

comment relier un fichier js avec html

Pour **relier un fichier JavaScript (.js) à un fichier HTML**, voici les méthodes simples et correctes 🐼

✔ Méthode 1 : Avec la balise <script> (la plus utilisée)

1 Structure des fichiers

```
projet/  
├── index.html  
└── script.js
```

2 Fichier index.html

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
  <meta charset="UTF-8">  
  <title>Mon projet</title>  
</head>  
<body>  
  
  <h1>Bonjour JavaScript</h1>  
  <button onclick="direBonjour()">Clique ici</button>  
  
  <!-- Lien vers le fichier JS -->  
  <script src="script.js"></script>  
</body>  
</html>
```

3 Fichier script.js

```
function direBonjour() {  
  alert("Bonjour depuis le fichier JavaScript !");  
}
```

★ Important :

→ Mets la balise <script> **juste avant** </body> pour éviter les erreurs de chargement.

✔ Méthode 2 : Charger le JS dans le <head> avec defer (recommandée)

```
<head>  
  <meta charset="UTF-8">  
  <title>Mon projet</title>
```

```
<script src="script.js" defer></script>
</head>
```

✦ defer signifie :

- ✓ le HTML se charge d'abord
- ✓ le JS s'exécute après

✓ Méthode 3 : Si le JS est dans un dossier

Structure

```
projet/
├── index.html
├── js/
│   └── main.js
```

Lien dans HTML

```
<script src="js/main.js"></script>
```

✗ Erreurs fréquentes à éviter

- ✗ Oublier .js
- ✗ Mauvais chemin (src)
- ✗ Écrire du JS avant que le HTML soit chargé
- ✗ Mettre <script> sans src

🔍 Vérification

- Ouvre la page dans le navigateur
- Appuie sur **F12** → **Console**
- S'il y a une erreur, elle s'affiche ici

Si tu veux, je peux aussi te montrer :

- ✓ un exemple avec **DOM**
- ✓ un projet **HTML + CSS + JS**
- ✓ comment relier JS avec **Bootstrap**
- ✓ ou avec **PHP**

☞ Dis-moi ce que tu veux faire exactement 😊

pourquoi ca ne fonctionne pas let age= 5 switch(age){ case age<=12: case 12<age<=13: case 13<age<=18: console.log('vous êtes mineur'); break case 18<age<=20: case 20<age<=30: console.log('vous êtes majeur') break default: console.log('adulte') }

Bonne question 🐼

Ton code **ne fonctionne pas** à cause de **la façon dont switch marche en JavaScript**.

❌ Pourquoi ton switch ne marche pas ?

1 switch ne gère PAS les conditions (<=, >, etc.)

En JavaScript :

```
switch (age) {  
  case 12:  
  case 13:  
}
```

☞ switch compare **des valeurs exactes** avec ===

Mais toi tu écris :

```
case age <= 12:
```

⚠ Problème :

- age <= 12 retourne **true ou false**
 - switch(age) compare age (ex: 5) avec true ou false
 - 5 === true ❌ → jamais vrai
-

2 Les expressions comme 12 < age <= 13 sont FAUSSES en JS

```
12 < age <= 13
```

JavaScript lit ça comme :

```
(12 < age) <= 13
```

→ (12 < age) donne true ou false

→ true <= 13 → 1 <= 13 → ✔ true (résultat trompeur)

✔ **Solution 1 (RECOMMANDÉE) : utiliser if / else**

```
let age = 5;
```

```
if (age <= 18) {  
  console.log("vous êtes mineur");  
} else if (age <= 30) {  
  console.log("vous êtes majeur");  
} else {  
  console.log("adulte");  
}
```

✓ Simple

✓ Clair

✓ Correct pour les intervalles

✓ Solution 2 : switch(true) (technique avancée)

Si tu veux **vraiment** utiliser switch :

