VIT Vidyalankar Institute of Technology Accredited A+ by NAAC	Department of Computer Engineering A.Y-2025-26

Semester	T.E. Semester VI Div C Batch – CMPN
Subject	ADBMS Lab
Subject Professor Incharge	Prof. Divya Nimbalkar
Assisting Teachers	Prof. Divya Nimbalkar

Student Name	Nilesh Patni	
Roll Number	24102C2001	
Grade and Subject Teacher's Signature		

Experiment Number	03
Experiment Title	Implementation of Role-Based Access Control (RBAC) with Row Level Security
Objectives (Skill Set / Knowledge Tested / Imparted)	• To secure the database of the Crowdsourced Traffic Incident Reporting System by implementing Role-Based • Access Control (RBAC) and Row-Level Security (RLS) so that different users access data according to their roles.

Aim: To create database roles, assign appropriate privileges, test role-based access by logging in as different users, and implement row-level security for sensitive data in the Crowdsourced Traffic Incident Reporting System.

Theory:

Role-Based Access Control (RBAC): -

Role-Based Access Control is a database security mechanism where permissions are assigned to roles instead of individual users. Users inherit privileges by being assigned a role. This simplifies security management and ensures controlled access to data.

Row-Level Security (RLS): -

Row-Level Security restricts access to specific rows in a table based on user identity or role. It ensures that users can view or modify only the data that belongs to them.

Roles in Crowdsourced Traffic Incident Reporting System: - • Admin: Manages users, incidents, and system data.

- Authority: Verifies incidents and updates incident status.
- User: Reports incidents and views only their own data.

Sensitive tables such as Incident and Feedback require restricted access to maintain data privacy and integrity.

