



**NAME:** Mary Franxine G. Nicol  
**COURSE, YEAR, &SECTION:** BSIS 2A

**SUBJECT:** WEB SYSTEM  
**PROFESOR:** Reymark Llagas

Scenario	Problem	Correct Answer
1	Using \$_POST even though the ID comes from the URL.	Use \$_GET['id'] and always check if it exists before using it.
2	Missing quotes around a string in SQL (first_name = \$fname).	Put the name inside quotes: first_name = '\$fname'.
3	Raw GET age value creates SQL injection risk.	Use prepared statements and bind the age parameter.
4	Inserting without validation—blank fields can be inserted.	Check if first and last name are not empty before inserting.
5	Typo in POST key (emial).	Use the correct POST key: email.
6	DELETE uses raw GET input. High risk of deleting everything.	Use intval(\$_GET['id']) or prepared statements.
7	Missing quotes around email + no error checking.	Wrap \$email in quotes and check if query succeeded.
8	Only one row printed because no while loop.	Use a while (mysqli_fetch_assoc()) loop to display all.
9	PHP reads POST but link sends GET.	Change to \$_GET['id'].
10	Wrong variable name used (\$aeg).	Use the correct variable \$age.
11	Form sends GET but PHP expects POST.	Match both to GET or both to POST.
12	Numeric ID placed inside quotes.	Remove quotes or cast \$id to (int).
13	UPDATE has no WHERE clause.	Add WHERE student_id = ? so only one row updates.
14	POST array fields not indexed correctly + missing quotes.	Use \$data['first_name'] and put all strings inside quotes.
15	Page number not validated, can break database.	Convert to integer and limit allowed values.

## EXPLANATION

### Scenario 1

The code used \$\_POST even though the ID comes from the URL. GET dapat nasa URL yung id, kaya ginamit \$\_GET para mabasa ng maayos at para maiwasan yung undefined index.



## Scenario 2

Strings in SQL must always be inside quotes. Pag hindi naka-quote, iniisip ng SQL na column name siya kaya lumalabas yung “unknown column” error.

## Scenario 3

Directly plugging GET values into SQL is super risky dahil pwede ito mag SQL injection. Prepared statements ensure na sanitized and safe yung input.

## Scenario 4

If walang validation, blank values or empty rows mapupunta sa database. Kaya dapat icheck muna if may laman bago mag-insert.

## Scenario 5

If mali spelling ng POST key, hindi makukuha ni PHP yung value. Kaya lumalabas undefined index. Correcting the key fixes it.

## Scenario 6

Using raw GET inside DELETE is dangerous pwede gamitin ng user para i-delete lahat ng records. Always cast the ID to integer or use a prepared statement.

## Scenario 7

Missing quotes causes SQL error, pero dahil walang error checking, “Updated!” pa rin ang output. Need to add proper validation to know if the query failed.

## Scenario 8

Mysqli\_fetch\_assoc() gets only 1 row kaya isa lang nalalabas. Using a while loop allows all data to be printed.

## Scenario 9

Link sends GET pero PHP nagbabasa ng POST, kaya undefined index. Dapat mag-match sila.

## Scenario 10

Wrong variable name breaks the SQL. Fixing \$aeg to \$age solves the problem.

## Scenario 11

Form uses GET but PHP reads POST, kaya walang nakuha. Both must use the same method.

## Scenario 12

ID is a number, so dapat walang quotes. Cleaner and safer pag integer cast.

## Scenario 13

No WHERE clause means buong table ma-uupdate. Adding WHERE restricts the update to one student.



REPUBLIC OF THE PHILIPPINES  
**BICOL UNIVERSITY**  
**POLANGUI**  
Polangui, Albay  
Email: [bupc-dean@bicol-u.edu.ph](mailto:bupc-dean@bicol-u.edu.ph)  




#### Scenario 14

POST array must be accessed properly (`$data['first_name']`), and SQL strings must be quoted correctly.

#### Scenario 15

User can set `?page=1000000000` and cause DB overload. Convert to int and add limits to avoid huge offsets.