```
let age = 5;
```

```
switch (true) {  
  case age <= 18:  
    console.log("vous êtes mineur");  
    break;  
  case age <= 30:  
    console.log("vous êtes majeur");  
    break;  
  default:  
    console.log("adulte");  
}
```

★ Ici :

- switch(true)
 - chaque case retourne true ou false
-

☐ À retenir (très important)

Cas	Utiliser
Comparaison exacte	switch
Intervalles / conditions if / else	

☐ Test rapide

```
let age = 25;  
// Résultat attendu : "vous êtes majeur"
```

Si tu veux, je peux :

- te montrer **quand utiliser switch**
- faire un **tableau d'âges propre**
- expliquer **switch vs if** avec schéma simple

☞ Dis-moi 😊

donne moi la manipulation de tableau avec js et php de base

Voici une **fiche claire et complète des manipulations de tableaux (arrays)** en **JavaScript** et en **PHP, niveau débutant**, avec les opérations de base ☞
(Très utile pour tes projets web et MVC que tu apprends)

☐ MANIPULATION DES TABLEAUX EN JAVASCRIPT

1 Créer un tableau

```
let fruits = ["pomme", "banane", "orange"];
```

Tableau vide :

```
let notes = [];
```

2 Accéder à un élément

```
console.log(fruits[0]); // pomme
```

★ Les index commencent à **0**

3 Modifier un élément

```
fruits[1] = "mangue";
```

4 Ajouter des éléments

➕ À la fin

```
fruits.push("ananas");
```

+ Au début

```
fruits.unshift("citron");
```

5 Supprimer des éléments

— À la fin

```
fruits.pop();
```

— Au début

```
fruits.shift();
```

6 Parcourir un tableau

↻ Avec for

```
for (let i = 0; i < fruits.length; i++) {  
  console.log(fruits[i]);  
}
```

↻ Avec forEach

```
fruits.forEach(fruit => {  
  console.log(fruit);  
});
```

7 Rechercher dans un tableau

```
fruits.includes("pomme"); // true
```

Index :

```
fruits.indexOf("orange"); // 2
```

8 Supprimer par index

```
fruits.splice(1, 1); // supprime 1 élément à l'index 1
```

9 Trier un tableau

```
fruits.sort();
```

Numérique :

```
let notes = [10, 3, 20];  
notes.sort((a, b) => a - b);
```

10 Tableau d'objets (très courant)

```
let users = [  
  { nom: "Jean", age: 20 },  
  { nom: "Marie", age: 17 }  
];
```

```
users.forEach(u => {  
  console.log(u.nom);  
});
```

MANIPULATION DES TABLEAUX EN PHP

1 Créer un tableau

```
$fruits = ["pomme", "banane", "orange"];
```

Tableau vide :

```
$notes = [];
```

2 Accéder à un élément

```
echo $fruits[0]; // pomme
```

3 Modifier un élément

```
$fruits[1] = "mangue";
```

4 Ajouter des éléments

+ À la fin

```
$fruits[] = "ananas";
```

OU

```
array_push($fruits, "citron");
```

5 Supprimer des éléments

— À la fin

```
array_pop($fruits);
```

— Au début

```
array_shift($fruits);
```

6 Parcourir un tableau

🔄 foreach (le plus utilisé)

```
foreach ($fruits as $fruit) {  
    echo $fruit . "<br>";  
}
```

Avec index :

```
foreach ($fruits as $index => $fruit) {  
    echo $index . " - " . $fruit;  
}
```

7 Rechercher dans un tableau

```
in_array("pomme", $fruits); // true
```

Index :

```
array_search("orange", $fruits);
```

8 Trier un tableau

```
sort($fruits);
```

Numérique :

```
sort($notes, SORT_NUMERIC);
```

9 Tableau associatif (clé → valeur)

```
$user = [  
    "nom" => "Jean",  
    "age" => 25  
];
```

```
echo $user["nom"];
```

Parcours :

```
foreach ($user as $cle => $valeur) {  
    echo "$cle : $valeur <br>";  
}
```

10 Tableau multidimensionnel