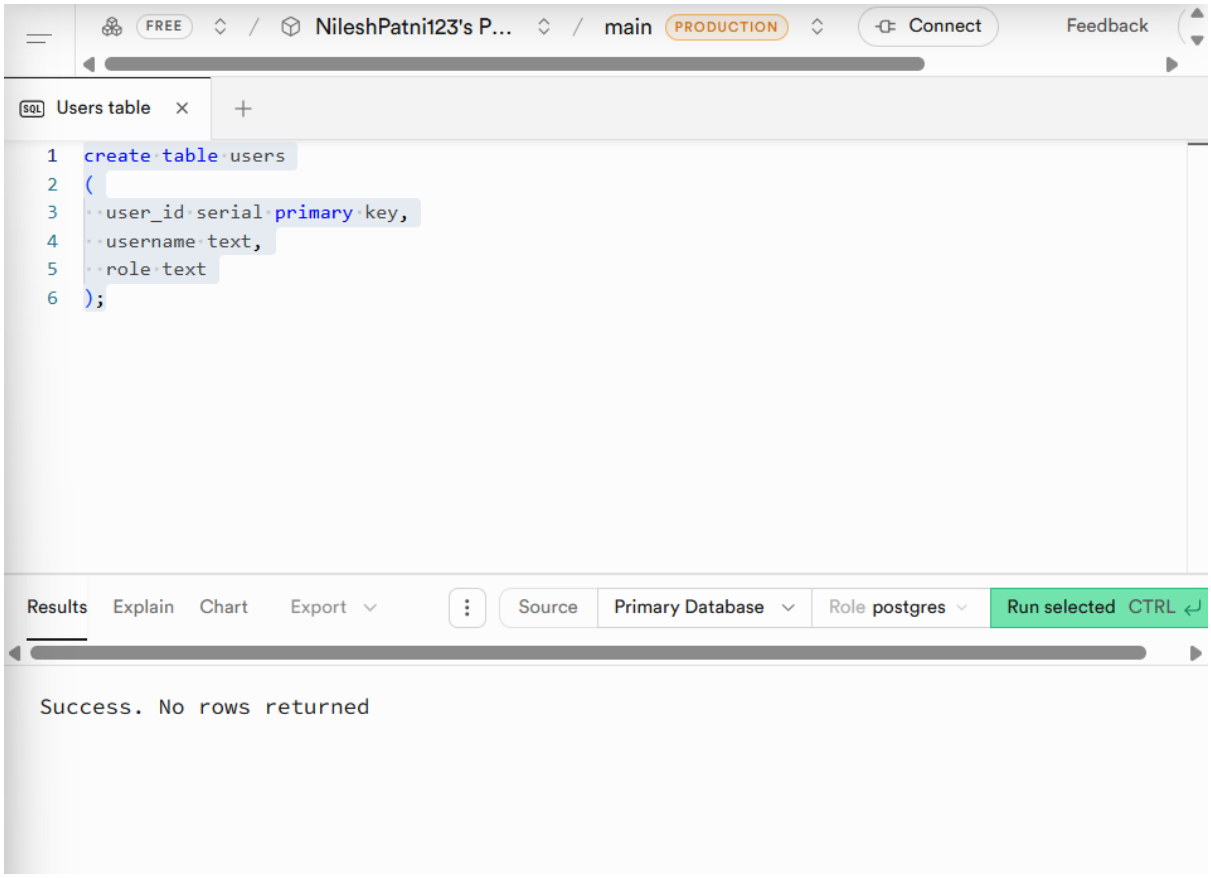
Lab Outcome Mapped: CO

CO Statement: Students will be able to design and implement database security using roles, privileges, and row-level security

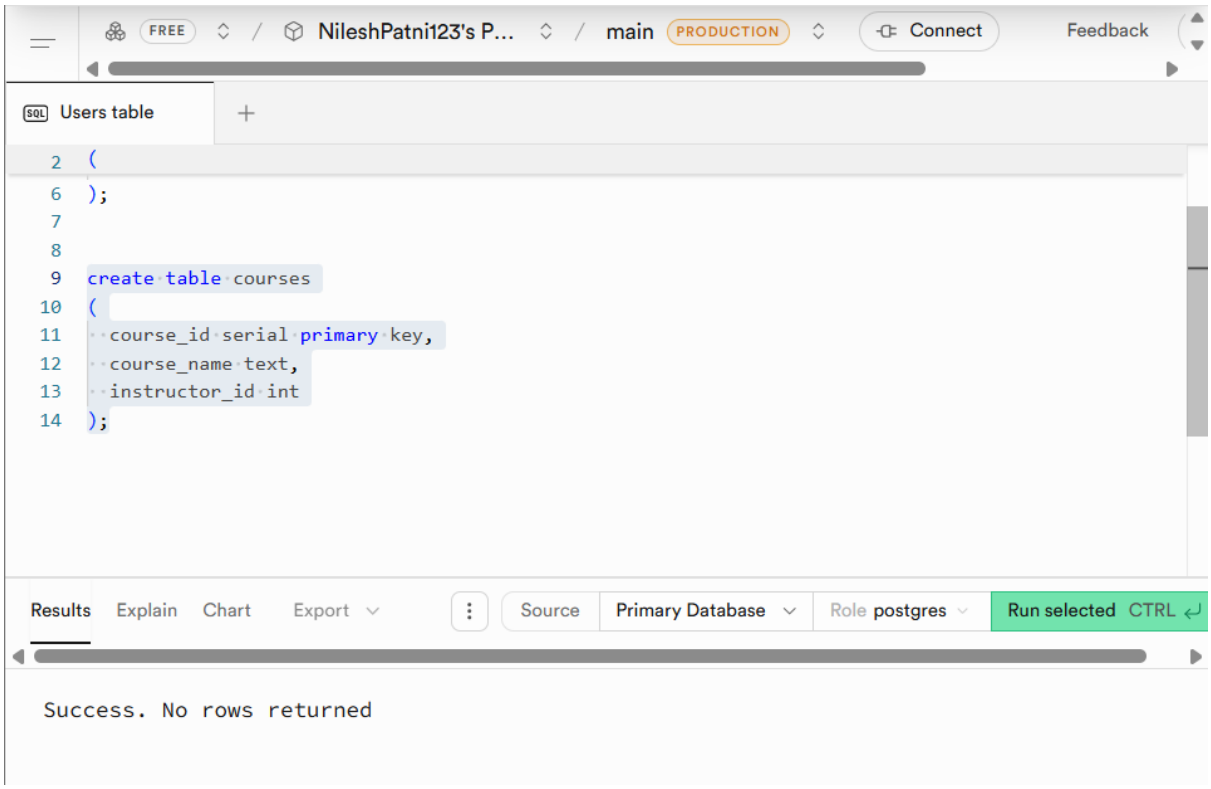
IMPLEMENTATION :

Create Tables (Database Schema)

