Foundations of Information Assurance

Lab Assignment 8 Report

Team 20

Sonam Ghatode

Vishal Maurya

11.20.2019
CY5010

Index

Section 1 : Introduction to databases and mysql

For this part of the lab, we will create a new database called “lab” and use it to store some data.

Screen Capture 1 → Take a screenshot of the output(for “show databases”):

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql> CREATE DATABASE lab
    -> ;
Query OK, 1 row affected (0.01 sec)

mysql>
```

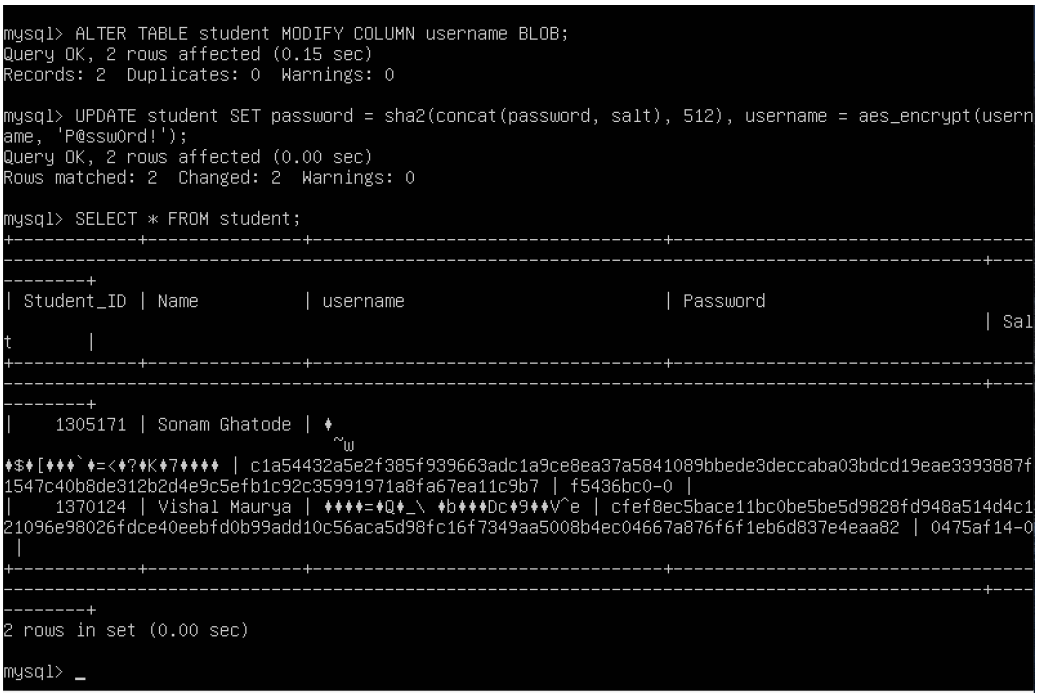
Screen Capture 2 → Take a screenshot of the student table(for “select * from student”)

Execute the command “mysql> SELECT * FROM student;” to verify that the new row has been successfully inserted into the table.

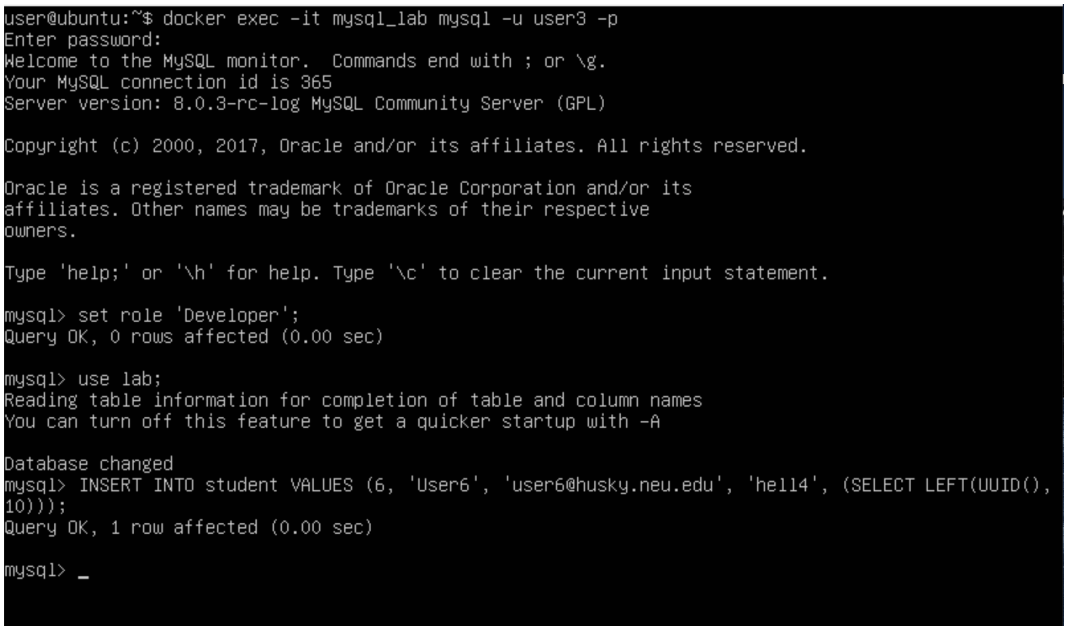
```
mysql> SELECT * FROM student;
+------------+--------------+-------------------------+----------+------------+
| Student_ID | Name         | Username                | Password | Salt       |
+------------+--------------+-------------------------+----------+------------+
|          1 | Sonam Ghatode | ghatode.s@husky.neu.edu | test     | 8f827739-0 |
|          2 | Vishal Maurya | maurya.v@husky.neu.edu  | 123      | ca53040c-0 |
+------------+--------------+-------------------------+----------+------------+
2 rows in set (0.00 sec)

mysql> _
```

Screen Capture 3 → Take a screenshot of the student table(for checking “username and password columns have been encrypted or not”)

Use a ‘trigger’ to ensure attribute encryption for all the rows inserted into “student” table.



Screen Capture 4 → Take a screenshot showing ‘insert’ commands and the output of student table:

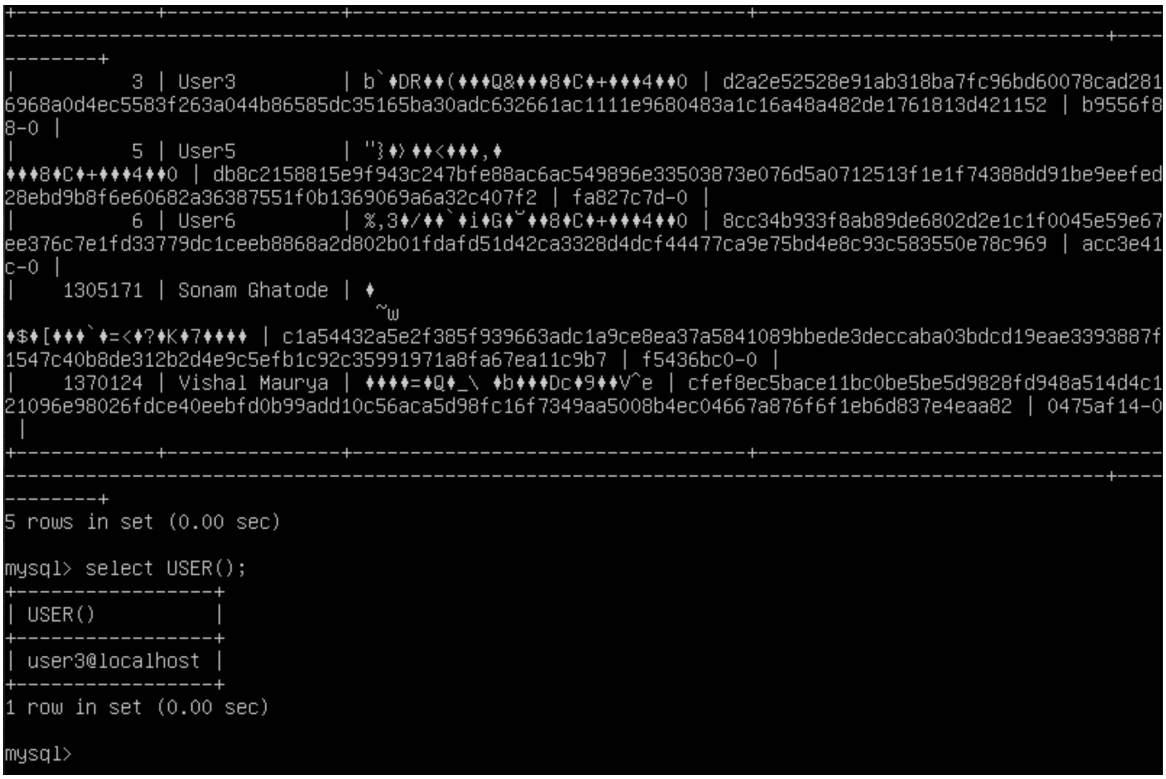


```
mysql> INSERT INTO student VALUES (6, 'User6', 'user6@husky.neu.edu', 'hell4', (SELECT LEFT(UUID(),
10)));
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM student;
+------------+--------------+--------------------------------+--------------------------------
--------------------------------------------------------------------------------------------+----
--------+
| Student_ID | Name         | username                       | Password
                                                                                           | Sal
t       |
+------------+--------------+--------------------------------+--------------------------------
--------------------------------------------------------------------------------------------+----
--------+
|          3 | User3        | b`♦DR♦♦(♦♦♦Q&♦♦♦8♦C♦+♦♦♦4♦♦0 | d2a2e52528e91ab318ba7fc96bd60078cad281
6968a0d4ec5583f263a044b86585dc35165ba30adc632661ac1111e9680483a1c16a48a482de1761813d421152 | b9556f8
8-0 |
|          5 | User5        | "}♦)♦♦<♦♦♦,♦
♦♦♦8♦C♦+♦♦♦4♦♦0 | db8c2158815e9f943c247bfe88ac6ac549896e33503873e076d5a0712513f1e1f74388dd91be9eefed
28ebd9b8f6e60682a36387551f0b1369069a6a32c407f2 | fa827c7d-0 |
|          6 | User6        | %,3♦/♦♦`♦i♦G♦~♦♦8♦C♦+♦♦♦4♦♦0 | 8cc34b933f8ab89de6802d2e1c1f0045e59e67
ee376c7e1fd33779dc1ceeb8868a2d802b01fdafd51d42ca3328d4dcf44477ca9e75bd4e8c93c583550e78c969 | acc3e41
c-0 |
|    1305171 | Sonam Ghatode | ♦
                                    ^w
♦$♦[♦♦♦`♦=<♦?♦K♦7♦♦♦♦ | c1a54432a5e2f385f939663adc1a9ce8ea37a5841089bbede3deccaba03bdcd19eae3393887f
1547c40b8de312b2d4e9c5efb1c92c35991971a8fa67ea11c9b7 | f5436bc0-0 |
|    1370124 | Vishal Maurya | ♦♦♦♦=♦Q♦_\ ♦b♦♦♦Dc♦9♦♦V^e | cfef8ec5bace11bc0be5be5d9828fd948a514d4c1
21096e98026fdce40eebfd0b99add10c56aca5d98fc16f7349aa5008b4ec04667a876f6f1eb6d837e4eaa82 | 0475af14-0
 |
+------------+--------------+--------------------------------+--------------------------------
--------------------------------------------------------------------------------------------+----
--------+
5 rows in set (0.00 sec)
```

Screen Capture 5 → Take a screenshot of the output(for SHOW GRANTS FOR 'user3' USING 'Developer';):

```
mysql> CREATE ROLE 'Developer';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON lab.student TO 'Developer';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE USER 'user3' IDENTIFIED BY 'User3Secret';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT 'Developer' TO 'user3';
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'user3' USING 'Developer';
+-------------------------------------------------------------------------+
| Grants for user3@%                                                      |
+-------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `user3`@`%`                                       |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `lab`.`student` TO `user3`@`%` |
| GRANT `Developer`@`%` TO `user3`@`%`                                    |
+-------------------------------------------------------------------------+
3 rows in set (0.00 sec)

mysql> _
```

Screen Capture 6 → Take a screenshot of the output showing student table and the current MYSQL user

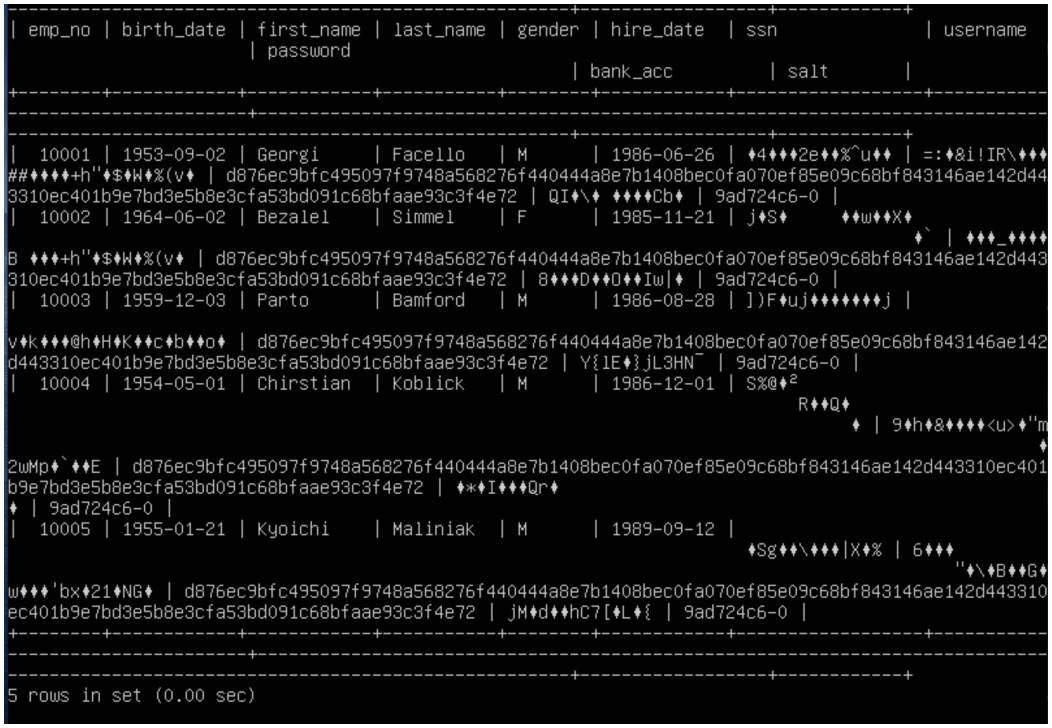


Section 2 : Hardening MySQL Database

Screen Capture 7 → Take a screenshot of the output of (show columns from employees) command

```
mysql> show columns from employees;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| emp_no     | int(11)       | NO   | PRI | NULL    |       |
| birth_date | date          | NO   |     | NULL    |       |
| first_name | varchar(14)   | NO   |     | NULL    |       |
| last_name  | varchar(16)   | NO   |     | NULL    |       |
| gender     | enum('M','F') | NO   |     | NULL    |       |
| hire_date  | date          | NO   |     | NULL    |       |
| ssn        | blob          | NO   |     | NULL    |       |
| username   | blob          | NO   |     | NULL    |       |
| password   | blob          | NO   |     | NULL    |       |
| bank_acc   | blob          | NO   |     | NULL    |       |
| salt       | varchar(10)   | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
11 rows in set (0.00 sec)

mysql> _
```

Screen Capture 8 → Take a screenshot of the output of (SELECT * FROM employees ORDER BY emp_no DESC LIMIT 5;) command showing protected values of ssn, username, bank account and password:



Link to Submitted image

vishalpmaurya/cy5010_database_lab_team_20

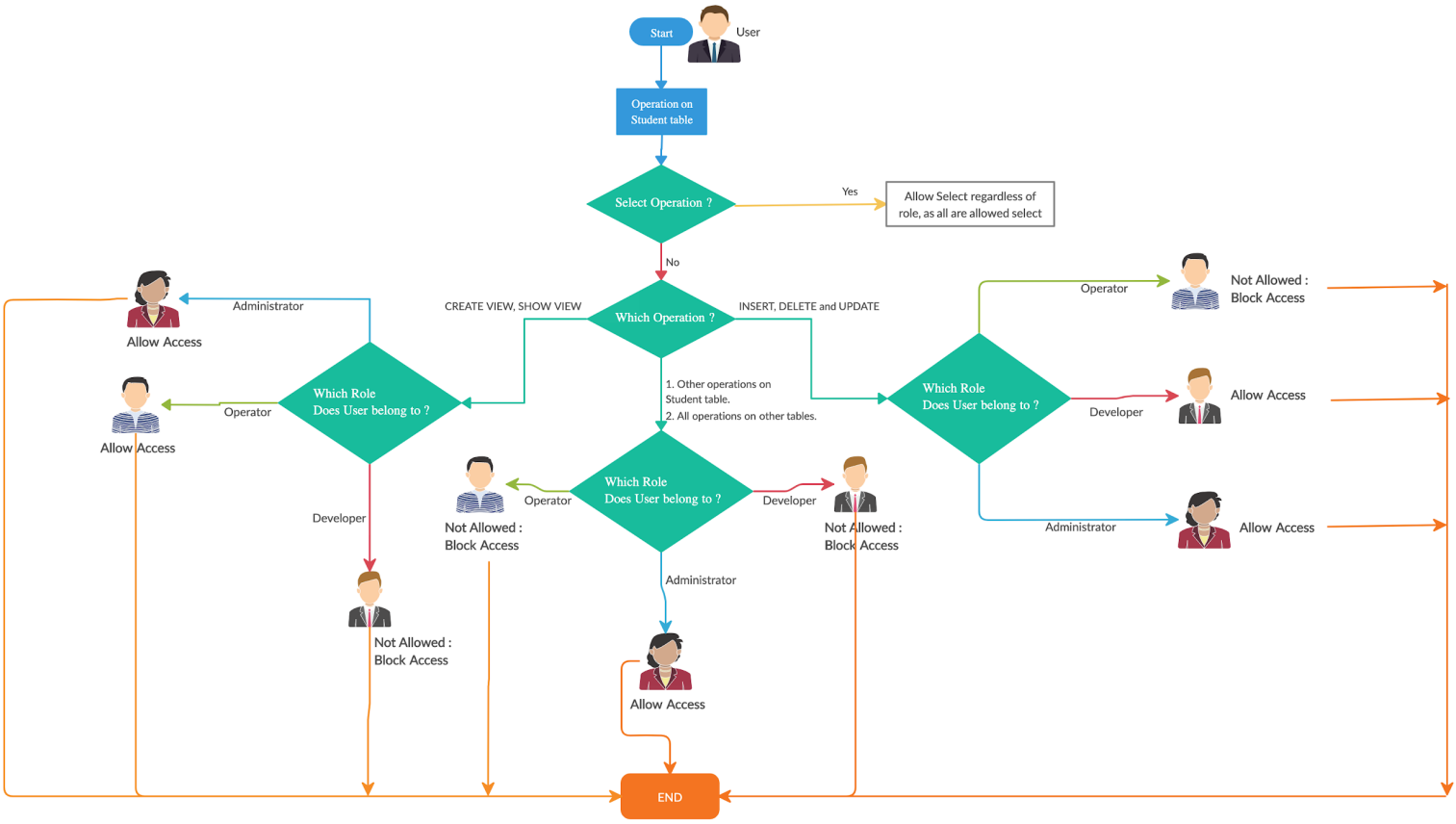
To download the container use :

docker pull vishalpmaurya/cy5010_database_lab_team_20

```
root@ubuntu:~# docker image list
REPOSITORY            TAG                 IMAGE ID            CREATED             SIZE
cy5010/database_lab   latest              383156d9c48d        2 years ago         1.47GB
root@ubuntu:~# docker ps -all
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS
            PORTS                 NAMES
a739f96cc543        cy5010/database_lab   "/entrypoint.sh mysq…"   2 days ago          Up 5 minutes
(healthy)   3306/tcp, 33060/tcp   mysql_lab
root@ubuntu:~# docker commit a739f96cc543 cy5010_database_lab_team_20
sha256:ffca8b77f21e1ddc9749ec7dd6c7e2cd772c7ecc40d79d8bb0af3ed4ef948c21
root@ubuntu:~#
```

```
root@ubuntu:~# docker image list
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
cy5010_database_lab_team_20   latest              ffca8b77f21e        5 minutes ago       1.71GB
cy5010/database_lab           latest              383156d9c48d        2 years ago         1.47GB
root@ubuntu:~# docker commit a739f96cc543 vishalpmaurya/cy5010_database_lab_team_20
sha256:0cfcc089a02aaf8d12f49bcd22a97e0b04705074335712af7e552081f03eee29
root@ubuntu:~# docker image list
REPOSITORY                                TAG                 IMAGE ID            CREATED
    SIZE
vishalpmaurya/cy5010_database_lab_team_20   latest              0cfcc089a02a        3 seconds ago
    1.71GB
cy5010_database_lab_team_20               latest              ffca8b77f21e        6 minutes ago
    1.71GB
cy5010/database_lab                       latest              383156d9c48d        2 years ago
    1.47GB
root@ubuntu:~# docker push vishalpmaurya/cy5010_database_lab_team_20:latest
The push refers to repository [docker.io/vishalpmaurya/cy5010_database_lab_team_20]
8523360cf3fa: Pushed
cdfb40237ff4: Pushed
0c39b2c234c8: Pushed
5fc883743fe6: Pushed
5c2f7c352a9c: Pushed
0302be4b1718: Pushed
latest: digest: sha256:386849a7b3f281c3ad657fe18772c681c95ec02cae1aed9469a8875dbc03fa99 size: 1581
root@ubuntu:~#
```

Simple flow chart to explain the role-based access control of users ‘user1’, ‘user2’ and ‘user3’



Due to the amount of contents in the Image, it is looking small. You can view the image online or download the image from this link :

https://vishalm-projects.s3.amazonaws.com/Flow_Chart_Database.jpg

Two database attacks and how they can be prevented :

Threats on database security can be divided into two different categories, physical and logical. Revelation of passwords, demolition of storage devices, stealing of important data by hackers constitute physical threats. Usual method to prevent this type of attacks is keep backup of every storage device. Logical threats are unauthorized access to information. These vulnerabilities are exploited by hackers to gain access and modify data or leak sensitive information.

1. Insider Threat:

A legitimate user who has or had access to the confidential information stored in the database is referred to as Insider Threat. Information can easily be transferred through electronic medium, printout or by direct conversation. It is difficult to prevent the data from Insider threat since it is very difficult to know the intentions of all employees working in a firm.

Insider Threats can be prevented by enforcing least privilege policy. A user should have only privileges that are absolutely necessary to carry out normal functioning of his/her job. For example, if an employee is working in sales domain, he does not need access of financial database or vice versa. It is always best if we segregate our databases according to the need and give privileges absolutely necessary to carry out normal operations. Another way is to enforce no media policy inside the workplace, this way electronic media could not be used to leak information.

2. SQL Injection:

SQL injection vulnerabilities occur when application code contains dynamic database queries which directly include user supplied input. For example, if username is being taken as an input and directed directly to a sql query without doing input sanitization, a query can be passed as an input and vulnerability can be exploited by having access to the database.

Preventing SQL injection is relatively straightforward:

1. Avoid the use of dynamic queries within applications. Use of prepared statements with parameterized queries will stop SQL injection.
2. Implement user input validation and sanitization before that input is passed to the application. This is a very worthwhile additional defence which also helps prevent many other attacks.

References:

[1] https://www.bcs.org/content-hub/top-ten-database-attacks/
[2] https://pdfs.semanticscholar.org/7273/d46cc418cdb82a54ad642c74031ba39fa012.pdf