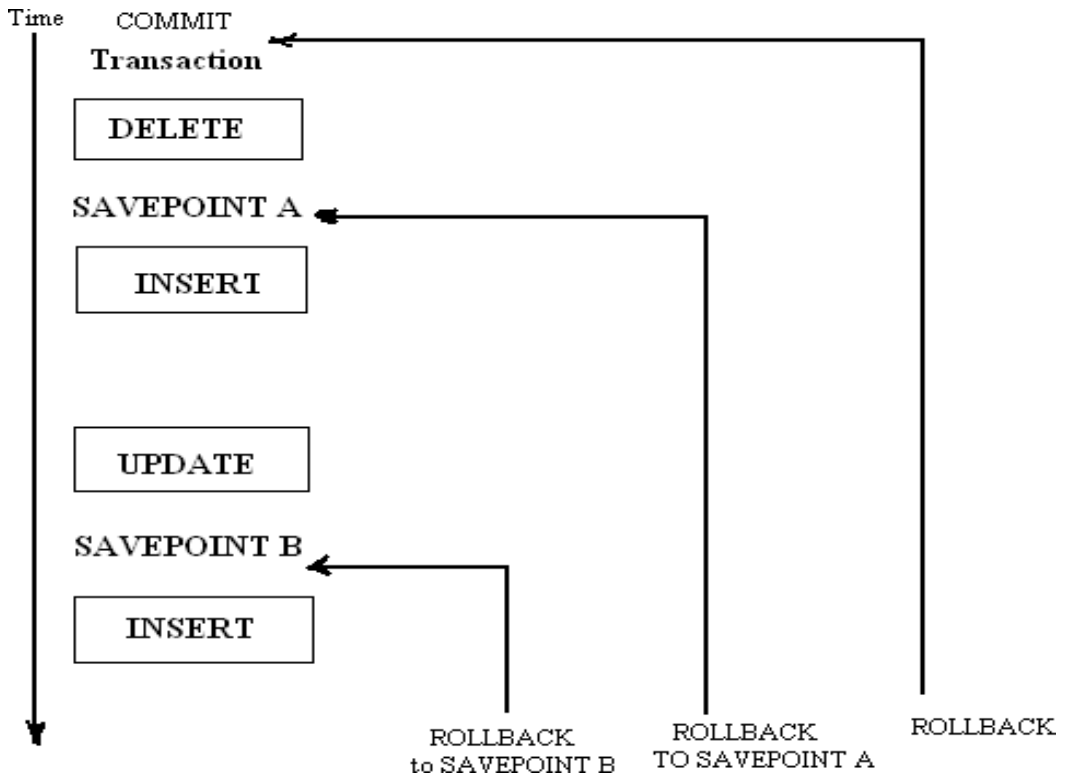
Syntax: MERGE INTO table_name table_alias
USING (table/ view/ sub_query) alias
ON (join condition)
WHEN MATCHED THEN
UPDATE SET
Col1 = col_val1,
Col2 = col2_val
WHEN NOT MATCHED THEN
INSERT (column_list)
VALUES (column_values);

e.g.: MERGE INTO copy_emp c
USING employees e
ON (c.employee-id = e.employee_id)
WHEN MATCHED THEN
UPDATE SET
…
WHEN NOT MATCHED THEN
INSERT VALUES……;

- **Database Transactions:**
  Consists of one of the following:
  - DML statement with one consistent change to the data
  - One DDL statement
  - One DCL statement

Begin when the first DML SQL statement is executed and end with a COMMIT or ROLLBACK statement is issued.

Controlling Transactions:



Control the logic of a transaction by using COMMIT, ROLLBACK, and SAVEPINT statements.
COMMIT: Ends the current transaction by making all pending data changes

SAVEPOINT: Marks a savepoint within the current transaction

ROLLBACK: ends the current transaction by discarding all pending data changes.