```
$users = [  
    ["nom" => "Jean", "age" => 20],  
    ["nom" => "Marie", "age" => 17]  
];
```

```
foreach ($users as $u) {  
    echo $u["nom"];  
}
```

❑ COMPARAISON RAPIDE JS vs PHP

Action	JavaScript	PHP
Ajouter	push()	\$arr[]
Supprimer	pop()	array_pop()
Parcourir	forEach	foreach
Rechercher	includes()	in_array()

🔗 Conseil pour toi (débutant → pro)



🔗 **JS** → manipulation côté navigateur

🔗 **PHP** → traitement serveur / base de données

🔗 Les **tableaux** sont partout : formulaires, API, MVC, JSON

Si tu veux :

- 📁 Exercice corrigé

- ☐ Exemple CRUD (tableau → base de données)
-  Lien JS ↔ PHP avec JSON
-  Exemple MVC simple

🔗 Dis-moi 👉

comment tester le code php l'afficher ver le navigateur? meme echo <!DOCTYPE html>
<html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport"
content="width=device-width, initial-scale=1.0"> <meta http-equiv="X-UA-Compatible"
content="ie=edge"> <title>Document</title> </head> <body> <h1>pratique avec php</h1>
<?php \$fruits=['pomme','mangue','banane'] echo \$fruits[0] ?> </body> </html>

Ton code est **presque correct**, mais il y a **2 problèmes essentiels** et **1 point très important** à connaître pour afficher du PHP dans le navigateur 🙋

❌ Pourquoi ton code ne s'affiche pas ?

1 Il manque des points-virgules ; en PHP

En PHP, **chaque instruction doit se terminer par ;**

❌ Ton code :

```
$fruits=['pomme','mangue','banane']
echo $fruits[0]
```

✓ Correction :

```
$fruits = ['pomme', 'mangue', 'banane'];
echo $fruits[0];
```

2 Le fichier doit être en .php (PAS .html)

❌ Mauvais :

index.html

✓ Correct :

index.php

🔗 Sinon, le navigateur **ignore complètement le PHP**

✓ Code PHP CORRECT ET FONCTIONNEL

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Test PHP</title>
</head>
<body>

  <h1>Pratique avec PHP</h1>

  <?php
    $fruits = ['pomme', 'mangue', 'banane'];
    echo $fruits[0];
  ?>

</body>
</html>
```

Résultat dans le navigateur :

Pratique avec PHP
pomme

⚠ Très important : PHP NE FONCTIONNE PAS sans serveur

PHP est exécuté **côté serveur**, pas directement par le navigateur.

✓ MÉTHODE 1 : Tester avec PHP intégré (le plus simple)

1 Vérifier que PHP est installé

Dans le terminal :

```
php -v
```

2 Aller dans le dossier du projet

```
cd chemin/vers/ton/projet
```

3 Lancer le serveur PHP

```
php -S localhost:8000
```

4 Ouvrir le navigateur

<http://localhost:8000/index.php>

✓ MÉTHODE 2 : Avec XAMPP / WAMP (Windows)

1 Installer **XAMPP**

2 Démarrer **Apache**

3 Mettre ton projet dans :

C:\xampp\htdocs\monprojet\

4 Ouvrir :

<http://localhost/monprojet/index.php>

✗ Erreurs fréquentes

✗ Oublier le ;

✗ Fichier en .html

✗ Ouvrir le fichier directement (file:///)

✗ Apache non démarré

□ Astuce débutant

Pour voir les erreurs PHP :

