NAME: TAGLE, KERWIN L. **SUBJECT:** IS Elect 2 Web Systems
COURSE, YEAR & SECTION: BSIS 2 **PROFESSOR:** REYMAR A. LLAGAS

Scenario 1 - Here's the catch:
So like, the value is coming from the URL, diba? So obviously we need to use GET, not POST. Kasi POST is for forms talaga, not for query strings. Kaya nagka "undefined index" siya, walang POST na nangyari at all.

Scenario 2 - Here's the catch:
Here the problem is super simple lang. You NEED quotes kapag string value yung hinahanap mo. Like, if you search for 'Ana', dapat naka-quote kasi hindi siya column name. Without quotes, SQL is like, "girl, sino si Ana? column ba yan?" haha laugh aq

Scenario 3 - Here's the catch:
Omy very kuan ito, kasi SQL injection is like the toxic ex of databases, very destructive. If you directly insert user input sa query, pwede nilang i-hack with "1 OR 1=1." So as an IS student keme, kailangan we use **prepared statements** para safe and secure yung query. Very cybersecurity queen lang ang atake.

Scenario 4 - Here's the catch:
Ito classic mistake na "bahala na si Batman." If walang validation, the system literally inserts empty data. Like hello, bakit ka nag-e-enroll ng student na walang name?? Kaya kailangan talaga ng empty() check kasi data integrity is life.

Scenario 5 - Here's the catch:
Grabe, ang sakit sa utak nito, simple typo pero wrecks the whole code. emial is not equal to email. PHP is very literal, parang siya yung classmate mong easily confused. So dapat correct spelling, always. Diba!

Scenario 6 - Here's the catch:
Using GET directly sa DELETE query is basically giving strangers your house keys ganon. Anyone can type ?id=0 OR 1=1 and boom goodbye to all students haha. So, we use intval() para sure na number lang talaga yung pumapasok.

Scenario 7 - Here's the catch:
This is so delulu-coded. The query literally fails but the system still claims "Updated!" Girl, huwag tayong assuming. Always check mysqli_query() result para sure na legit yung success message.

Scenario 8 - Here's the catch:
mysqli_fetch_assoc() only gets ONE row. If gusto mo ng all students, kailangan mo ng loop. Parang attendance: you don't just check the first student and say "OK na yan." Loops are life.

Scenario 9 - Here's the catch:
Hyperlinks naturally send GET, that's just how the internet works, like gravity. So if PHP expects POST, magagalit siya kasi wala naman siyang nareceive. Solution? Change it to GET. Easy life.

Scenario 10 - Here's the catch:
They used $aeg, which is like girl saan nanggaling yun?? As an IS student, variable consistency is a must. One wrong letter = chaos. Korek diba?

Scenario 11 - Here's the catch:
If the form uses GET but PHP expects POST, syempre hindi sila mag-match. Parang talking stage na hindi nagkakaintindihan charet. Dapat same method para may communication.

Scenario 12 - Here's the catch:
IDs are numeric, so hindi kailangan ng quotes around them. Quotes are for strings, not for numbers. Para hindi malito si SQL, gusto niya very organized.

Scenario 13 - Here's the catch:
This is literally the most nakakabaliw na scenario. If walang WHERE, it updates **ALL** rows. Like, lahat ng students biglang same email, universal email era?? Always include WHERE, walang exception.

Scenario 14 - Here's the catch:
They treated array keys like variables, like $data[first_name] Girl, NO. That only works sa imagination mo. It must be $data['first_name'] plus quotes around strings. Correct syntax = happy database.

Scenario 15 - Here's the catch:
If you let the user put ANY page number, they might type page=99999999999 and MySQL just collapses emotionally and mentally. So kailangan talaga ng validation and boundary checking para may control tayo over system performance.