State of the data before COMMIT or ROLLBACK:

- The previous sate of data can be recovered.
- The current user review the result of DML operations by using the SELECT statement
- Other user can‟t views the result of the DML statements by the current user.
- The affected rows are locked: other user can‟t change the data within affected rows.

Sate of data after COMMIT;

- Data change may be permanent in the database.
- The previous sate of the data is permanently lost
- All users can view the results.
- Locks on affected rows are released
- All savepoints are erased.

Committing Data:

- make the changes

e.g.: DELETE FROM employees
WHERE employee_id = 99999;

INSERT INTO departments
VALUES (290, „Corporate Tax“, NULL, 1700);

- Commit the changes

e.g.: COMMIT;
Commit complete.

Sate of data after ROLLBACK:

- Discard all pending changes by using the ROLLBACK statement.

e.g.: DELETE FROM copy_emp;
22 rows deleted
ROLLBACK;
Rollback complete.

LOCKING:

In an Oracle database, locks:

- Prevent destructive interaction between concurrent transactions
- Require no user action
- Automatically use the lowest level of restrictiveness
- Are held for the duration of the transaction
- Two types of locking: explicit locking and implicit locking
- Locks held until COMMIT and ROLLBACK.

Conclusion:

In this practical, learned how to use DML statements and control transactions.

LAB
ASSIGNMENT -5

Roll. No. B74	Name: Akash Kinage
Class: SE-B	Batch: B4
Date of Experiment: 11-03-2022	Date of Submission: 13-03-2022
Grade:	

1. Add the first row of data to the MY_EMPLOYEE table from the following sample Data. Do not list the column in the INSERT clause.

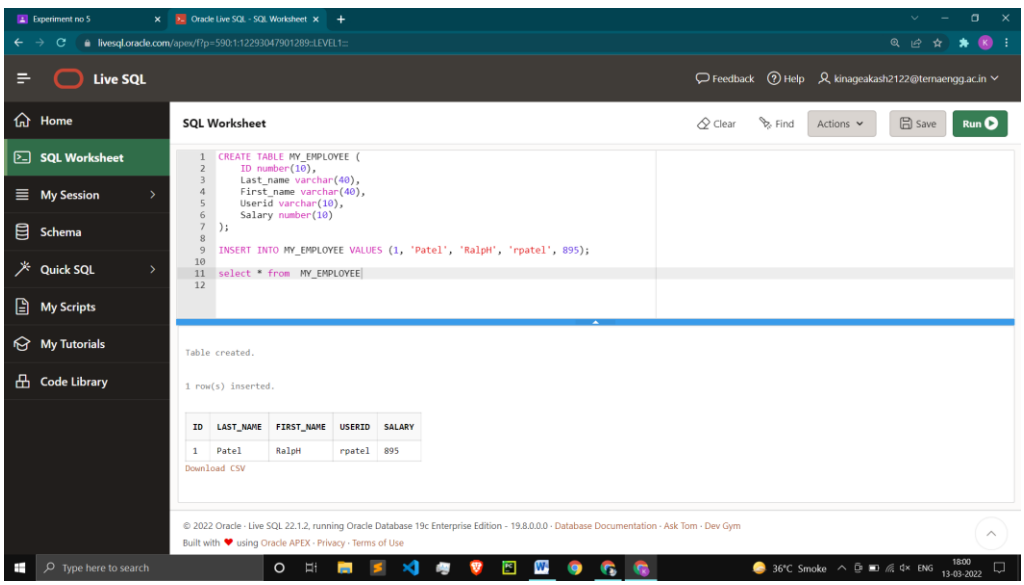
ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	Rpatel	895
2	Dance	Betty	Bdancs	860
3	Biri	Ben	Bbiri	1100
4	Newman	Chad	Cnewman	750
5	Ropeburn	Audrey	Aropebur	1550

Query:

```
CREATE TABLE MY_EMPLOYEE (
   ID number(10),
   Last_name varchar(40),
   First_name varchar(40),
   Userid varchar(10),
   Salary number(10)
);

INSERT INTO MY_EMPLOYEE VALUES (1, 'Patel', 'RalpH', 'rpatel', 895);
select * from  MY_EMPLOYEE
```

