🧩 1. SQL que usas en tus ejercicios (resumen completo)

Tus ejercicios PHP trabajan con:

- Tabla **profesor**
- Tabla **curso**
- Tabla **puntos**
- (A veces alumnos)

Aquí están todas las SQL típicas:

✔ Login

```
SELECT *
FROM profesor
WHERE usuario = '$usuario' AND clave = '$clave';
```

✔ Obtener profesor por DNI

```
SELECT *
FROM profesor
WHERE dni = '$dni';
```

✔ Insertar puntos

```
INSERT INTO puntos (dni, puntos)
VALUES ('$dni', $puntos);
```

✔ Mostrar tabla de puntos

```
SELECT *
FROM puntos
WHERE dni = '$dni';
```

✔ Calcular puntos totales

```
SELECT SUM(puntos) AS total
FROM puntos
WHERE dni = '$dni';
```

✔️ Actualizar puntos

```
UPDATE puntos
SET puntos = puntos + $extra
WHERE dni = '$dni';
```

✔️ Eliminar puntos

```
DELETE FROM puntos
WHERE dni = '$dni';
```

🧩 2. SQL aplicadas a la tabla CURSO (nuevo que me diste)

Tu consulta original:

```
SELECT *
FROM curso
WHERE codigocurso = '$codCurso';
```

🧩 3. Cómo usar una variable dentro de una consulta SQL (PHP)

Depende del tipo:

🔷 Si la variable es número:

```
$sql = "SELECT * FROM curso WHERE codigocurso = $codCurso";
```

🔷 Si es texto:

```
$sql = "SELECT * FROM curso WHERE codigocurso = '$codCurso'";
```

🧩 4. Forma correcta usando Prepared Statements (lo que corregimos)

Tu versión tenía errores, así que aquí está el **código final y correcto**:

```
$codCurso = $_POST['codCurso'];
```

```
// Consulta correcta con placeholder
$sqlCodCurso = "
    SELECT *
    FROM curso
    WHERE codigocurso = ?
";

// Preparar sentencia
$stmt = $conn->prepare($sqlCodCurso);

// Vincular parámetro (i = integer)
$stmt->bind_param("i", $codCurso);

// Ejecutar
$stmt->execute();

// Obtener resultado
$resultado = $stmt->get_result();

// Comprobar si existe el curso
if ($resultado->num_rows == 0) {
    echo "El curso no existe";
} else {
    $fila = $resultado->fetch_assoc();
    echo "Curso encontrado: " . $fila['nombrecurso'];
}
```

🧩 5. Errores corregidos

❌ Antes (incorrecto)

- Usabas `$sql` en lugar de `$sqlCodCurso`.
- Ponías el valor dentro del SQL aunque luego usabas `bind_param`.
- La consulta tenía comillas mal colocadas.
- Hacías `if ($resultado = 0)` → **asignación**, no comparación.
- `$resultado` no es un número, es un objeto.

✔️ Ahora (correcto)

- La consulta usa `?` para prepared statement.
- Se usa `bind_param("i", $codCurso)` correctamente.
- Se ejecuta sin riesgo de SQL injection.
- Se valida con `num_rows`.

🧩 6. SQL útiles para ejercicios futuros

⭐ Ver ejercicios completados:

```
SELECT ejercicio, puntos
FROM puntos
WHERE dni = '$dni';
```

⭐ Ranking:

```
SELECT dni, SUM(puntos) AS total
FROM puntos
GROUP BY dni
ORDER BY total DESC;
```

⭐ Media de puntos:

```
SELECT AVG(puntos) AS media
FROM puntos
WHERE dni = '$dni';
```

⭐ Comprobar si un ejercicio ya se hizo:

```
SELECT *
FROM puntos
WHERE dni = '$dni' AND ejercicio = '$ejercicio';
```

🚀 RESUMEN AVANZADO DE CONSULTAS MySQL

🧩 1. Consultas SELECT avanzadas

✔️ Seleccionar columnas específicas

```
SELECT nombre, apellidos, email
FROM profesor;
```

✔️ Renombrar columnas (AS)

```
SELECT nombre AS docente, dni AS identificacion
FROM profesor;
```

✔️ Ordenar resultados (ORDER BY)

```
SELECT *
FROM curso
ORDER BY nombrecurso ASC;
```

✔️ Filtrar por rango (BETWEEN)

```
SELECT *
FROM puntos
WHERE puntos BETWEEN 5 AND 10;
```

✔️ Filtrar por múltiples valores (IN)

```
SELECT *
FROM curso
WHERE codigocurso IN (1, 2, 5);
```

✔️ Buscar por coincidencias (LIKE)

```
SELECT *
FROM profesor
WHERE nombre LIKE 'Mar%';  -- empieza por Mar
```

