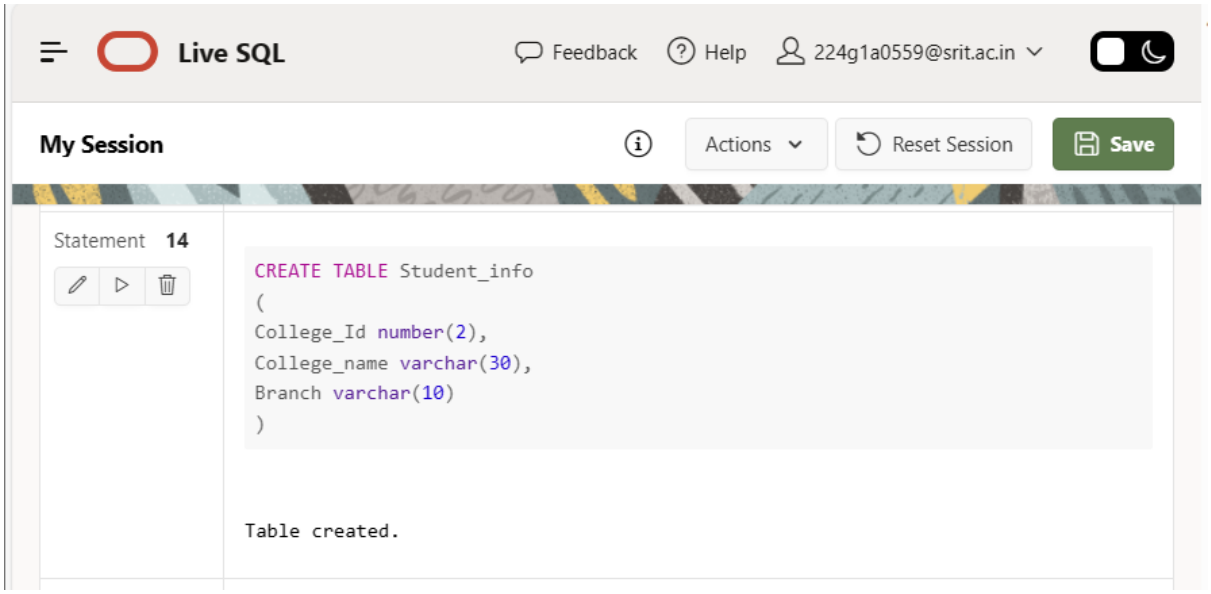
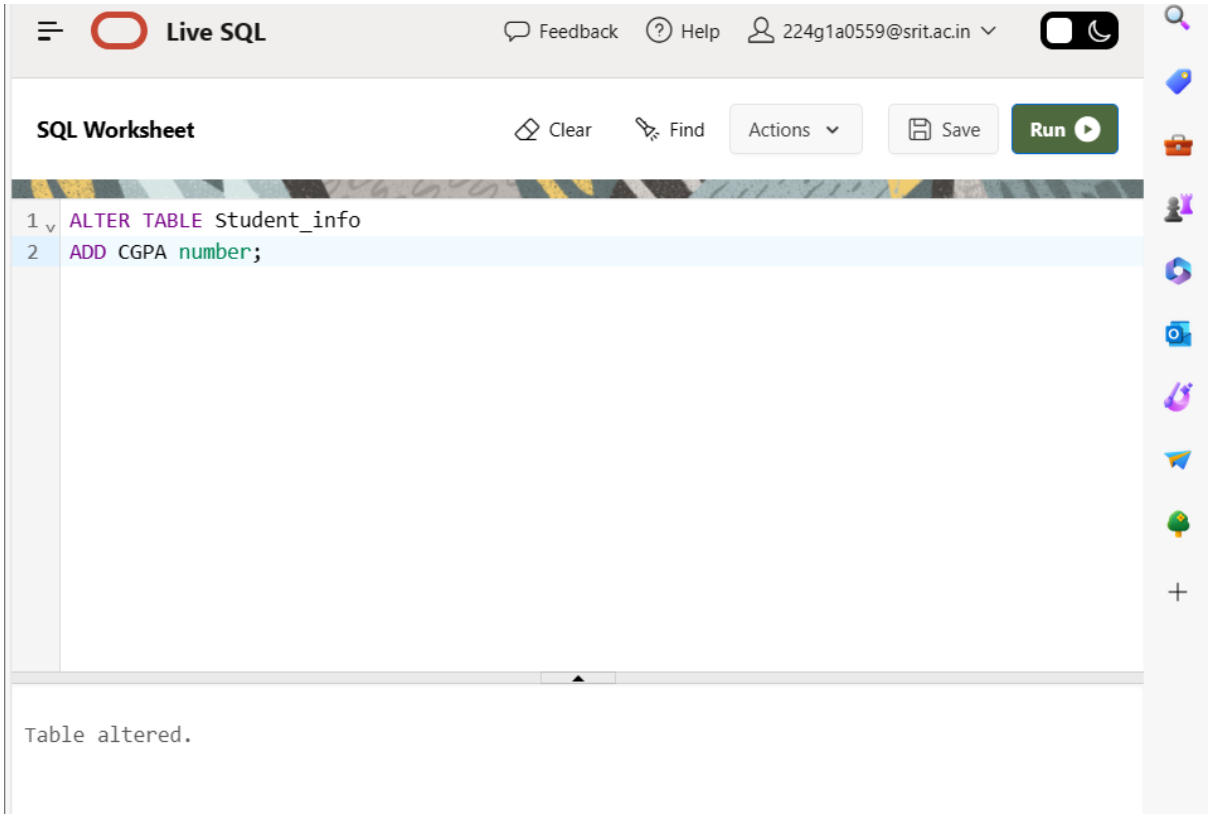
EXPERIMENT – 1

1.CREATE Table:



2.ALTER Table:



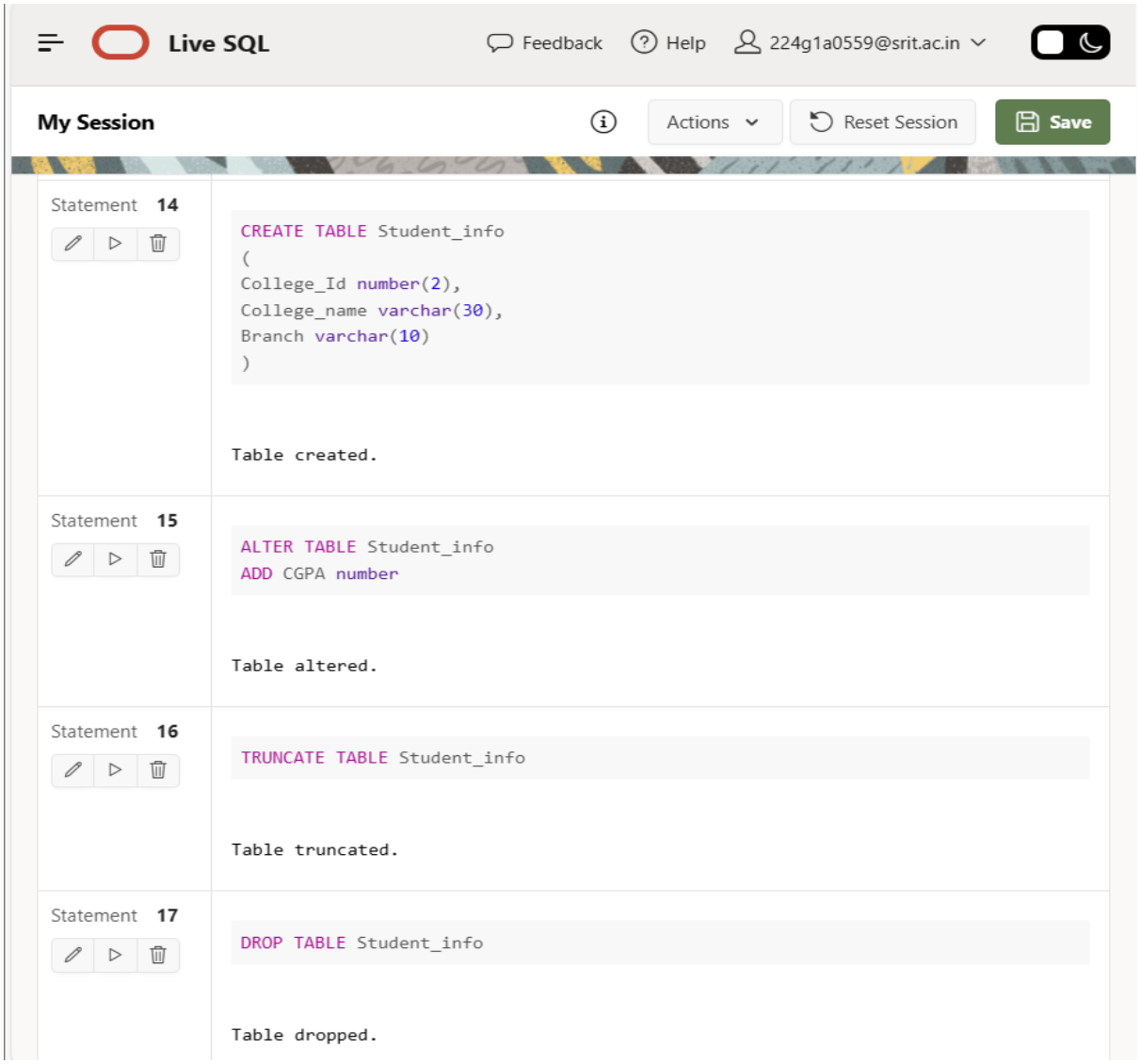
3.DROP Table:

Statement 17

```
DROP TABLE Student_info
```

```
Table dropped.
```

4. TRUNCATE Table:

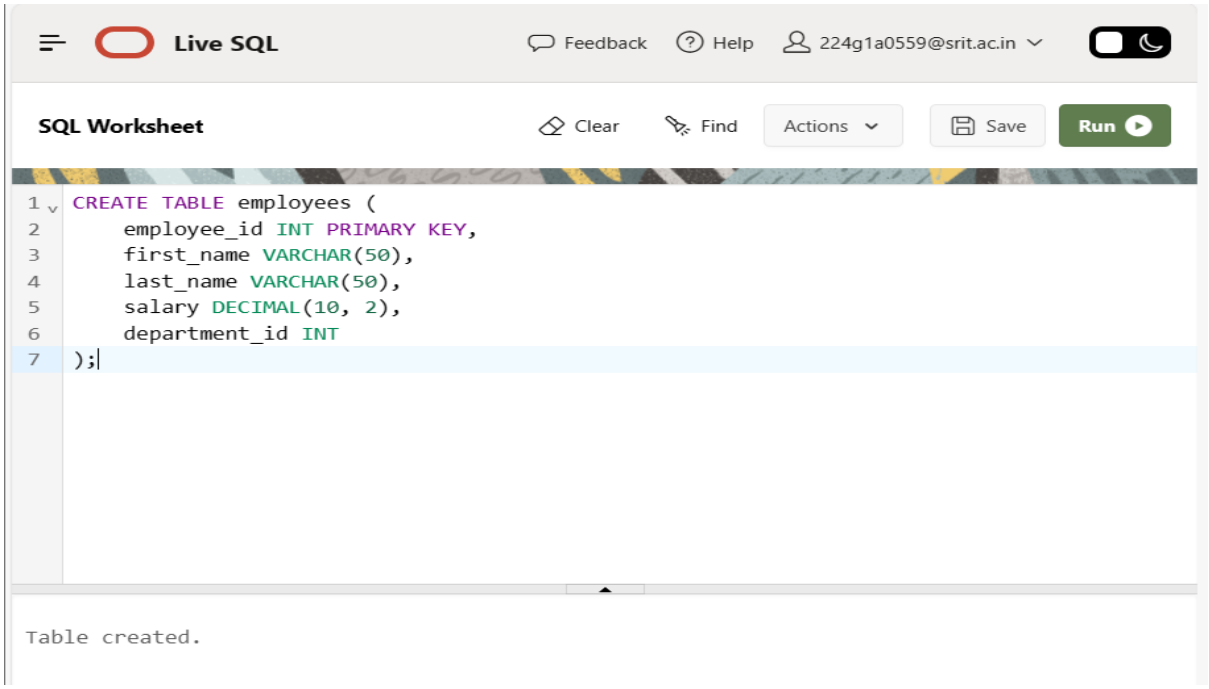


Conclusion:

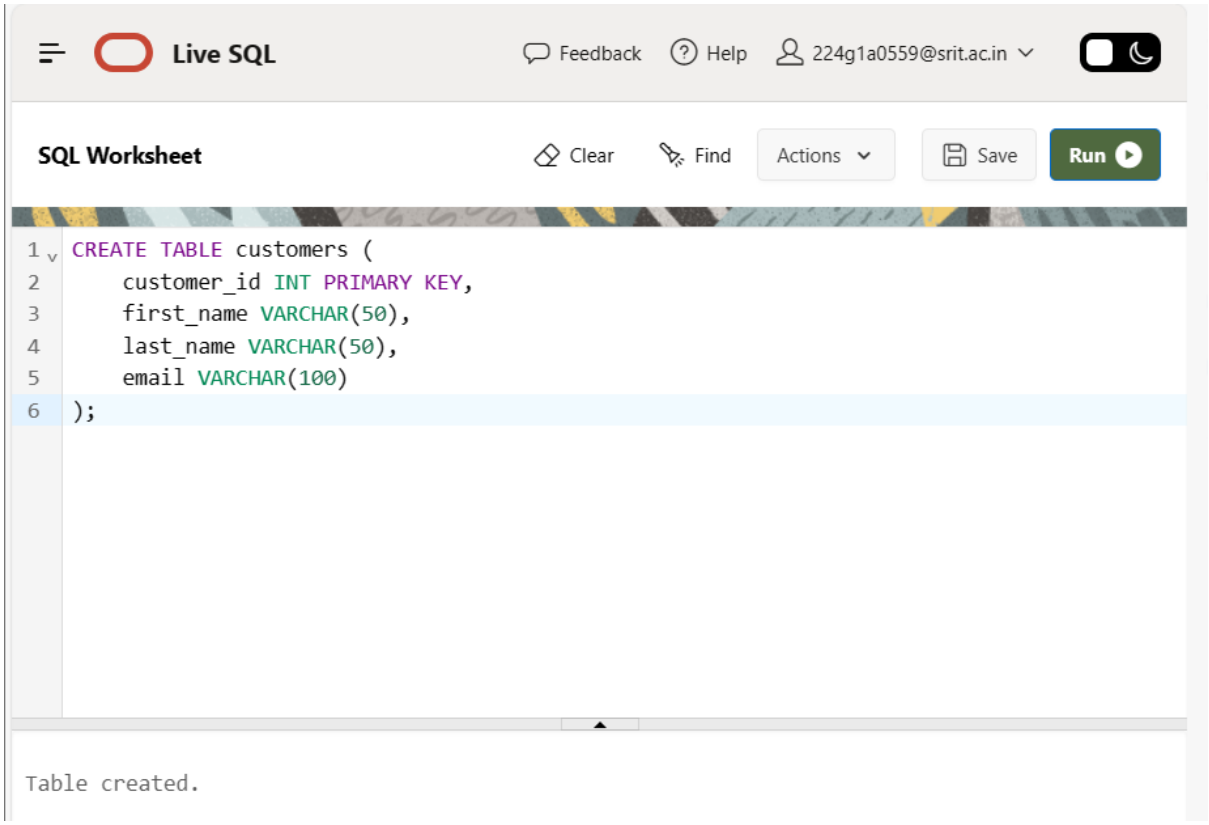
In this lab, we successfully practiced SQL Queries to CREATE TABLES for various databases using DDL commands.

Experiment – 2

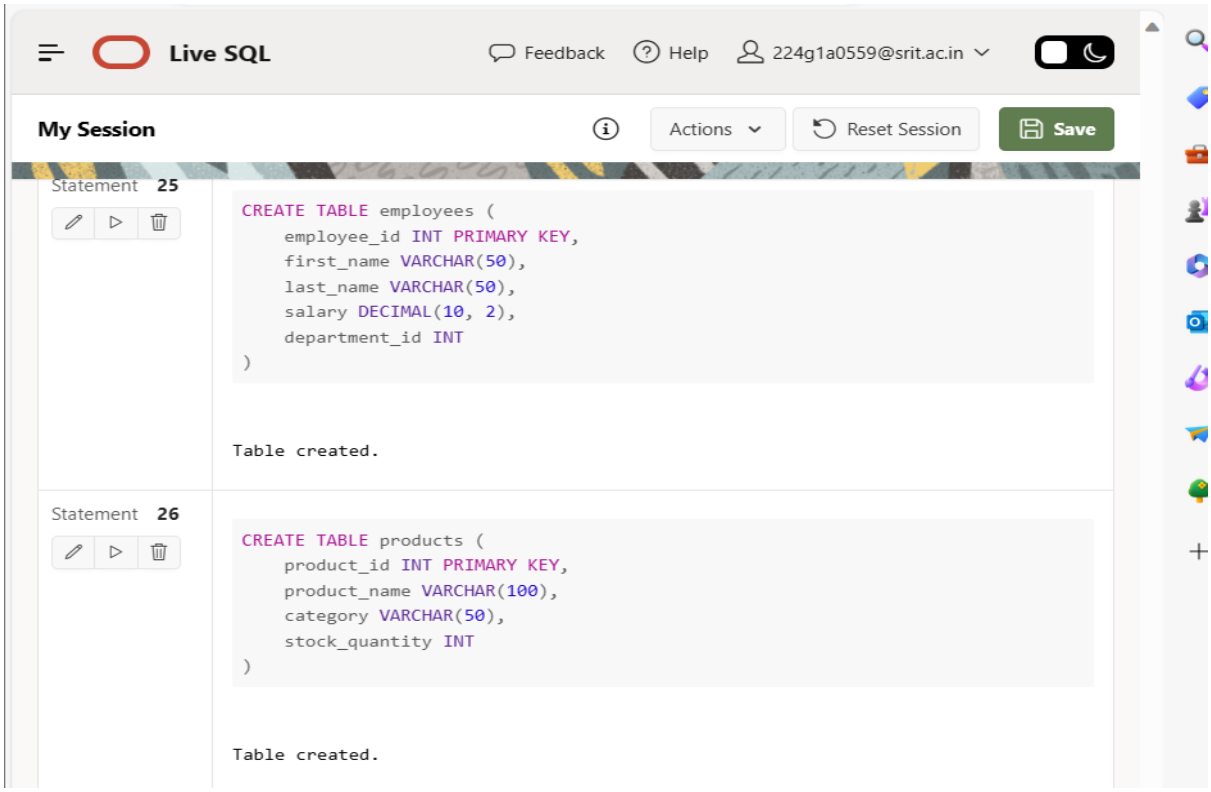
1.CREATE TABLE for Employees:



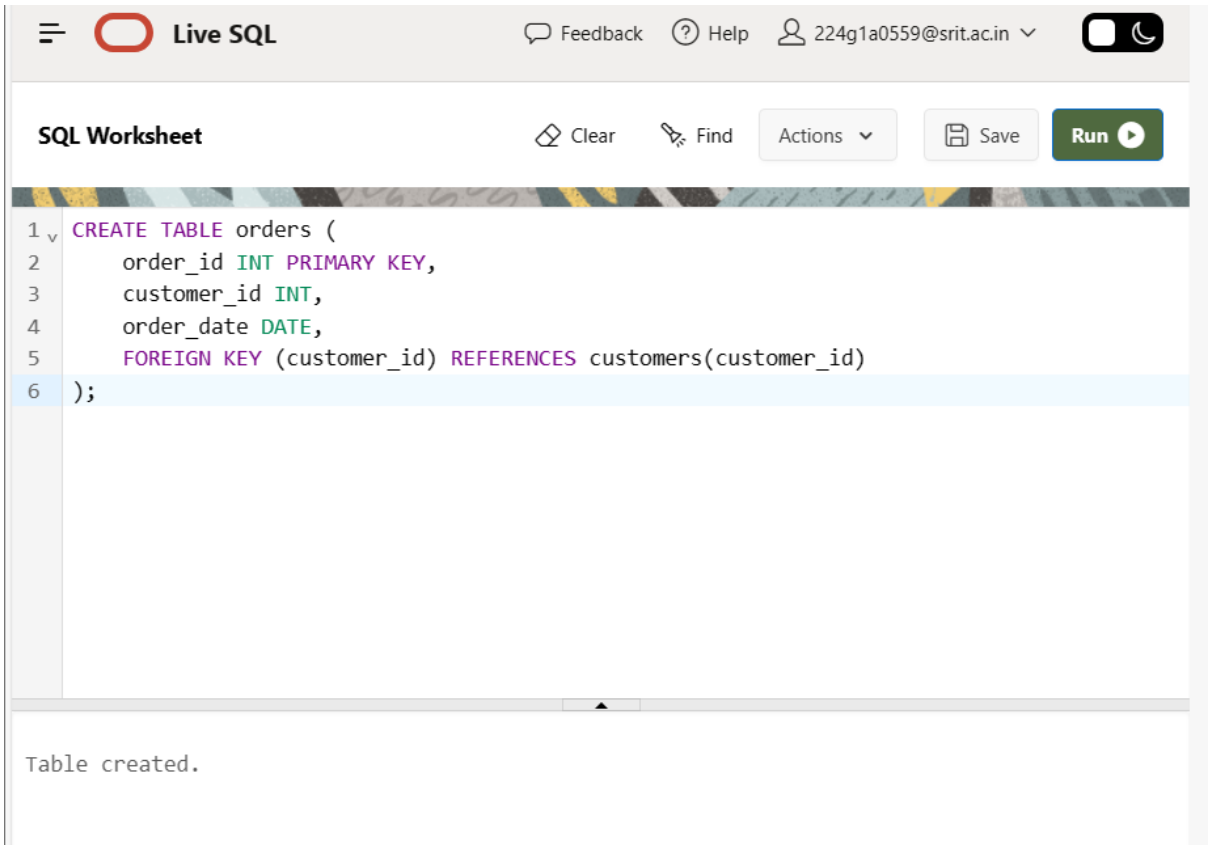
2.CREATE TABLE for customers:



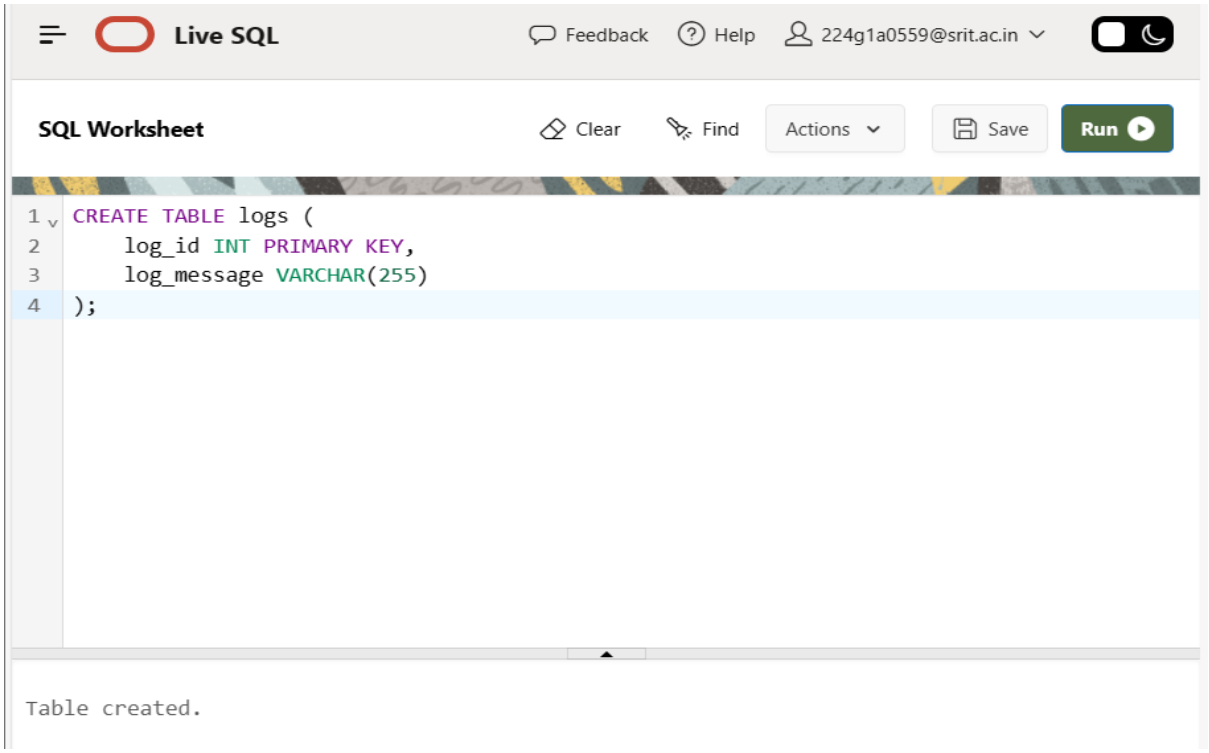
3.CREATE TABLE for products:



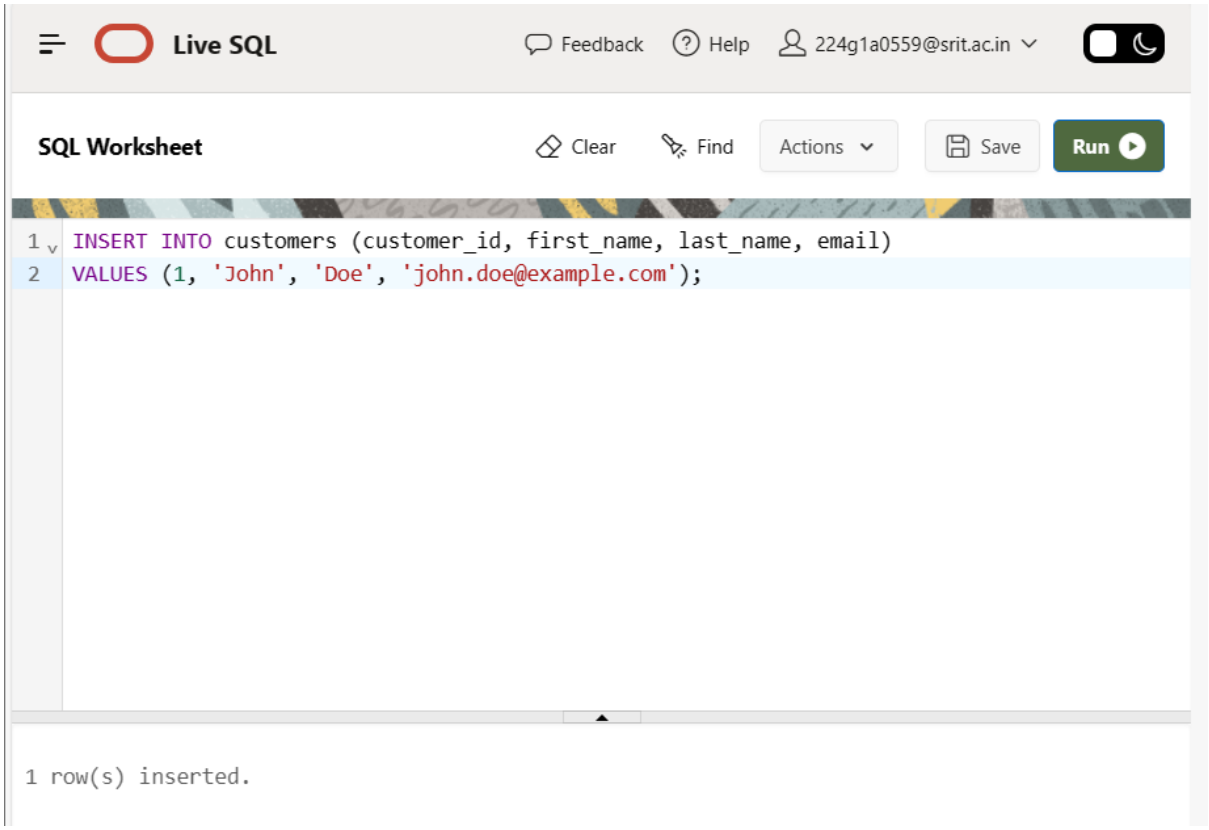
4.CREATE TABLE for orders:



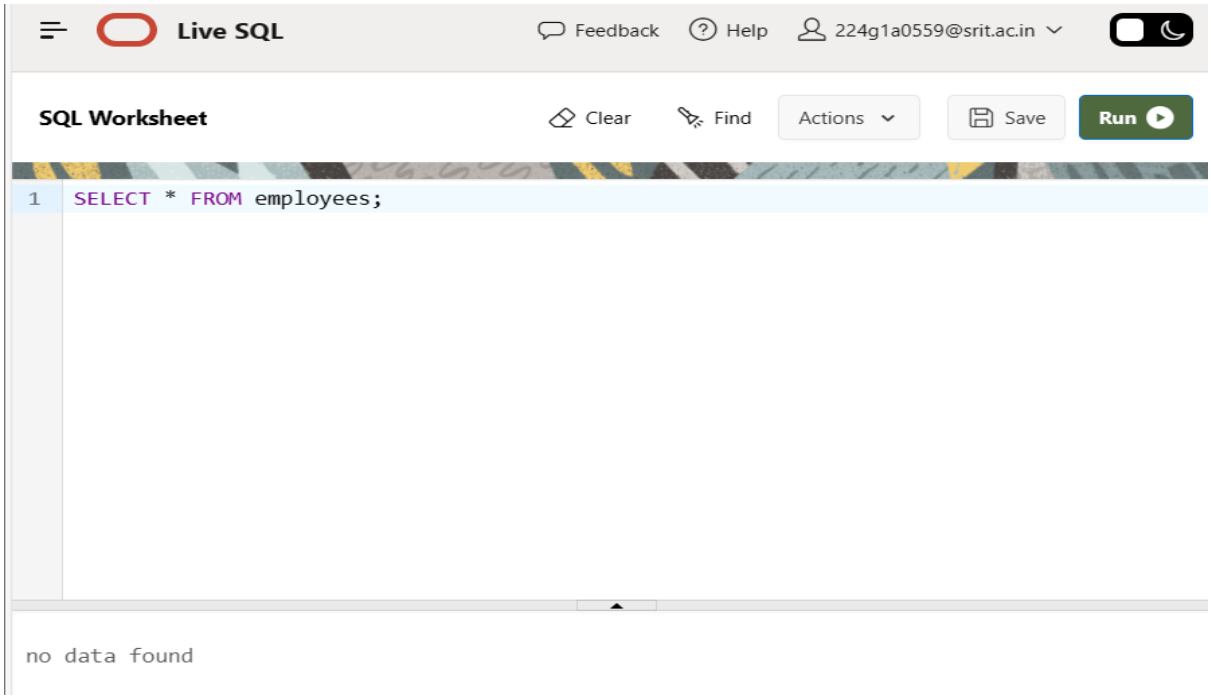
5.CREATE TABLE for logs:



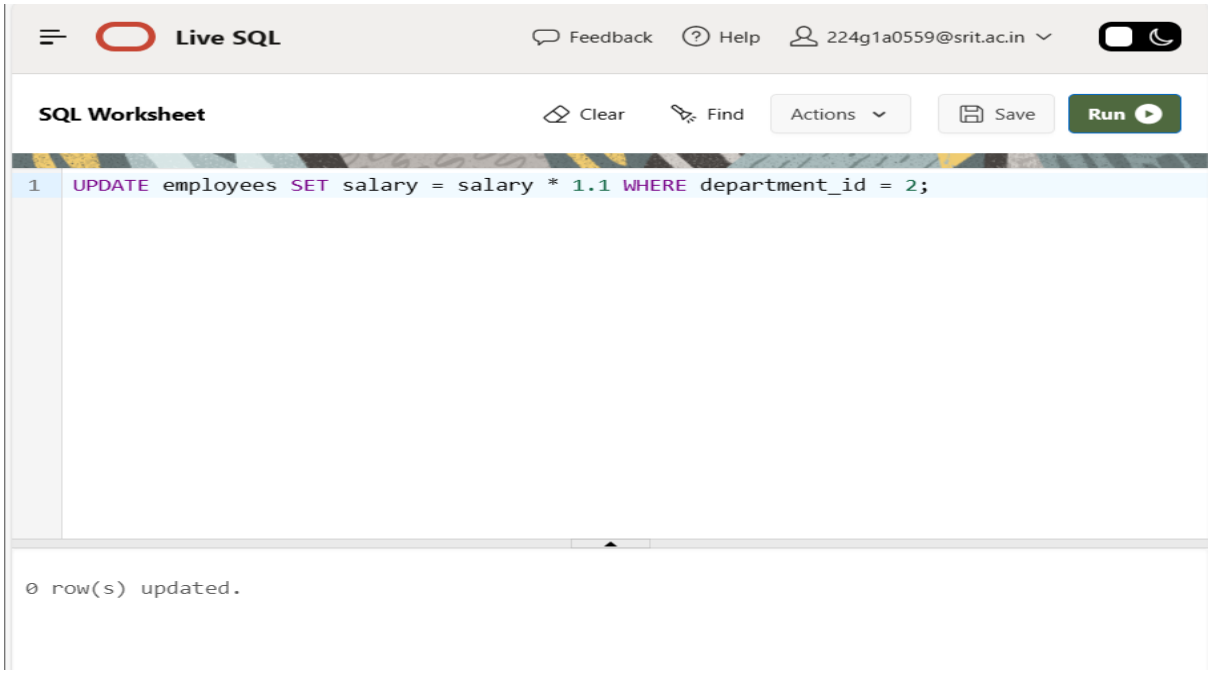
6.INSERT:



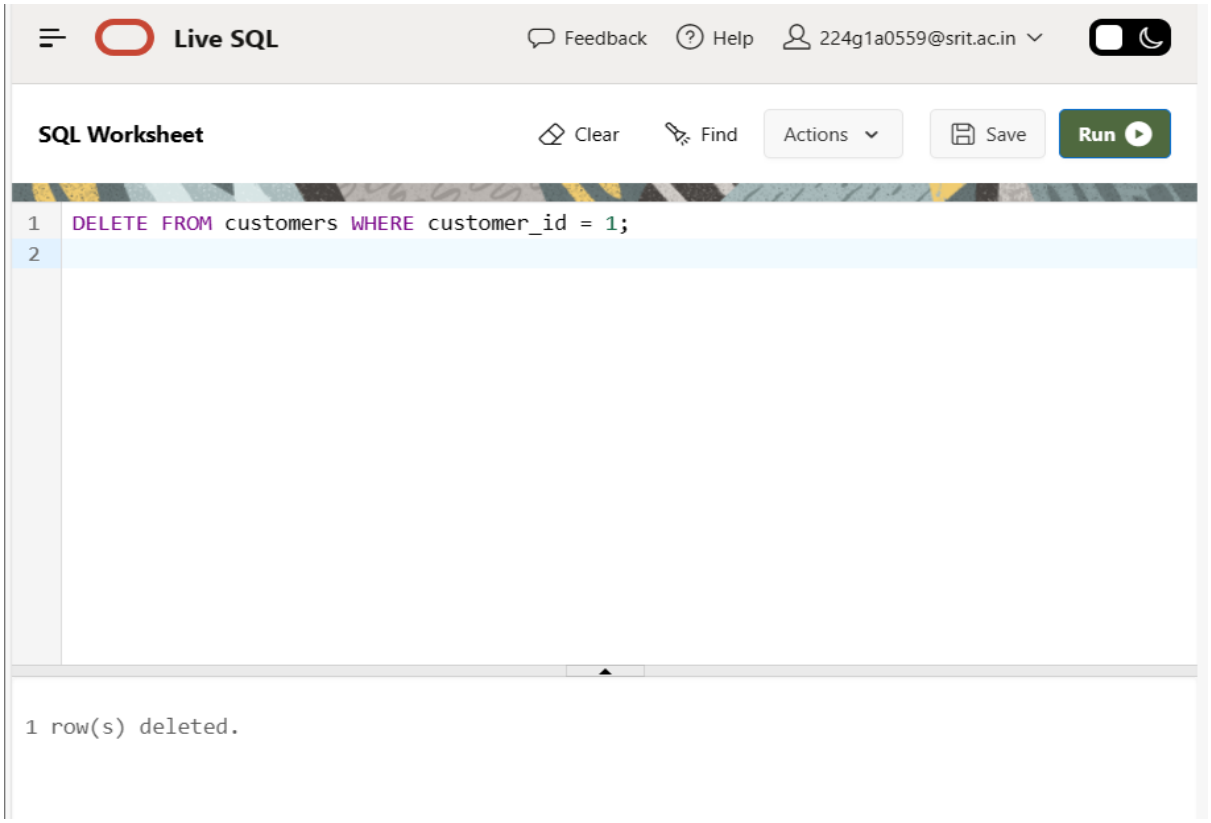
5.SELECT:



6.UPDATE:



7.DELETE:

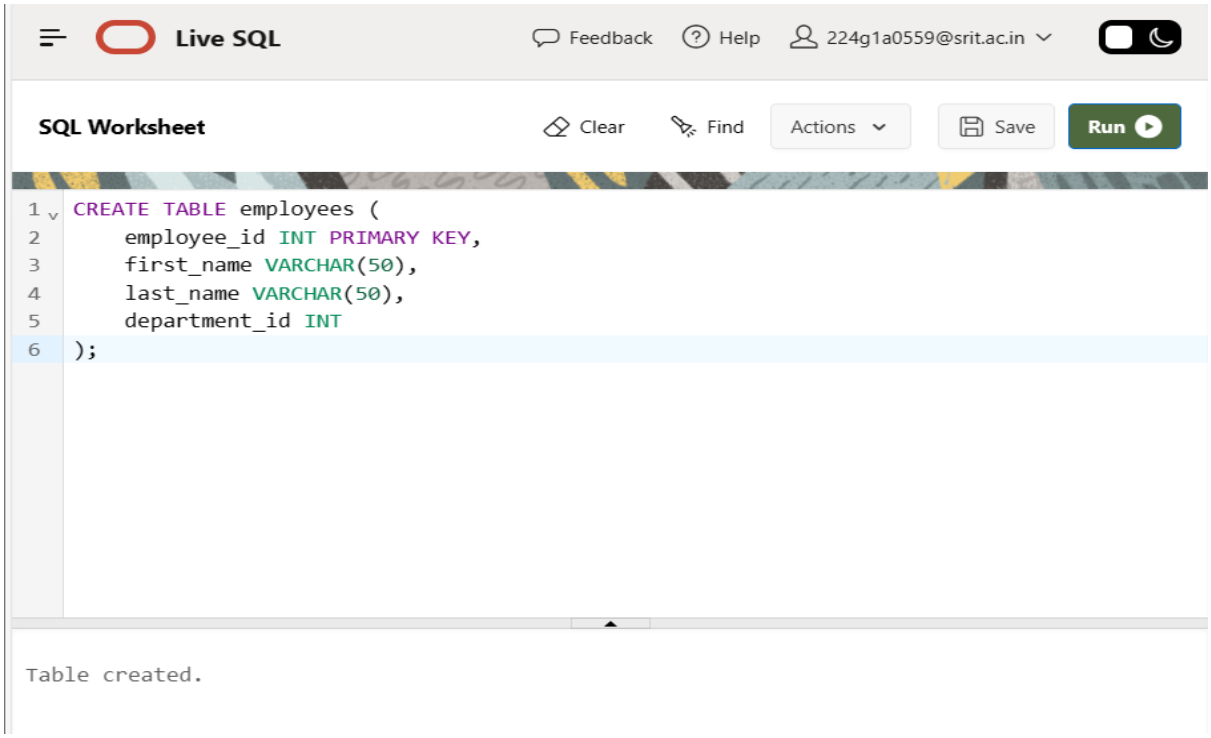


Conclusion:

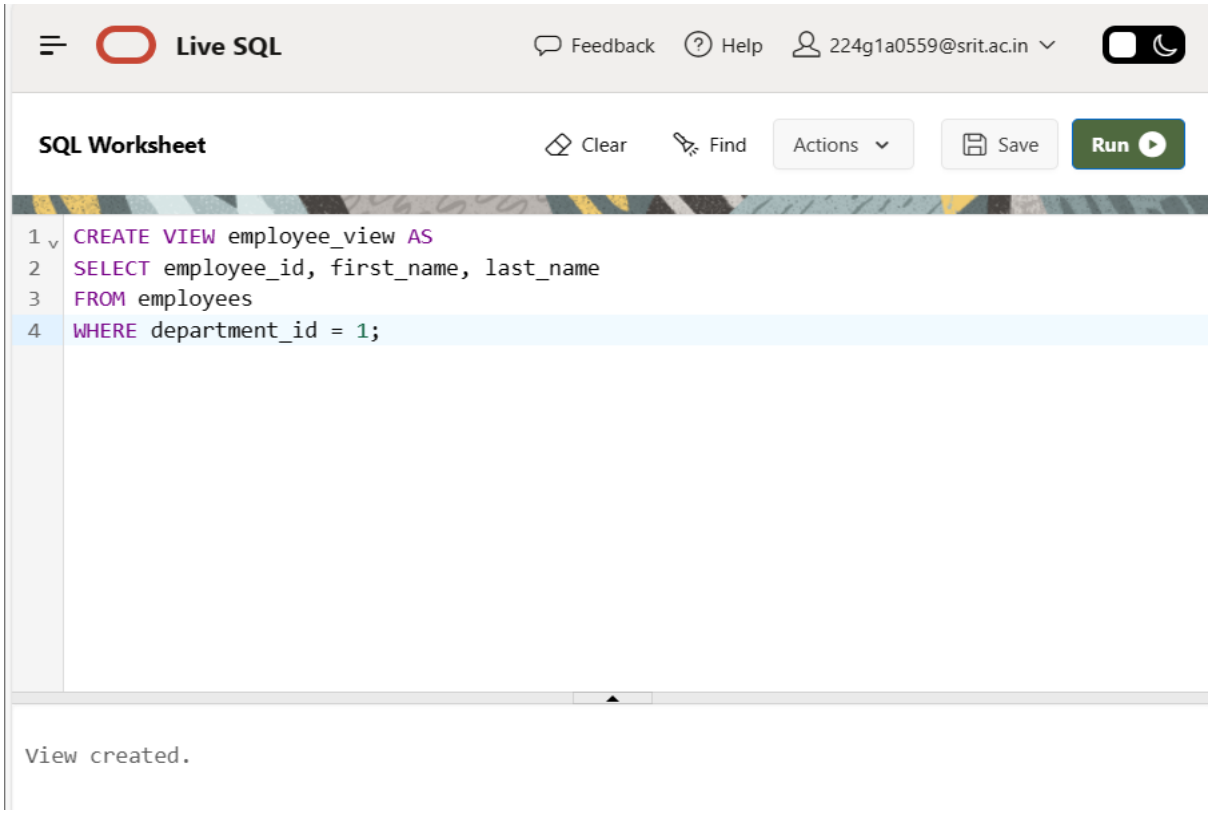
In this lab, we successfully practiced SQL Queries using DML commands (SELECT ,UPDATE,INSERT,DELETE).

EXPERIMENT – 3

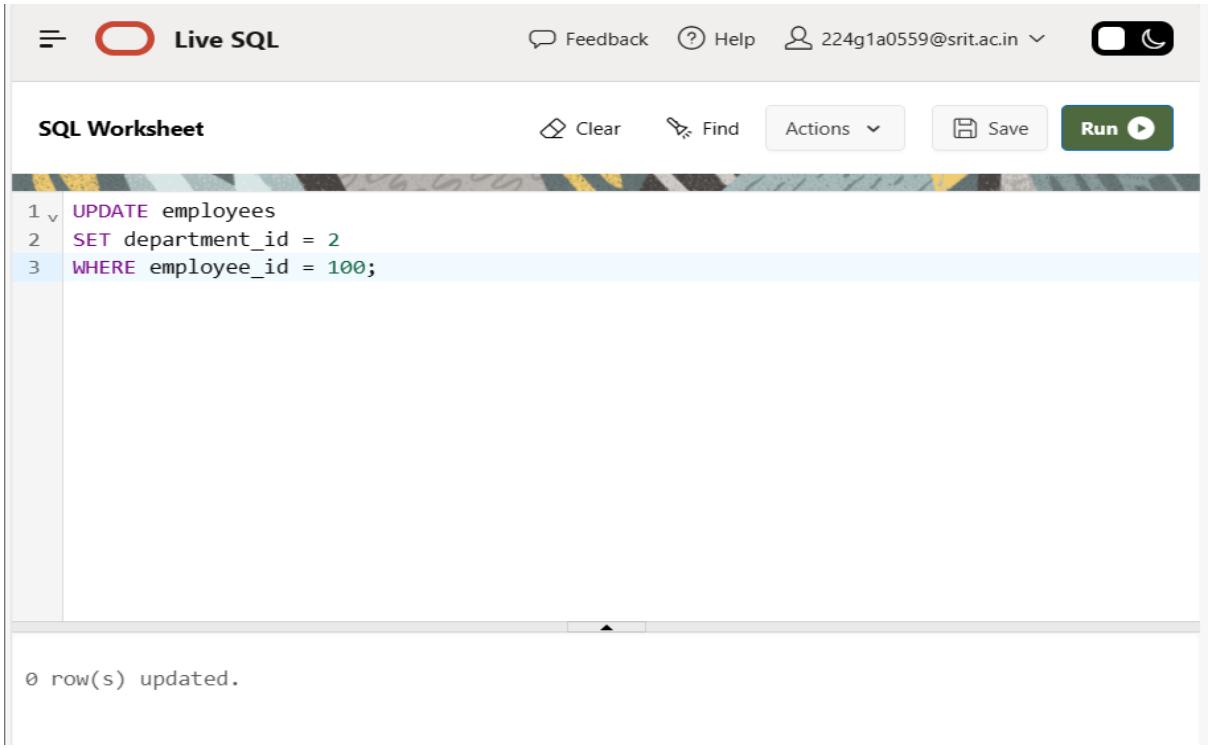
1.CREATE TABLE:



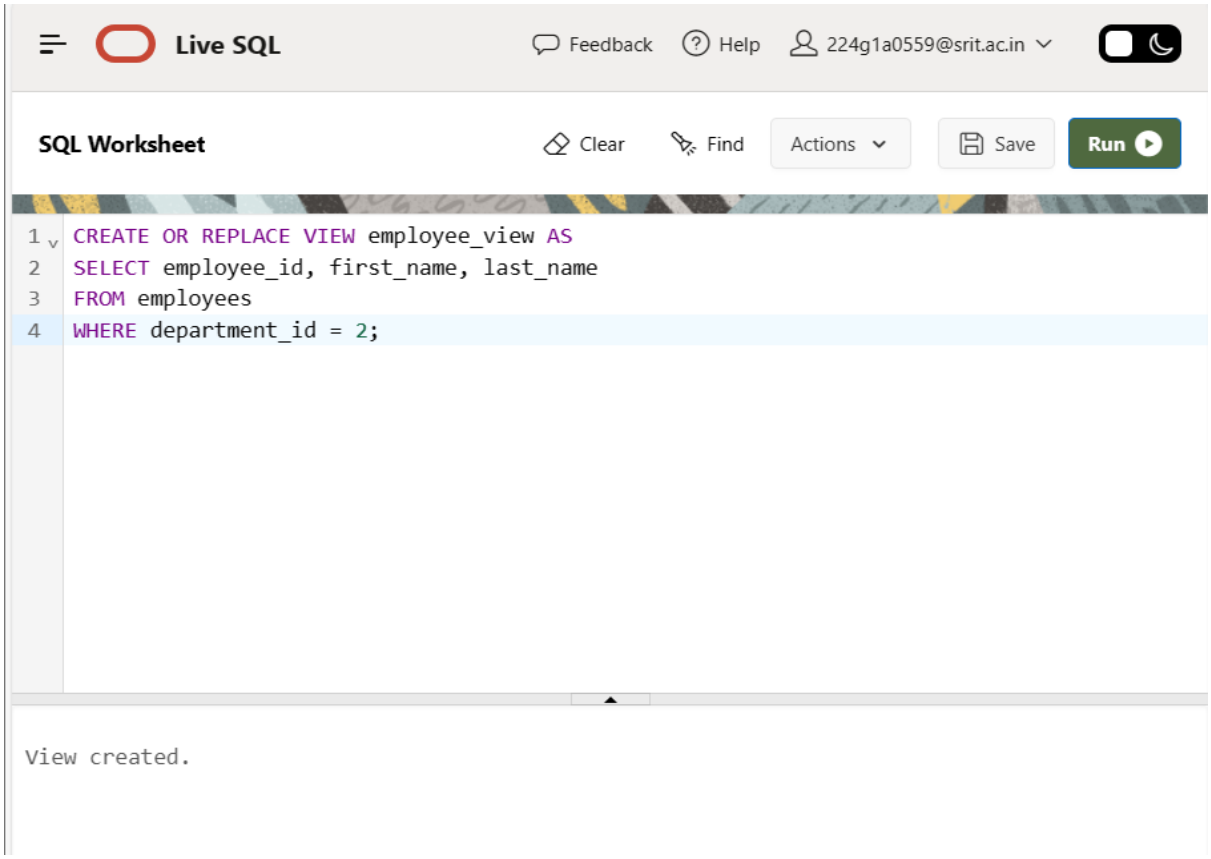
2.CREATING A VIEW:



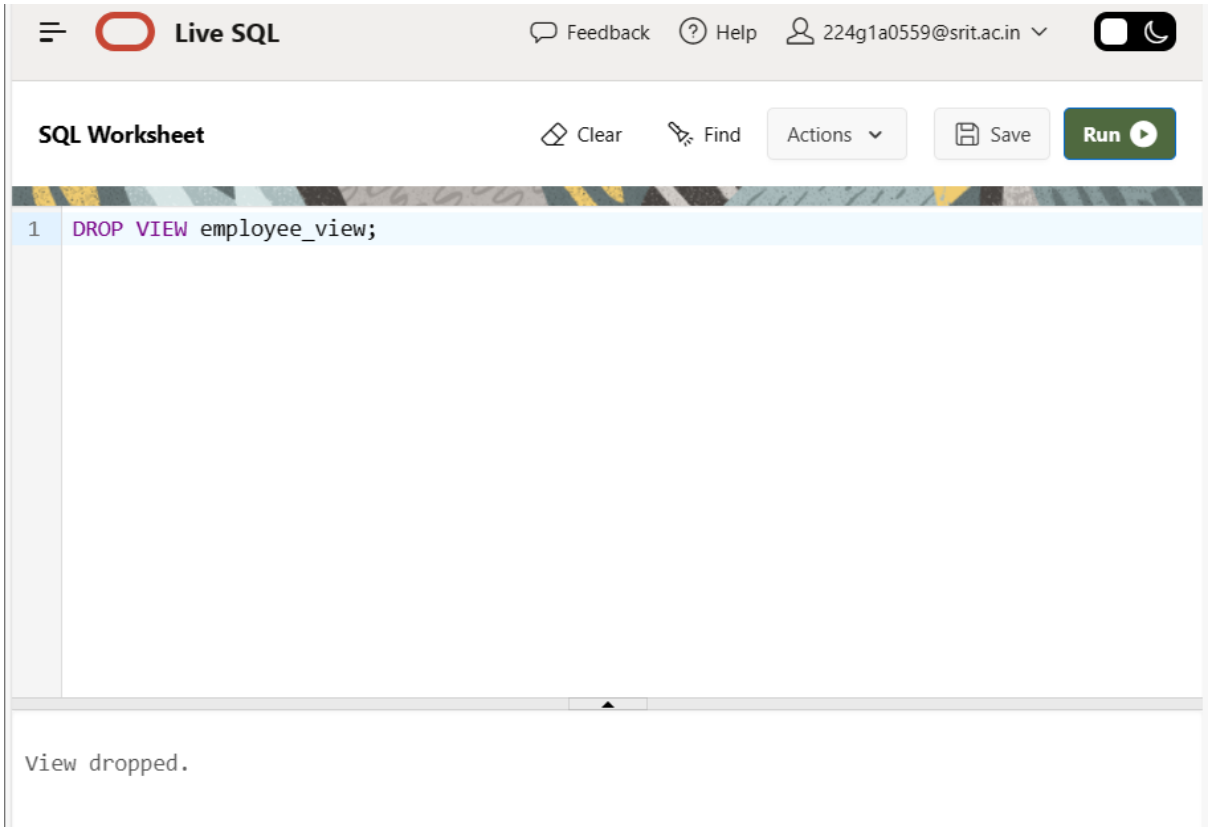
3.UPDATING A VIEW:



4.ALTERING A VIEW:



5.DELETING A VIEW:

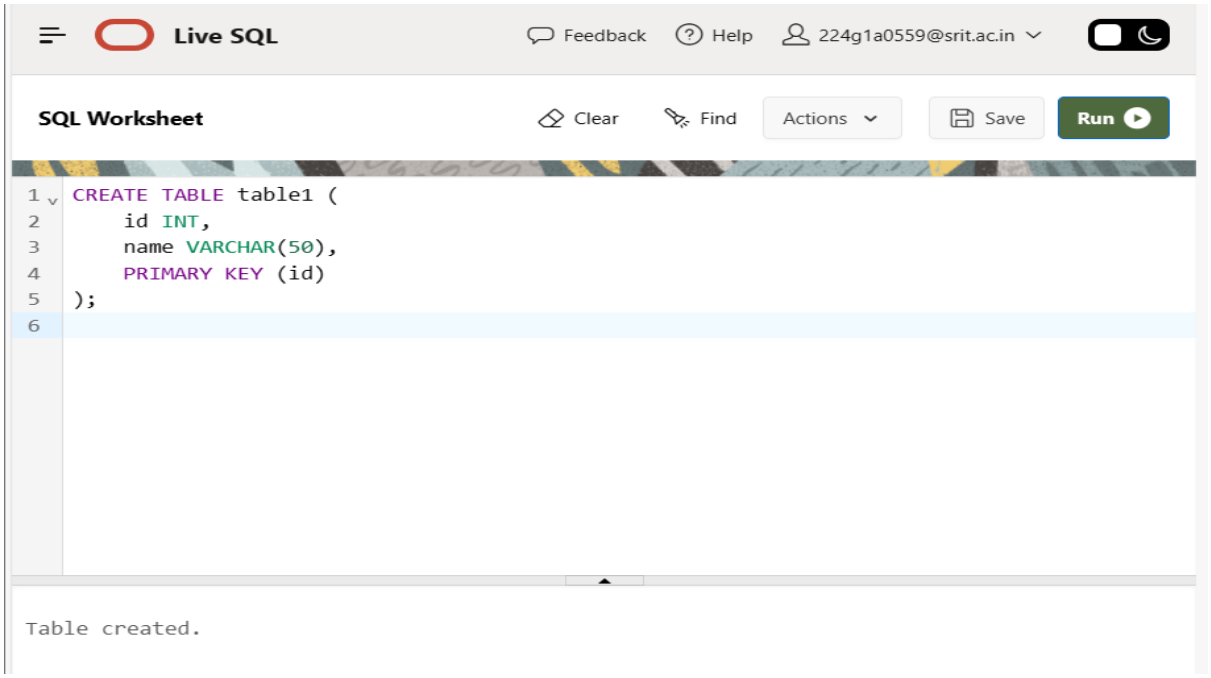


Conclusion:

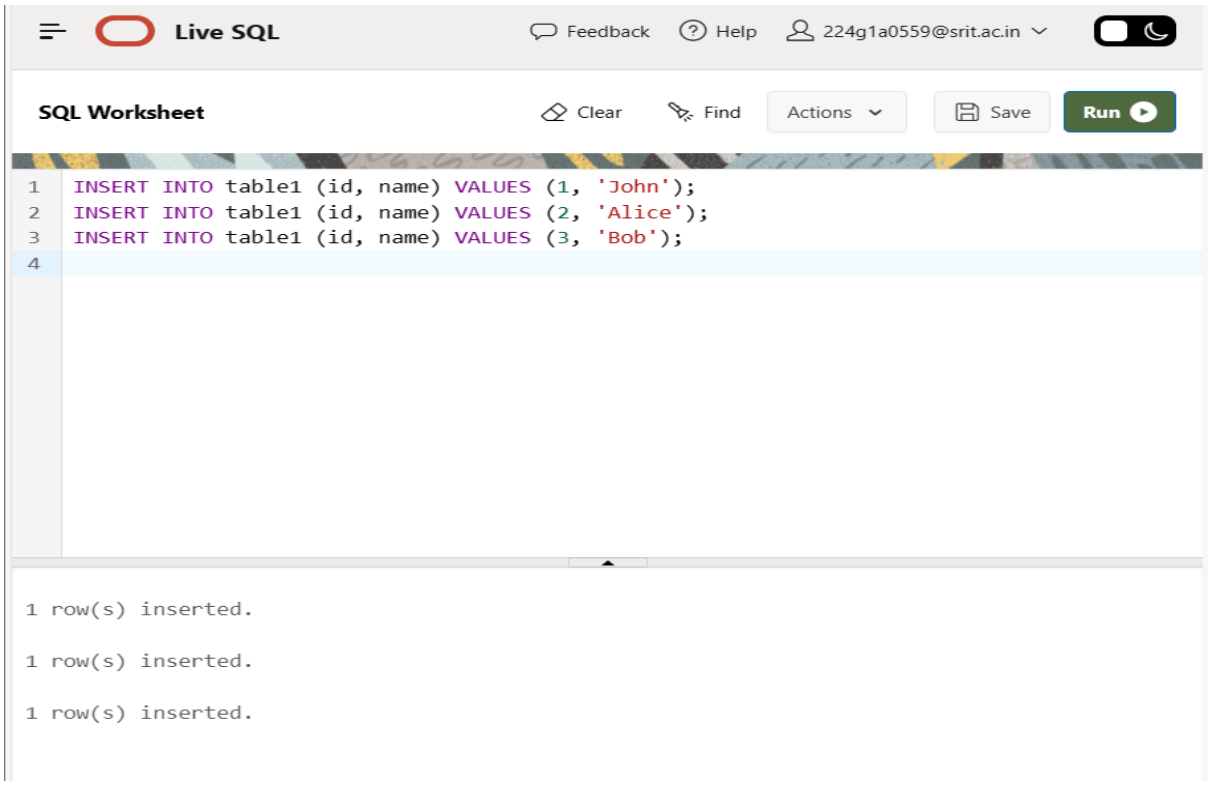
In this lab, we successfully practiced SQL Queries to create VIEWS for various data base (CREATE VIEW, UPDATE VIEW, ALTER VIEW, DELETE VIEW).

EXPERIMENT – 4

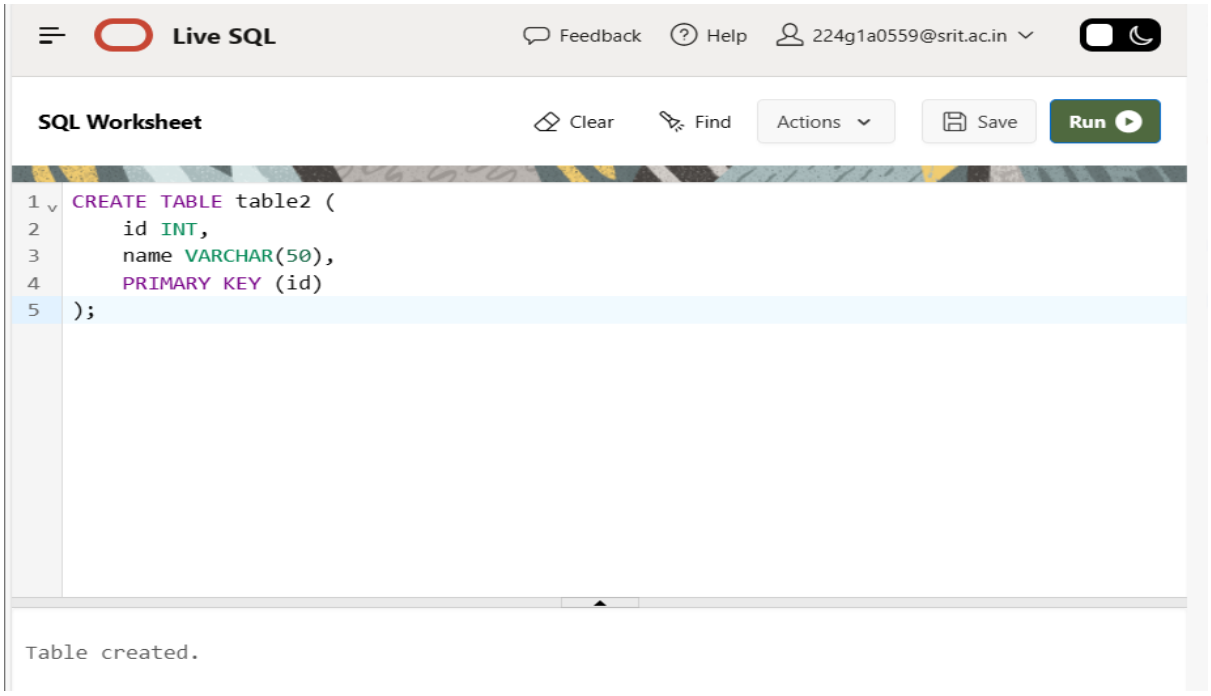
1.creating a table1:



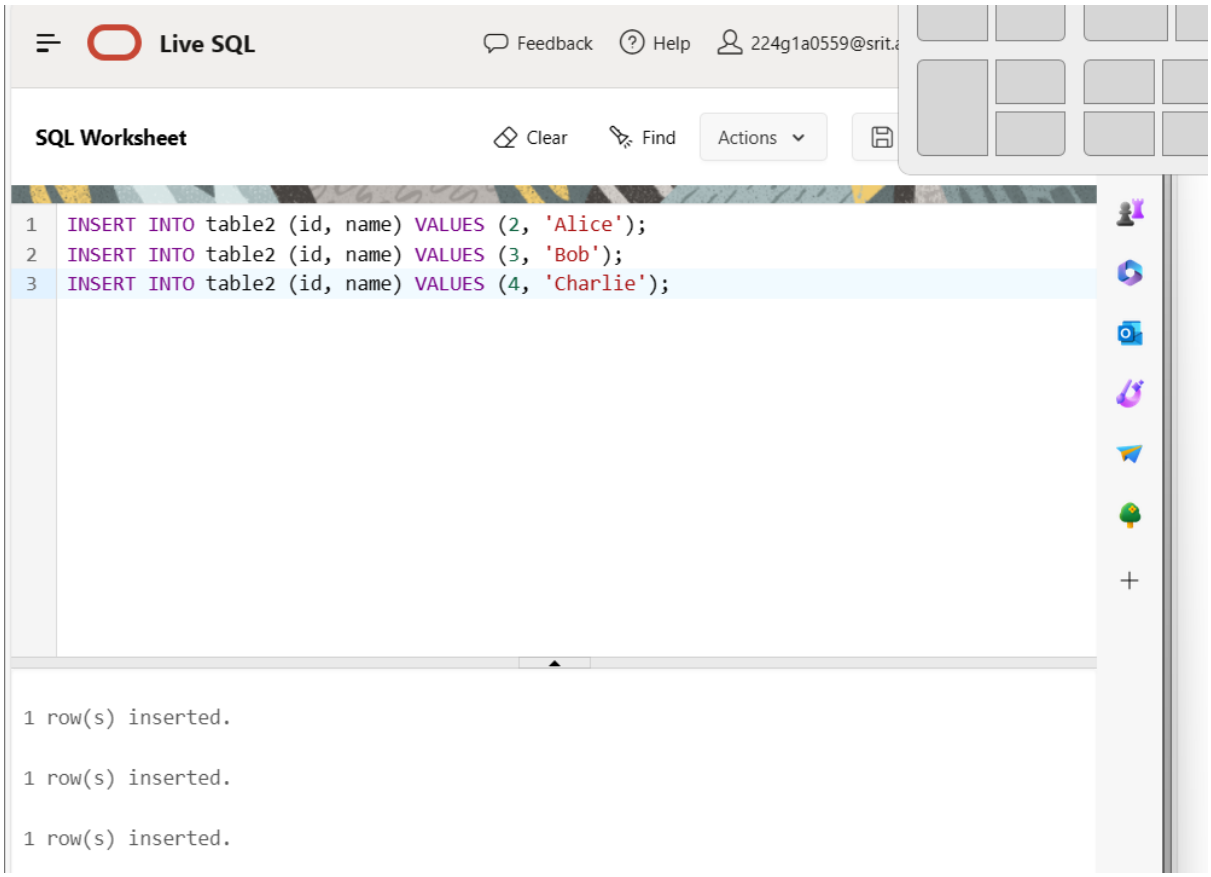
2.inserting VALUES into table1:



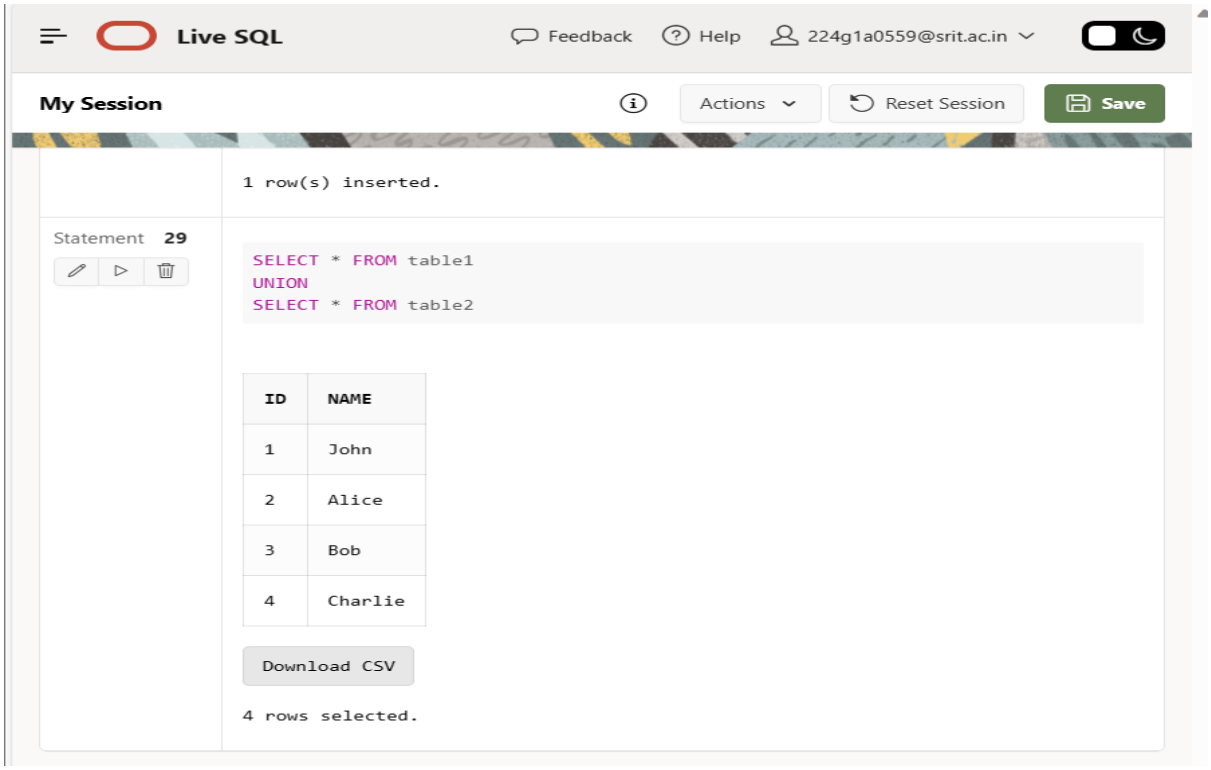
3.creating a table2:



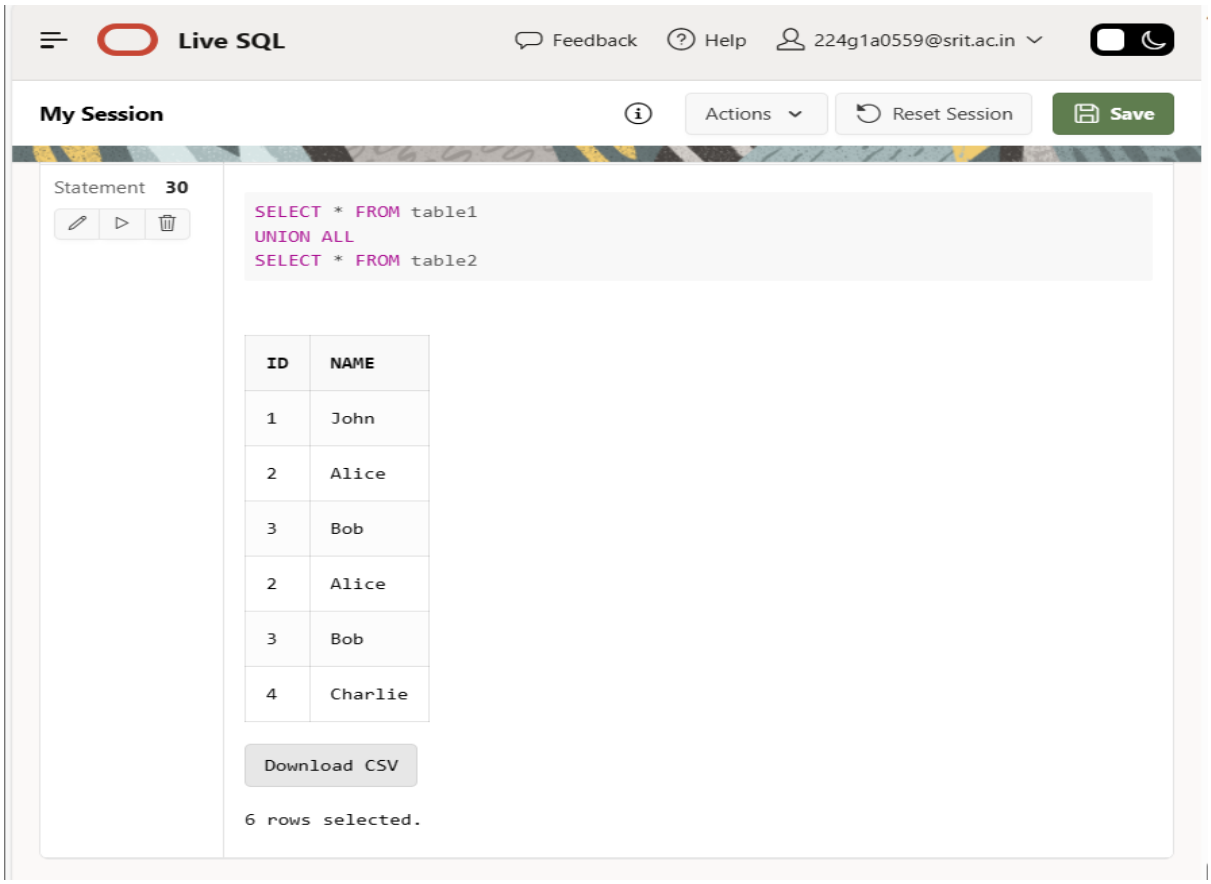
4.Inserting VALUES into table2:



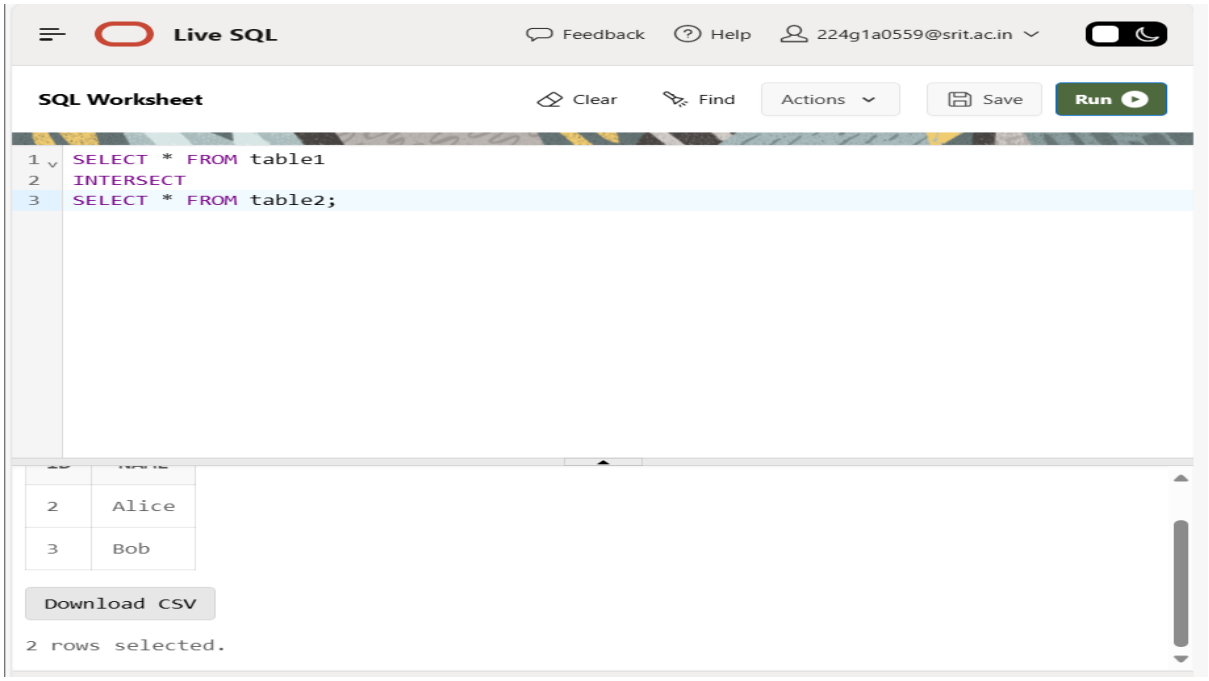
5.UNION:



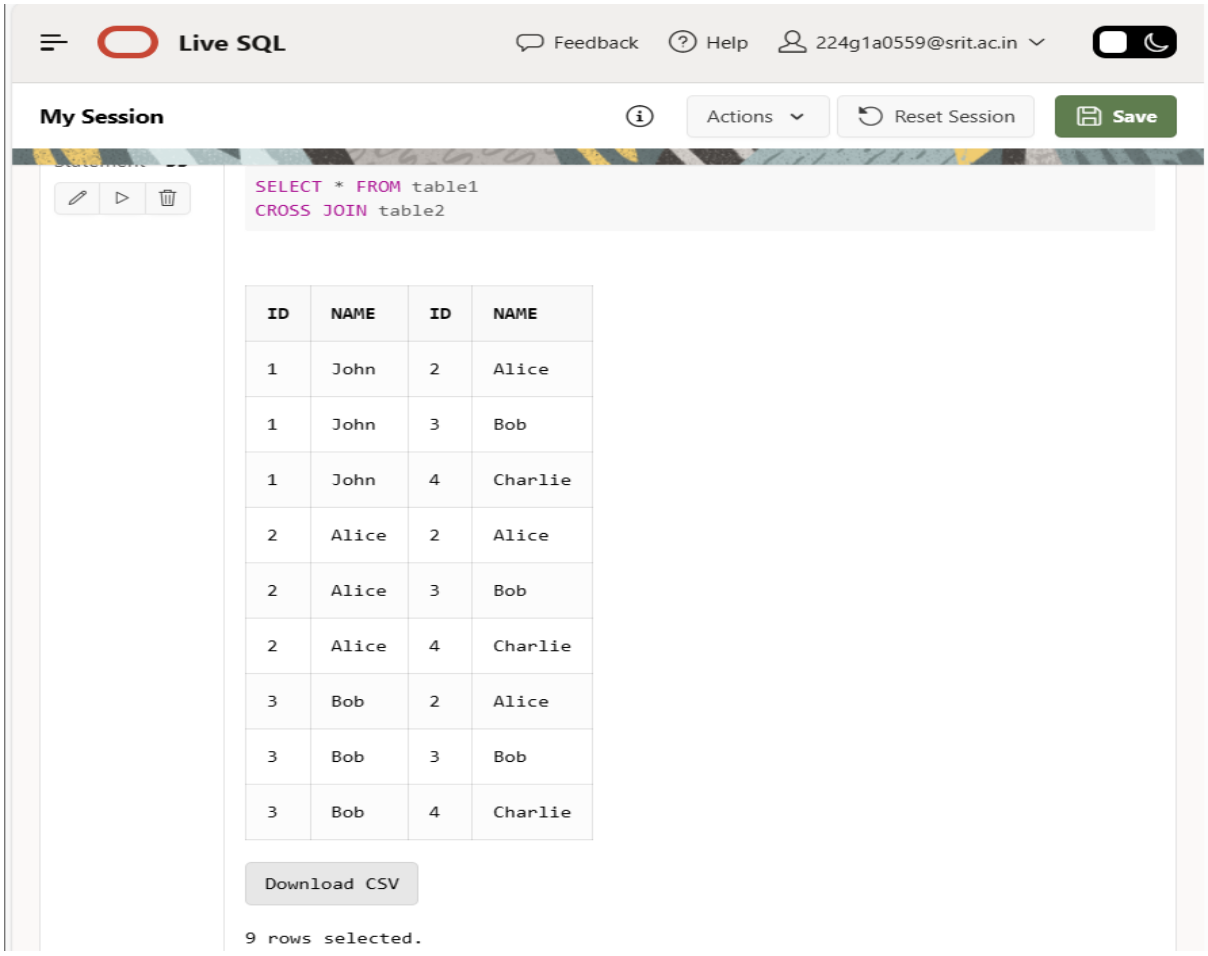
6.UNION ALL:



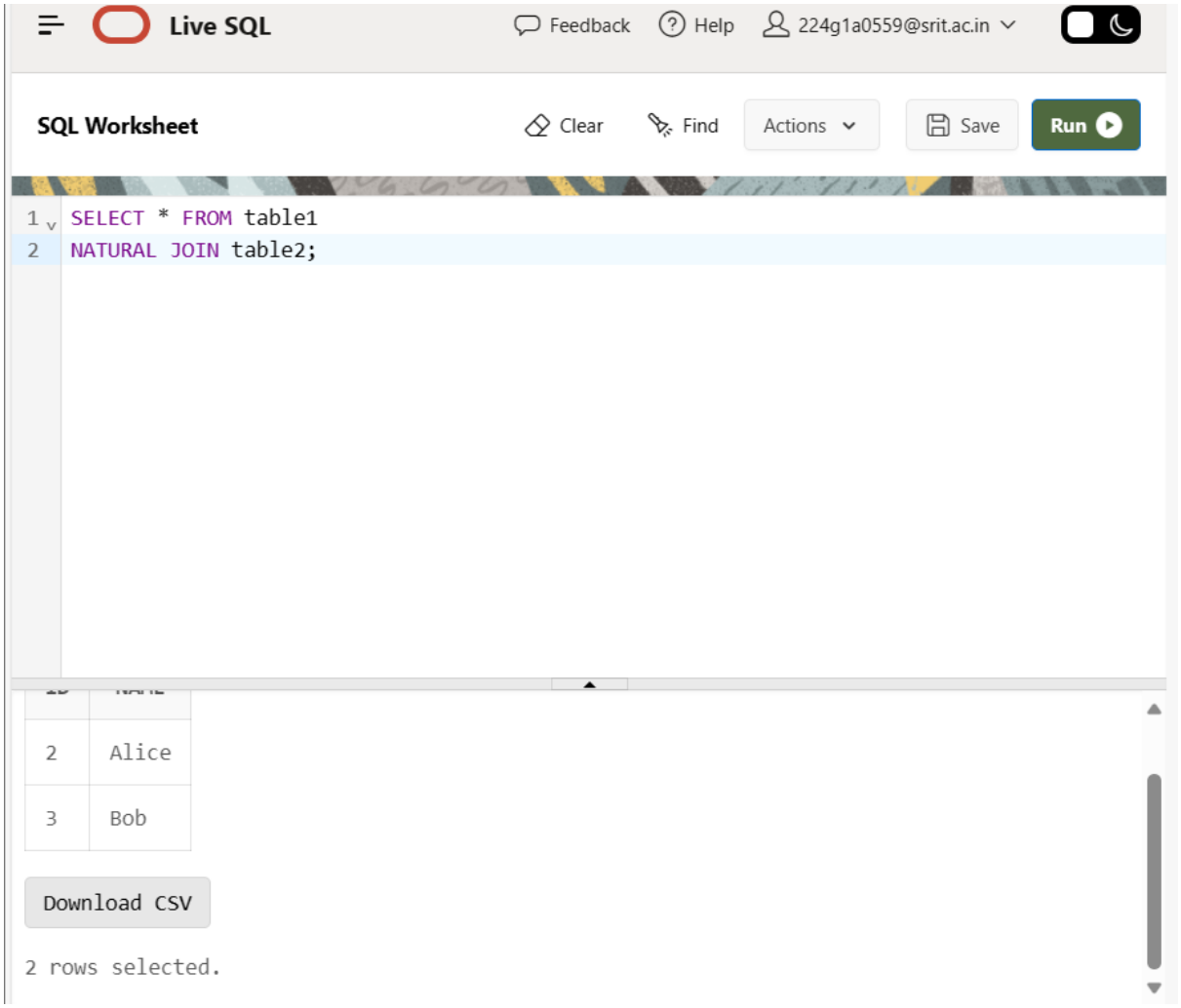
7.INTERSECT:



8.CROSS JOIN:



9.NATURAL JOIN:

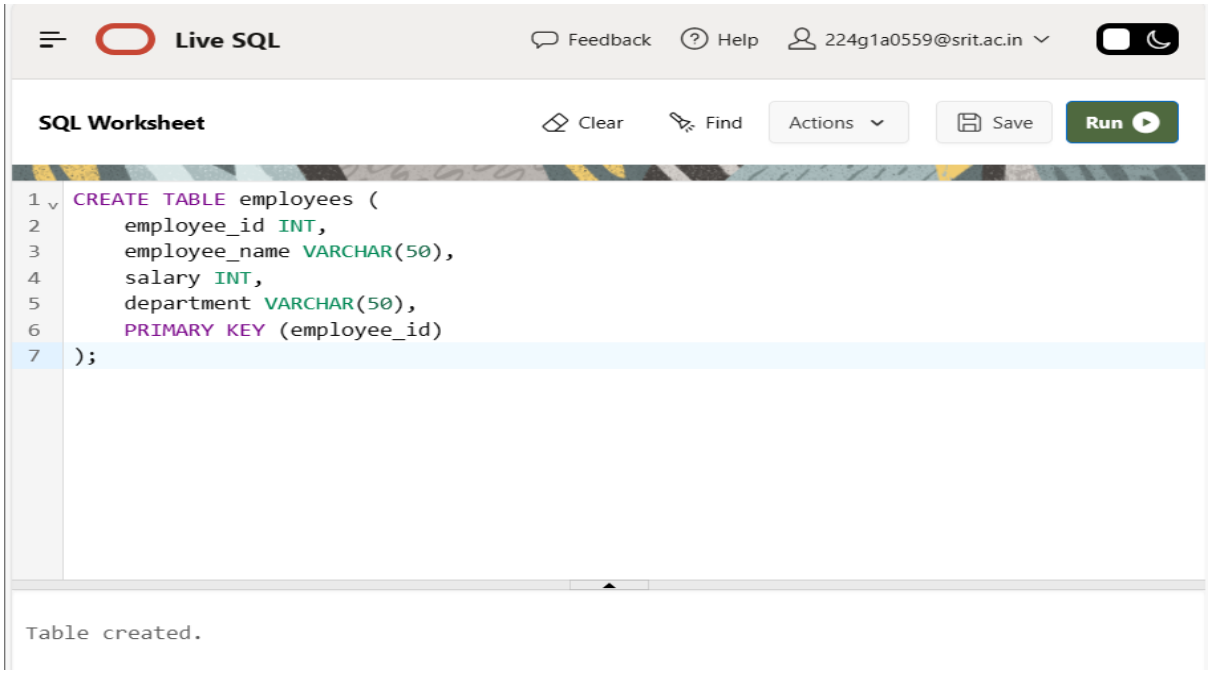


Conclusion:

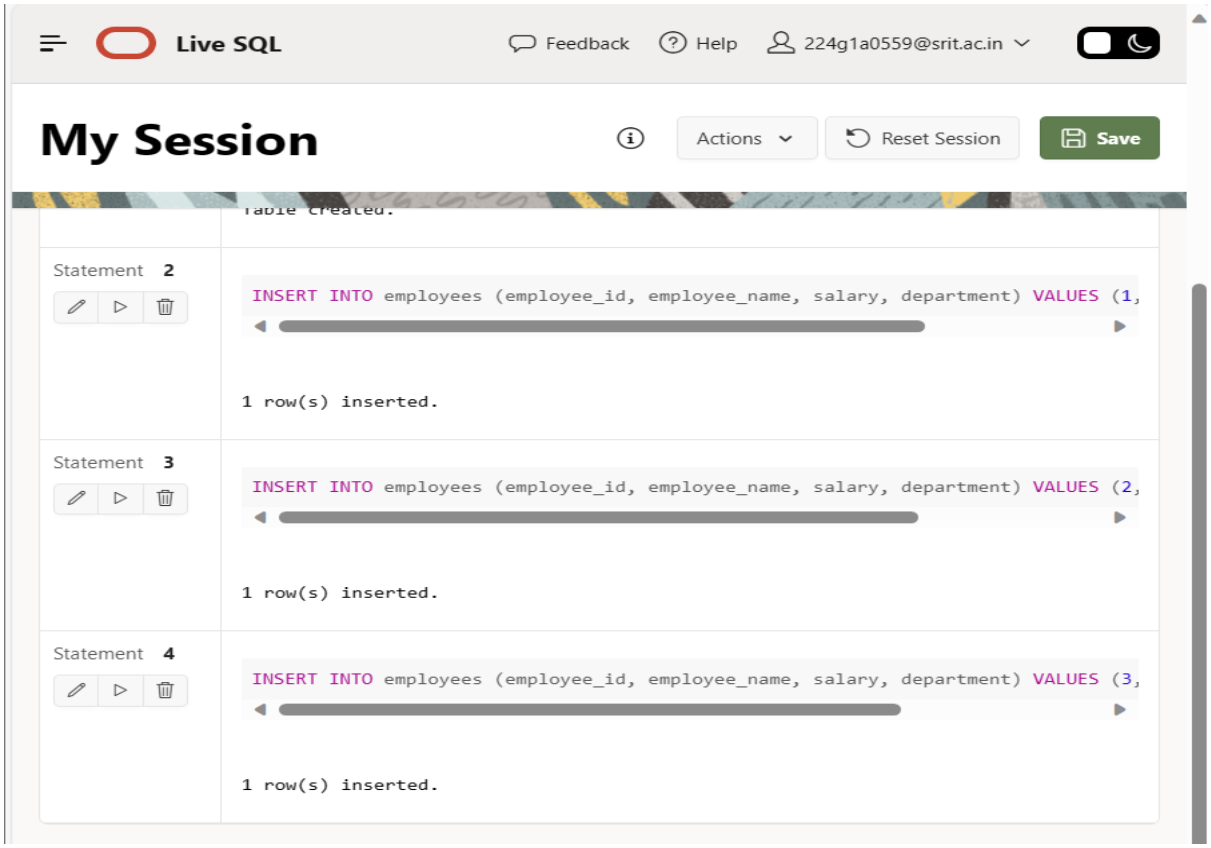
In this lab, we successfully practiced how to perform SQL Queries on RELATIONAL SET OPERATIONS (UNION, UNION ALL, CROSS JOIN, NATURAL JOIN).

EXPERIMENT – 5

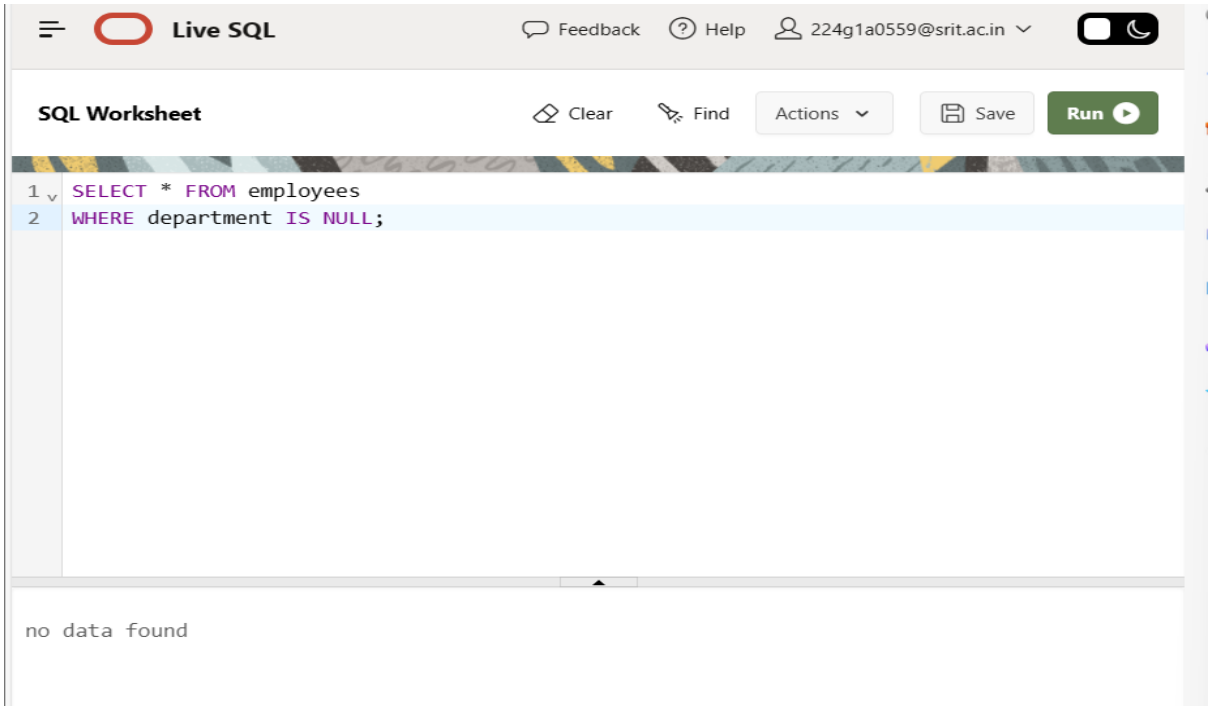
1.creating a table:



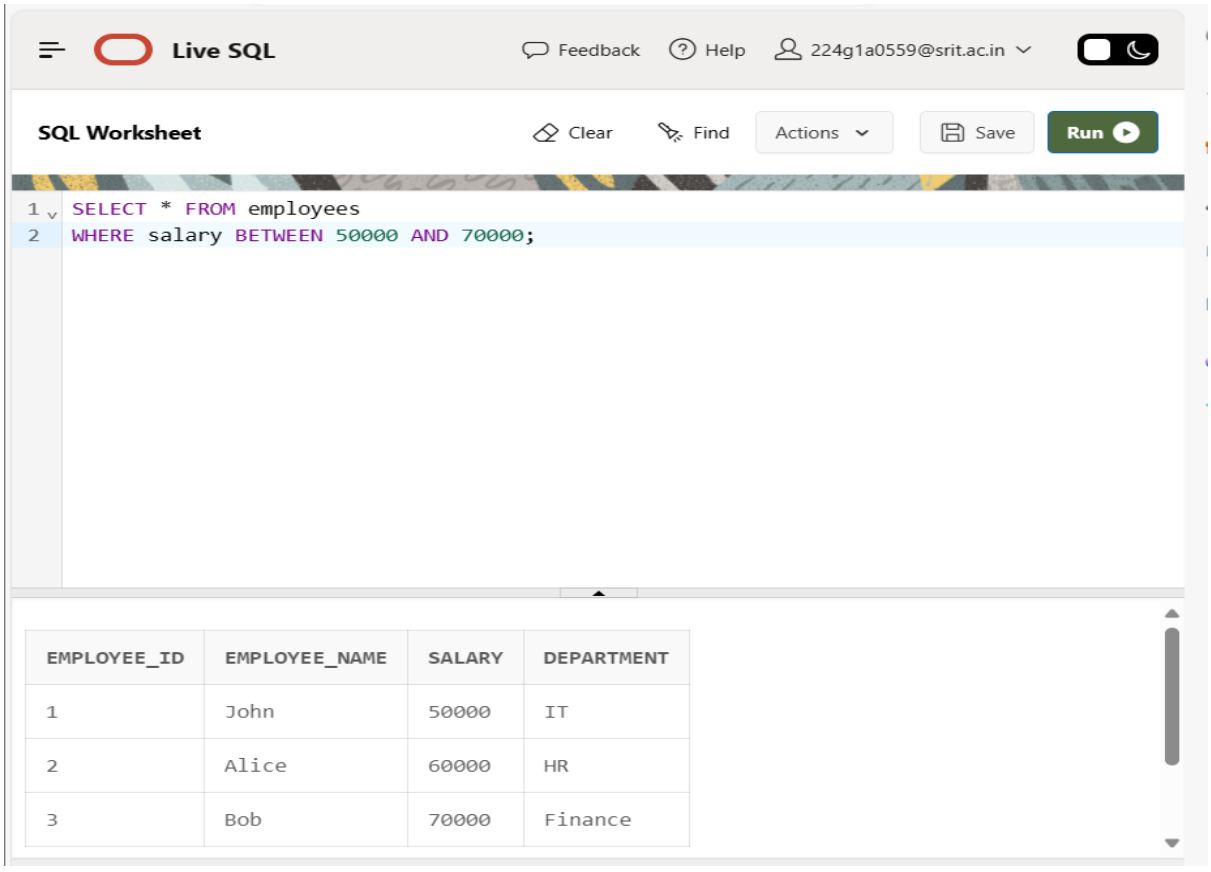
2.Inserting VALUES into table:



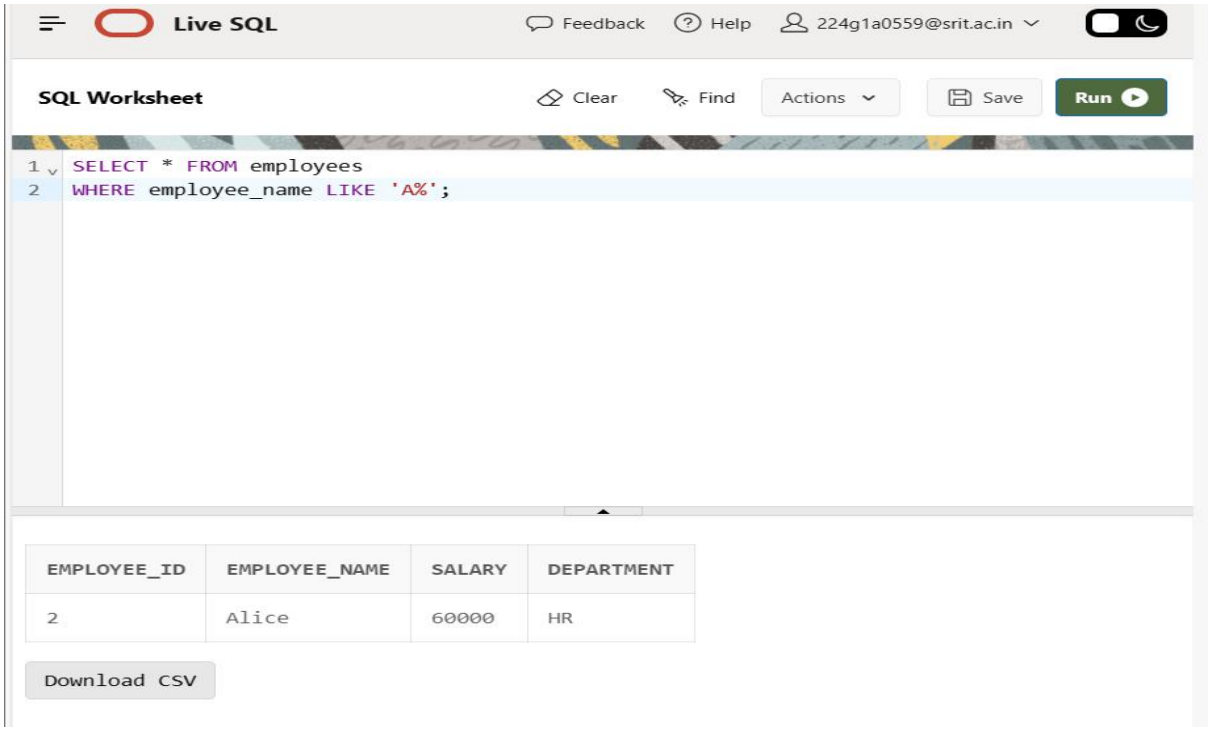
3.IS NULL:



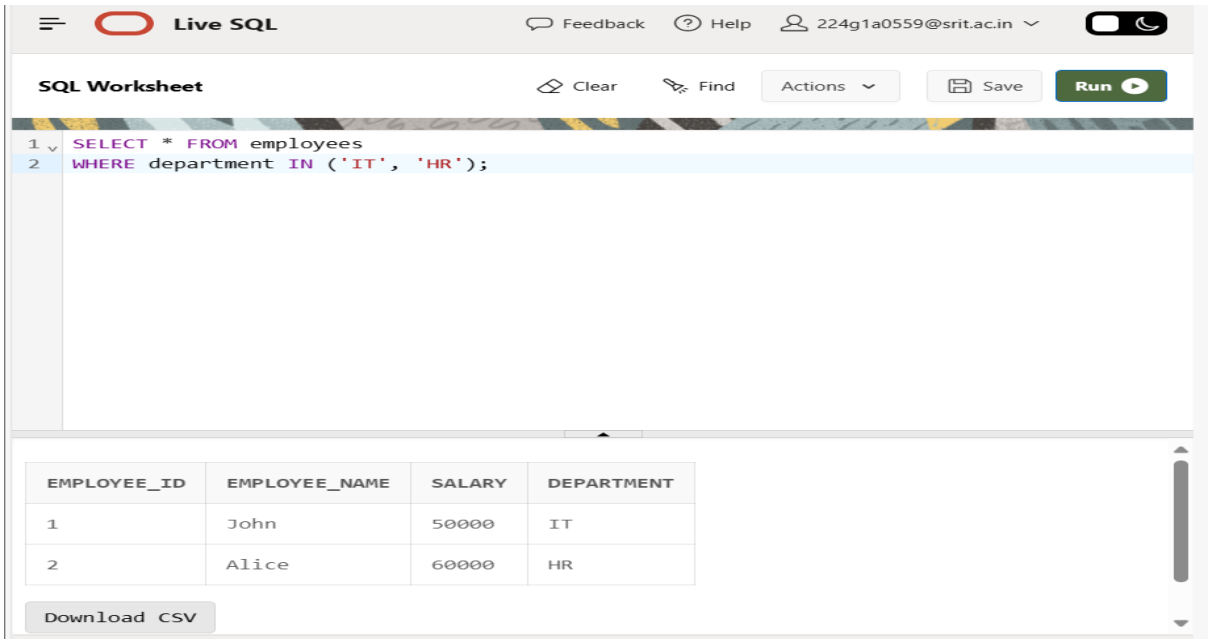
4.BETWEEN:



5.LIKE:



6.IN:

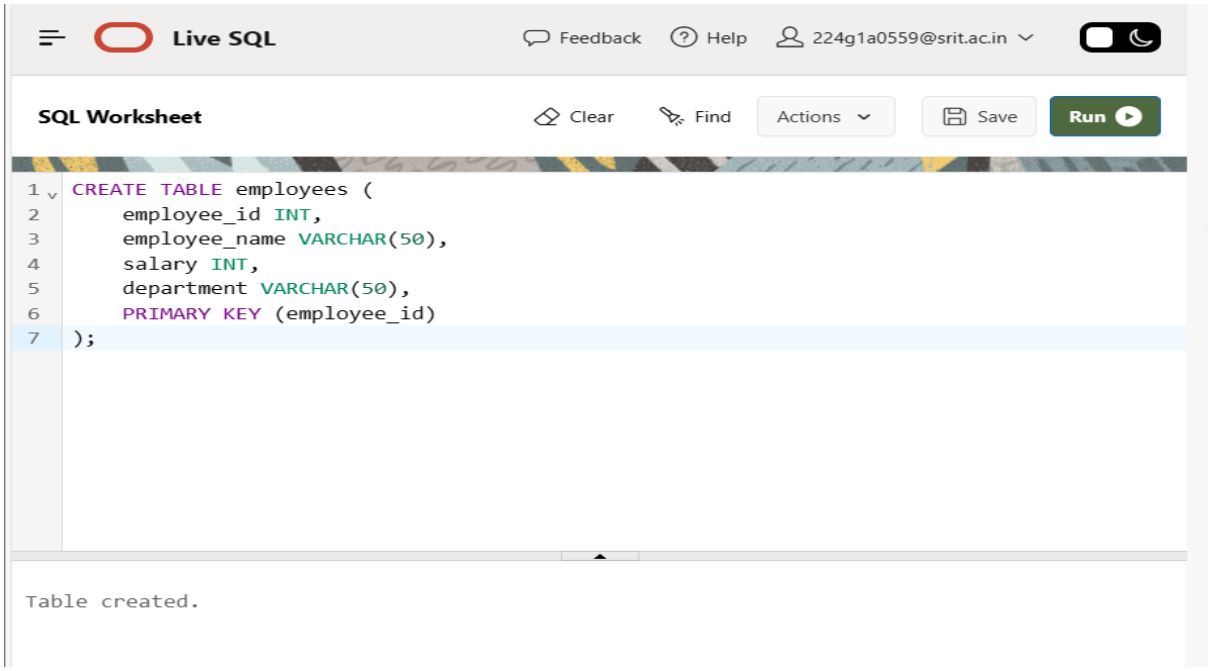


Conclusion:

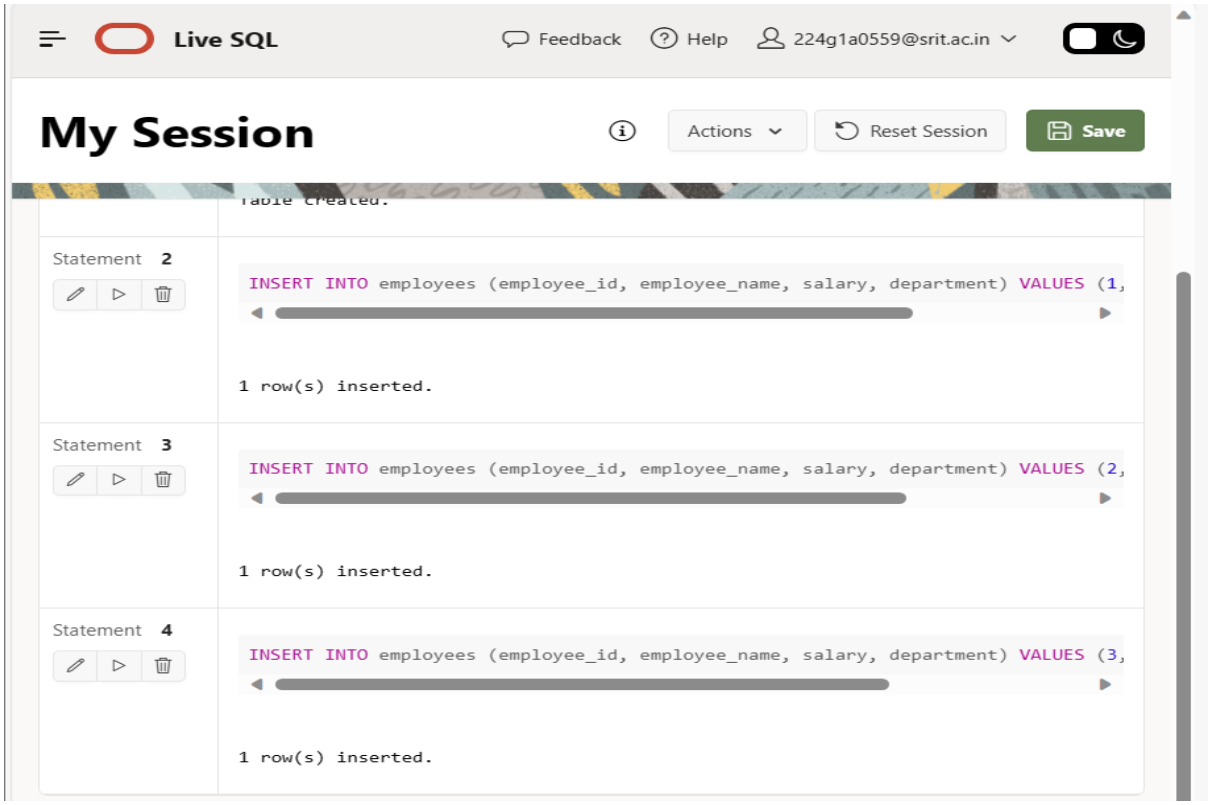
In this lab, we successfully practiced SQL Queries on SPECIAL OPERATIONS (IS NULL, BETWEEN, LIKE, IN).

EXPERIMENT – 6

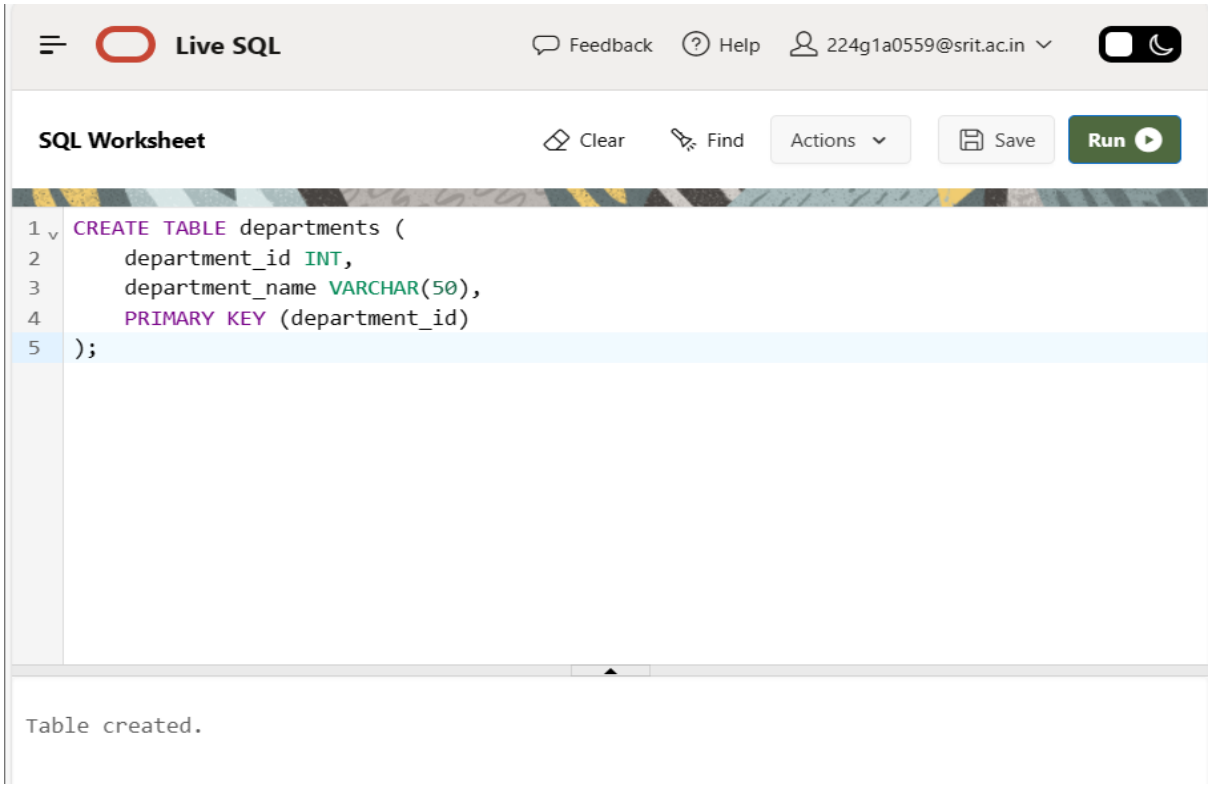
1.creating a employee table:



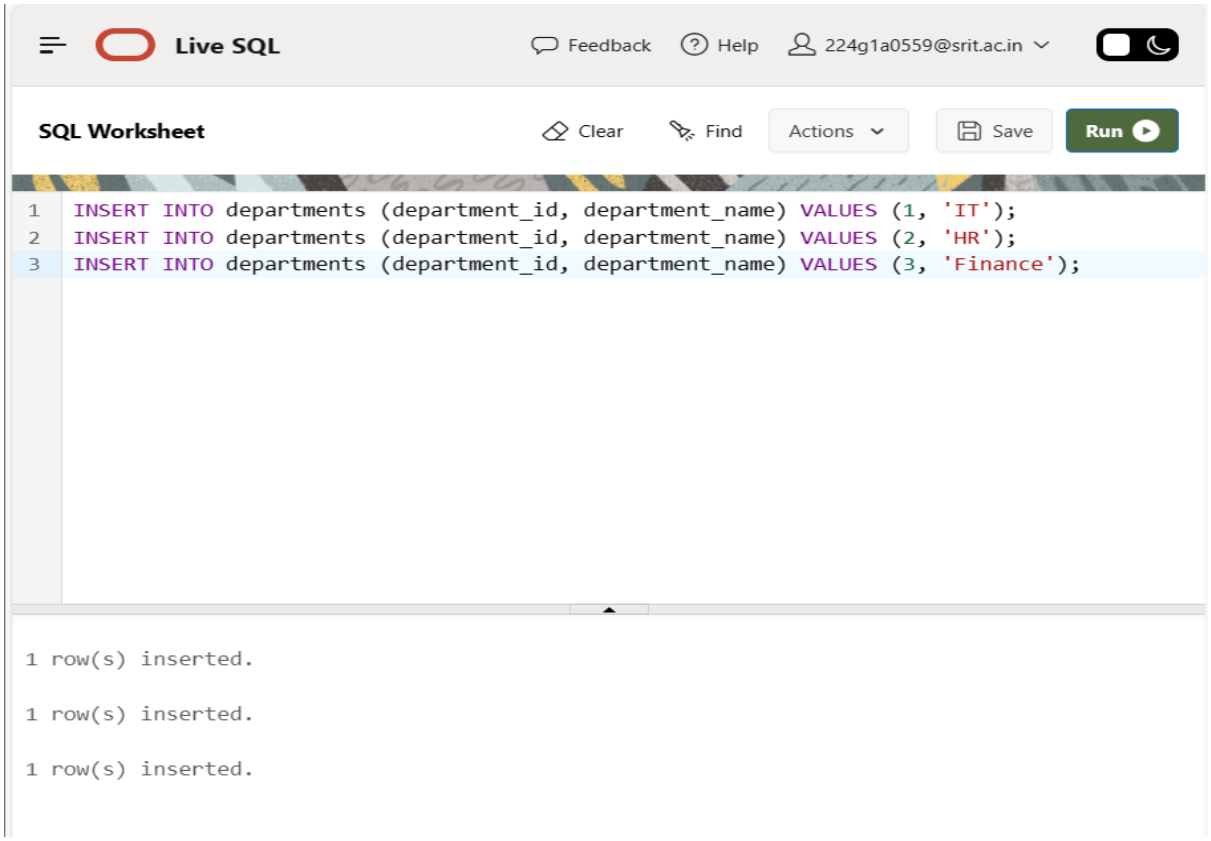
2.Inserting VALUES into employee:



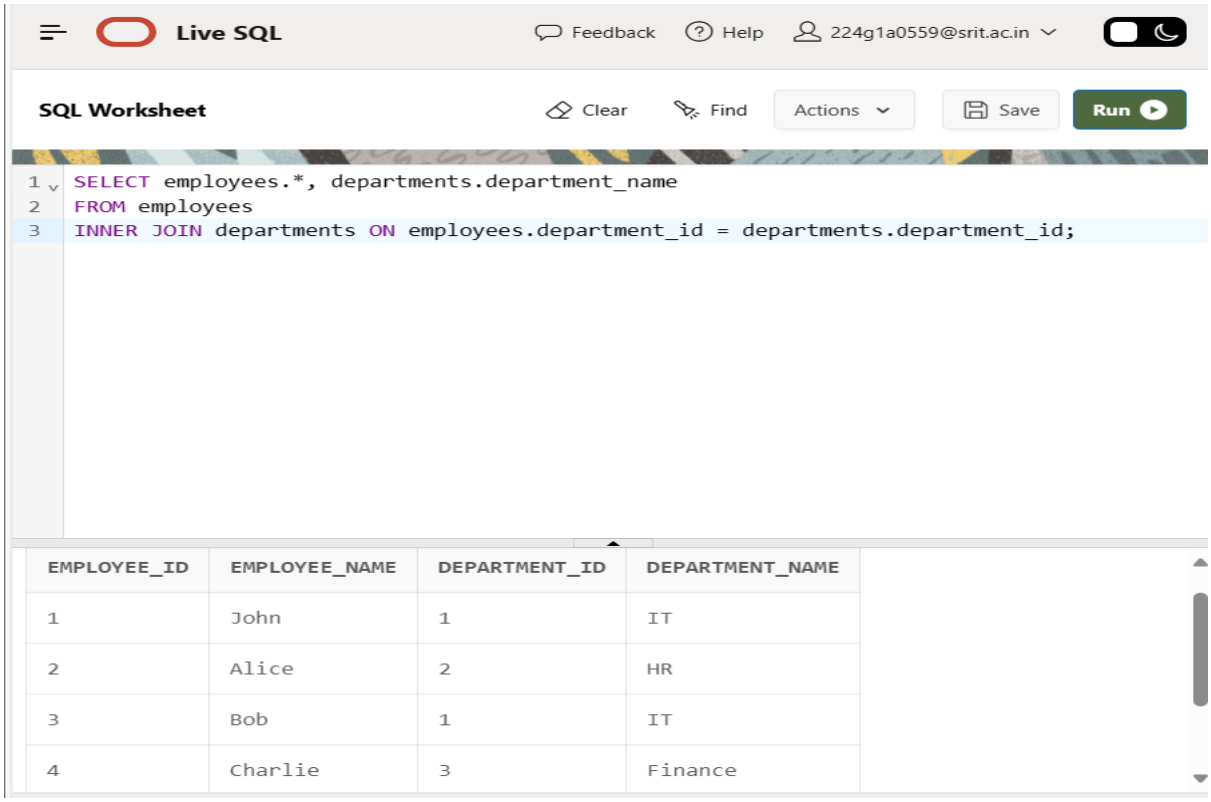
3.Creating a department table:



4.Inserting VALUES into table:

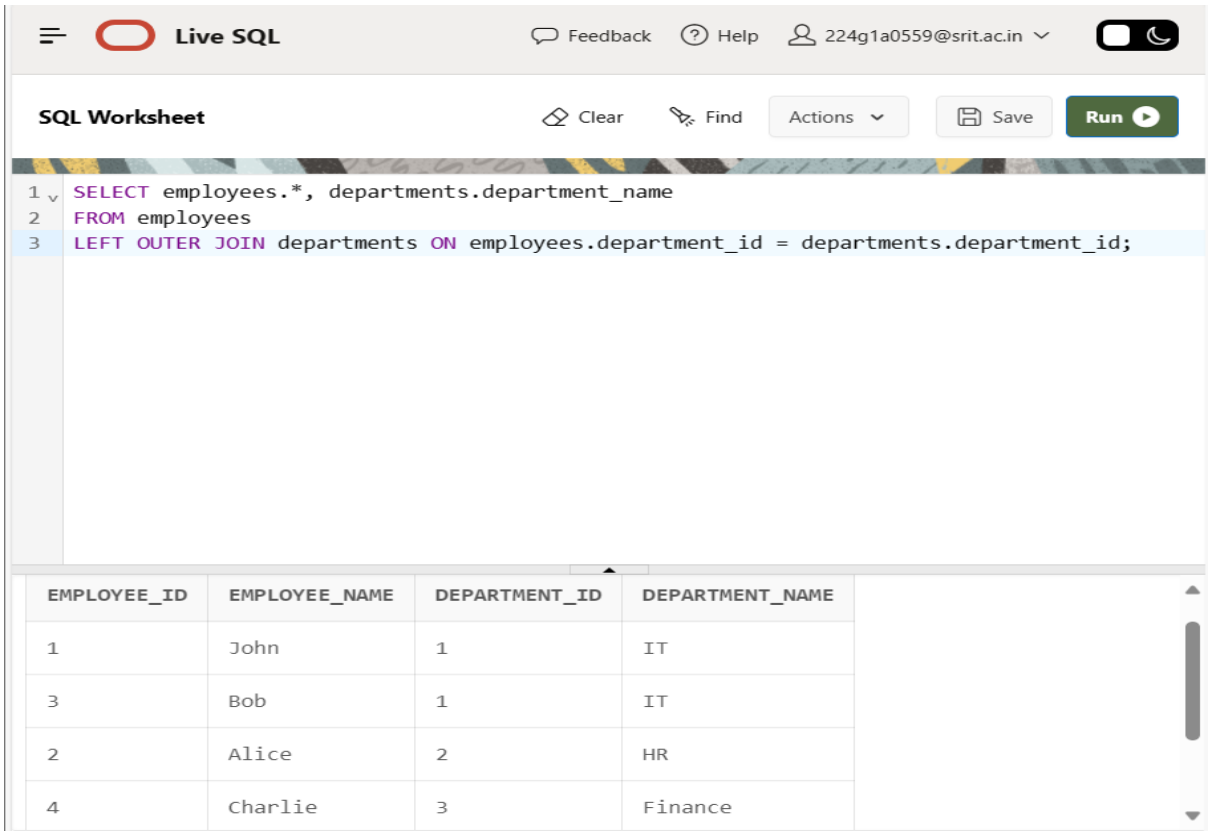


5.INNER JOIN (OR EQUI JOIN):



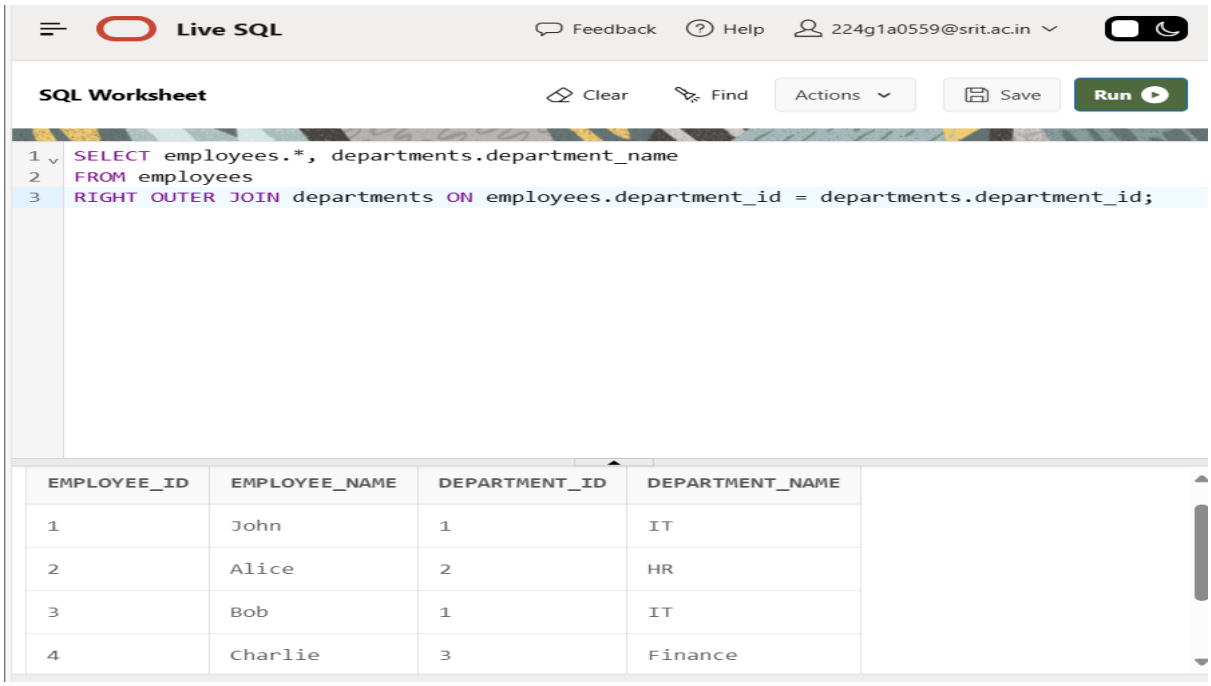
EMPLOYEE_ID	EMPLOYEE_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	John	1	IT
2	Alice	2	HR
3	Bob	1	IT
4	Charlie	3	Finance

6.LEFT OUTER JOIN:

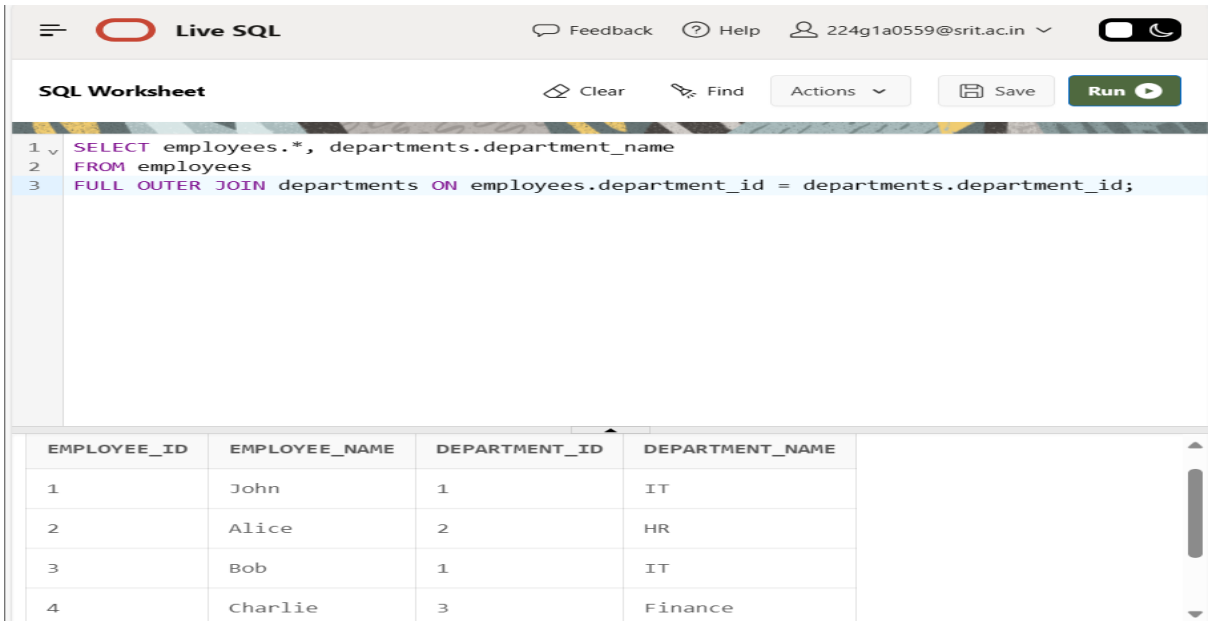


EMPLOYEE_ID	EMPLOYEE_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	John	1	IT
3	Bob	1	IT
2	Alice	2	HR
4	Charlie	3	Finance

7.RIGHT OUTER JOIN:



8.FULL OUTER JOIN:

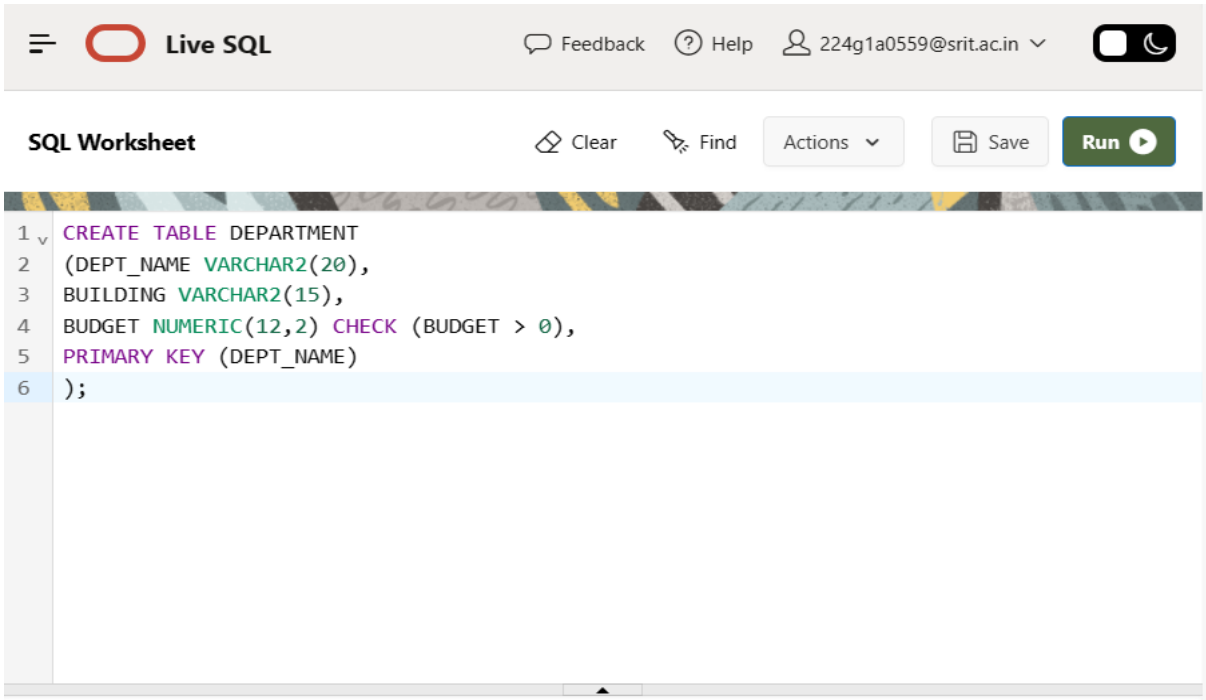


Conclusion:

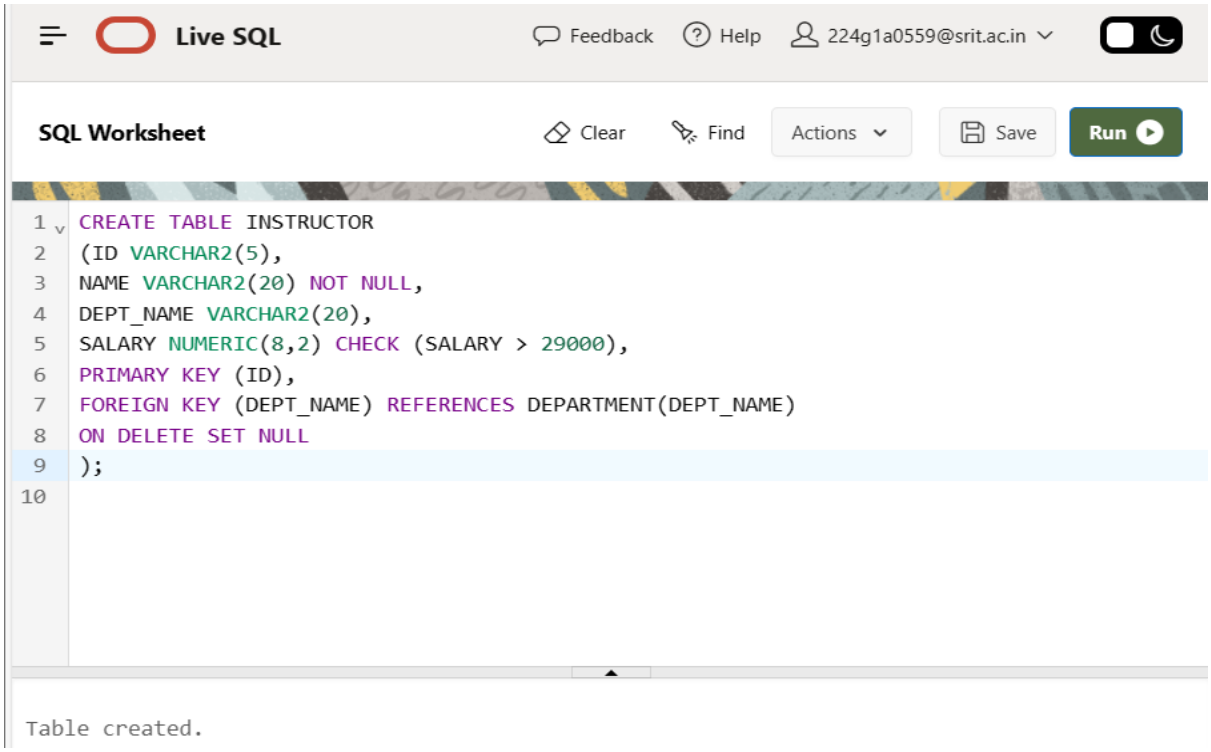
In this lab, we successfully practiced SQL Queries on JOIN OPERATIONS (CONDITIONAL JOIN, EQUI JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN).

EXPERIMENT – 7

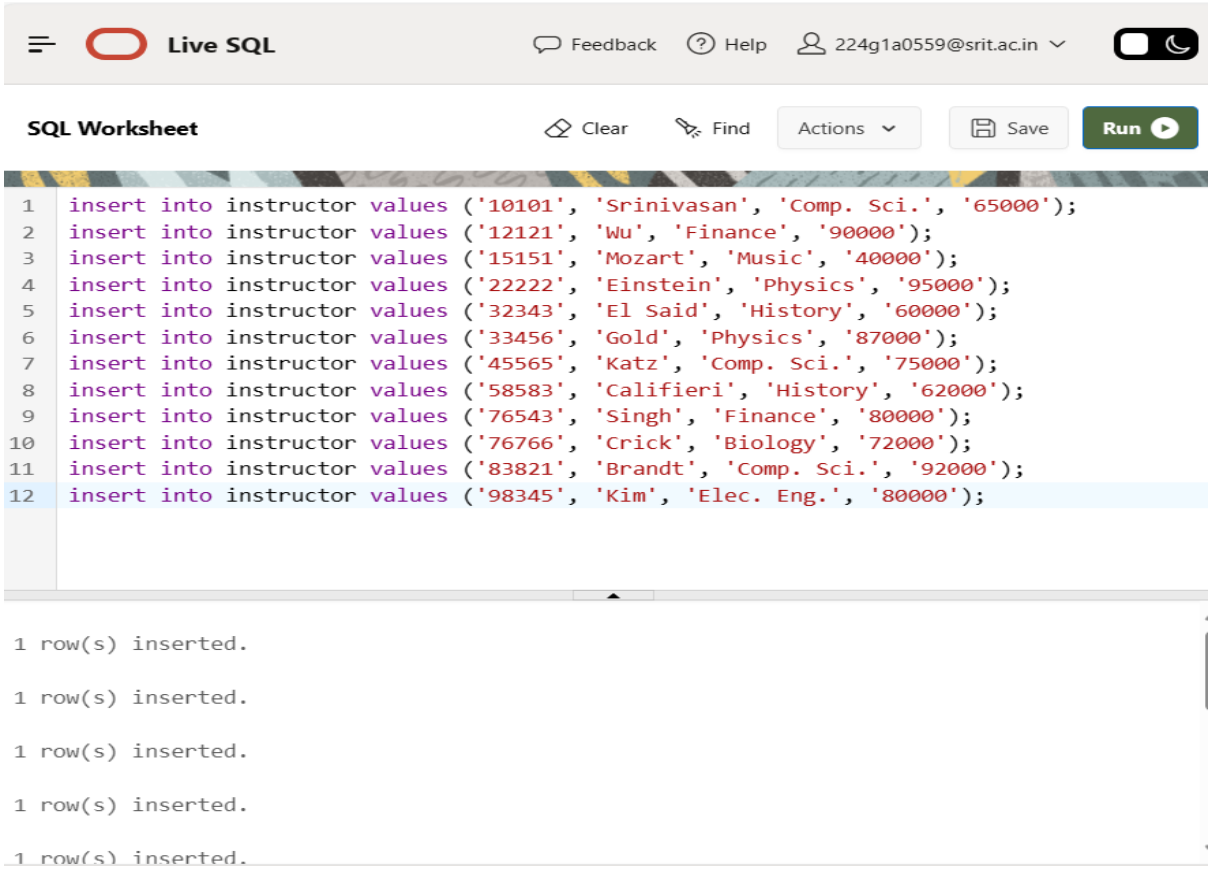
1.Create DEPARTMENT Table:



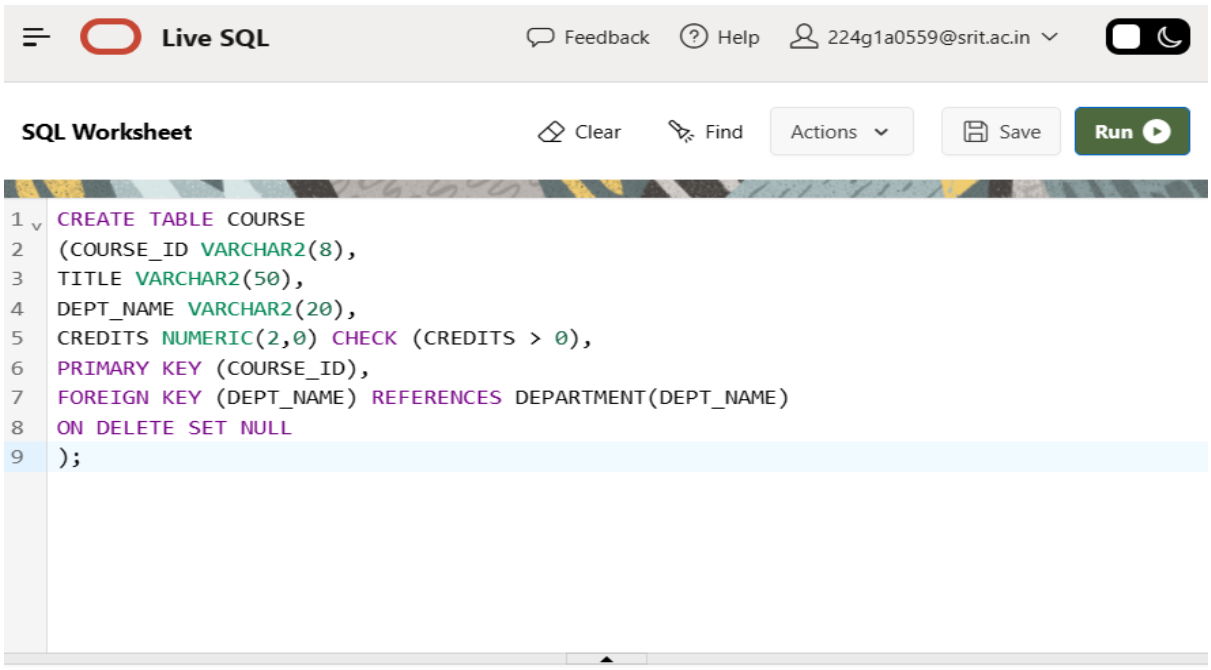
2.Create INSTRUCTOR Table:



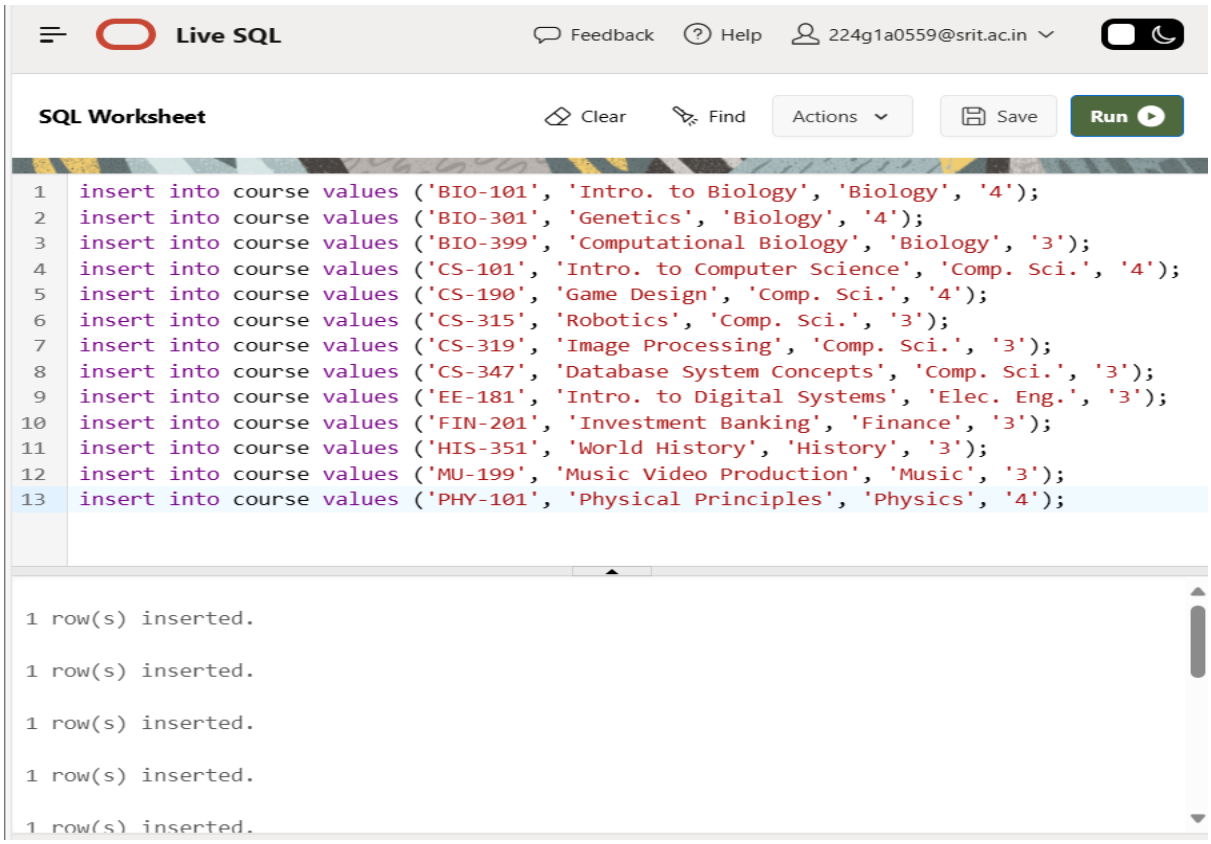
3.Instances of instructor:



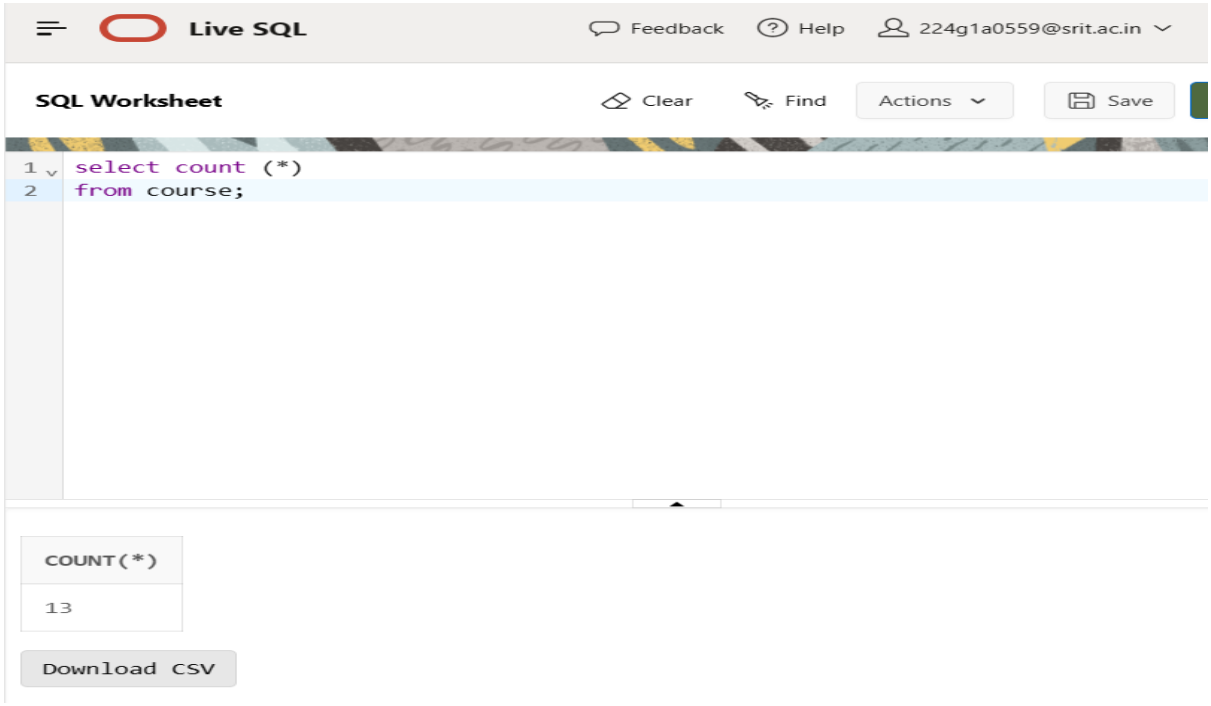
4.Create COURSE Table:

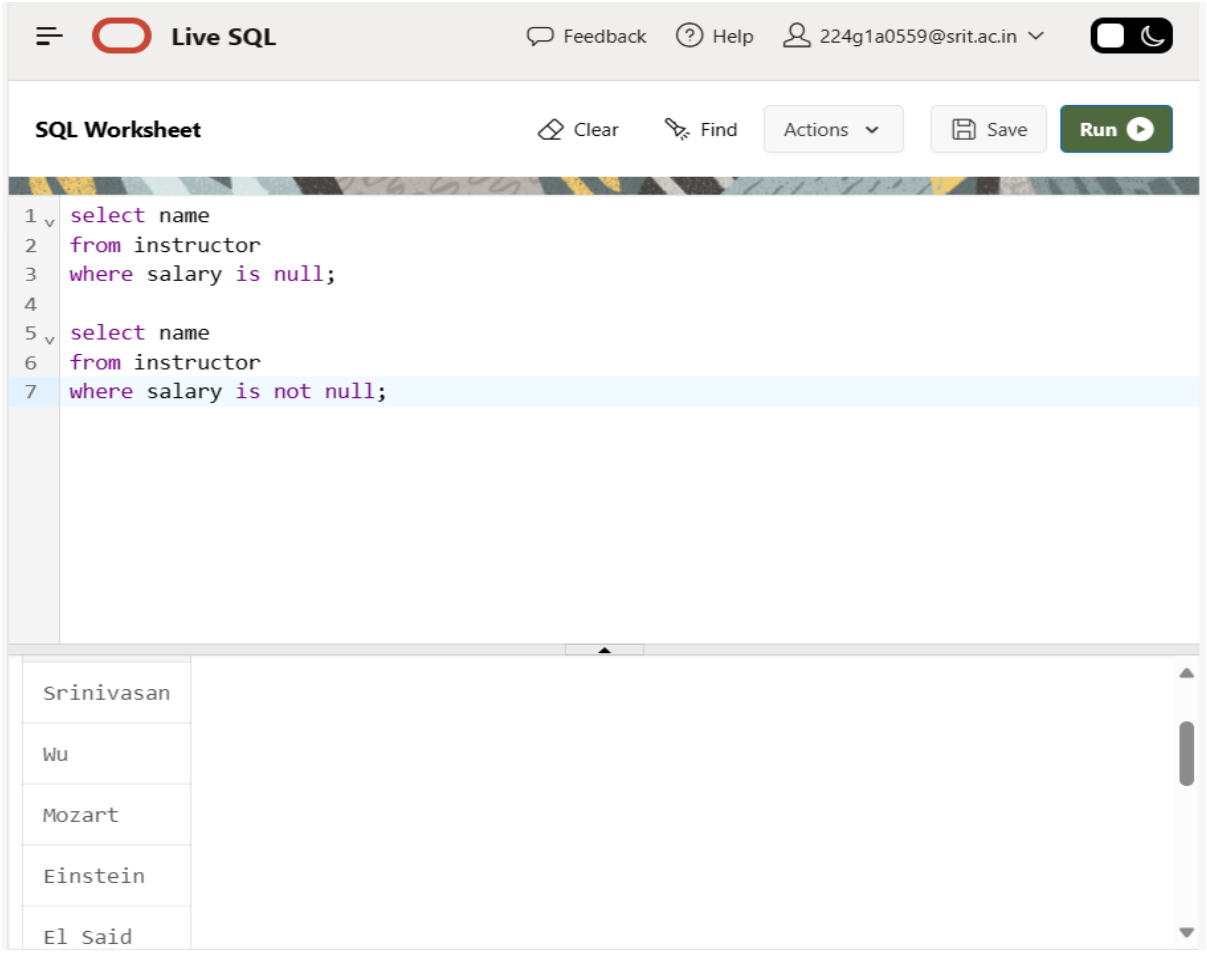


5.Instances of COURSE Table:



To find no. of courses taught in the university:



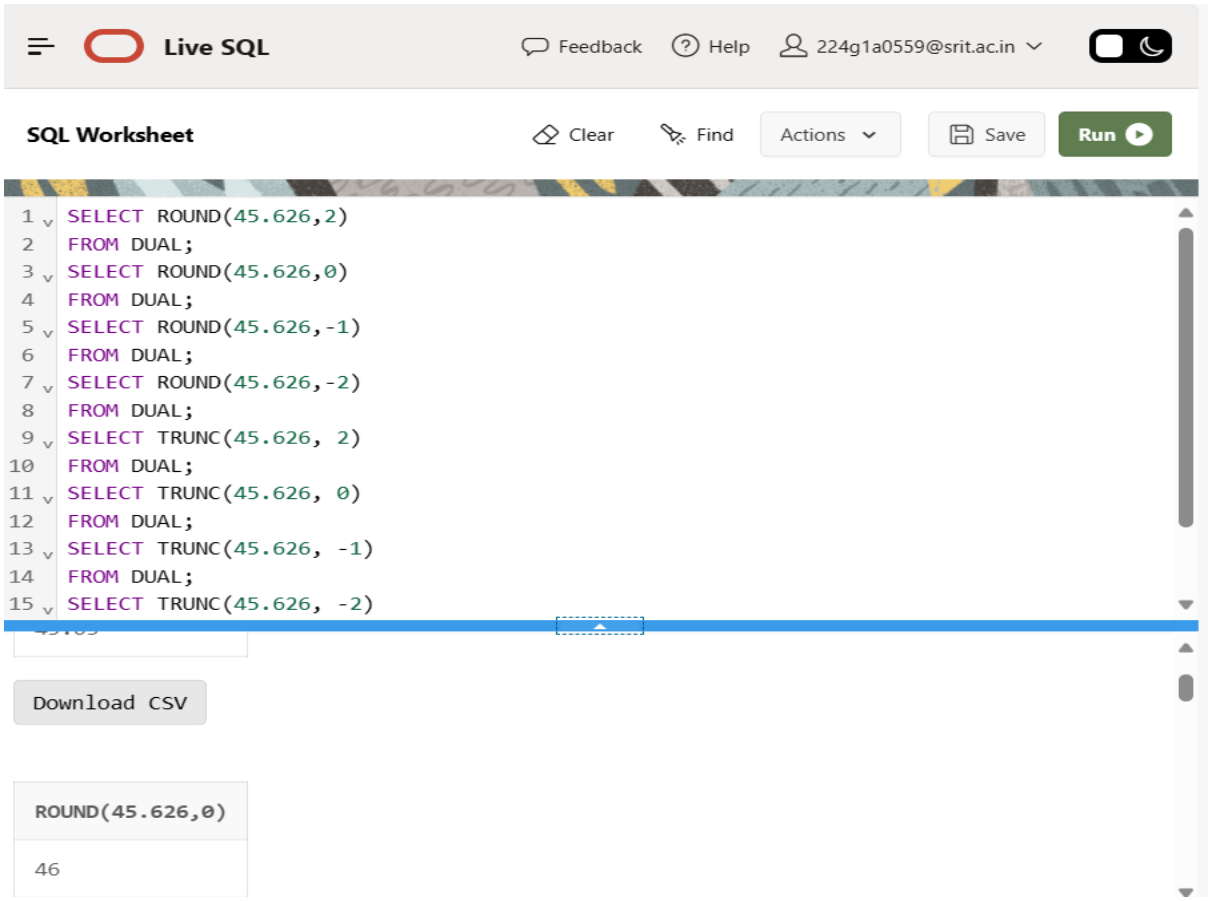


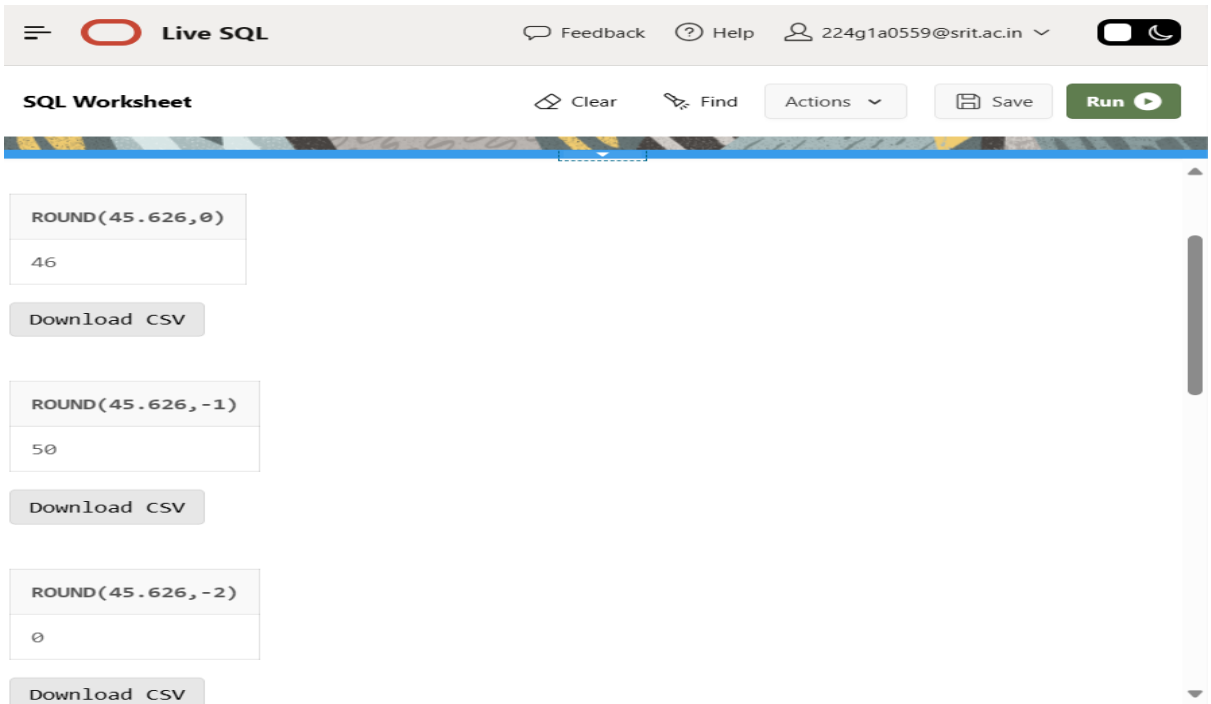
Conclusion:

In this lab, we successfully practiced SQL Queries to perform AGGREGATE Operations.

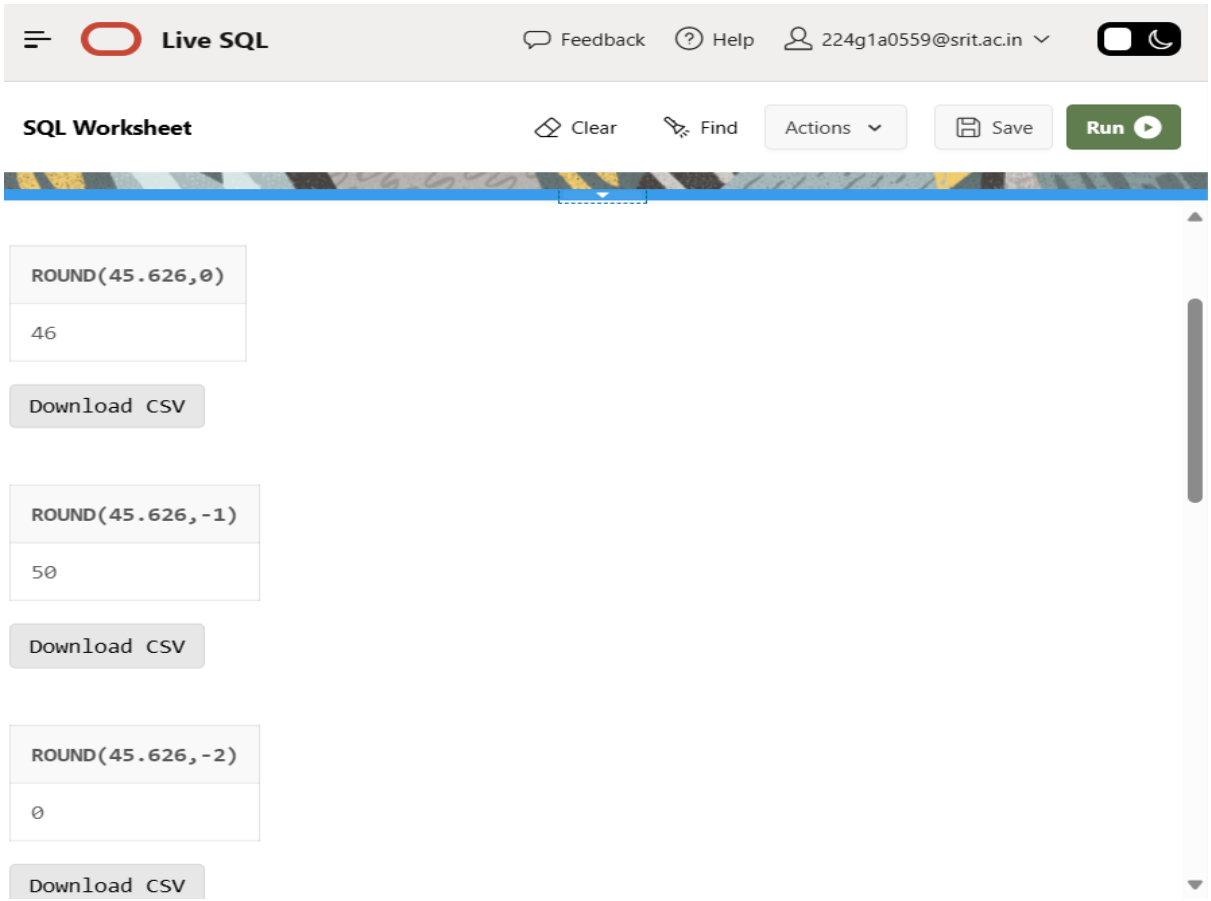
EXPERIMENT – 8

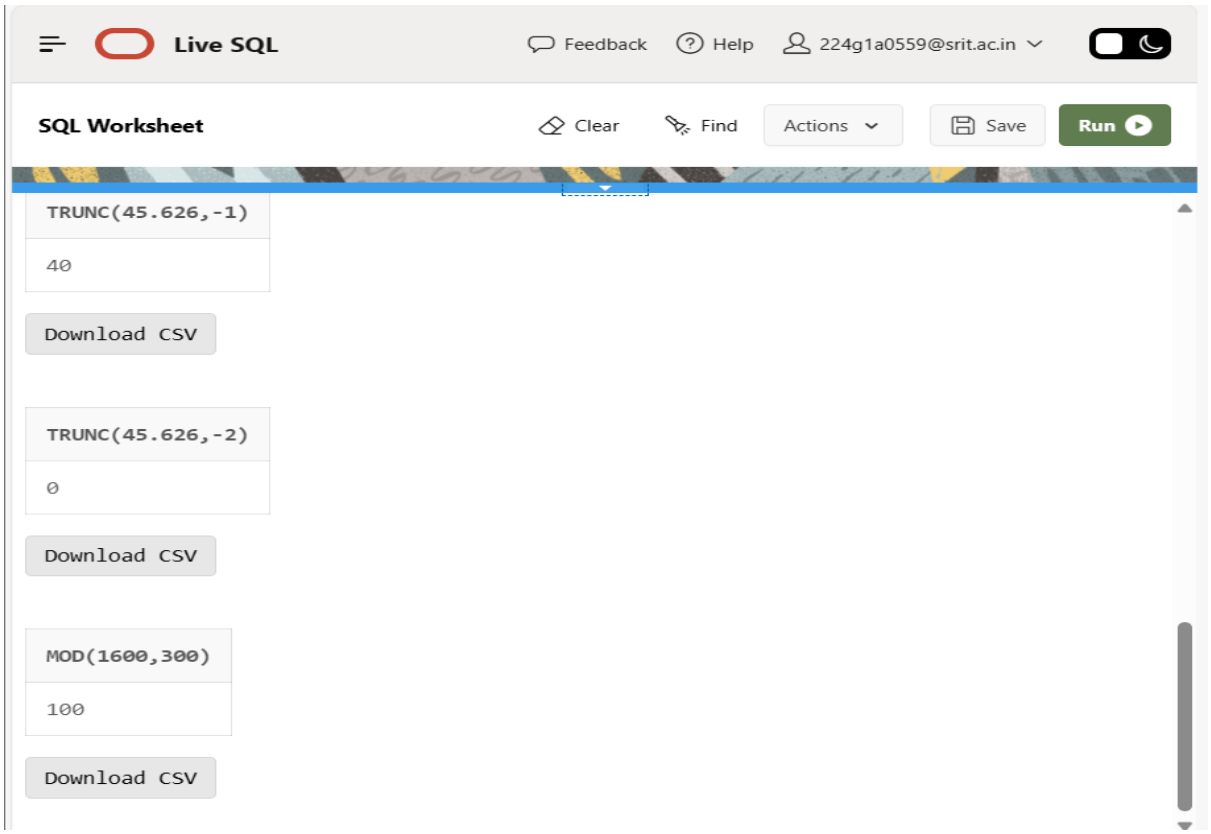
1.NUMBER Functions:



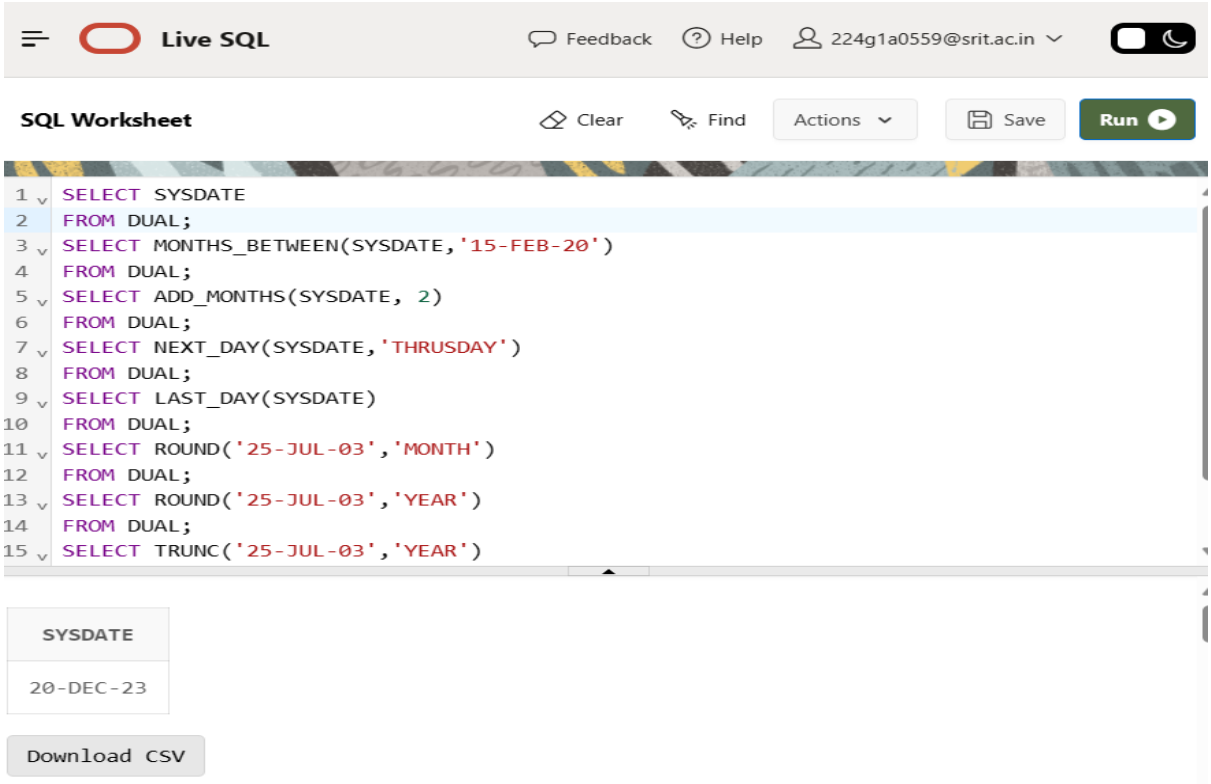


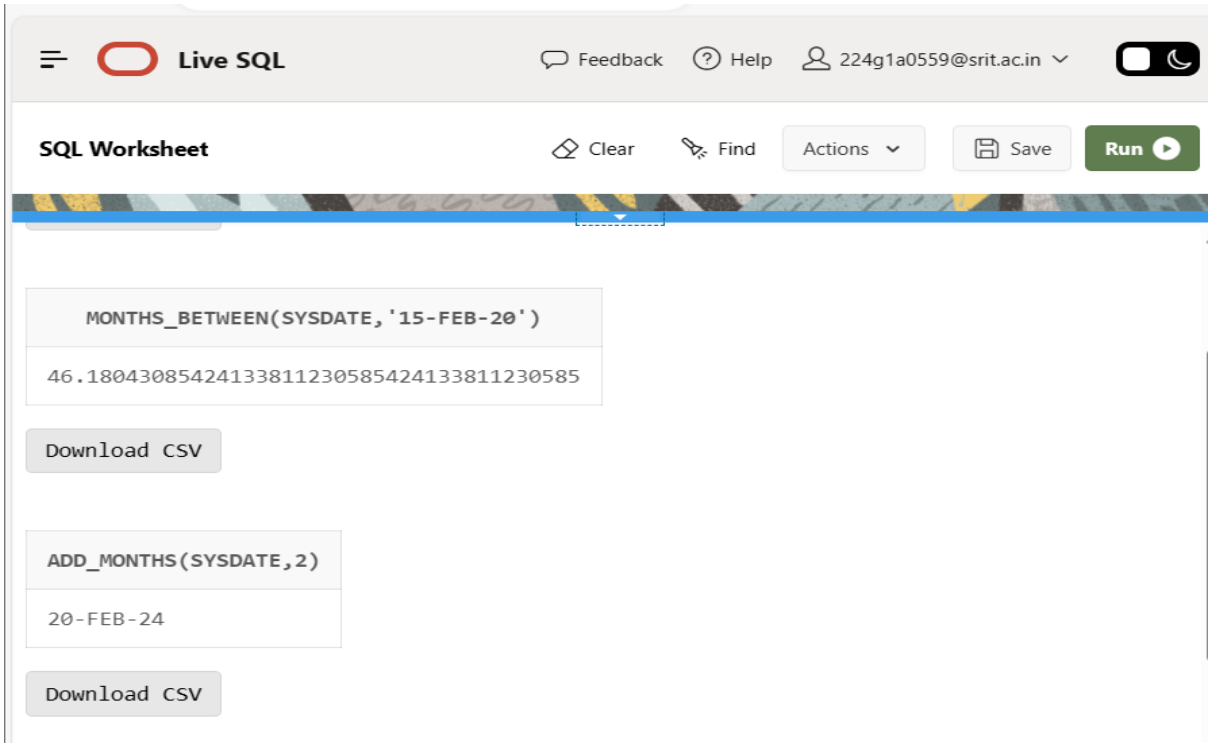
224g1a0559(CSE) – Neha Anjum





2.Date Functions:



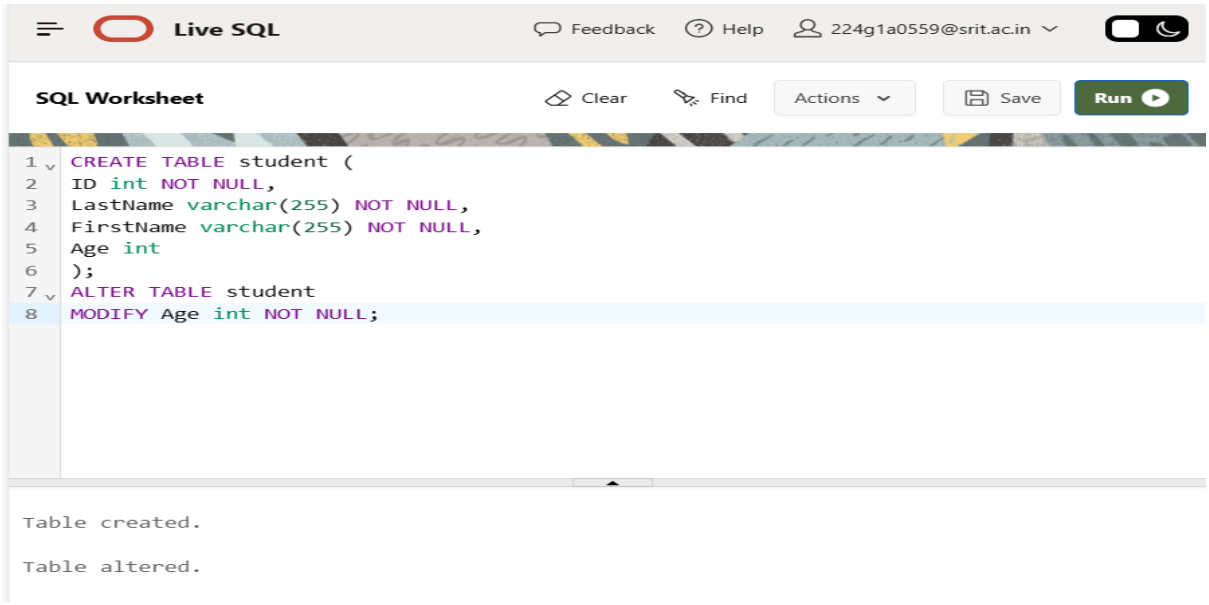


Conclusion:

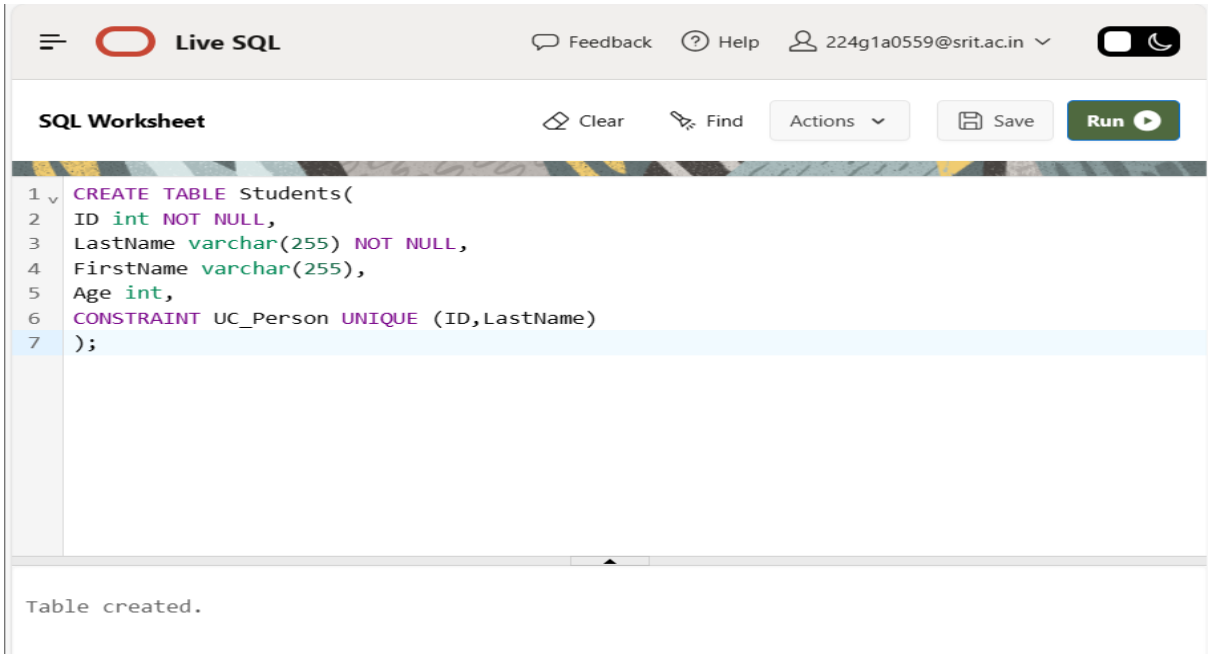
In this lab, we successfully practiced SQL Queries to perform ORACLE BUILT-IN Functions.

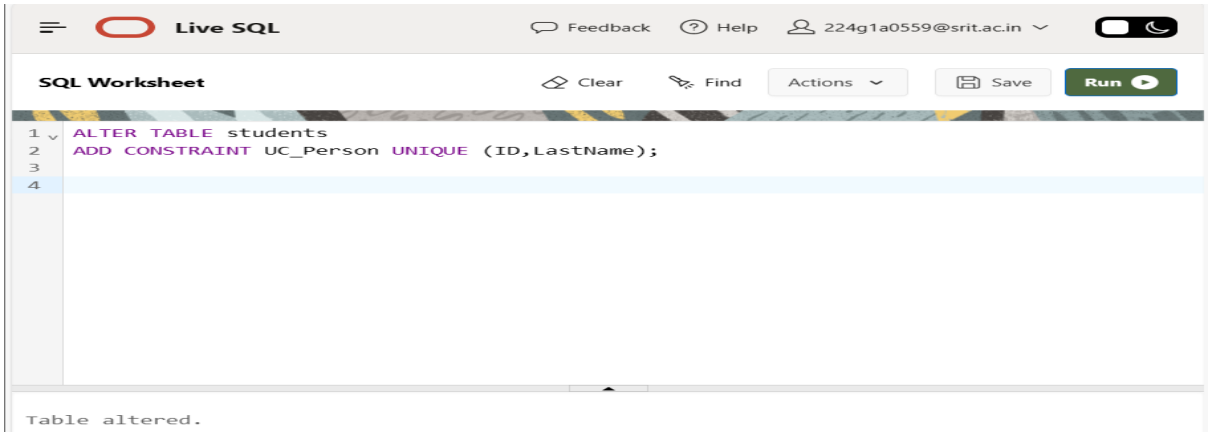
EXPERIMENT – 9

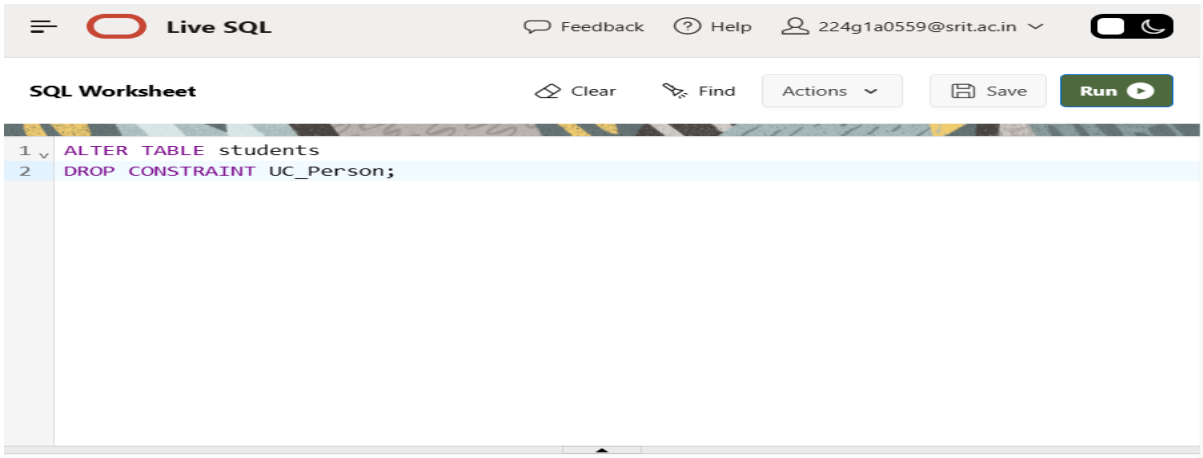
1.NOT NULL CONSTRAINT Example:



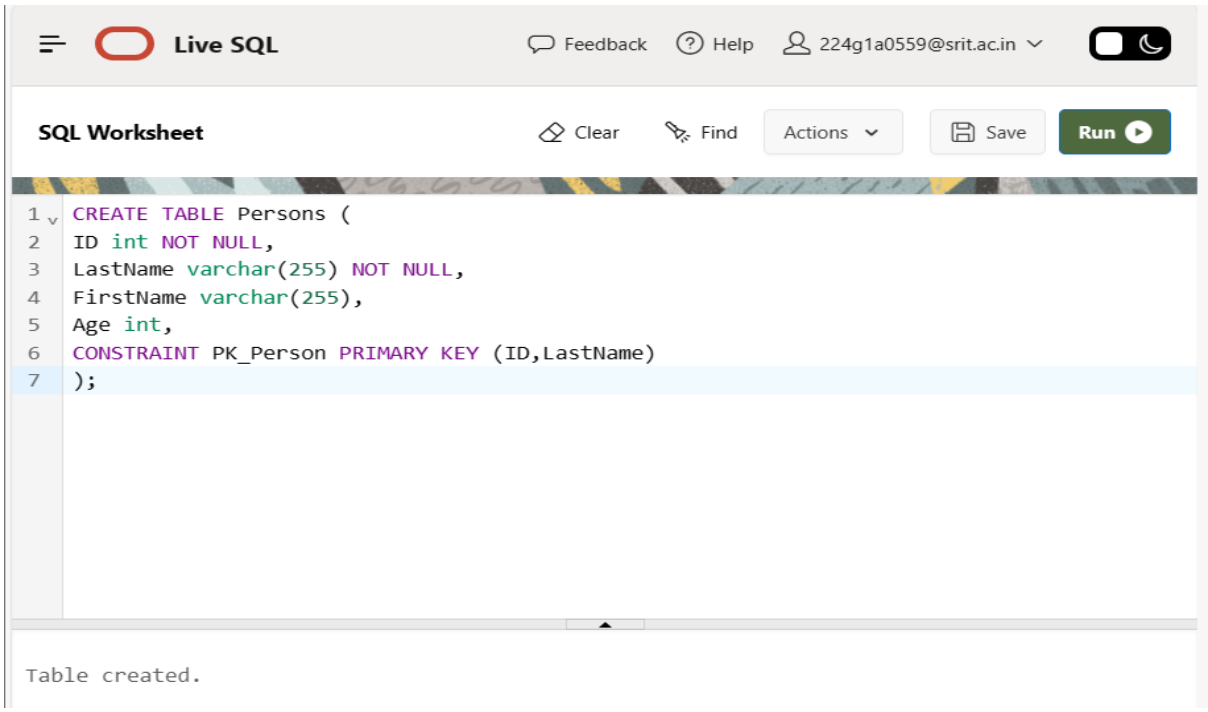
2.UNIQUE CONSTRAINT Example:

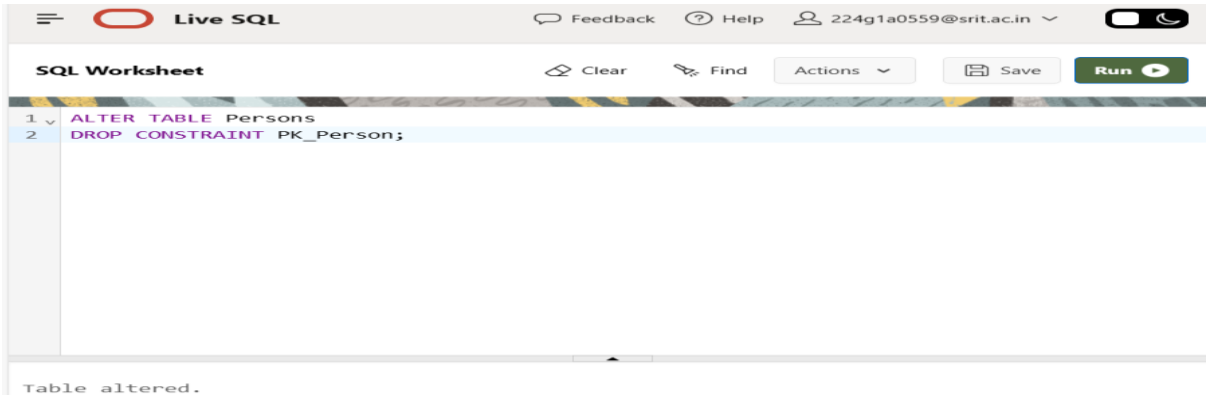




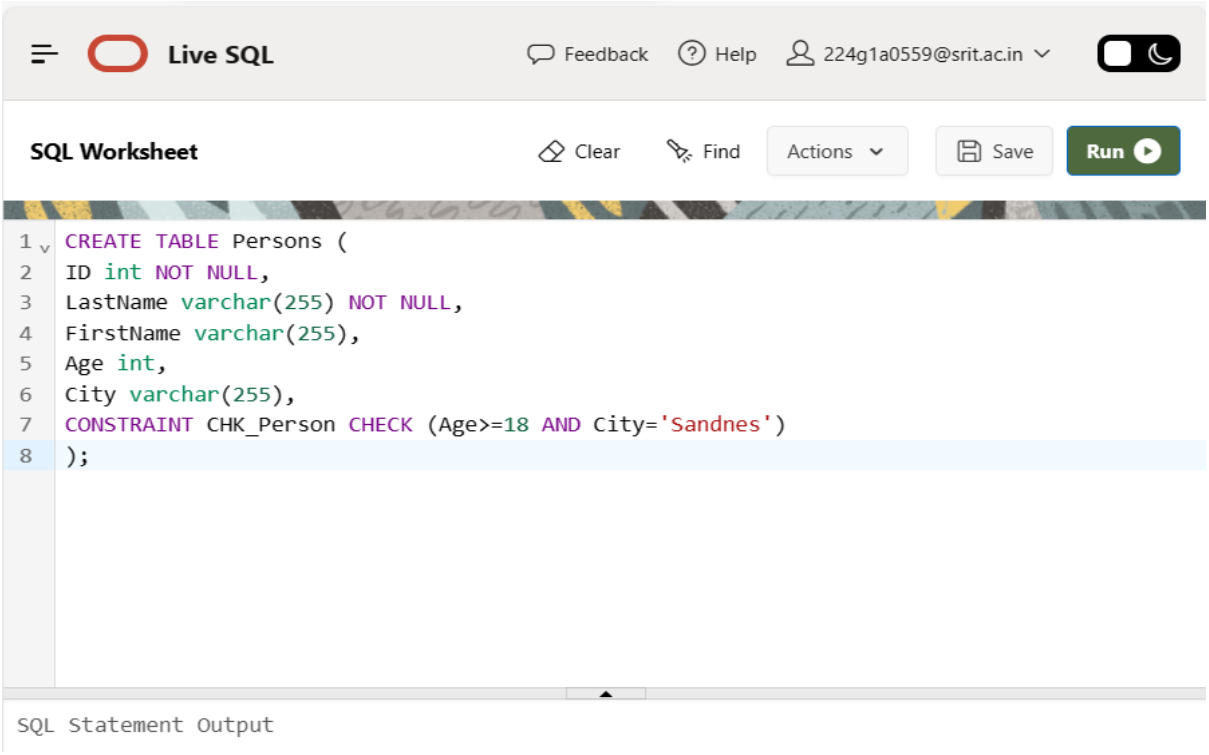


3.PRIMARY KEY CONSTRAINT Example:





4.DEFAULT CONSTRAINTS Example:

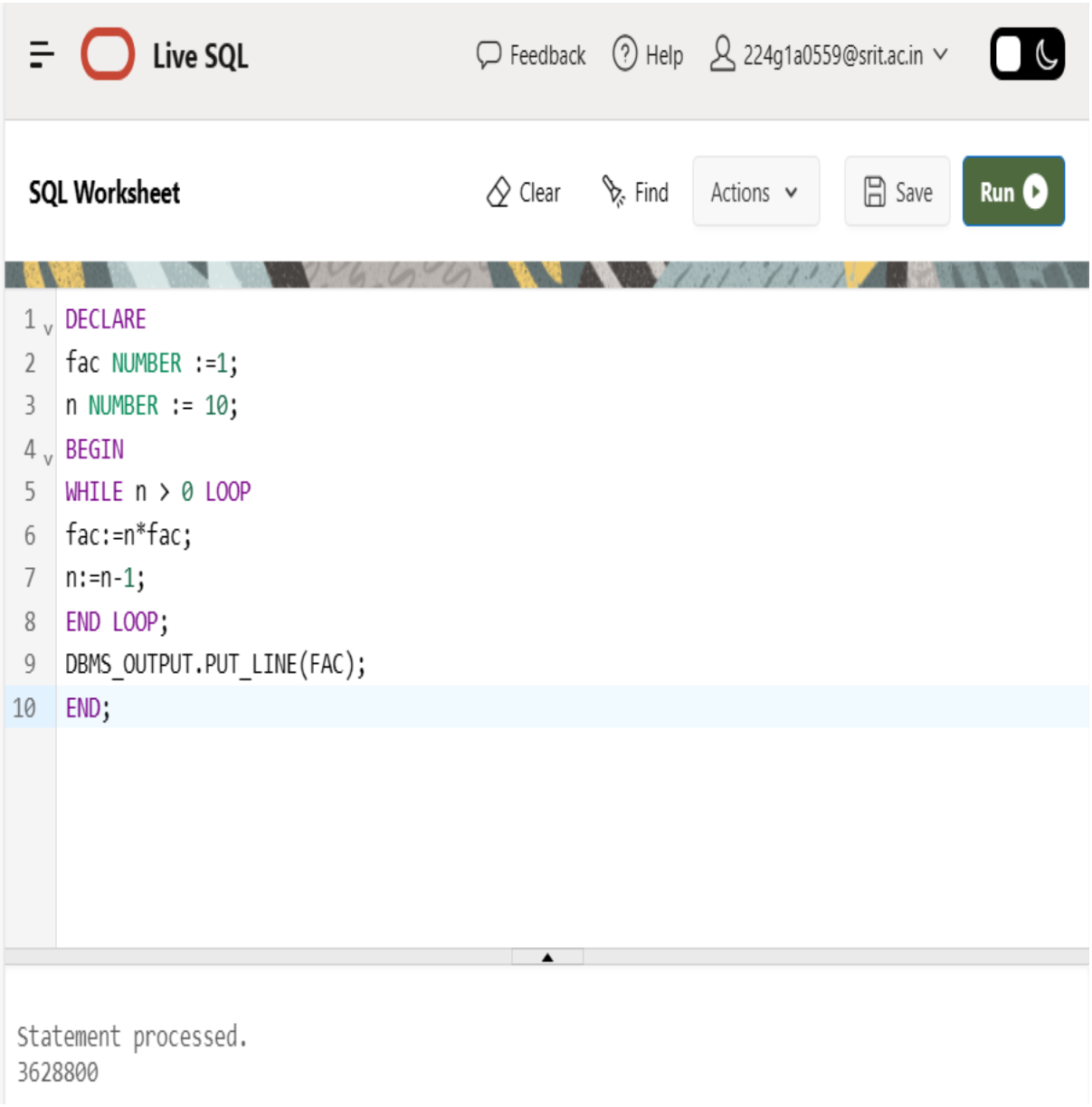


Conclusion:

In this lab, we successfully executed SQL Queries to perform KEY CONSTRAINTS.

Experiment – 10

1.Factorial Program:

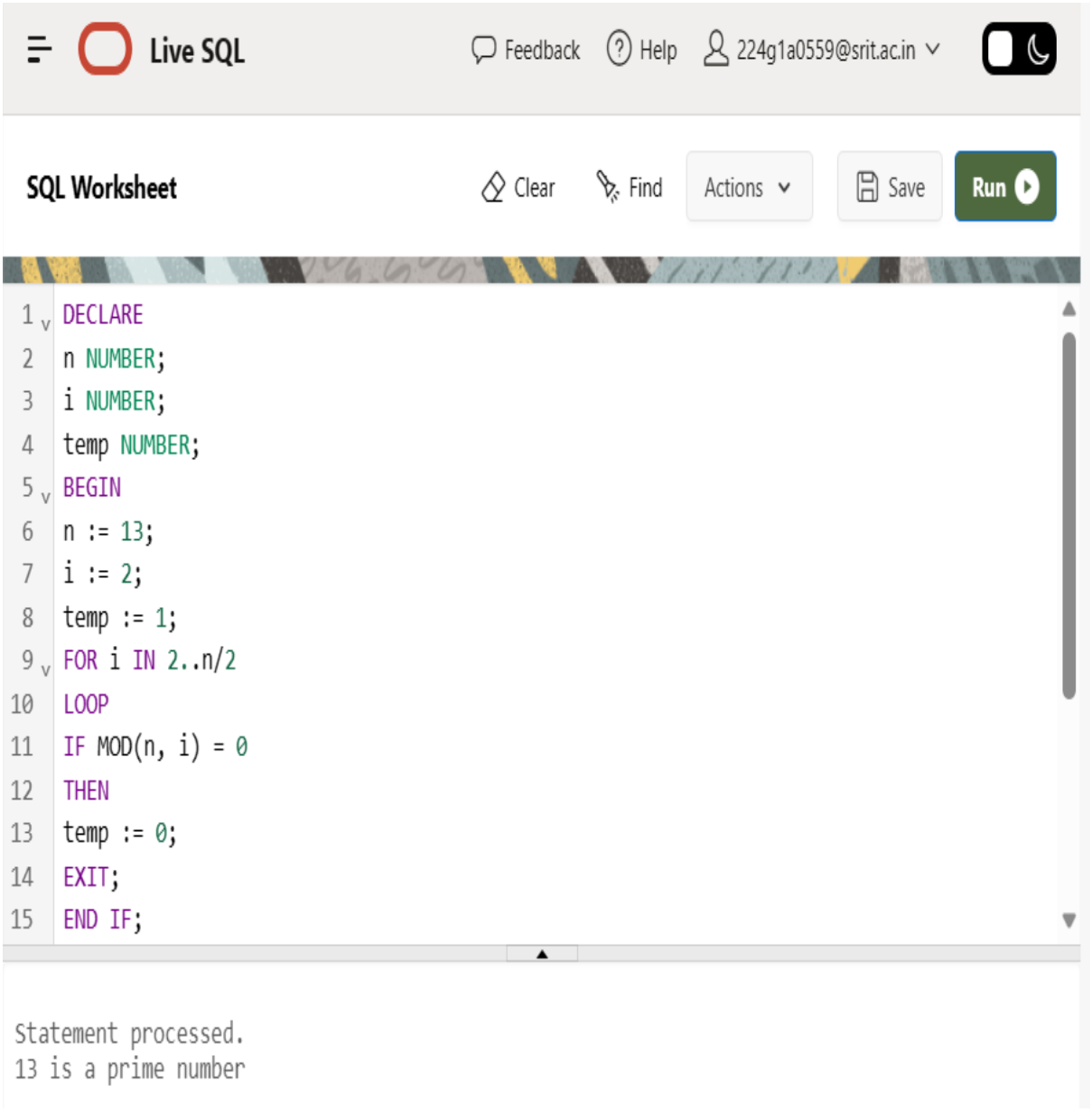


Conclusion:

In this lab, We successfully practiced a program for calculating the factorial of a given number.

Experiment- 11

1.Prime number or Not Program:

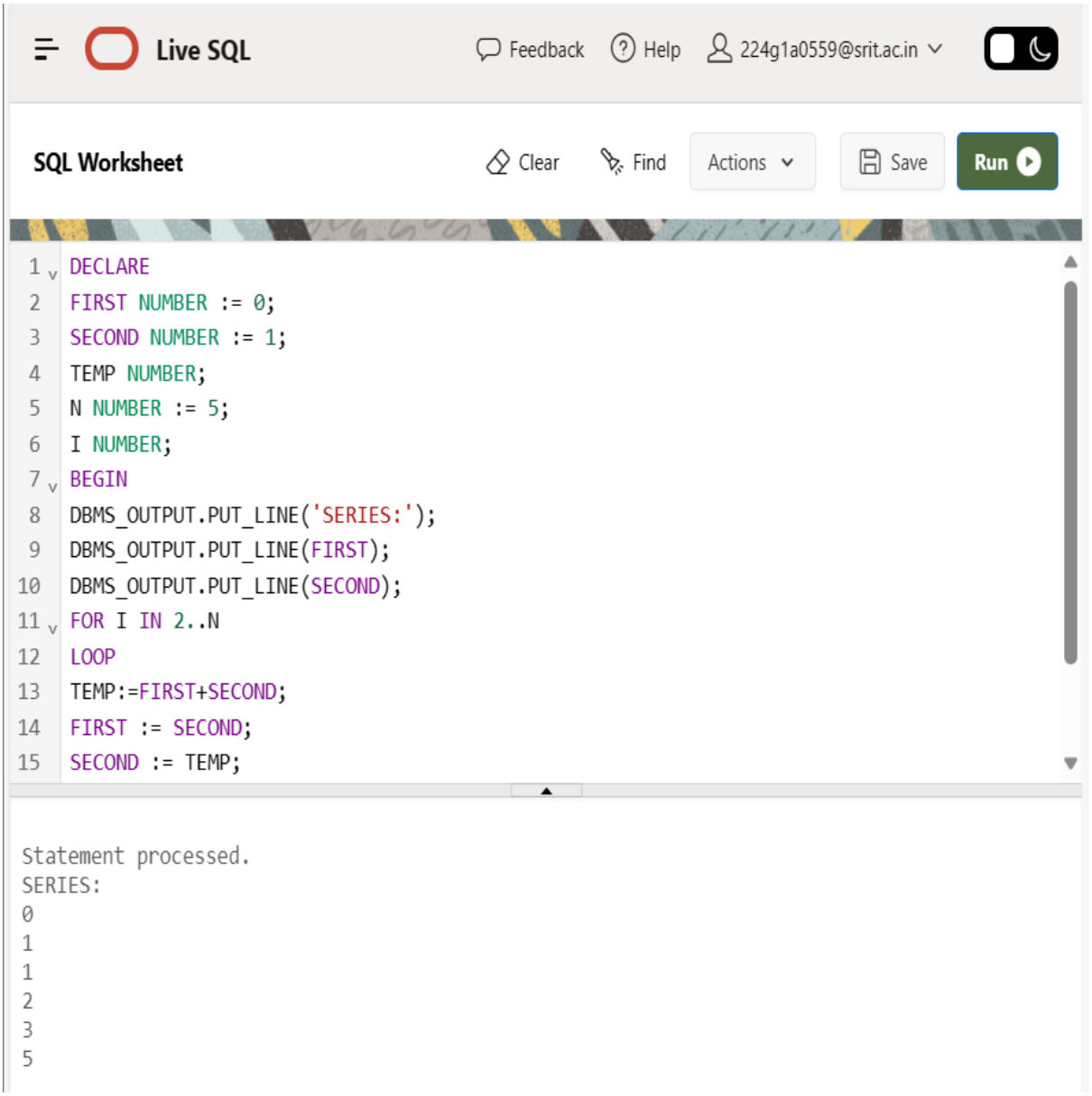


Conclusion:

In this lab, we Successfully practiced a program for Finding the given Number is Prime number or not.

Experiment- 12

1. Fibonacci Series Program:

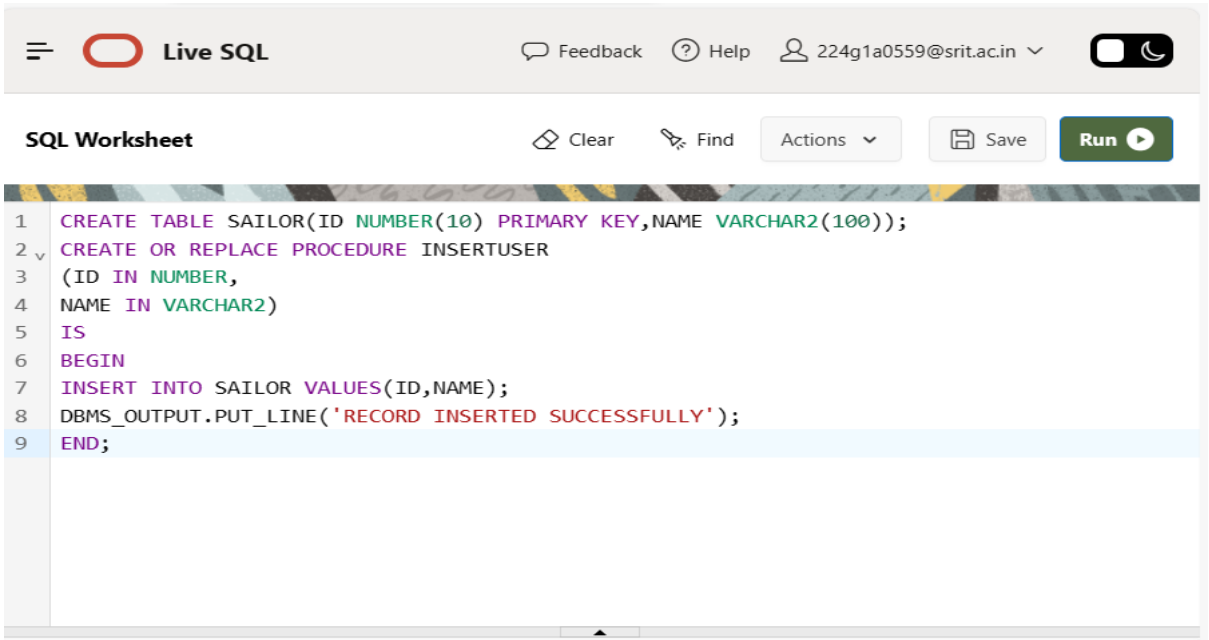


Conclusion:

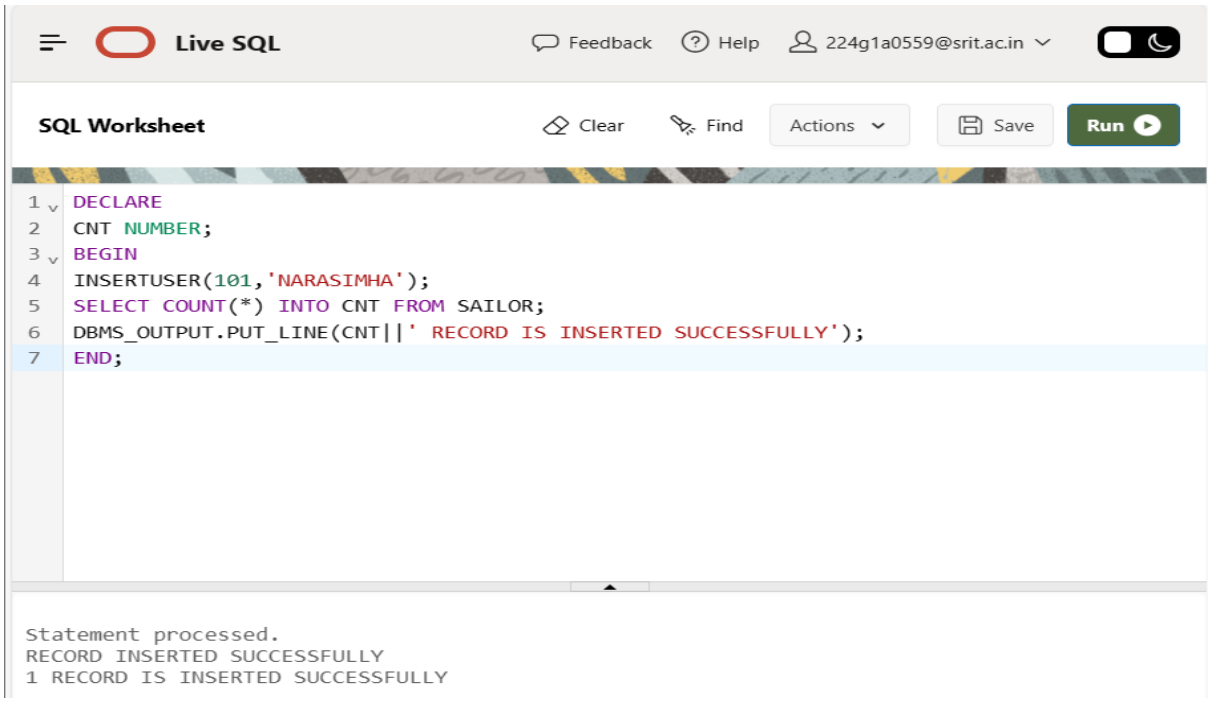
In this lab, we Successfully practiced a program for displaying Fibonacci Series up to an integer.

Experiment- 13

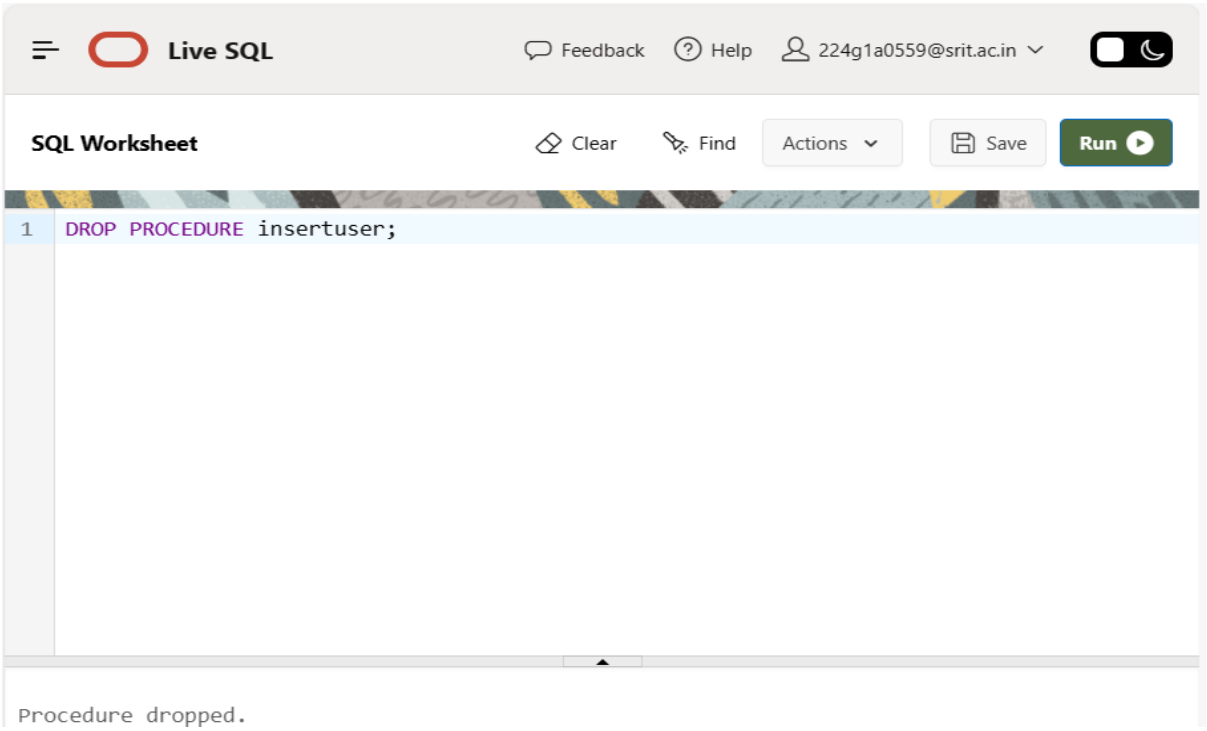
1.CREATE TABLE:



2.Execution Procedure:



3.DROP Procedure:

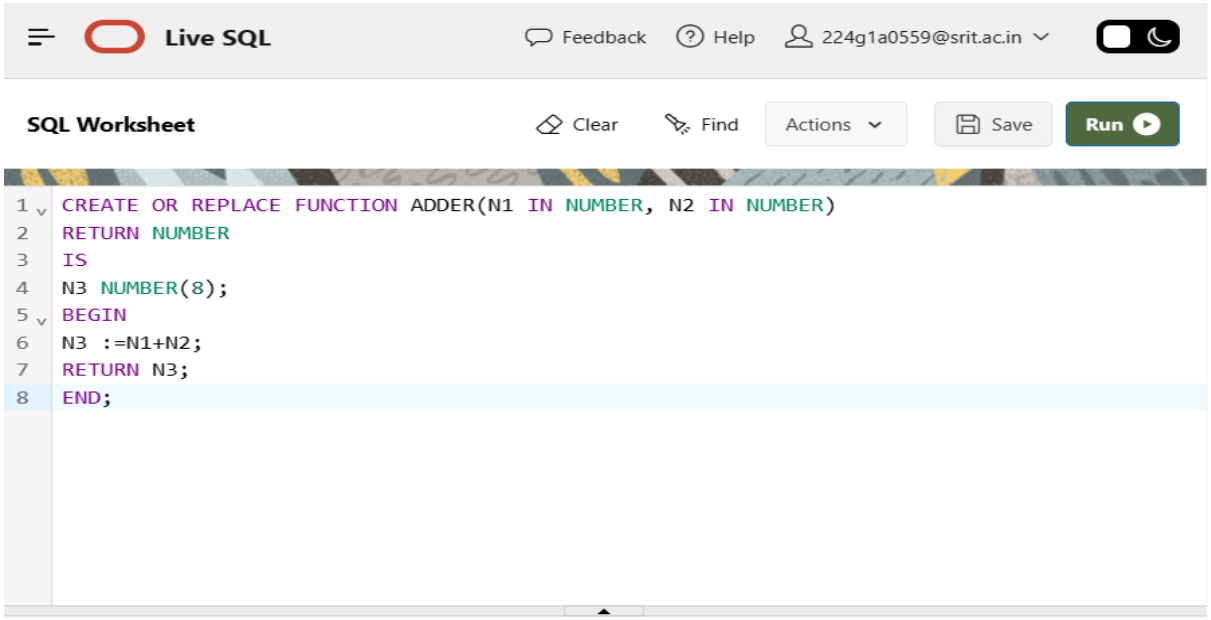


Conclusion:

In this lab, we successfully practiced How to write a program to implement Stored Procedure on table.

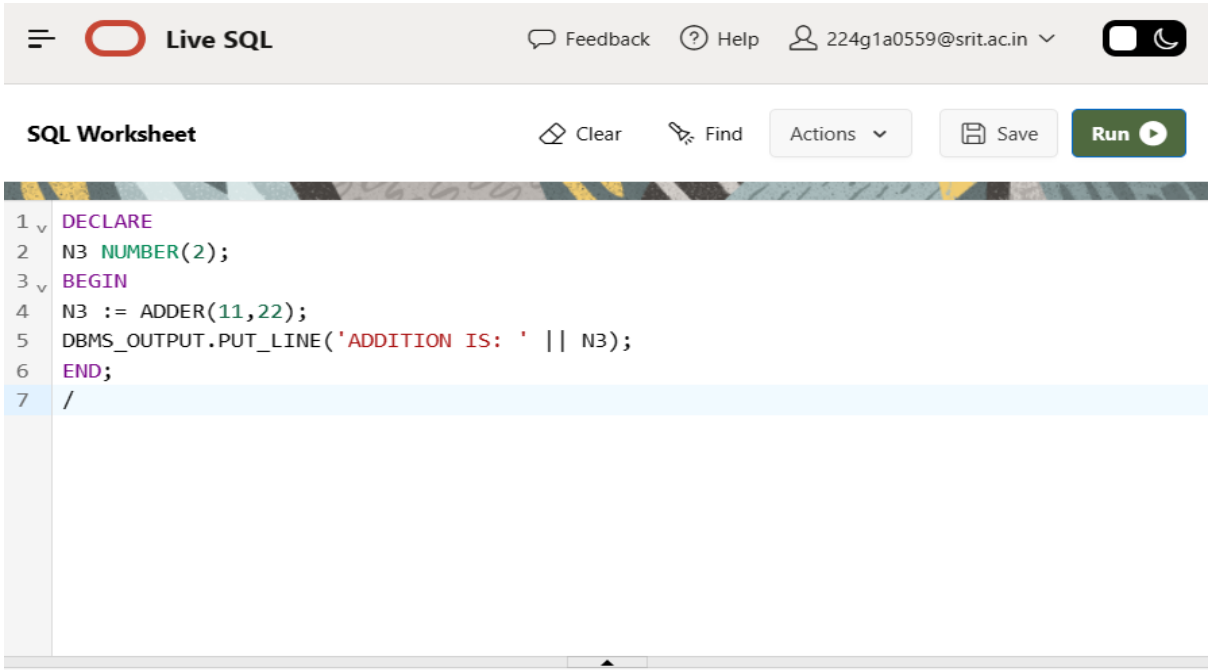
Experiment- 14

1.PL/SQL Function:



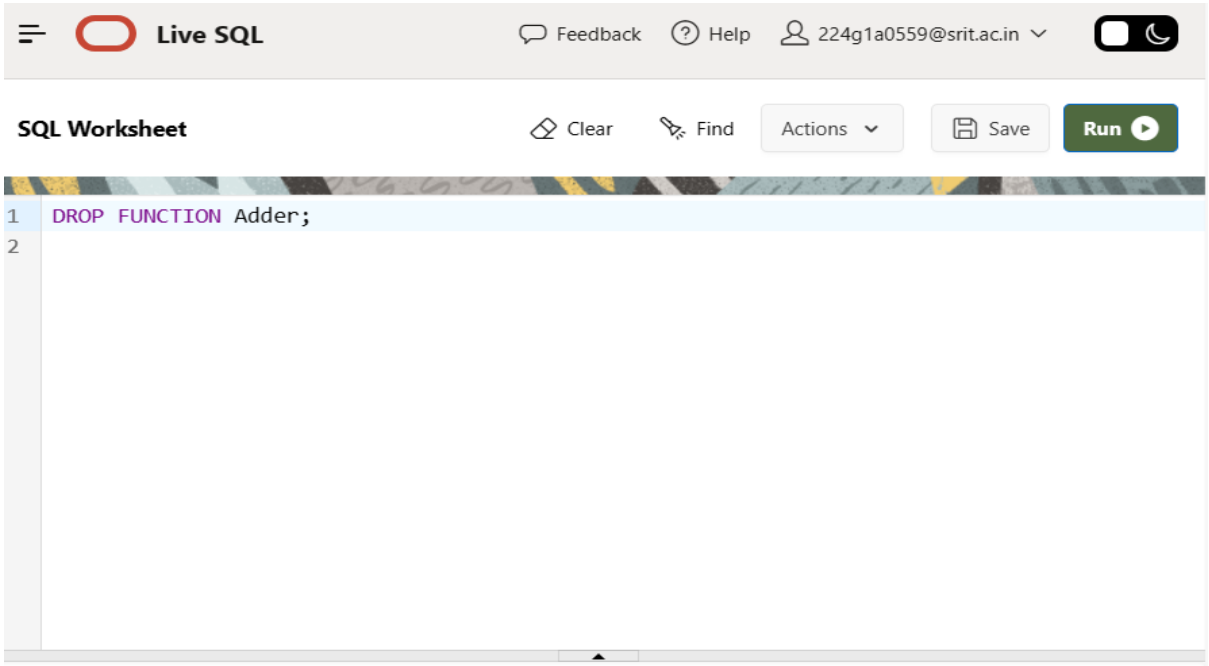

```
Function created.
```

2.Execution Procedure:



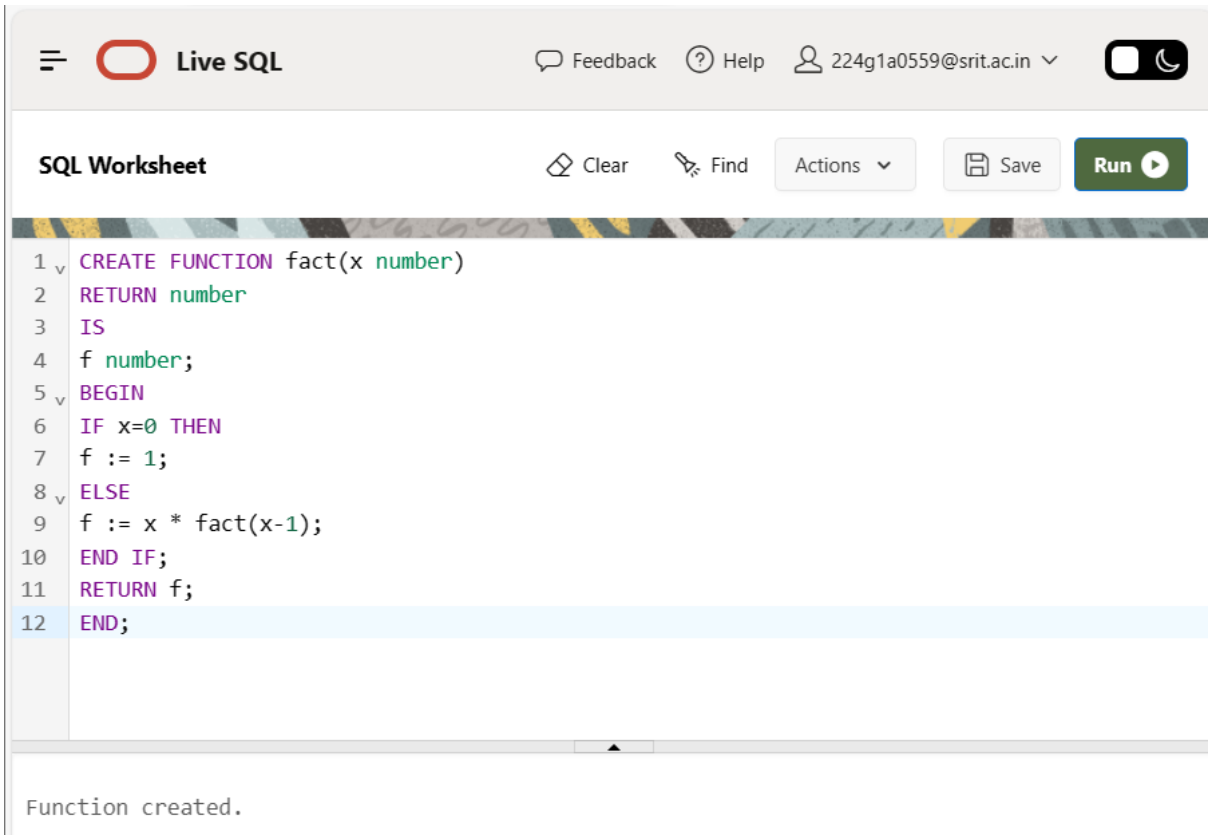

```
Statement processed.
ADDITION IS: 33
```

3.DROP Function:



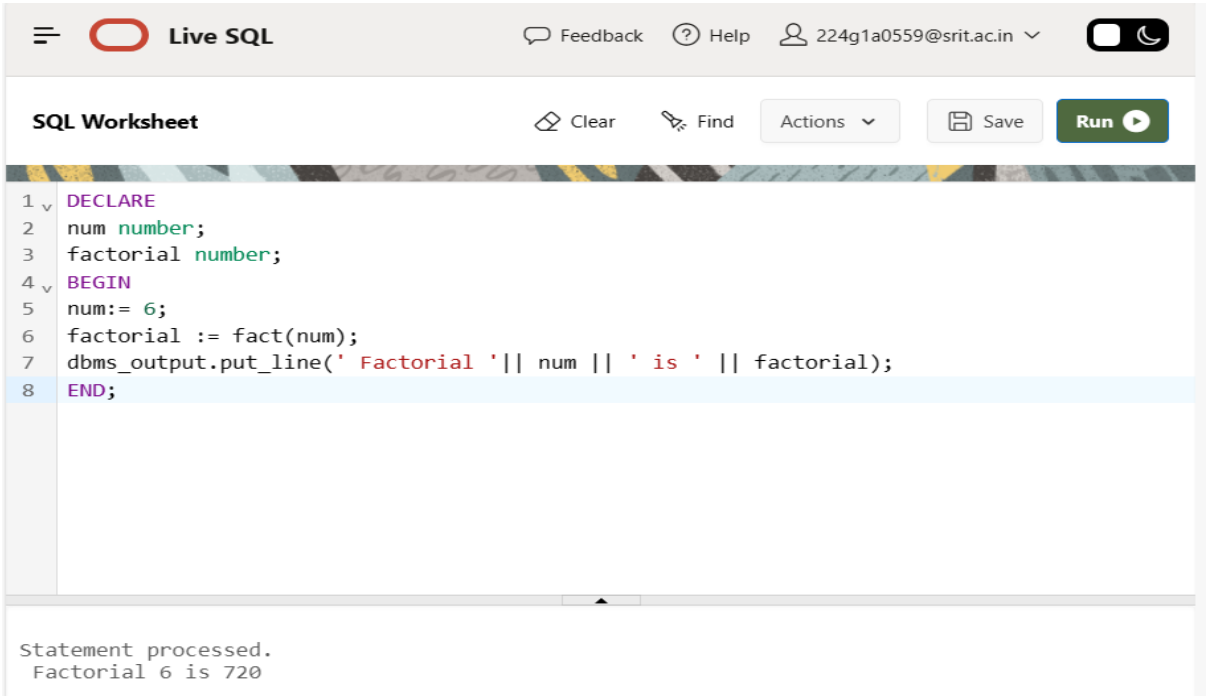

```
Function dropped.
```

4.Recursive Function:

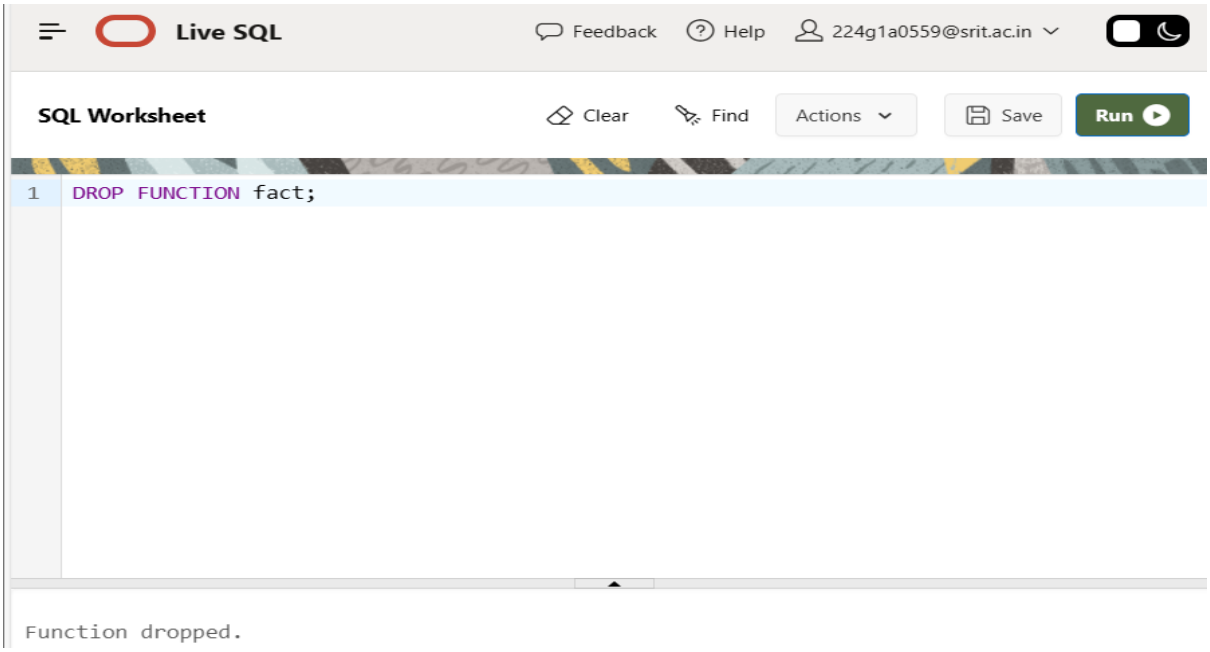



```
Function created.
```

5.Execution Procedure:



6.DROP Function:

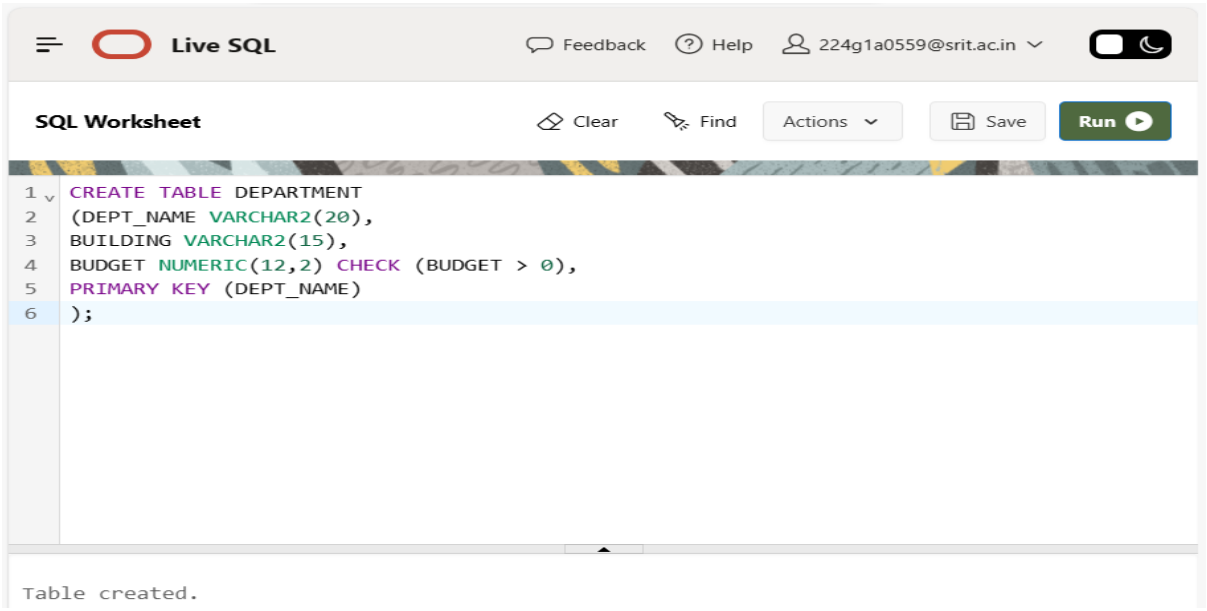


Conclusion:

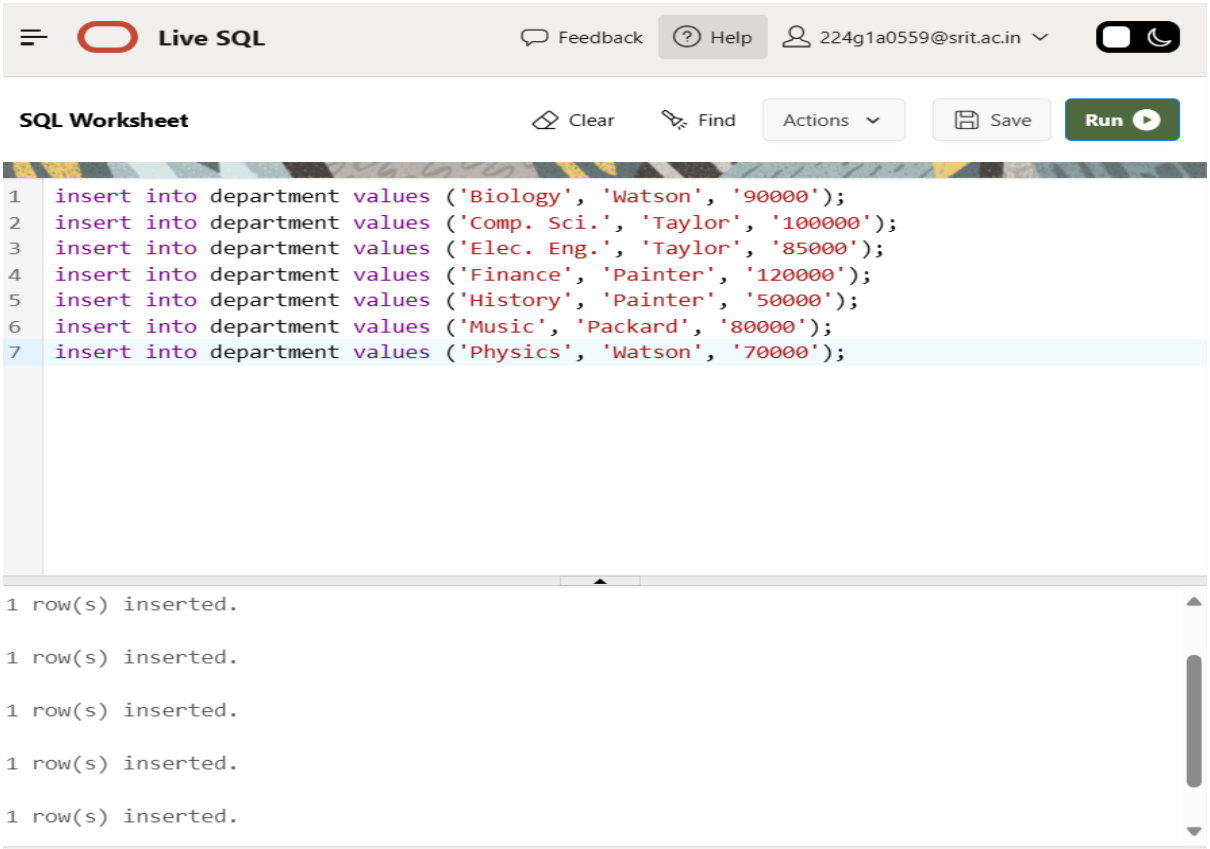
In This Lab, we successfully practiced PL/SQL Program to implement Stored Function on Table.

Experiment – 15

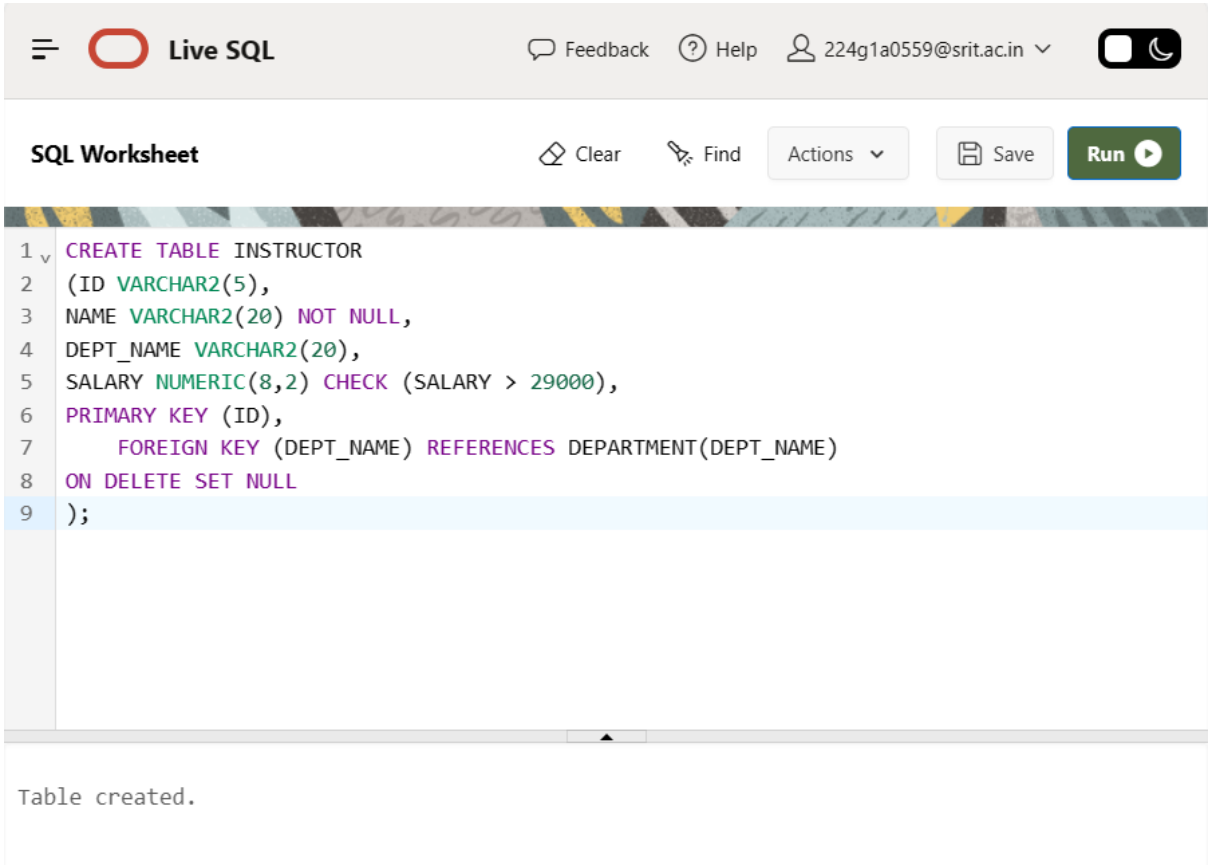
1.Create Department Table



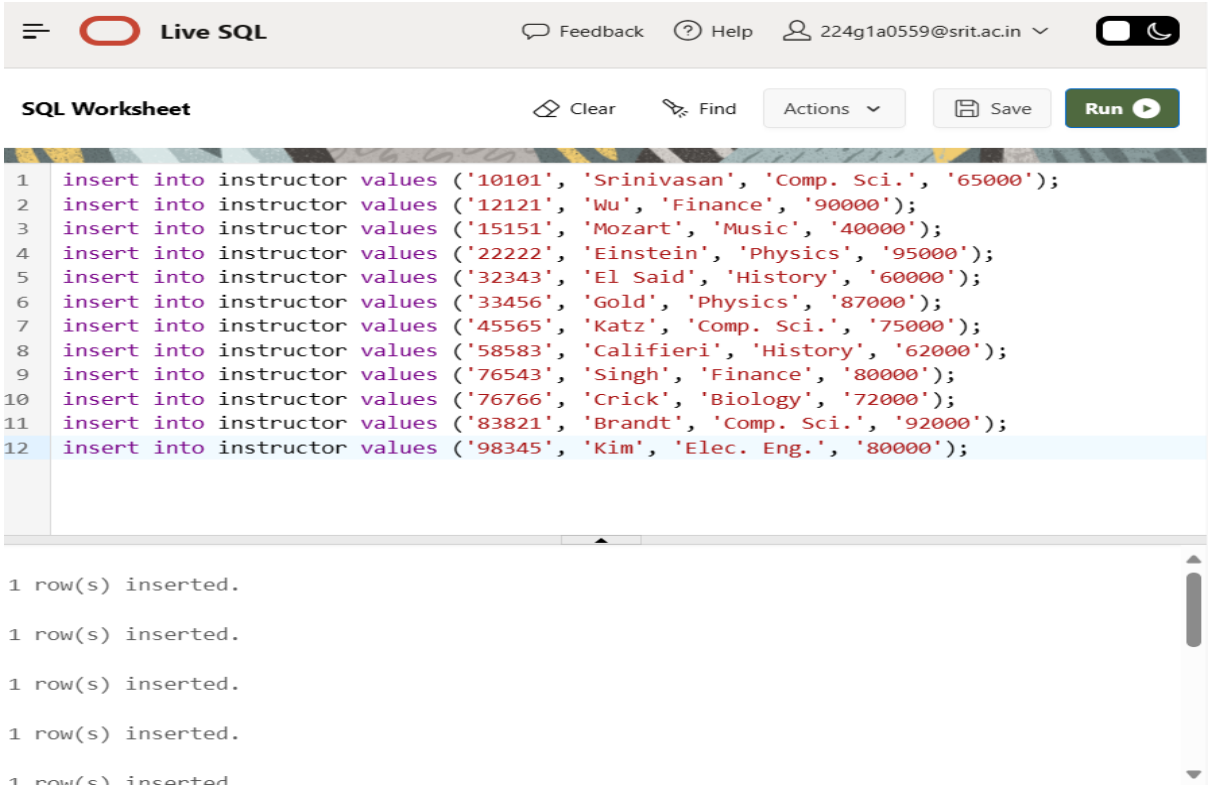
2.Instances of Department Table



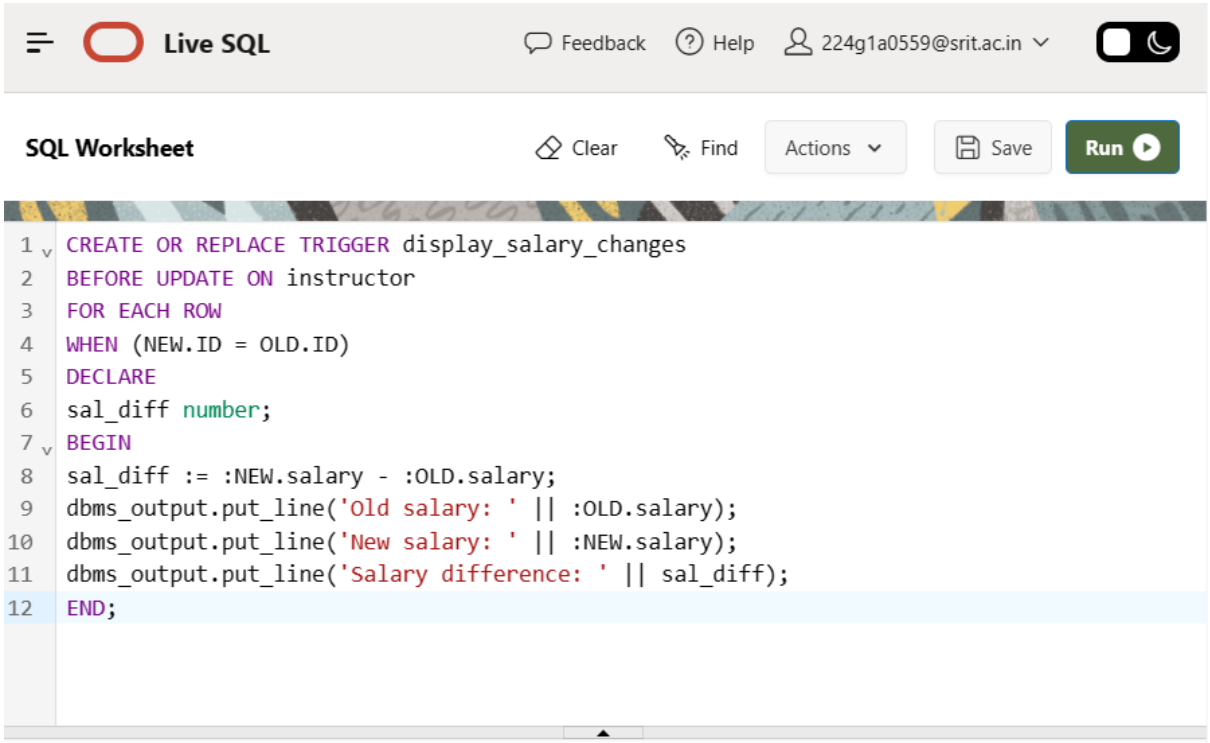
3.Create Instructor Table



4.Instances of Instructor Table

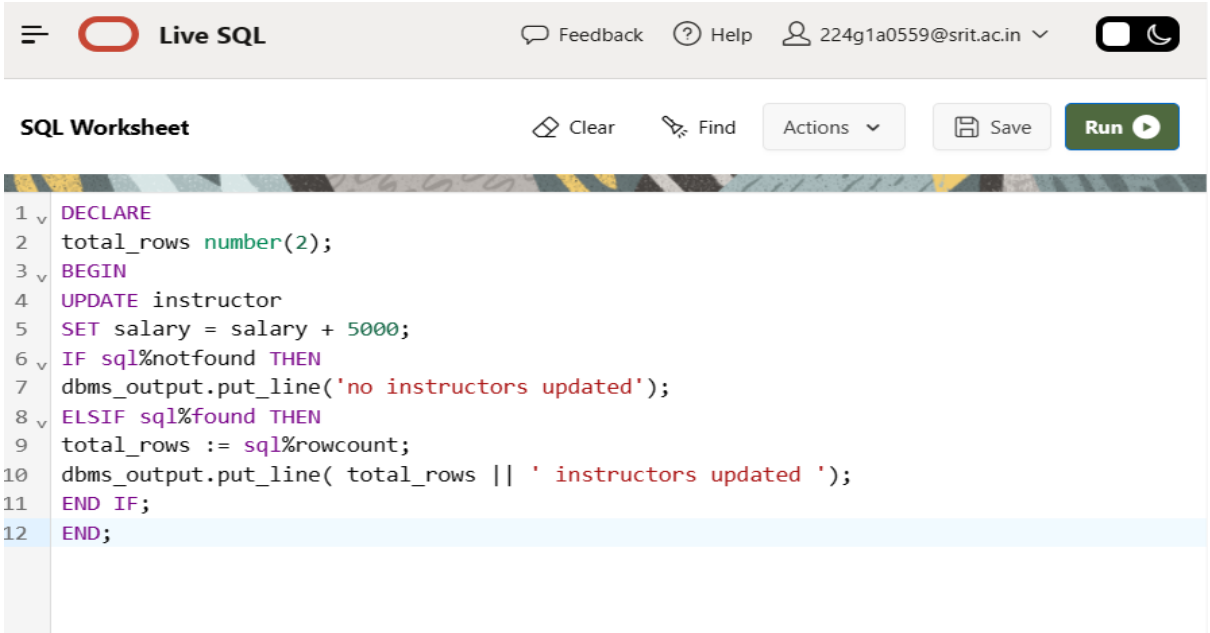


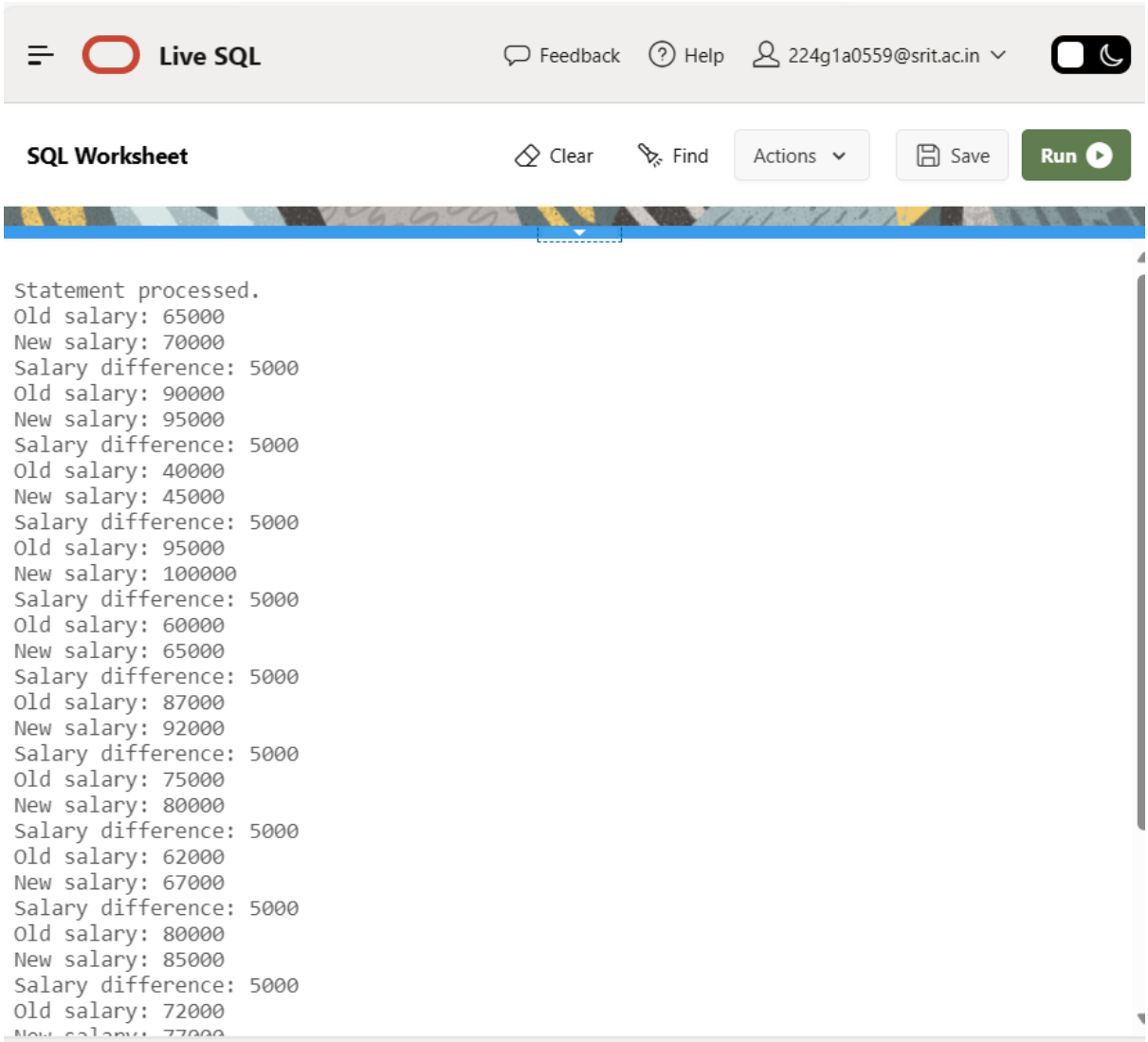
5.Example to Create Trigger

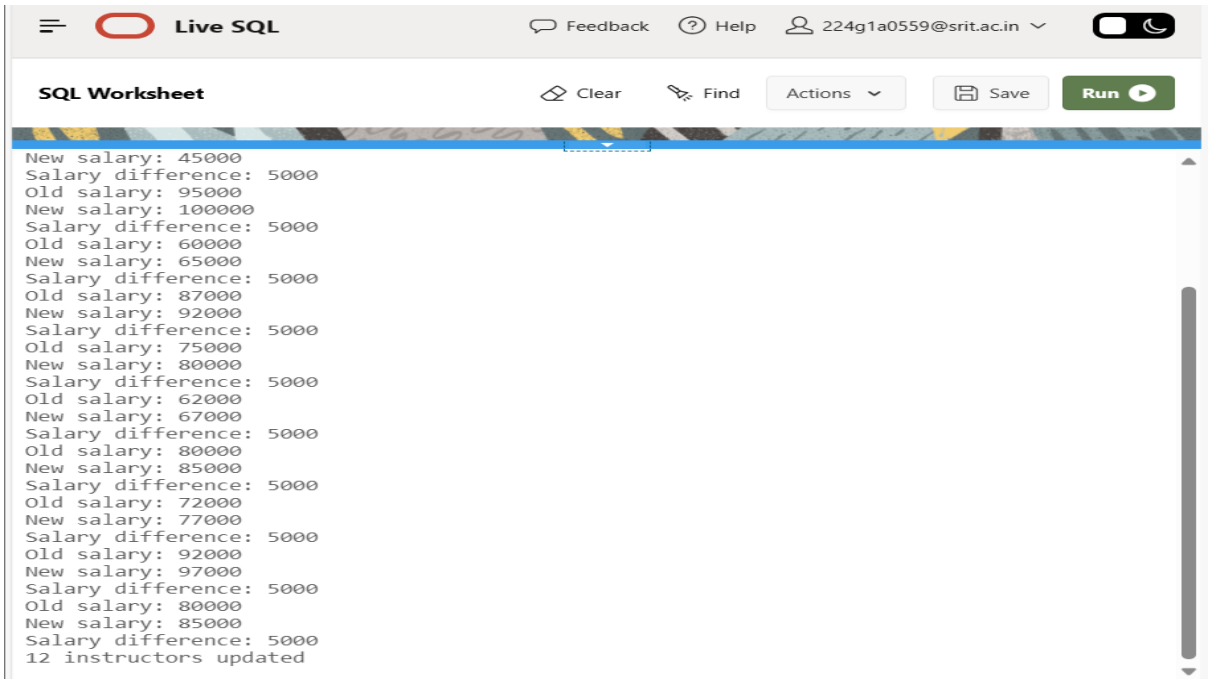



```
Trigger created.
```

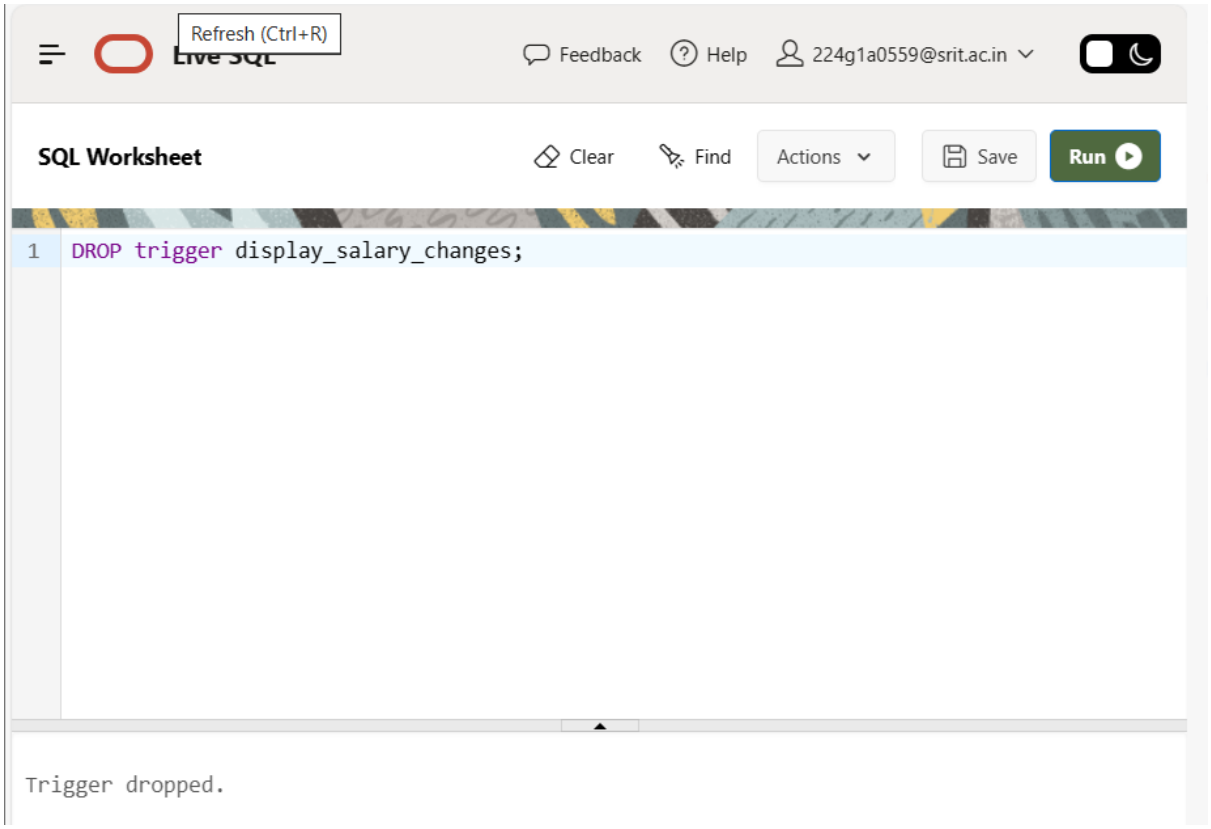
6.PL/SQL Program to Execute a trigger




Live SQL
Feedback
Help
224g1a0559@srit.ac.in
SQL Worksheet
Clear
Find
Actions
Save
Run
Statement processed.
Old salary: 65000
New salary: 70000
Salary difference: 5000
Old salary: 90000
New salary: 95000
Salary difference: 5000
Old salary: 40000
New salary: 45000
Salary difference: 5000
Old salary: 95000
New salary: 100000
Salary difference: 5000
Old salary: 60000
New salary: 65000
Salary difference: 5000
Old salary: 87000
New salary: 92000
Salary difference: 5000
Old salary: 75000
New salary: 80000
Salary difference: 5000
Old salary: 62000
New salary: 67000
Salary difference: 5000
Old salary: 80000
New salary: 85000
Salary difference: 5000
Old salary: 72000


Live SQL
Feedback
Help
224g1a0559@srit.ac.in
SQL Worksheet
Clear
Find
Actions
Save
Run
New salary: 45000
Salary difference: 5000
Old salary: 95000
New salary: 100000
Salary difference: 5000
Old salary: 60000
New salary: 65000
Salary difference: 5000
Old salary: 87000
New salary: 92000
Salary difference: 5000
Old salary: 75000
New salary: 80000
Salary difference: 5000
Old salary: 62000
New salary: 67000
Salary difference: 5000
Old salary: 80000
New salary: 85000
Salary difference: 5000
Old salary: 72000
New salary: 77000
Salary difference: 5000
Old salary: 92000
New salary: 97000
Salary difference: 5000
Old salary: 80000
New salary: 85000
Salary difference: 5000
12 instructors updated

7.Drop the Trigger

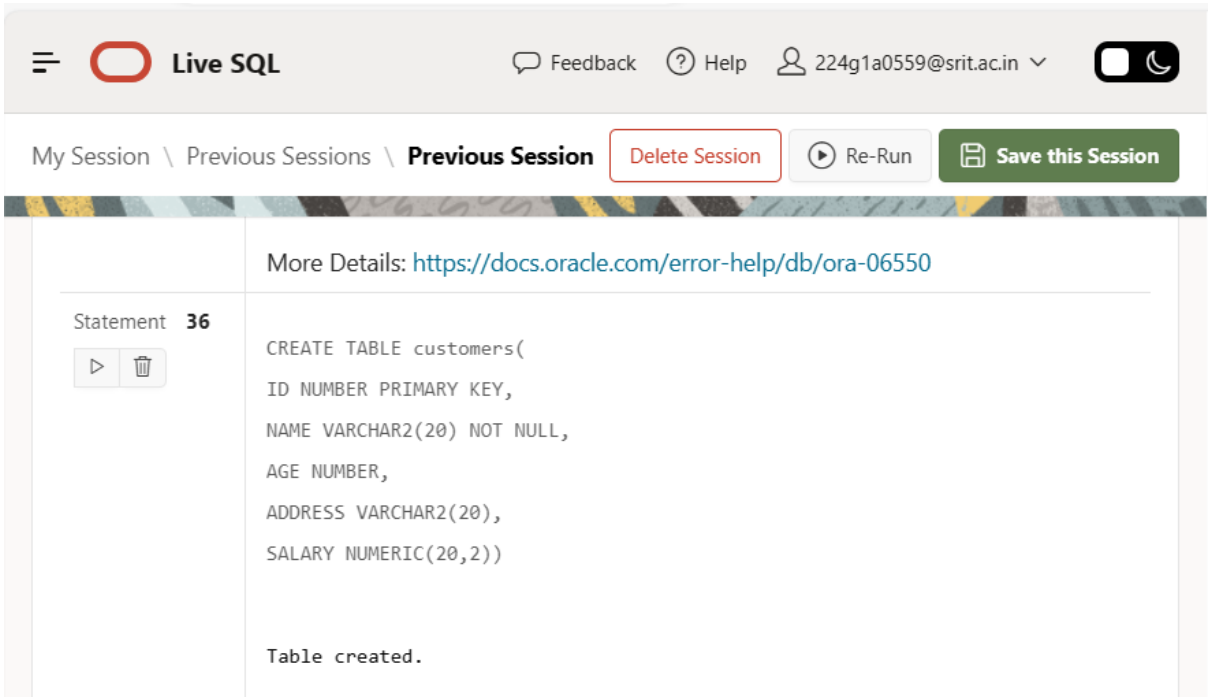


Conclusion:

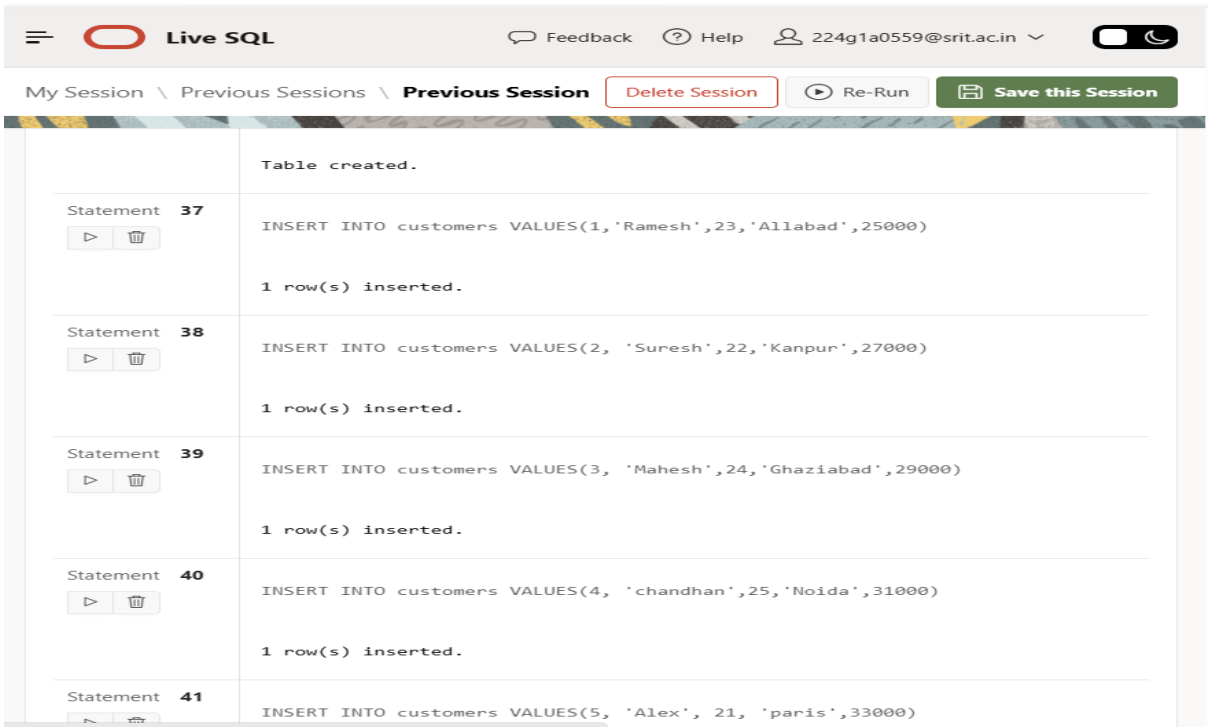
In this lab, We successfully Executed PL/SQL program to implement Trigger on Table.

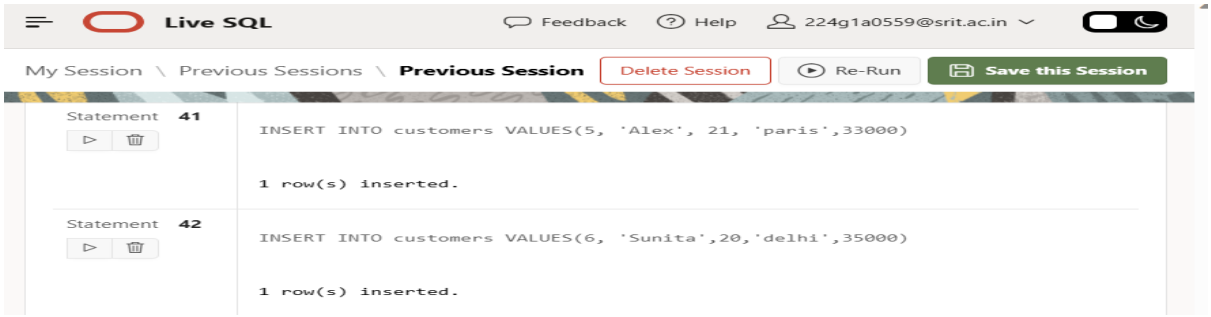
Experiment – 16

1.Create Table

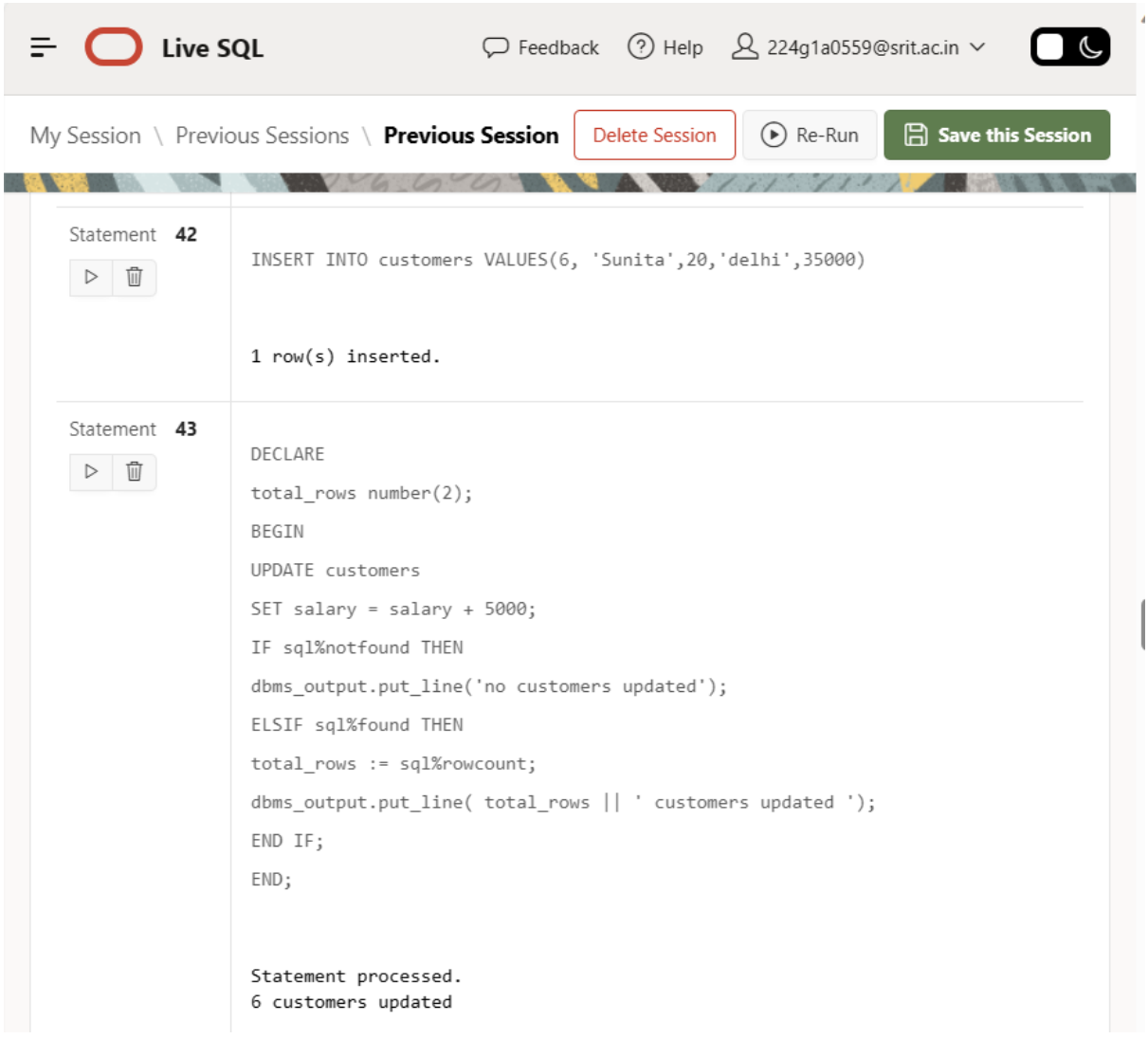


2.Instances of Customers

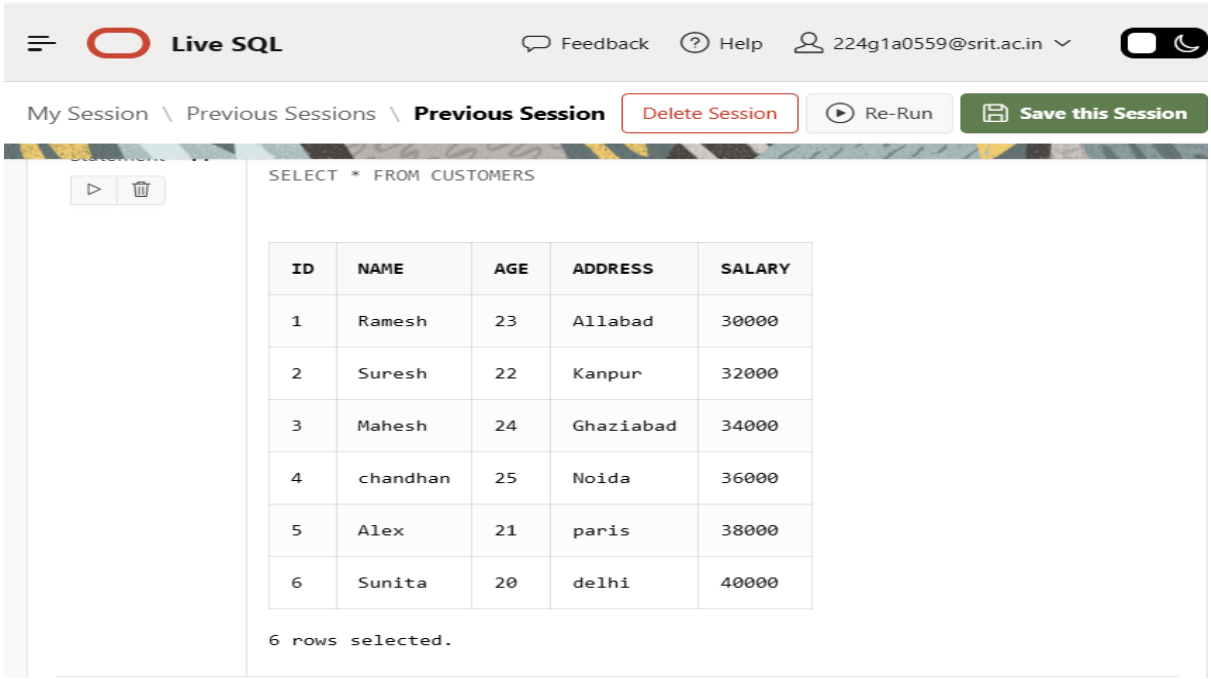




3.Create Update Procedure



4.Display Created Customers Table



ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allabad	30000
2	Suresh	22	Kanpur	32000
3	Mahesh	24	Ghaziabad	34000
4	chandhan	25	Noida	36000
5	Alex	21	paris	38000
6	Sunita	20	delhi	40000

6 rows selected.

5.PL/SQL Program using Explicit Cursors


```
c_id customers.id%type;
c_name customers.name%type;
c_addr customers.address%type;
CURSOR c_customers is
SELECT id, name, address FROM customers;
BEGIN
OPEN c_customers;
LOOP
FETCH c_customers into c_id, c_name, c_addr;
EXIT WHEN c_customers%notfound;
dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
END LOOP;
CLOSE c_customers;
END;
/
```

```
Statement processed.
1 Ramesh Allabad
2 Suresh Kanpur
3 Mahesh Ghaziabad
4 chandhan Noida
5 Alex paris
6 Sunita delhi
```

Conclusion:

In this lab, we successfully Practiced PL/SQL Program to implement Cursor On table.