Sql Mitigation Assignment

Steps

- From the description, the database is vulnerable, so tried logging in.
- I successfully logged in using the command: user_id = ‘ or ‘1’=’1’ -- and password = j
- In order to fix this, I parameterized both the user_id and password queries.
- I assigned both parameters to a variable called ‘bind_param’ and passed it to the execute function.
- After which the I was unable to log in using same command as seen in the picture below

```
evidencemonday@evidencemonday-TM1703:~/Documents/Study/Securtiy Testing/sectest_homework_2$ python3 main.py
Welcome to this vulnerable database reader
You have to login first
Insert your user-id
' or '1'='1' --
Insert your password
j

===Logged-in=====

Here is joe123 data:
user-id= joe123
first_name= Joe
last_name= Doe
phone +1234

Here is ferrari data:
user-id= ferrari
first_name= Massimo
last_name= Ferrari
phone +1111

Here is smith data:
user-id= smith
first_name= John
evidencemonday@evidencemonday-TM1703:~/Documents/Study/Securtiy Testing/sectest_homework_2$ python3 solution_main.py
Welcome to this vulnerable database reader
You have to login first
Insert your user-id
' or '1'='1' --
Insert your password
j
Wrong credentials
evidencemonday@evidencemonday-TM1703:~/Documents/Study/Securtiy Testing/sectest_homework_2$
```

Fig.1 output from vulnerable and modified python code.

Note: main.py is the vulnerable python code and solution_main.py is the modified python code.

```
retrieve_user = "SELECT * FROM credentials WHERE user_id = ? and
password = ?;"
bind_param = (user_id, password,)
cursor = conn.execute(retrieve_user, bind_param)

entries = cursor.fetchall()
if len(entries) > 0:
    print("\n===Logged-in=====")
    print_data(user_id)
else:
    print("Wrong credentials")
```

Fig 2. Modified code in solution_main.py