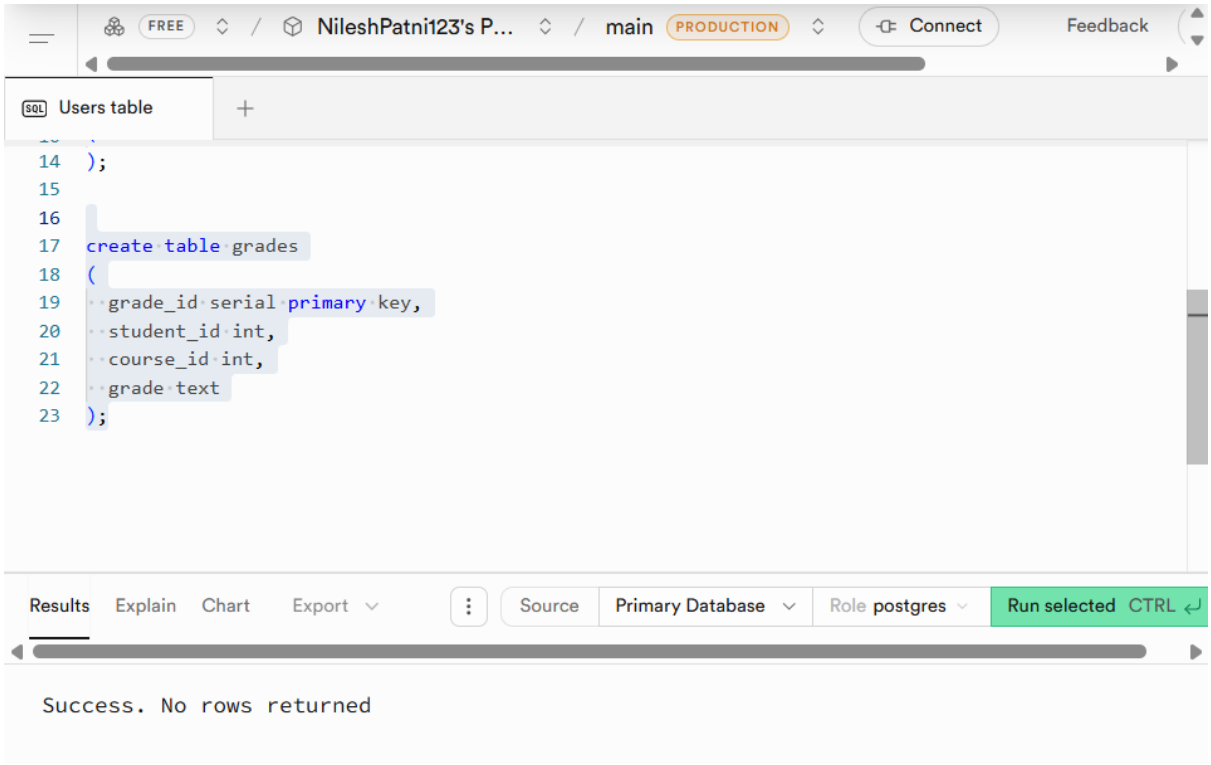
-- Users table :