Output:

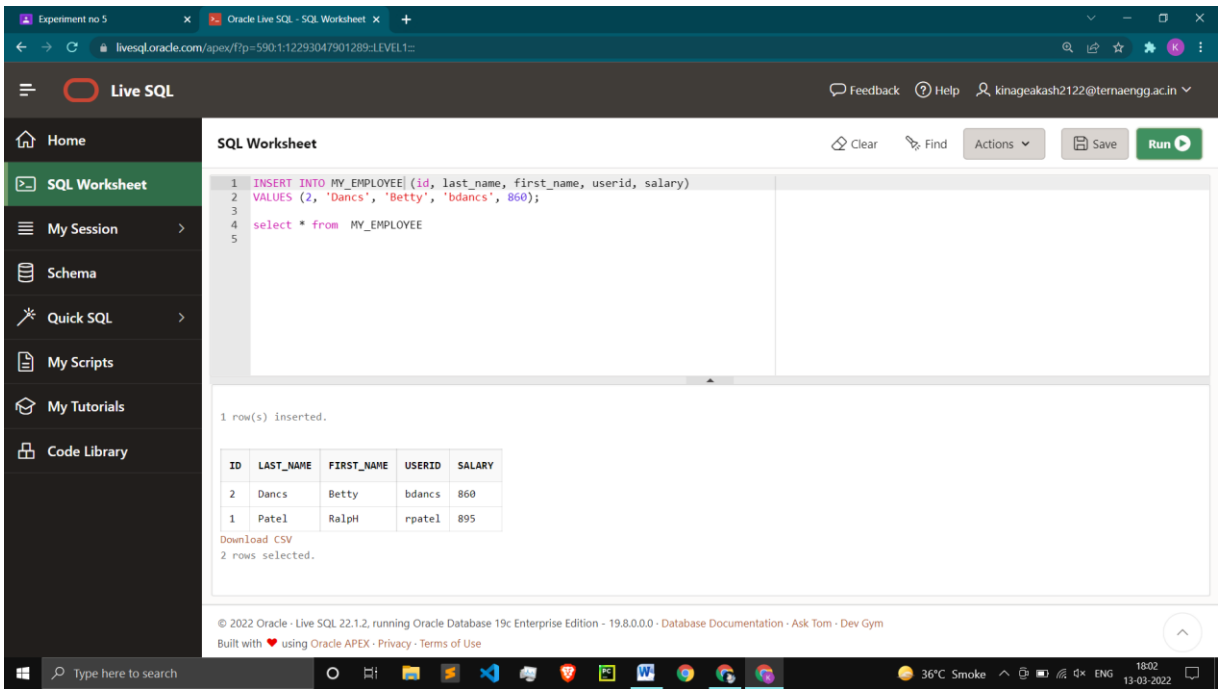


2. Populate the MY_EMPLOYEE table with the second row of sample data from the preceding list. This time, list the column explicitly in the INSERT clause.