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);
?>
```

🔗 Prochaine étape pour toi

Tu peux maintenant :

- ✓ afficher des variables
- ✓ tester des tableaux
- ✓ connecter PHP + HTML
- ✓ passer au **MVC** (Controller → View)

Si tu veux :

- CRUD PHP simple
- PHP + formulaire HTML

- PHP + MySQL
- Structure MVC débutant

☞ Dis-moi 🔥

qu'est ce qui différencie let, const et var avec js

Excellente question 👉

Comprendre **var** / **let** / **const** est **fondamental en JavaScript**. Voici une **explication claire, simple et complète**, avec exemples concrets.

● **var** (ancienne façon – à éviter)

✓ **Caractéristiques**

- **Portée : fonction**
- Peut être **redéclarée**
- Peut être **réaffectée**
- Subit le **hoisting** (montée en mémoire)

✗ **Problèmes**

- Comportements imprévisibles
- Bugs difficiles à détecter

Exemple

```
var x = 10;
```

```
if (true) {  
  var x = 20;  
}
```

```
console.log(x); // 20 🤖 (écrase la valeur)
```

□ **let** (moderne – recommandé)

✓ **Caractéristiques**

- **Portée : bloc {}**
- Peut être **réaffectée**
- ✗ Ne peut pas être redéclarée dans le même scope
- Hoisting sans initialisation

Exemple

```
let age = 20;

if (true) {
  let age = 15;
  console.log(age); // 15
}

console.log(age); // 20 ✓
```

❑ const (valeur constante)

✓ Caractéristiques

- **Portée : bloc {}**
- ✗ Ne peut pas être réaffectée
- ✗ Ne peut pas être redéclarée
- Doit être **initialisée immédiatement**

Exemple

```
const PI = 3.14;
PI = 4; // ✗ Erreur
```

⚠ IMPORTANT : const et les objets/tableaux

⚠ const bloque la **référence**, pas le contenu.

```
const fruits = ["pomme", "banane"];
fruits.push("mangue"); // ✓ autorisé
fruits = []; // ✗ interdit
```

📊 TABLEAU COMPARATIF

Critère	var	let	const
Portée	fonction	bloc	bloc
Réaffectation	oui	oui	✗ non
Redéclaration	oui	✗ non	✗ non
Hoisting	oui	partiel	partiel
Utilisation moderne	✗ non	✓ oui	✓ oui

🔗 Quand utiliser quoi ?

- ✓ Par défaut → **const**
- ✓ Besoin de changer la valeur → **let**
- ✗ Éviter **var**

Exemple recommandé

```
const API_URL = "https://example.com";  
let compteur = 0;  
  
compteur++;
```

□ Résumé simple

- 🔒 **const** = ne change pas
 - 🔄 **let** = peut changer
 - ⚠️ **var** = ancien et risqué
-

Si tu veux, je peux :

- faire un **exercice corrigé**
- montrer les erreurs courantes
- expliquer le **hoisting visuellement**
- comparer avec **PHP (\$var, const)**

☞ Dis-moi 😊

je veis faire une test avec js de base, php de base et codeIgniter, que dois je maiotriser ?

Très bonne initiative 👍

Si tu veux **réussir un test** avec **JS de base + PHP de base + CodeIgniter**, voici **exactement ce que tu dois maîtriser, sans te disperser, du minimum vital au bonus.**

(Je te parle comme pour un test technique réel)

🎯 OBJECTIF DU TEST

Montrer que tu sais :

- comprendre une logique
 - manipuler des données
 - structurer un petit projet
 - utiliser CodeIgniter correctement
-

□ 1 JAVASCRIPT DE BASE (OBLIGATOIRE)

◆ À maîtriser absolument

- ✓ Variables : let, const
- ✓ Conditions : if / else, switch
- ✓ Boucles : for, while, forEach
- ✓ Tableaux & objets
- ✓ Fonctions
- ✓ DOM (document.getElementById, querySelector)
- ✓ Événements (onclick, addEventListener)

◆ Exemples de test

