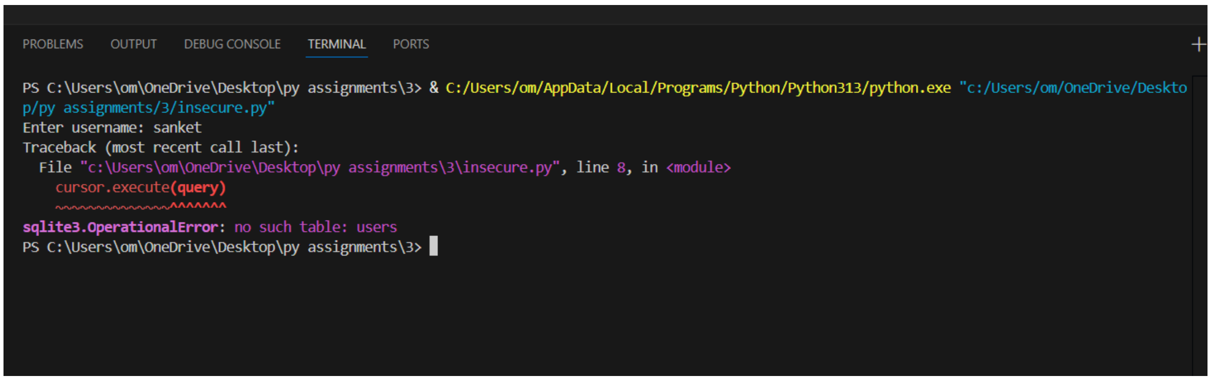
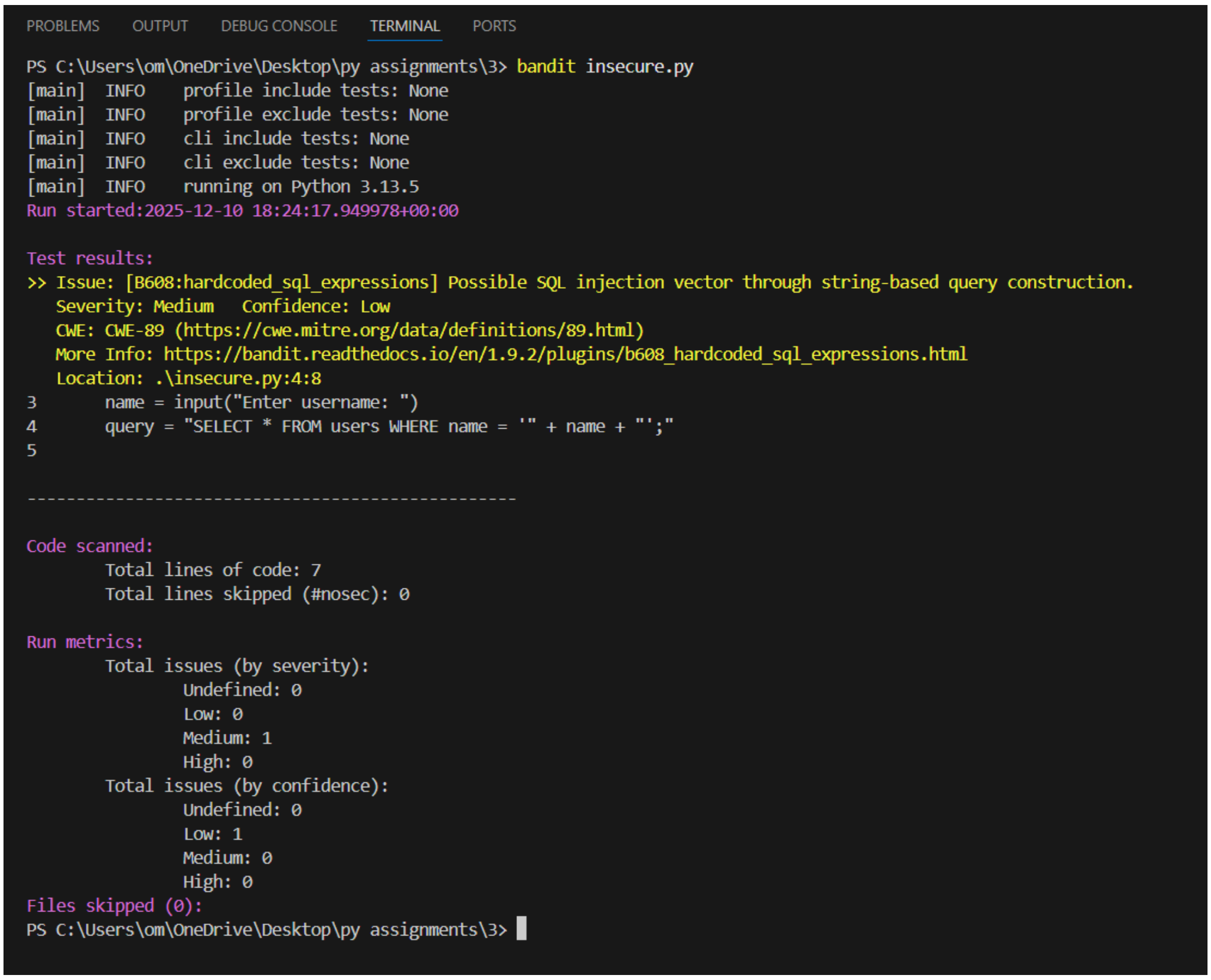
● Use tools like static analyzers or manual inspection methods.

```
import sqlite3

name = input("Enter username: ")

query = "SELECT * FROM users WHERE name = '" + name + "';"  # vulnerable to SQL Injection

conn = sqlite3.connect("test.db")

cursor = conn.cursor()

cursor.execute(query)

print(cursor.fetchall())

conn.close()
```



```
PS C:\Users\om\OneDrive\Desktop\py assignments\3> & C:/Users/om/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/om/OneDrive/Deskto
p/py assignments/3/insecure.py"
Enter username: sanket
Traceback (most recent call last):
  File "c:\Users\om\OneDrive\Desktop\py assignments\3\insecure.py", line 8, in <module>
    cursor.execute(query)
    ~~~~~~~~~~~~~~^^^^^^^
sqlite3.OperationalError: no such table: users
PS C:\Users\om\OneDrive\Desktop\py assignments\3>
```

● Use tools like static analyzers or manual inspection methods.



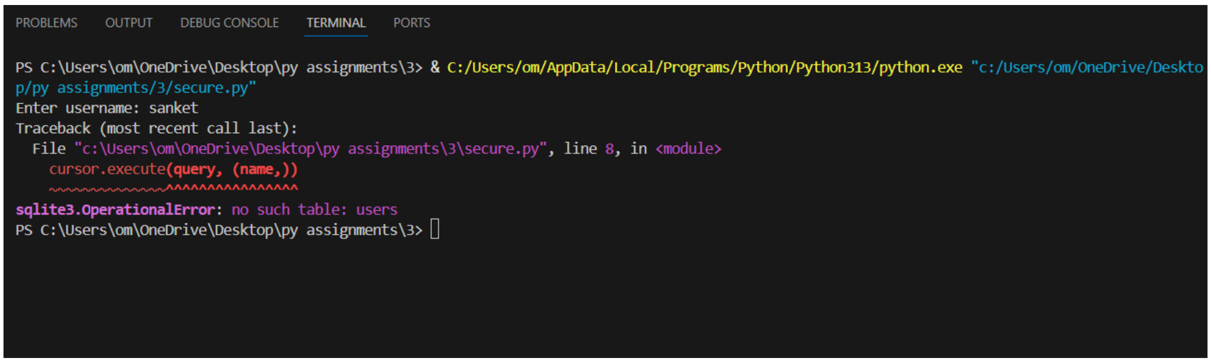
```
PS C:\Users\om\OneDrive\Desktop\py assignments\3> bandit insecure.py
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.13.5
Run started:2025-12-10 18:24:17.949978+00:00

Test results:
>> Issue: [B608:hardcoded_sql_expressions] Possible SQL injection vector through string-based query construction.
   Severity: Medium   Confidence: Low
   CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)
   More Info: https://bandit.readthedocs.io/en/1.9.2/plugins/b608_hardcoded_sql_expressions.html
   Location: .\insecure.py:4:8
3       name = input("Enter username: ")
4       query = "SELECT * FROM users WHERE name = '" + name + "';"
5

--------------------------------------------------

Code scanned:
        Total lines of code: 7
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 1
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 1
                Medium: 0
                High: 0
Files skipped (0):
PS C:\Users\om\OneDrive\Desktop\py assignments\3>
```

● Provide recommendations and best practices for secure coding.

- Use **parameterized queries**
- Validate user inputs
- Avoid string concatenation in SQL
- Use least privilege access
- Use static analysis tools regularly

● Document findings and suggest remediation steps for safer code.

```
import sqlite3


name = input("Enter username: ")

query = "SELECT * FROM users WHERE name = ?"


conn = sqlite3.connect("test.db")

cursor = conn.cursor()

cursor.execute(query, (name,))

print(cursor.fetchall())

conn.close()
```



```
PS C:\Users\om\OneDrive\Desktop\py assignments\3> & C:/Users/om/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/om/OneDrive/Desktop/py assignments/3/secure.py"
Enter username: sanket
Traceback (most recent call last):
  File "c:\Users\om\OneDrive\Desktop\py assignments\3\secure.py", line 8, in <module>
    cursor.execute(query, (name,))
    ~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^
sqlite3.OperationalError: no such table: users
PS C:\Users\om\OneDrive\Desktop\py assignments\3>
```