Query:

```
INSERT INTO MY_EMPLOYEE (id, last_name, first_name, userid, salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

Output:



3. Change the name of employee 3 to Drexler.

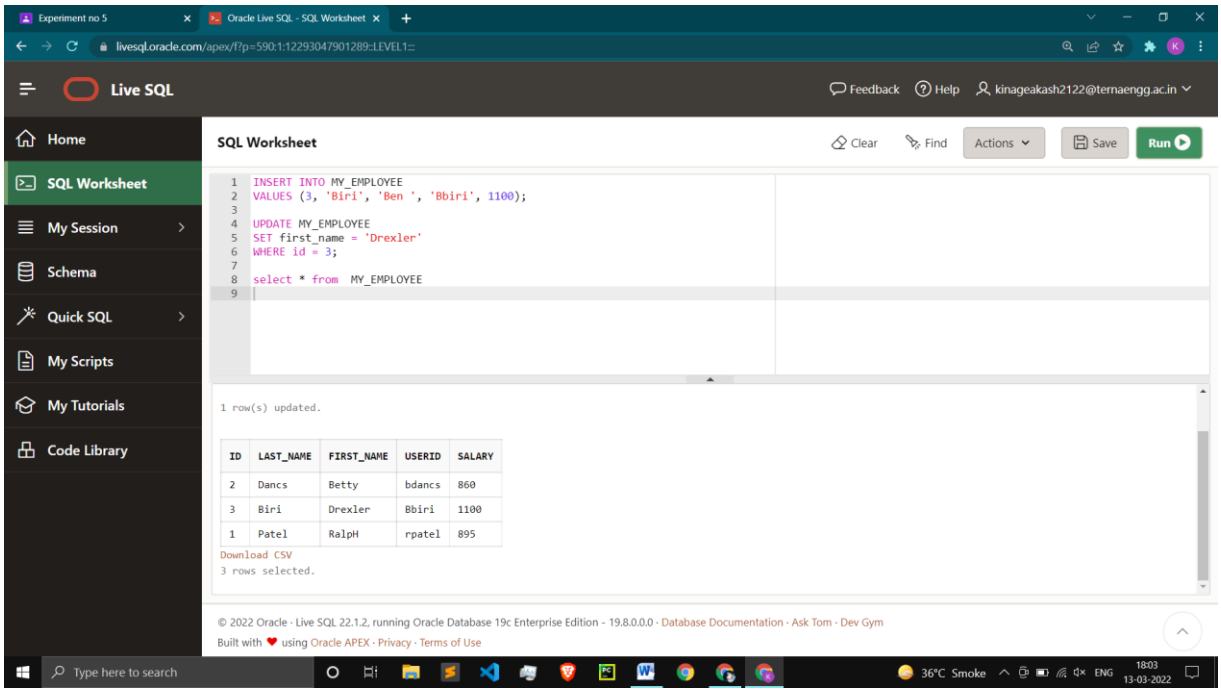
Query:

```
INSERT INTO MY_EMPLOYEE
VALUES (3, 'Biri', 'Ben ', 'Bbiri', 1100);

UPDATE MY_EMPLOYEE
SET first_name = 'Drexler'
WHERE id = 3;

select * from  MY_EMPLOYEE
```

Output:

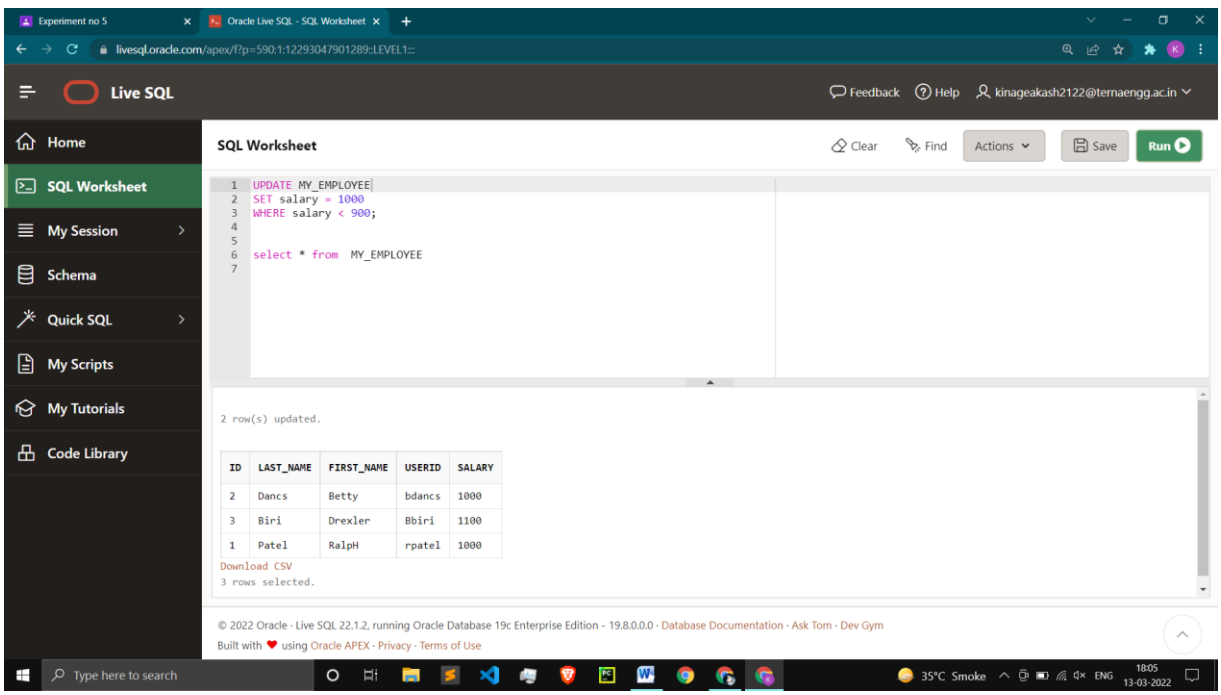


4. Change the salary to 1000 for all employee with a salary less than 900.

Query:

```
UPDATE my_employeee
SET salary = 1000
WHERE salary < 900;
```

Output:

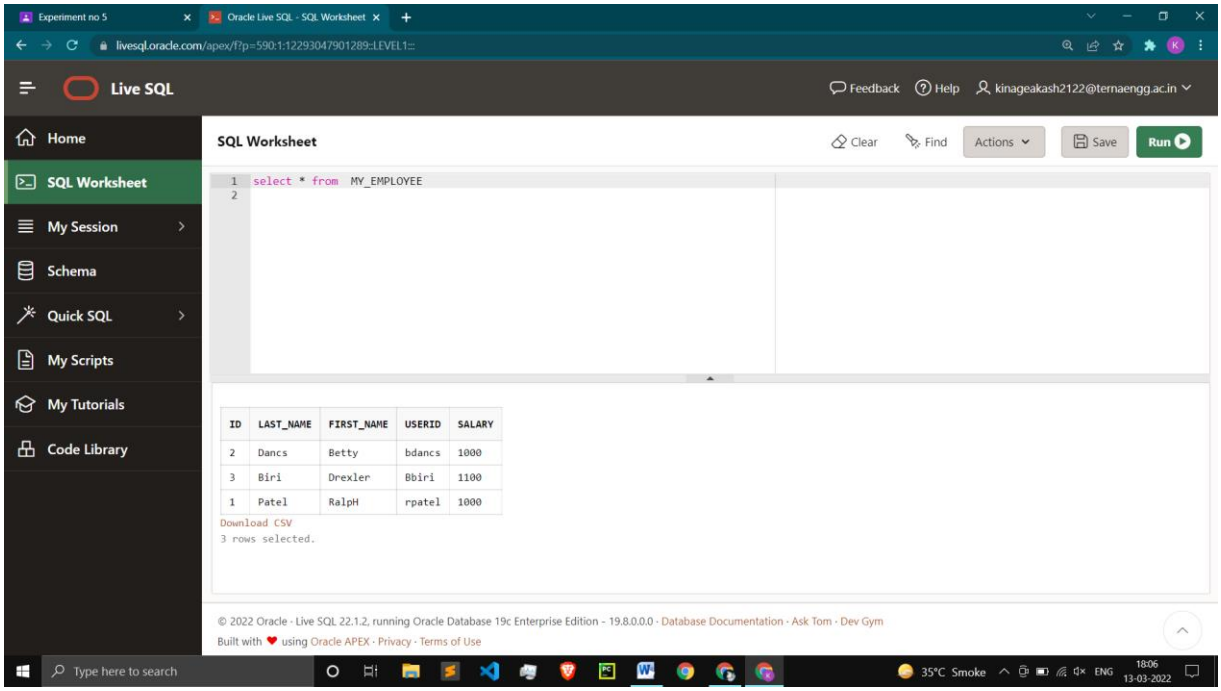


5. Confirm your addition to the table.

Query:

```
select * from  MY_EMPLOYEE;
```

Output:

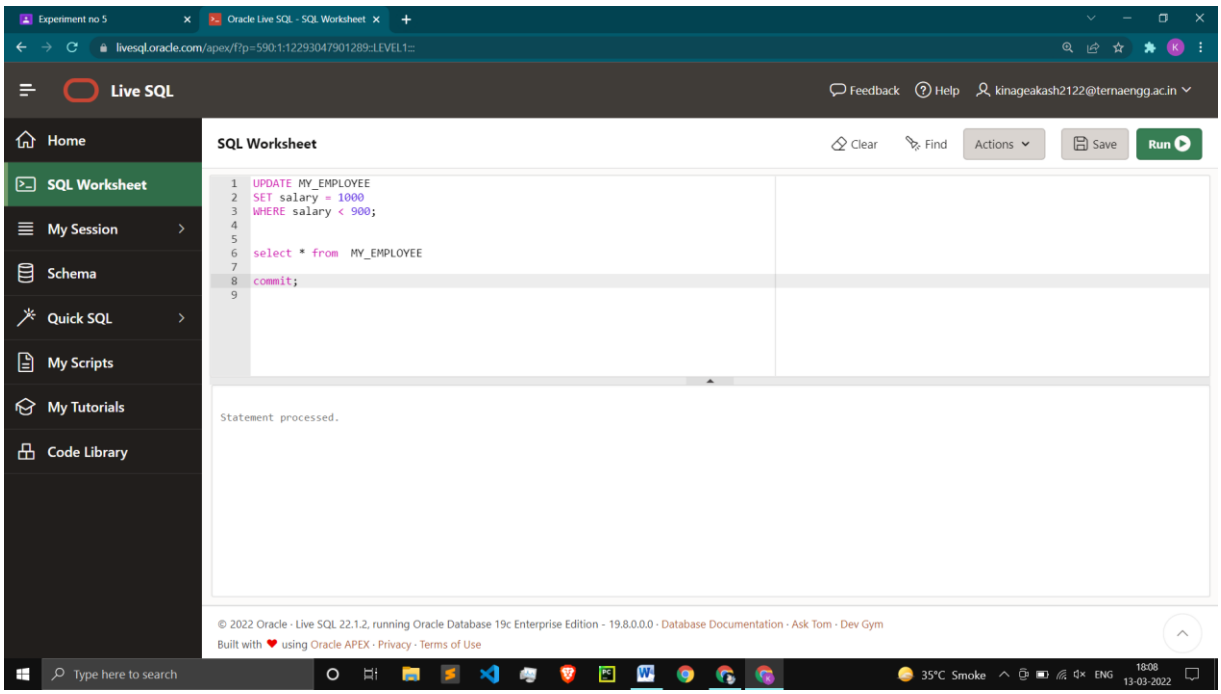


6. Commit all pending changes. Control data transaction to the MY_EMPLOYEE table.

Query:

```
Commit;
```

Output:

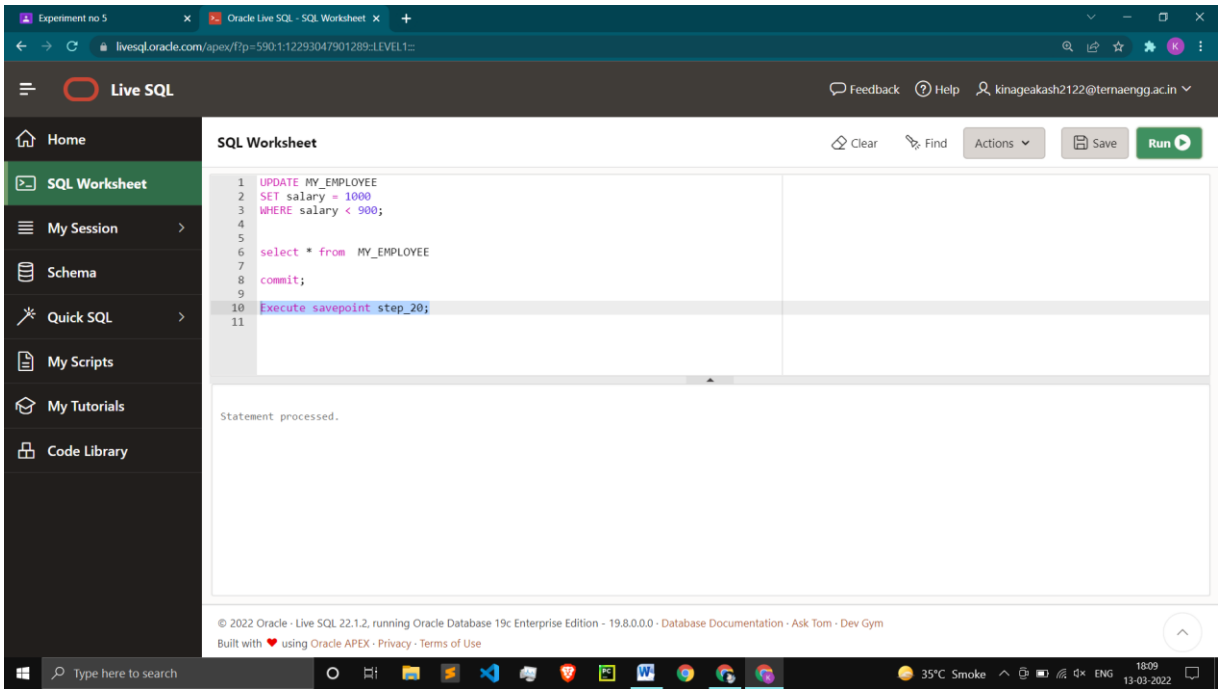


7. Mark an intermediate point in the processing of the transaction.

Query:

Execute savepoint step_20;

Output:

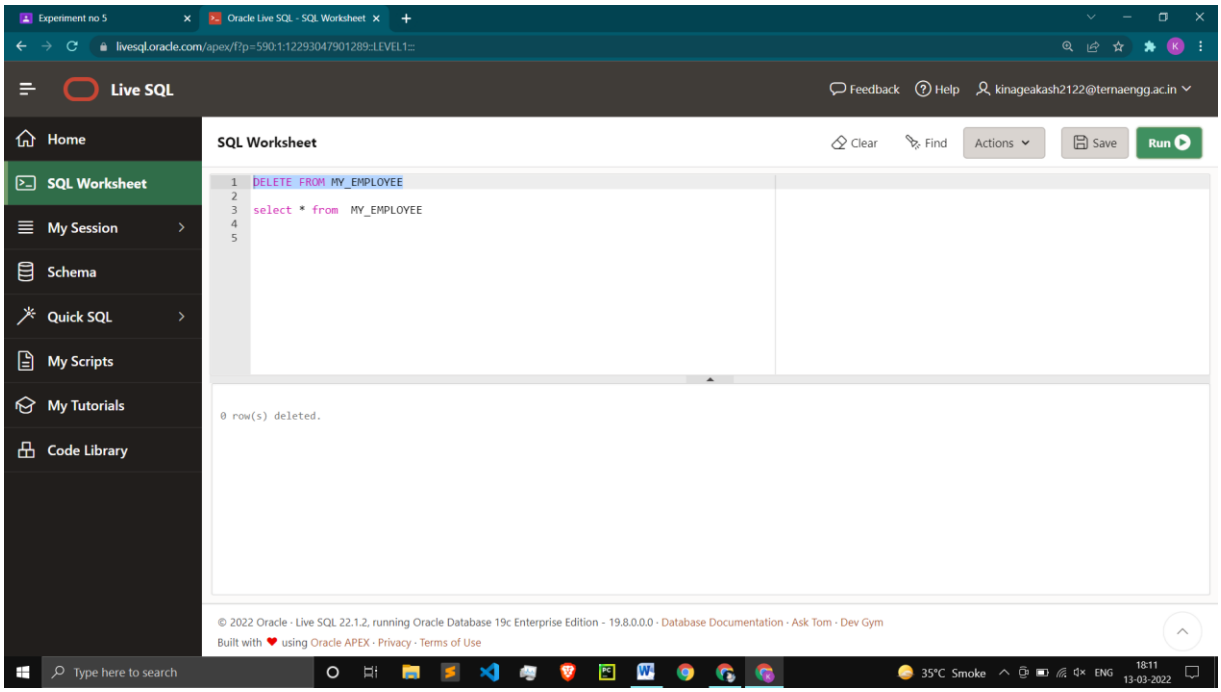


8. Empty the entire table.