```
let notes = [10, 12, 15];
```

```
let somme = 0;  
notes.forEach(n => somme += n);
```

```
console.log(somme);
```

◆ DOM

```
<button id="btn">Cliquer</button>  
<script>  
document.getElementById("btn").onclick = () => {  
    alert("OK");  
}  
</script>
```

□ 2 PHP DE BASE (INDISPENSABLE)

◆ À maîtriser absolument

- ✓ Syntaxe PHP
- ✓ Variables & types
- ✓ echo, print_r, var_dump
- ✓ Conditions & boucles
- ✓ Tableaux (simples, associatifs, multidimensionnels)
- ✓ Fonctions
- ✓ Inclusion de fichiers (include, require)
- ✓ Traitement formulaire (\$_POST, \$_GET)

◆ Exemple

```
$users = ["Jean", "Marie"];
```



```
foreach ($users as $u) {  
    echo $u;  
}
```

□ **3 PHP + HTML (FORMULAIRES)**

◆ Très fréquent en test

- Créer un formulaire
- Récupérer les données
- Valider les champs
- Afficher les erreurs

```
<form method="post">  
    <input name="nom">  
    <button>Envoyer</button>  
</form>
```

```
<?php  
if ($_POST) {  
    echo $_POST['nom'];  
}  
?>
```

□ **4 BASE DE DONNÉES (IMPORTANT)**

◆ À maîtriser

- ✓ MySQL / MariaDB
- ✓ CRUD (Create, Read, Update, Delete)
- ✓ SELECT, INSERT, UPDATE, DELETE
- ✓ Connexion PDO

```
$db = new PDO("mysql:host=localhost;dbname=test", "root", "");
```

□ **5 CODEIGNITER (TRÈS IMPORTANT)**

◆ À maîtriser ABSOLUMENT

- ✓ Architecture MVC
- ✓ Controllers
- ✓ Views
- ✓ Models
- ✓ Routes
- ✓ Helpers

- ✓ Validation
 - ✓ Base URL
-

◆ Structure CodeIgniter

```
app/  
├── Controllers/  
├── Models/  
└── Views/
```

◆ Controller

```
class User extends BaseController {  
    public function index() {  
        return view('users');  
    }  
}
```

◆ Model

```
class UserModel extends Model {  
    protected $table = 'users';  
}
```

◆ View

```
<h1>Liste utilisateurs</h1>
```

□ 6 CRUD SIMPLE AVEC CODEIGNITER (TRÈS FRÉQUENT)

- ✓ Liste
- ✓ Ajouter
- ✓ Modifier
- ✓ Supprimer

☞ Même sans finir, **la structure compte énormément.**

□ 7 BONUS QUI FAIT LA DIFFÉRENCE 🌟

- ★ JavaScript + formulaire (validation client)
 - ★ Sécurité (htmlspecialchars, password_hash)
 - ★ Bootstrap basique
 - ★ Commentaires dans le code
 - ★ Bonne organisation des fichiers
-

✗ CE QU'ILS N'ATTENDENT PAS

- ✗ React
 - ✗ API complexe
 - ✗ Sécurité avancée
 - ✗ Optimisation
-

☐ STRATÉGIE POUR RÉUSSIR

- ☞ Commence simple
 - ☞ Code propre
 - ☞ MVC respecté
 - ☞ Fonctionnel avant beau
 - ☞ Explique ton raisonnement
-

✦ PLAN DE RÉVISION (3 JOURS)

📅 Jour 1

- JS de base
- PHP de base
- Tableaux & boucles

📅 Jour 2

- Formulaires PHP
- CRUD PHP + MySQL

📅 Jour 3

- CRUD CodeIgniter
 - Routes & MVC
-

Si tu veux, je peux :

- 🔥 te donner un **sujet de test réel**
- ☐ te faire un **mini projet CodeIgniter**
- 📁 une **check-list à réviser**
- ☐ des **exercices corrigés**

🔗 Dis-moi ce que tu veux exactement 🗨️

donne moi des exemple a chaque partie

Parfait 👍

Voici **des exemples concrets pour CHAQUE PARTIE : JS de base, PHP de base, PHP + formulaire, BDD, CodeIgniter (MVC + CRUD simple).**

🔗 Ce sont **exactement le type d'exemples qu'on te demande en test.**

☐ 1. JAVASCRIPT DE BASE – EXEMPLES

◆ Variables & conditions

