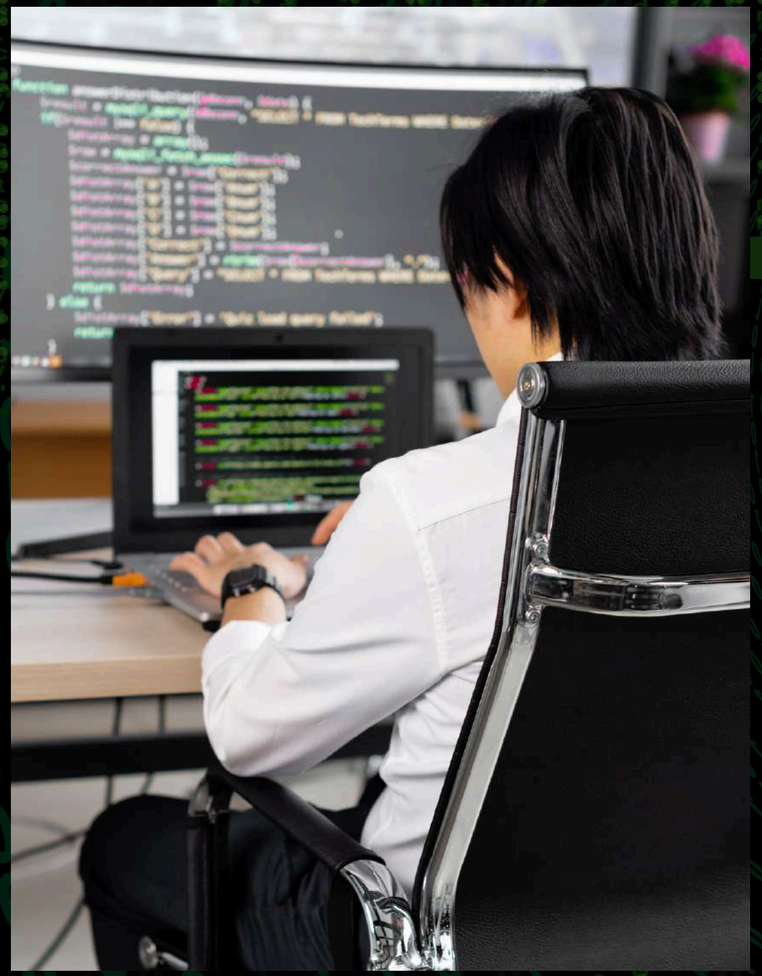
# PROJECT P0

# SQL INJECTION ATTACK

GUIDED BY:-
MR.ZAKIR HUSSAIN

PREPARED BY:-
SRIKESH PUJAR

# TABLE OF CONTENT



# 1. WHAT IS SQL INJECTION ?

It is a cyber attack where malicious SQL code is injected into a web application’s database.

Common attack goals include:

1)Extracting sensitive data.

2)Modifying or deleting data.

3)Gaining unauthorized acccess.

# 2. TYPES OF SQL INJECTION ATTACKS

1 **Classic SQL Injection**: .

Classic SQL Injection involves injecting SQL code into input fields like login forms, allowing attackers to interact directly with the database, retrieve data, or make modifications.

Example:

```
SELECT * FROM users WHERE username = 'admin' -- ' AND password =     'password';
```

2 **Blind SQL Injection**: Attacker infers data through conditional responses.

Example: Checking if a condition is true or false by making the application behave differently.

Query:

```
SELECT * FROM users WHERE id = 1 AND 1=1; -- True
SELECT * FROM users WHERE id = 1 AND 1=2; -- False
```

Result: The attacker infers whether the condition is true or false by how the application responds.

3 **Time-based SQL Injection**: Exploiting the time it takes the database to respond.

Example: Using the SLEEP() function to cause a delay if a condition is true.

Query:

```
SELECT * FROM users WHERE IF(username='admin', SLEEP(5), 0);
```

Result: If the username is 'admin', the response will be delayed by 5 seconds, confirming the condition.