Query:

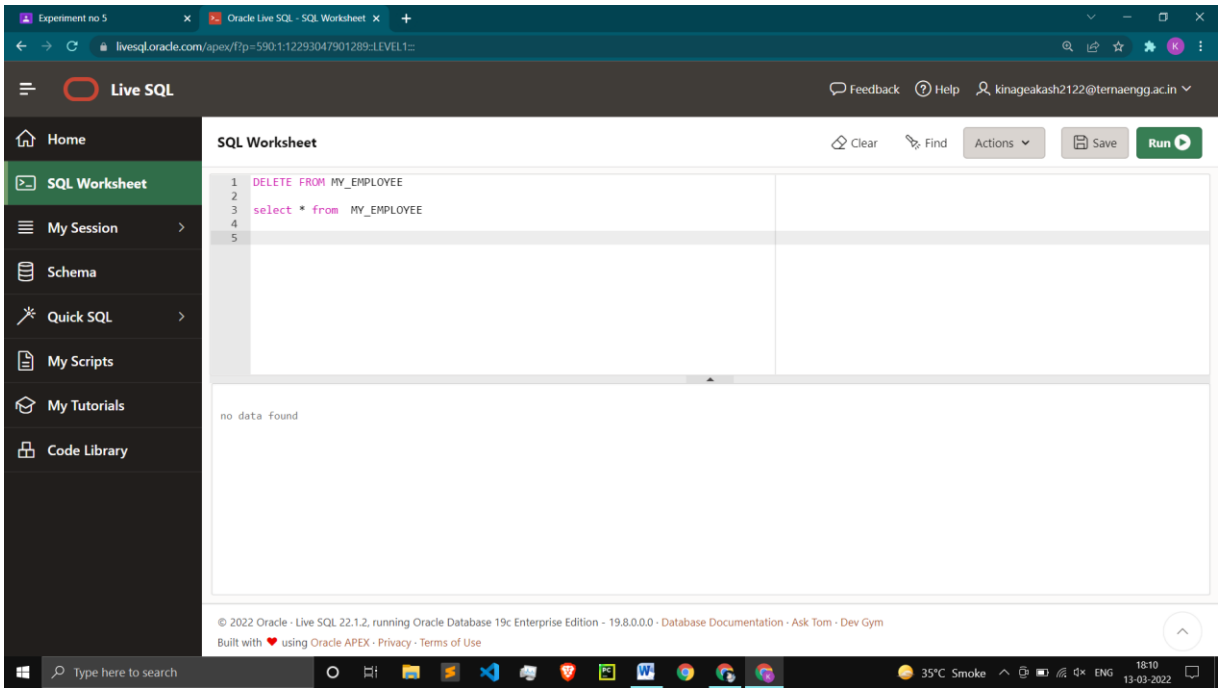
DELETE FROM MY_EMPLOYEE

Output:



9. Confirm that table is empty.

Output:



10. Discard the recent DELETE operation without discarding the earlier INSET Operation.

Query:

```
Rollback;
```

Output: