Practical exercises and questions:

In the following section there will be four exercises about Cross Site Scripting and SQL injection. We suggest you to follow the exercises in the order in which they are presented.

Exercise 1: In the following exercise you will experience what happens if an application includes request data in the immediate response in an unsafe way.
You can find the lab here: https://portswigger.net/web-security/cross-site-scripting/reflected/lab-html-context-nothing-encoded

Objective: The goal of the exercise is to perform a Cross Site Scripting attack that calls the alert function. The only thing you know is that this lab contains a simple reflected Cross Site Scripting vulnerability in the search functionality.

A suggestion in case you are lost: *Try to exploit the search functionality, you might be able to inject a script and make it execute.*

Exercise 2: In the following exercise you will experience what happens if an application includes request data in the next responses in an unsafe way.
You can find the lab here: https://portswigger.net/web-security/cross-site-scripting/stored/lab-html-context-nothing-encoded

Objective: The goal of the exercise is to submit a comment that calls the alert function when a post is viewed. The only thing you know is that this lab contains a simple stored Cross Site Scripting vulnerability in the comment functionality.

A suggestion in case you are lost: *Try to exploit the comment functionality, you might be able to inject a script and make it execute.*

Exercise 3: In the following exercise you will experience what happens if the input of a query is not filtered and the results can be easily manipulated.

You can find the lab here: https://portswigger.net/web-security/sql-injection/lab-retrieve-hidden-data

Objective: The goal of the exercise is to perform a SQL injection attack that causes the application to display details of all products in any cathegory, both released and unreleased. The only thing you know is that this labs contains a SQL injection vulnerability in the product category filter. Keep in mind that a query when selecting a category looks something like this: SELECT * FROM products WHERE category = 'Gifts' AND released = 1

A suggestion in case you are lost: *Try to intercept the legitimate query and understand how it works. You might be able to modify it to make the application give different results.*

Exercise 4: In the following exercise you will experience what happens if the login function is vulnerable and can be easily manipulated.
You can find the lab here: https://portswigger.net/web-security/sql-injection/lab-login-bypass

Objective: The goal of the exercise is to perform a SQL injection attack that logs you into the application as the administrator user. The only thing you know is that this lab contains a SQL injection vulnerability in the login function.

A suggestion in case you are lost: *Try to intercept the legitimate request and mess with the parameters. You might be able to modify those to get administrator access.*

Questions:
In this little section you will find some simple questions to test your comprehension of the topic.

1. What is the main vehicle for injection attacks?

2. Which is the main cause of injection attacks?

3. Briefly explain how an XSS attack can occur.

4. Briefly explain the possible consequences of a SQL injection attack.

5. How can SQLi vulnerability be detected?