## 4 **Out-of-band SQL Injection:** Sending data to an external server.

Out-of-band SQL Injection uses external channels like HTTP or DNS to send data, effective when database responses are restricted.

Example:

The attacker may use a DNS-based approach to send sensitive data to a remote server.

# 3. Preventing SQL Injection Attacks

Key Prevention Techniques:

1) Prepared Statements

2) Parameterized Queries.

3)Input Validation and Sanitization.

4) Use of ORMs (Object-Relational Mappers).

5) Escaping Special Characters.

6) Database Privilege Management: Implement the least privilege principle.

# 4. Database and Table Structure Overview

## Database 1

User_Authentication: Stores user-related information and authentication details.

## Tables

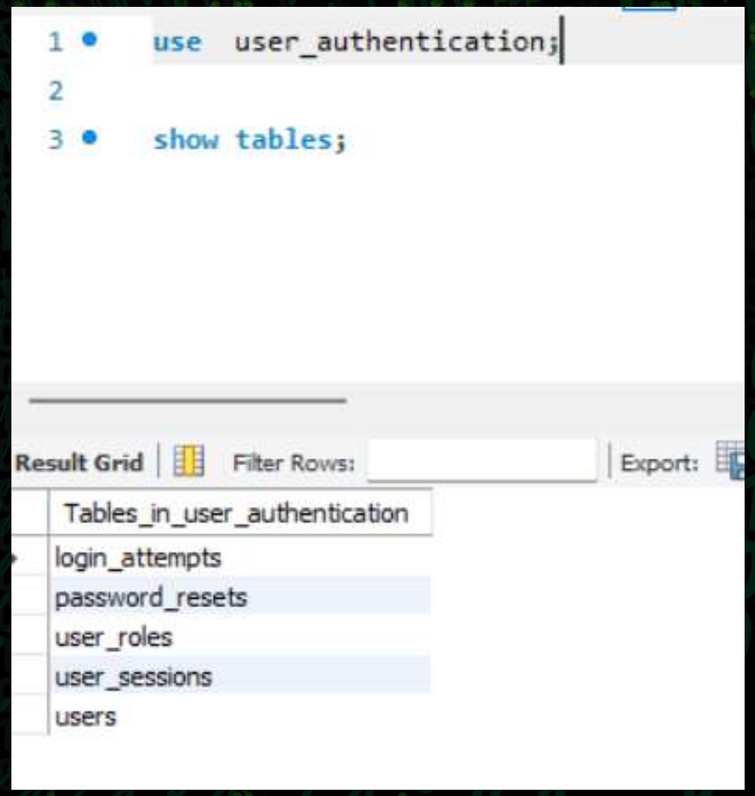
1)users: Contains user credentials like ID, username, password, and email.

2)user_roles: Stores user roles with role names and descriptions.

3)login_attempts: Tracks login attempts with username, date, and success status.

4)user_sessions: Logs user sessions including session start and end times.

5)password_resets: Keeps records of password reset attempts and associated tokens.

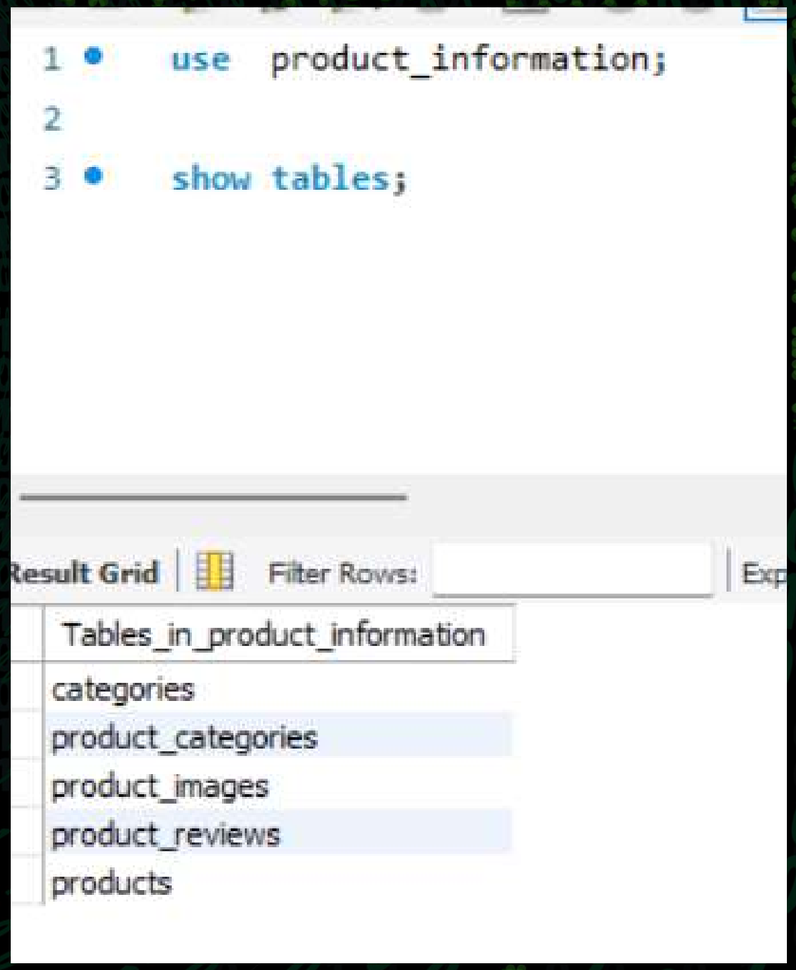


## Database 2

Product_Information: Holds product data along with categories, reviews, and images

## Tables

1)products: Stores product details like ID, name, description, and price.
2)categories: Holds category information with category names and descriptions.
3)product_categories: Links products to categories with references to product and category IDs.
4)product_reviews: Tracks reviews for products with review dates and ratings.
5)product_images: Stores URLs of product images linked to product IDs

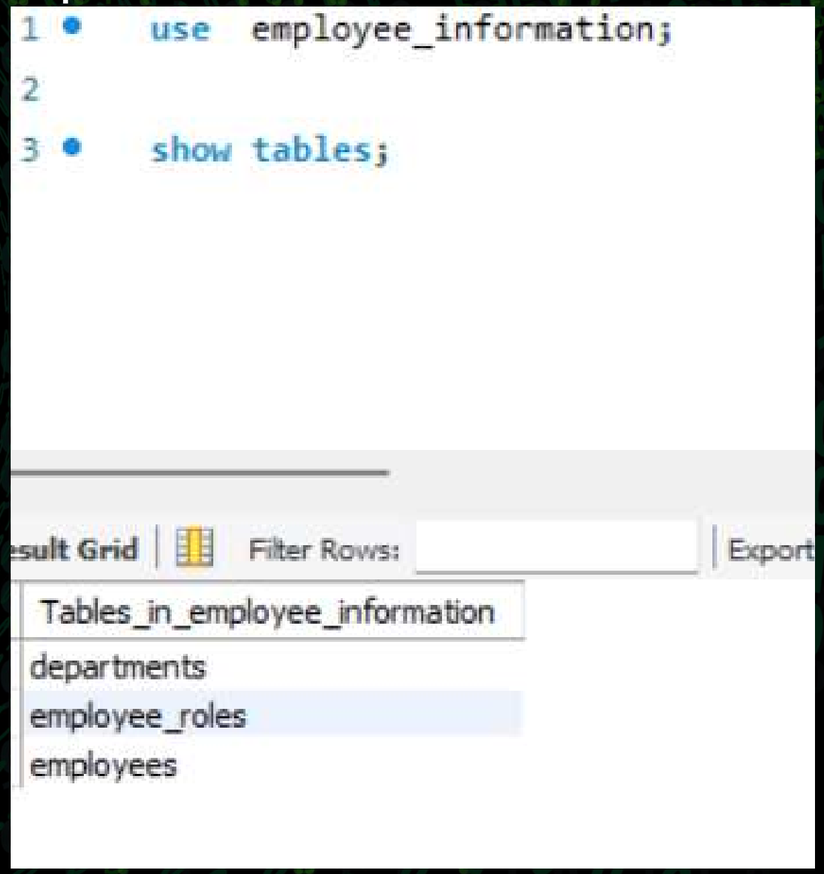


## Database 3

Employee_Information: Contains records of employees and their respective roles.

## Tables

1) employees: Contains employee details including name, department, and salary.
2) departments: Stores department information such as names and descriptions.
3) employee_roles: Defines employee roles along with role names and descriptions

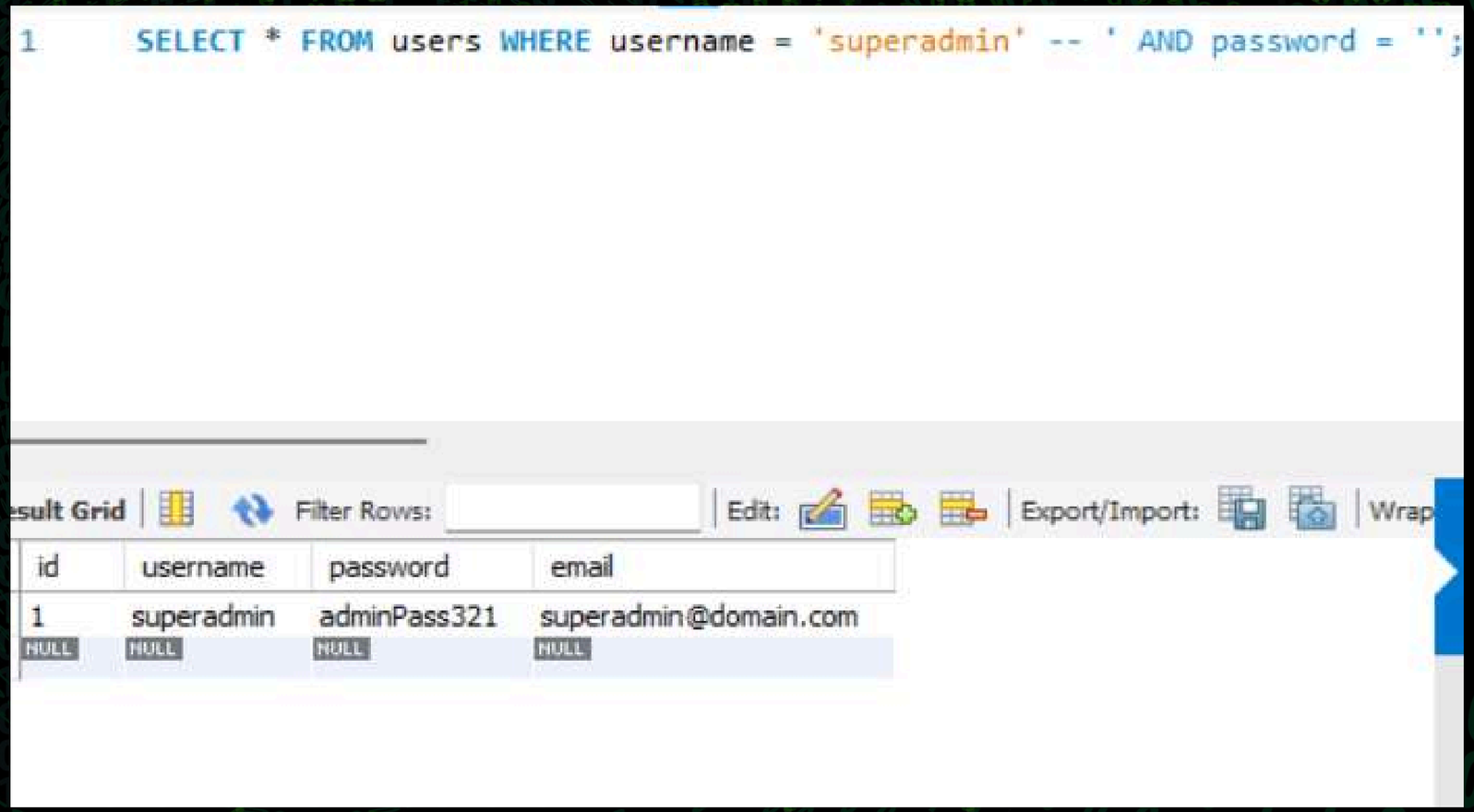


# 5. Executions and Results

**1) Simple SQL Injection Results**:

After executing the simple SQL Injection query:
SELECT * FROM users WHERE username = 'admin' -- ' AND password = '';

You will notice that the query logs in as admin without needing the actual password. This confirms that the application is vulnerable to SQL Injection.

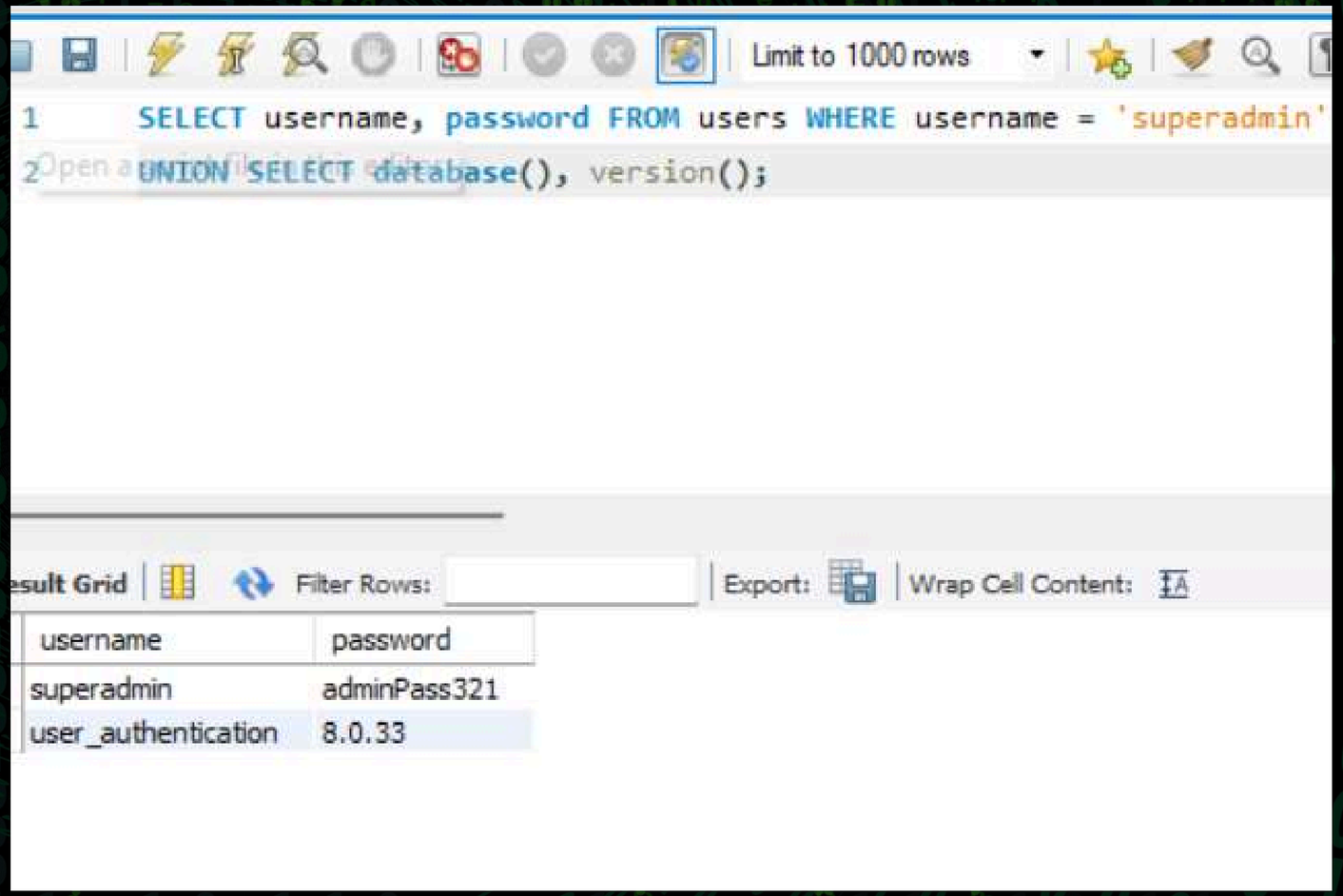


## 2) Union-Based Injection Results

The union-based SQL Injection query:

SELECT username, password FROM users WHERE username = 'superadmin' UNION SELECT database(), version();

should return the username and password of the admin user, as well as information about the database and MySQL version.

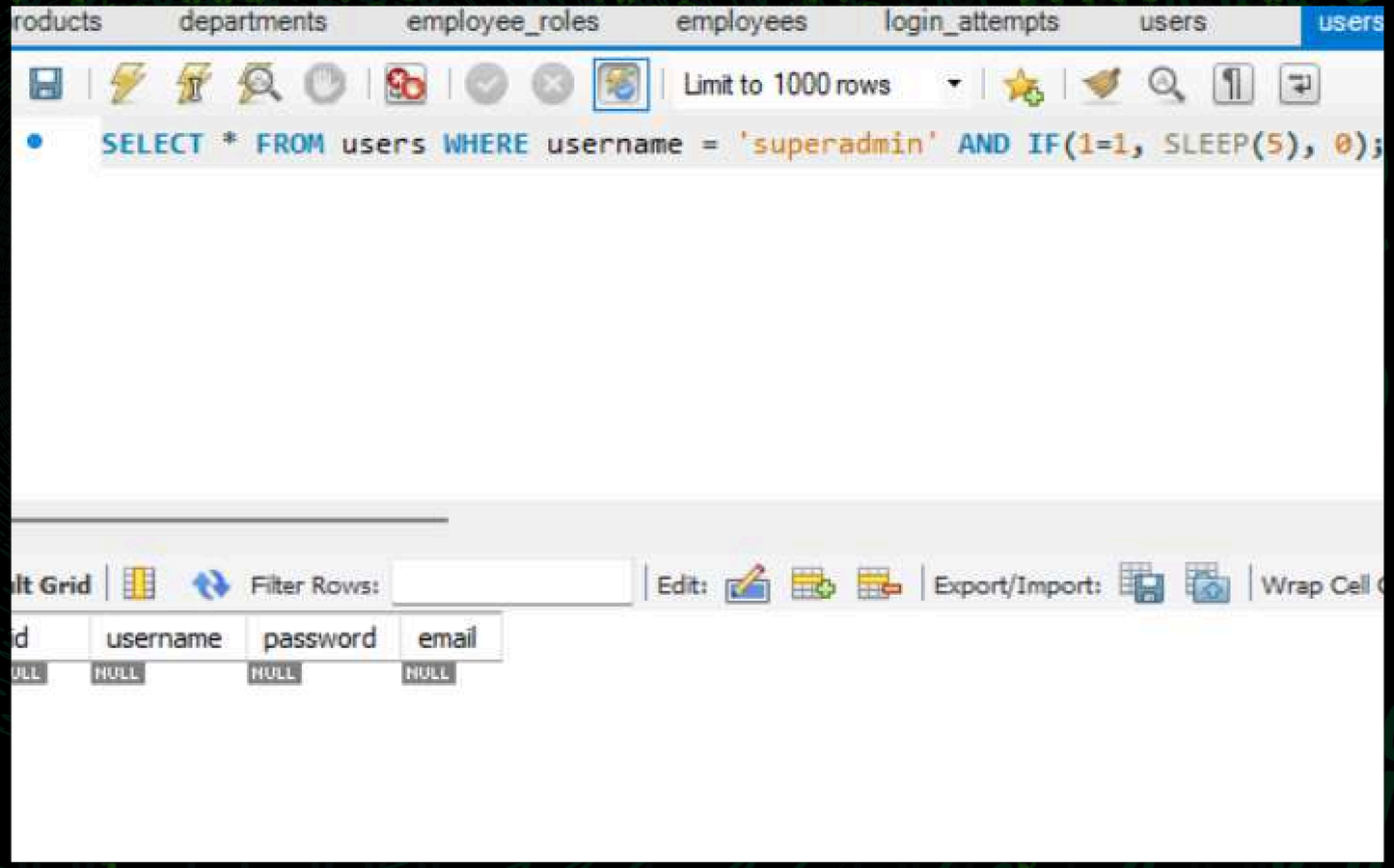


## 3) Blind SQL Injection Results:

When executing the blind SQL Injection query:

SELECT * FROM users WHERE username = 'admin' AND IF(1=1, SLEEP(5), 0);

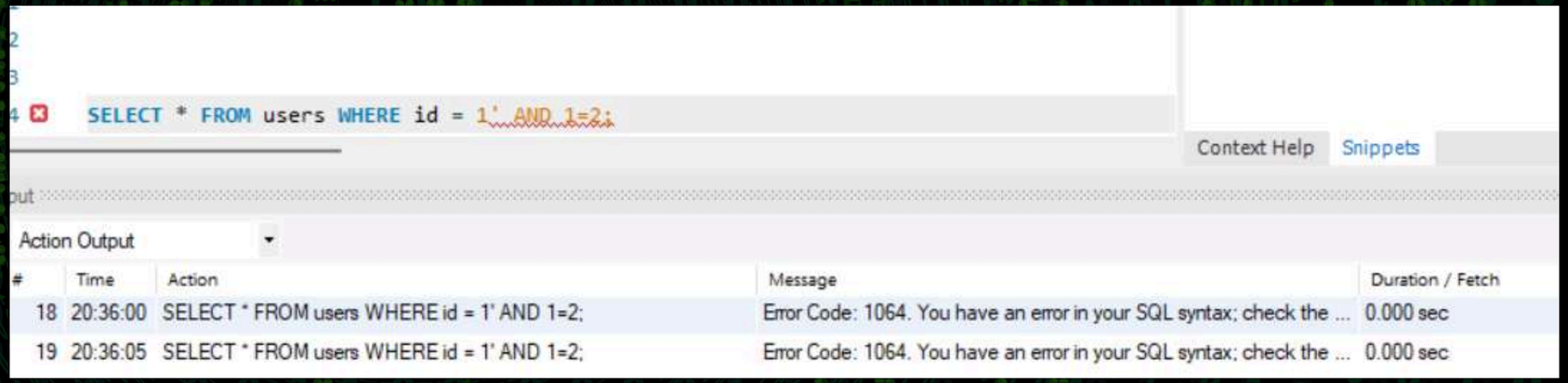
you will observe that the query causes a delay in the application's response. This proves that the attacker can infer information from the database without directly seeing query results.



## 4) Error-Based SQL Injection Results:

When running the error-based query:
SELECT * FROM users WHERE id = 1' AND 1=2;

you might receive an error message revealing the internal structure of the database.



# 6. Conclusion

- SQL Injection is still one of the most widespread vulnerabilities, allowing attackers to execute unauthorized queries with severe consequences like data theft and system compromise.
- Using prepared statements and parameterized queries is a key defense, ensuring that user inputs are treated as data rather than executable SQL code.
- Proper input validation and escaping of special characters are crucial to minimizing the risk of SQL Injection by filtering and sanitizing user data.
- Limiting database access rights can significantly reduce the impact of a successful attack, restricting the potential damage.
- Ongoing vigilance with tools like vulnerability scanners, code reviews, and security best practices is essential to safeguard applications from SQL Injection threats.