```
let age = 17;
```

```
if (age < 18) {  
  console.log("Mineur");  
} else {  
  console.log("Majeur");  
}
```

◆ Tableaux & boucles

```
let fruits = ["pomme", "mangue", "banane"];
```

```
fruits.forEach(fruit => {  
  console.log(fruit);  
});
```

◆ DOM & événements

```
<button id="btn">Cliquer</button>  
<p id="msg"></p>
```

```
<script>  
document.getElementById("btn").addEventListener("click", function () {  
  document.getElementById("msg").innerText = "Bouton cliqué";  
});
```

</script>

❑ 2 PHP DE BASE – EXEMPLES

◆ Variables & echo

```
<?php
$nom = "Jean";
echo "Bonjour " . $nom;
?>
```

◆ Conditions & boucles

```
<?php
$age = 20;

if ($age >= 18) {
    echo "Majeur";
} else {
    echo "Mineur";
}
?>
```

◆ Tableaux

```
<?php
$fruits = ["pomme", "mangue", "banane"];

foreach ($fruits as $fruit) {
    echo $fruit . "<br>";
}
?>
```

❑ 3 PHP + FORMULAIRE (TRÈS IMPORTANT EN TEST)

```
<form method="post">
    <input type="text" name="nom" placeholder="Votre nom">
    <button type="submit">Envoyer</button>
</form>
```

```
<?php
if ($_POST) {
    $nom = htmlspecialchars($_POST['nom']);
    echo "Nom saisi : " . $nom;
}
?>
```

- ✓ Récupération
 - ✓ Sécurisation
 - ✓ Affichage
-

❑ 4 BASE DE DONNÉES – EXEMPLE CRUD SIMPLE (PDO)

◆ Connexion

```
<?php
$pdo = new PDO("mysql:host=localhost;dbname=test", "root", "");
?>
```

◆ INSERT

```
$pdo->query("INSERT INTO users (nom) VALUES ('Jean')");
```

◆ SELECT

```
$users = $pdo->query("SELECT * FROM users");

foreach ($users as $user) {
    echo $user['nom'] . "<br>";
}
```

◆ DELETE

```
$pdo->query("DELETE FROM users WHERE id = 1");
```

❑ 5 CODEIGNITER – EXEMPLES CLÉS (MVC)

◆ Structure

```
app/
├── Controllers/User.php
├── Models/UserModel.php
└── Views/users.php
```

◆ Controller

```
namespace App\Controllers;
```

```
class User extends BaseController
{
    public function index()
    {
        return view('users');
    }
}
```

◆ Model

```
namespace App\Models;
use CodeIgniter\Model;

class UserModel extends Model
{
    protected $table = 'users';
    protected $allowedFields = ['nom'];
}
```

◆ View

```
<h1>Liste des utilisateurs</h1>
```

◆ Route

```
$routes->get('/users', 'User::index');
```

□ 6 MINI CRUD CODEIGNITER (AJOUT)

Controller

```
public function add()
{
    $model = new \App\Models\UserModel();
    $model->save([
        'nom' => $this->request->getPost('nom')
    ]);
    return redirect()->to('/users');
}
```

View

```
<form method="post" action="/add-user">
    <input name="nom">
    <button>Ajouter</button>
</form>
```

Route

```
$routes->post('/add-user', 'User::add');
```

☐ CE QUE LE CORRECTEUR REGARDE

- ✓ Logique correcte
 - ✓ Code lisible
 - ✓ MVC respecté
 - ✓ Données bien récupérées
 - ✓ Pas d'erreurs PHP
-

🔗 CONSEIL FINAL POUR LE TEST

Même si ce n'est pas parfait, montre que tu comprends la structure.

Si tu veux maintenant :

- 📁 un sujet de test réel
- ☐ simulation d'examen
- 🛠️ mini projet CodeIgniter complet
- ? questions/réponses fréquentes en test

👉 Dis-moi 🔥

donne moi des type de requete possible avec des jointures aussi

Très bien 👍

Voici une **fiche claire et complète des types de requêtes SQL**, avec **jointures (JOIN)**, **niveau test technique / débutant → intermédiaire**, exactement ce qu'on attend de toi avec **PHP / CodeIgniter / MySQL**.

☐ BASE : TABLES D'EXEMPLE

users

id

nom

email

orders

id
user_id
total
date_commande

1 REQUÊTES DE BASE

◆ SELECT (lire)

```
SELECT * FROM users;
```

◆ INSERT (ajouter)

```
INSERT INTO users (nom, email)  
VALUES ('Jean', 'jean@gmail.com');
```

◆ UPDATE (modifier)

```
UPDATE users  
SET nom = 'Paul'  
WHERE id = 1;
```

◆ DELETE (supprimer)

```
DELETE FROM users  
WHERE id = 1;
```

2 REQUÊTES AVEC CONDITIONS

◆ WHERE

```
SELECT * FROM users WHERE nom = 'Jean';
```

◆ AND / OR

```
SELECT * FROM users  
WHERE nom = 'Jean' AND email LIKE '%gmail%';
```

◆ BETWEEN

```
SELECT * FROM orders
```

WHERE total BETWEEN 100 AND 500;

◆ IN

```
SELECT * FROM users  
WHERE id IN (1, 3, 5);
```

3 TRI & LIMITES

◆ ORDER BY

```
SELECT * FROM users ORDER BY nom ASC;
```

◆ LIMIT

```
SELECT * FROM users LIMIT 5;
```

4 FONCTIONS D'AGRÉGATION

◆ COUNT

```
SELECT COUNT(*) FROM users;
```

◆ SUM

```
SELECT SUM(total) FROM orders;
```

◆ GROUP BY

```
SELECT user_id, SUM(total) AS total_commande  
FROM orders  
GROUP BY user_id;
```

5 JOINTURES (TRÈS IMPORTANT)

∞ INNER JOIN (le plus utilisé)

☞ Donne les données **liées dans les deux tables**

```
SELECT users.nom, orders.total
```

```
FROM users
INNER JOIN orders ON users.id = orders.user_id;
```

∞ LEFT JOIN

☞ Tous les utilisateurs, **même sans commande**

```
SELECT users.nom, orders.total
FROM users
LEFT JOIN orders ON users.id = orders.user_id;
```

∞ RIGHT JOIN (moins utilisé)

```
SELECT users.nom, orders.total
FROM users
RIGHT JOIN orders ON users.id = orders.user_id;
```

∞ FULL JOIN

⚠ **Pas supporté par MySQL**
(simulé avec UNION)

6 JOIN + CONDITION

```
SELECT users.nom, orders.total
FROM users
INNER JOIN orders ON users.id = orders.user_id
WHERE orders.total > 100;
```

7 JOIN + GROUP BY

```
SELECT users.nom, COUNT(orders.id) AS nb_commandes
FROM users
LEFT JOIN orders ON users.id = orders.user_id
GROUP BY users.id;
```

8 JOIN SUR PLUSIEURS TABLES

Tables :

users → orders → payments
SELECT users.nom, orders.total, payments.mode

```
FROM users
INNER JOIN orders ON users.id = orders.user_id
INNER JOIN payments ON orders.id = payments.order_id;
```

9 SOUS-REQUÊTES (SUBQUERY)

```
SELECT nom
FROM users
WHERE id IN (
    SELECT user_id FROM orders WHERE total > 500
);
```

10 UNION (fusion de résultats)

```
SELECT nom FROM clients
UNION
SELECT nom FROM fournisseurs;
```

□ UTILISATION AVEC PHP (PDO)

```
$sql = "
SELECT users.nom, orders.total
FROM users
INNER JOIN orders ON users.id = orders.user_id
";
```