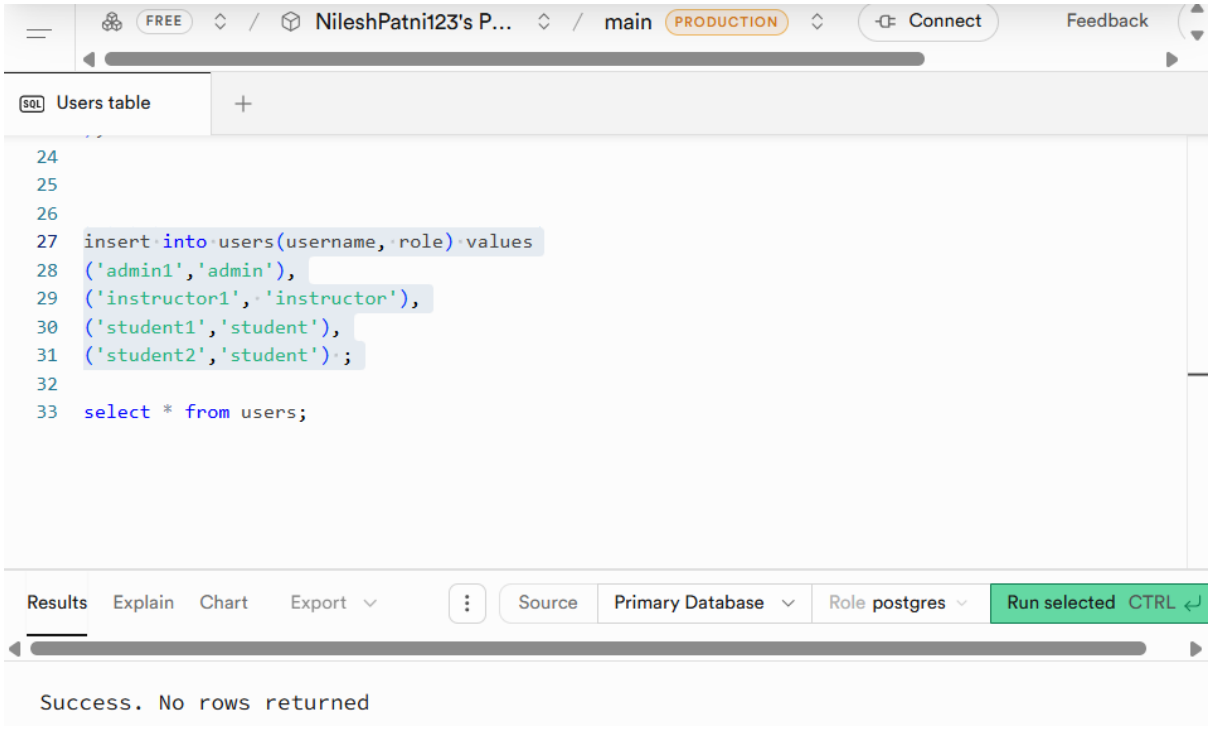
-- Courses table :

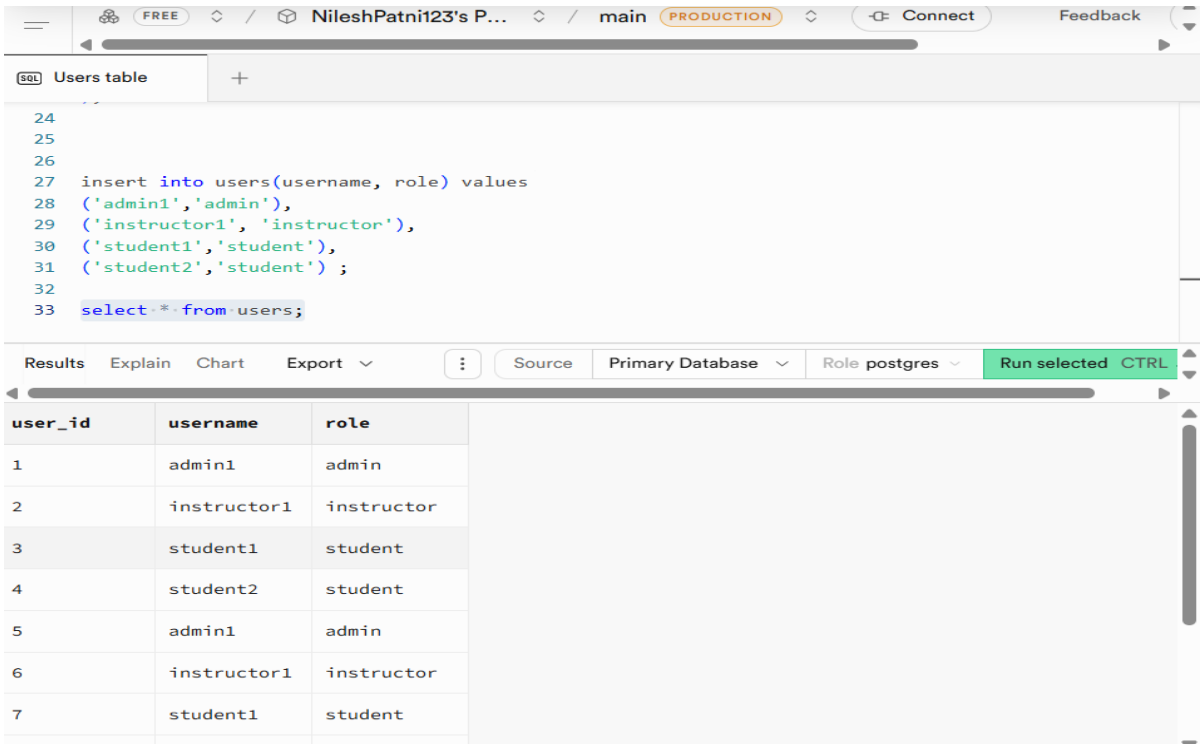


-- Grades table (Sensitive) :



Insert Sample Data :



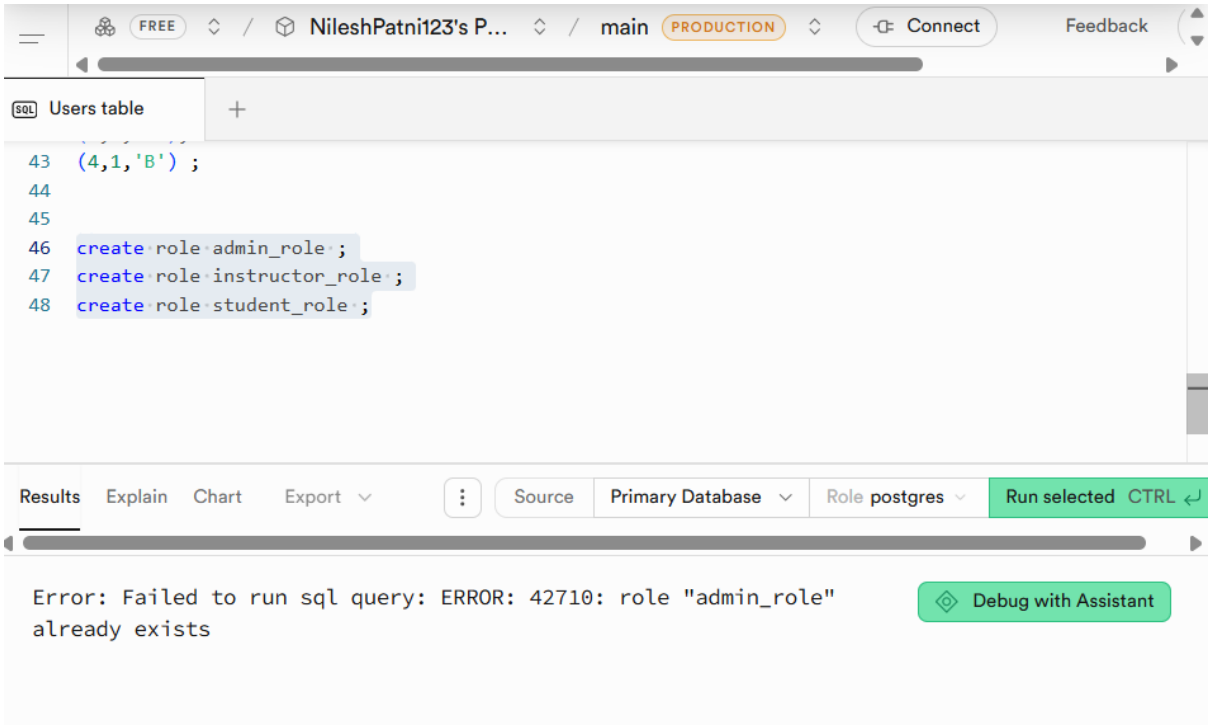


user_id	username	role
1	admin1	admin
2	instructor1	instructor
3	student1	student
4	student2	student
5	admin1	admin
6	instructor1	instructor
7	student1	student


FREE
NileshPatni123's P...
main
PRODUCTION
Connect
Feedback
Users table
33 select * from users;
34
35
36
37 insert into courses(course_name, instructor_id)
38 values ('Machine Learning',2) ;
Results
Explain
Chart
Export
Source
Primary Database
Role postgres
Run selected CTRL
Success. No rows returned


FREE
NileshPatni123's P...
main
PRODUCTION
Connect
Feedback
Users table
38 values ('Machine Learning',2) ;
39
40
41 insert into grades(student_id, course_id, grade) values
42 (3,1,'A'),
43 (4,1,'B') ;
Results
Explain
Chart
Export
Source
Primary Database
Role postgres
Run selected CTRL
Success. No rows returned

Create Roles (RBAC) :



Grant Privileges to Roles :

Admin – Full access

Instructor – View courses, update grades

```
grant select on courses to instructor_role;
grant select , update on grades to instructor_role;
```

Results Explain Chart Export Source Primary Database Role postgres Run selected CTRL

```
Success. No rows returned
```

Student – Read-only access

```
grant select on courses to student_role;
grant select on grades to student_role;
```

Results Explain Chart Export Source Primary Database Role postgres Run selected CTRL

```
Success. No rows returned
```

Test Role-Based Access Control :

Instructor Test :

```
SET ROLE instructor_role;
SELECT * FROM grades;
UPDATE grades SET grade = 'A+' WHERE student_id = 3;
```

Results Explain Chart Export Source Primary Database Role postgres Run selected CTRL

```
Error: Failed to run sql query: ERROR: 42501: permission denied to
set role "instructor_role"
```

Debug with Assistant

```
64
65  SET ROLE instructor_role;
66  SELECT * FROM grades;
67  UPDATE grades SET grade = 'A+' WHERE student_id = 3;
68
```

Results | Explain | Chart | Export | Source | Primary Database | Role postgres | Run selected CTRL

grade_id	student_id	course_id	grade
1	3	1	A
2	4	1	B

```
64
65  SET ROLE instructor_role;
66  SELECT * FROM grades;
67  UPDATE grades SET grade = 'A+' WHERE student_id = 3;
68
```

Results | Explain | Chart | Export | Source | Primary Database | Role postgres | Run selected CTRL

Success. No rows returned

Student Test :

```
72
73  set role student_role;
74  select * from courses;
```

Results | Explain | Chart | Export | Source | Primary Database | Role postgres | Run selected CTRL

Error: Failed to fetch (api.supabase.com)

Debug with Assistant

```
70  delete from grades ;
71
72
73  set role student_role;
74  select * from courses;
```

Results Explain Chart Export Source Primary Database Role postgres Run selected CTRL

course_id	course_name	instructor_id
1	Machine Learning	2

```
76
77  UPDATE grades SET grade='A' WHERE student_id=3;
78  select * from grades;
79
```

Results Explain Chart Export Source Primary Database Role postgres Run selected CTRL

Success. No rows returned

Enable Row-Level Security (RLS) :

```
80
81
82  ALTER TABLE grades ENABLE ROW LEVEL SECURITY;
83
```

Results Explain Chart Export Source Primary Database Role postgres Run selected CTRL

Success. No rows returned

Student-Specific Grade Access (RLS Policy) :

Students can see only their own grades

```
CREATE POLICY student_grade_policy
ON grades
FOR SELECT
TO student_role
USING (student_id = current_setting('app.current_student')::INT);
```

Results Explain Chart Export | Source: Primary Database | Role: postgres | Run selected CTRL

```
Success. No rows returned
```

Set current student context:

```
SET app.current_student = '3';
```

Results Explain Chart Export | Source: Primary Database | Role: postgres | Run selected CTRL

```
Success. No rows returned
```

Student sees only his/her row :

```
SET ROLE student_role;
SELECT * FROM grades;
```

Results Explain Chart Export | Source: Primary Database | Role: postgres | Run selected CTRL

```
Success. No rows returned
```

Instructor RLS Policy (Full Access) :

```
100
101 CREATE POLICY instructor_grade_policy
102 ON grades
103 FOR ALL
104 TO instructor_role
105 USING (true);
106
```

Results Explain Chart Export | Source | Primary Database | Role postgres | Run selected CTRL

Success. No rows returned

```
108 insert into grades(student_id, course_id, grade) values
109 (3,1,'A'),
110 (4,1,'B') ;
111 select * from grades;
112
113 UPDATE grades SET grade='A' WHERE student_id=3;
114 select * from grades;
```

Results Explain Chart Export | Source | Primary Database | Role postgres | Run selected CTRL

grade_id	student_id	course_id	grade
4	4	1	B
6	4	1	B
3	3	1	A
5	3	1	A

Conclusion : -

In this experiment, Role-Based Access Control (RBAC) and Row-Level Security (RLS) were successfully implemented for the Crowdsourced Traffic Incident Reporting System. Different roles were created and assigned appropriate privileges to control access to sensitive data. Row-level security ensured that users could access only their own incident and feedback records. This experiment demonstrates effective database security implementation and highlights the importance of access control in real-world database applications.

---------- END ----------