Scenario 1 — Using $_POST instead of $_GET
Fix:

```
php
<?php
$conn = mysqli_connect("localhost", "root", "", "class_db");

$id = $_GET['id']; // Changed from $_POST to $_GET

$sql = "SELECT * FROM students WHERE student_id = $id"; // Fixed column name
$res = mysqli_query($conn, $sql);
$r = mysqli_fetch_assoc($res);

echo $r['first_name'];
?>
```

Explanation: The code was using $_POST but the URL parameter uses GET method. Also fixed column name fro

Scenario 2 — Missing quotes in SQL when using POST
Fix:

```
php
<?php
$conn = mysqli_connect("localhost","root","", "class_db");

$fname = $_POST['fname'];

$sql = "SELECT * FROM students WHERE first_name = '$fname'"; // Added quotes
$res = mysqli_query($conn, $sql);
?>
```

Explanation: String values in SQL need to be enclosed in quotes.

Scenario 3 — SQL injection vulnerability
Fix:

```
php
<?php
$conn = mysqli_connect("localhost","root","", "class_db");

$age = $_GET['age'];
$sql = "SELECT * FROM students WHERE age = ?";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "i", $age);
mysqli_stmt_execute($stmt);
$res = mysqli_stmt_get_result($stmt);
?>
```

Explanation: Used prepared statements to prevent SQL injection.

Scenario 4 — Forgetting to validate empty POST field
Fix:

```
php
<?php
$conn = mysqli_connect("localhost","root","", "class_db");

$first = $_POST['fname'];
$last = $_POST['lname'];

if(!empty($first) && !empty($last)) { // Added validation
    $sql = "INSERT INTO students (first_name,last_name) VALUES ('$first', '$last')";
    mysqli_query($conn, $sql);
    echo "Inserted!";
} else {
    echo "Please fill all fields!";
}
?>
```

Scenario 5 — Wrong key name in POST
Fix:

```
php
<?php
$conn = mysqli_connect("localhost","root","", "class_db");
```

```
$email = $_POST['email']; // Fixed spelling

$sql = "SELECT * FROM students WHERE email='$email'";
$res = mysqli_query($conn, $sql);
?>
```

Scenario 6 — Unsafe direct use of GET in DELETE
Fix:

```
php
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$id = intval($_GET['id']); // Sanitize input

$sql = "DELETE FROM students WHERE student_id = $id"; // Fixed column name
mysqli_query($conn, $sql);
?>
```

Scenario 7 — Query fails but script continues
Fix:

```
php
<?php
$conn = mysqli_connect("localhost","root","", "class_db");

$id = $_POST['id'];
$email = $_POST['email'];

$sql = "UPDATE students SET email='$email' WHERE student_id=$id"; // Added quotes and fixed column
$res = mysqli_query($conn, $sql);

if($res) {
    echo "Updated!";
} else {
    echo "Error: " . mysqli_error($conn);
}
?>
```

Scenario 8 — Missing mysqli_fetch_assoc loop
Fix:

```
php
<?php
$conn = mysqli_connect("localhost","root","", "class_db");

$res = mysqli_query($conn, "SELECT * FROM students");

while($row = mysqli_fetch_assoc($res)) { // Added loop
    echo $row['email'] . "<br>";
}
?>
```

Scenario 9 — Using GET but link sends POST
Fix:

```
php
<?php
$id = $_GET['id']; // Changed to GET
?>
<a href="view.php?id=3">View Student</a>
```

Scenario 10 — Wrong variable used in SQL
Fix:

```
php
<?php
$age = $_POST['age'];
$sql = "SELECT * FROM students WHERE age = $age"; // Fixed variable name
?>
```

Scenario 11 — Mismatched method

Fix:

```
php
// Option 1: Change form method to POST
<form method="POST" action="save.php">
    <input name="email">
</form>

// Option 2: Change PHP to use GET
$email = $_GET['email'];
```

Scenario 12 — Numeric GET used inside quotes
Fix:

```
php
<?php
$id = intval($_GET['id']); // Cast to integer
$sql = "SELECT * FROM students WHERE student_id = $id"; // Removed quotes
?>
```

Scenario 13 — Missing WHERE clause in UPDATE
Fix:

```
php
<?php
$newEmail = $_POST['email'];
$id = $_POST['id']; // Added ID parameter
$sql = "UPDATE students SET email='$newEmail' WHERE student_id=$id"; // Added WHERE
mysqli_query($conn,$sql);
?>
```

Scenario 14 — Using POST array incorrectly
Fix:

```
php
<?php
$data = $_POST;

$sql = "INSERT INTO students (first_name, last_name, email)
    VALUES ('{$data['first_name']}', '{$data['last_name']}', '{$data['email']}')";
?>
```

Scenario 15 — GET parameter used without sanitization
Fix:

```
php
<?php
$page = isset($_GET['page']) ? max(0, intval($_GET['page'])) : 0; // Validate and sanitize
$limit = 5;
$offset = $page * $limit;

$sql = "SELECT * FROM students LIMIT $offset, $limit";
?>
```