🧩 2. Funciones agregadas AVANZADAS

✔ SUM – sumar valores

```
SELECT SUM(puntos) AS total
FROM puntos
WHERE dni = '12345678A';
```

✔ AVG – media

```
SELECT AVG(puntos) AS media
FROM puntos;
```

✔ COUNT – contar filas

```
SELECT COUNT(*) AS totalCursos
FROM curso;
```

✔ MAX / MIN

```
SELECT MAX(puntos) AS mayor,
       MIN(puntos) AS menor
FROM puntos;
```

🧩 3. GROUP BY (agrupaciones)

✔ Puntos totales por usuario

```
SELECT dni, SUM(puntos) AS total
FROM puntos
GROUP BY dni;
```

✔ Ejercicios realizados por usuario

```
SELECT dni, COUNT(*) AS ejerciciosRealizados
FROM puntos
GROUP BY dni;
```

✔ Media de puntos por curso

```
SELECT codigocurso, AVG(puntos) AS mediaCurso
FROM puntos
GROUP BY codigocurso;
```

🧩 4. HAVING (filtros sobre grupos)

✔ Usuarios con más de 20 puntos acumulados

```
SELECT dni, SUM(puntos) AS total
FROM puntos
GROUP BY dni
HAVING total > 20;
```

✔ Ocultar cursos sin alumnos

```
SELECT codigocurso, COUNT(*) AS inscritos
FROM alumnos
GROUP BY codigocurso
HAVING inscritos > 0;
```

🧩 5. JOINS avanzados

✔ INNER JOIN (coincidencias exactas)

Cursos con su profesor:

```
SELECT curso.nombrecurso, profesor.nombre
FROM curso
INNER JOIN profesor ON curso.dni_profesor = profesor.dni;
```

✔ LEFT JOIN (todo de la izquierda, aunque no haya coincidencia)

Cursos aunque no tengan alumnos:

```
SELECT curso.nombrecurso, alumnos.nombre
FROM curso
LEFT JOIN alumnos ON curso.codigocurso = alumnos.codigocurso;
```

✔ RIGHT JOIN (menos común)

```
SELECT *
FROM puntos
RIGHT JOIN alumnos ON puntos.dni = alumnos.dni;
```

✔ JOIN de tres tablas

```
SELECT profesor.nombre, curso.nombrecurso, puntos.puntos
FROM profesor
JOIN curso ON profesor.dni = curso.dni_profesor
JOIN puntos ON profesor.dni = puntos.dni;
```

🧩 6. Subconsultas (SUBQUERIES)

✔ Usuario con mayor puntuación

```
SELECT dni
FROM puntos
GROUP BY dni
ORDER BY SUM(puntos) DESC
LIMIT 1;
```

✔ Cursos completados por un profesor usando subconsulta

```
SELECT *
FROM curso
WHERE codigocurso IN (
    SELECT codigocurso
    FROM puntos
    WHERE dni = '12345678A'
);
```

🧩 7. Modificación avanzada (INSERT, UPDATE, DELETE)

✔ Insertar desde otra tabla

```
INSERT INTO historial_puntos (dni, puntos)
SELECT dni, puntos FROM puntos;
```

✔ Update con JOIN

```
UPDATE curso
JOIN profesor ON curso.dni_profesor = profesor.dni
SET curso.nombrecurso = CONCAT('Curso de ', profesor.nombre);
```

✔ Update con condición avanzada

```
UPDATE puntos
SET puntos = puntos + 5
WHERE dni IN (
    SELECT dni FROM profesor WHERE categoria = 'experto'
);
```

✔ Delete condicional

```
DELETE FROM puntos
WHERE puntos = 0;
```

🧩 8. Consultas para seguridad y optimización

✔ Evitar duplicados

```
SELECT *
FROM puntos
WHERE dni = '$dni' AND ejercicio = '$ejercicio';
```

✔ Limitar número de resultados (paginación)

```
SELECT *
FROM curso
LIMIT 10 OFFSET 20; -- desde el registro 21
```

✔ Crear índices (mejora rendimiento)

```
CREATE INDEX idx_dni ON profesor(dni);
```

🧩 9. Consultas para sistemas de ranking, cursos, gamificación

✔ TOP 10 mejores usuarios

```
SELECT dni, SUM(puntos) AS total
FROM puntos
GROUP BY dni
ORDER BY total DESC
LIMIT 10;
```

✔ Últimos cursos creados

```
SELECT *
FROM curso
ORDER BY fecha_creacion DESC
LIMIT 5;
```

✓ Cursos no realizados aún

```
SELECT *
FROM curso
WHERE codigocurso NOT IN (
    SELECT codigocurso FROM puntos WHERE dni = '$dni'
);
```