```
$stmt = $pdo->prepare($sql);
$stmt->execute();
$result = $stmt->fetchAll();
```

```
foreach ($result as $row) {
    echo $row['nom'] . " - " . $row['total'];
}
```

□ AVEC CODEIGNITER (Query Builder)

```
$db = \Config\Database::connect();

$builder = $db->table('users');
$builder->select('users.nom, orders.total');
$builder->join('orders', 'orders.user_id = users.id', 'left');
$query = $builder->get();

$result = $query->getResult();
```

❑ À SAVOIR POUR UN TEST

- ✓ INNER JOIN = données liées
 - ✓ LEFT JOIN = tout à gauche
 - ✓ GROUP BY souvent avec COUNT/SUM
 - ✓ Toujours une condition ON correcte
 - ✓ MySQL ≠ FULL JOIN
-

🔗 RÉSUMÉ EXPRESS

Besoin	Requête
Lire	SELECT
Ajouter	INSERT
Modifier	UPDATE
Supprimer	DELETE
Lier tables	JOIN
Compter	COUNT
Total	SUM

Si tu veux :

- 📁 exercices SQL corrigés
- ❑ sujets de test avec JOIN
- 🏗️ schémas visuels de jointures
- 🛠️ CRUD CodeIgniter avec JOIN

👉 Dis-moi 🗨️

et comment a propos de jquery

Très bien 👍

Voici l'essentiel de **jQuery** à maîtriser **pour un test technique**, avec **exemples simples et concrets**, surtout pour **PHP / CodeIgniter**.

❑ 1 C'est quoi jQuery ?

👉 **jQuery est une bibliothèque JavaScript** qui simplifie :

- la manipulation du DOM
- les événements
- l'AJAX (communication avec PHP)

- les animations simples

✦ Aujourd'hui, **JS pur** peut suffire, mais **jQuery** est encore très utilisé en test et projets existants.

□ 2 Comment inclure jQuery

✓ **CDN (le plus courant en test)**

```
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
```

⚠ **Toujours avant ton script perso**

```
<script src="script.js"></script>
```

□ 3 Structure de base jQuery

```
$(document).ready(function () {  
    // code jQuery ici  
});
```

☞ Signifie : *exécuter le code quand la page est chargée*

Version courte :

```
$(function () {  
    // code  
});
```

□ 4 Sélection des éléments (TRÈS IMPORTANT)

```
$("#id");    // id  
$(".classe"); // classe  
$("p");      // balise
```

Exemple

```
$("#titre").text("Bonjour jQuery");
```

□ 5 Événements (clic, change, submit)

◆ **Click**

```
$("#btn").click(function () {
```

```
    alert("Bouton cliqué");
});
```

◆ **Change**

```
$("#age").change(function () {
    console.log($(this).val());
});
```

◆ **Submit (formulaire)**

```
$("#form").submit(function (e) {
    e.preventDefault(); // empêche rechargement
    alert("Formulaire envoyé");
});
```

□ **6 Manipulation du contenu**

```
$("#msg").text("Texte simple");
$("#msg").html("<b>Texte HTML</b>");
$("#msg").val(); // input
```

□ **7 Ajouter / supprimer des éléments**

```
$("#liste").append("<li>Nouvel élément</li>");
$("#item").remove();
```

□ **8 CSS avec jQuery**

```
$("#box").css("color", "red");
```

Plusieurs styles :

```
$("#box").css({
    color: "white",
    background: "blue"
});
```

□ **9 AJAX (TRÈS IMPORTANT AVEC PHP / CI)**

◆ **Envoi vers PHP**

```
$.ajax({
    url: "traitement.php",
    method: "POST",
```

```
data: { nom: "Jean" },
success: function (reponse) {
    console.log(reponse);
}
});
```

◆ Version courte

```
$.post("traitement.php", { nom: "Jean" }, function (data) {
    $("#resultat").html(data);
});
```

□ 10 Exemple complet : jQuery + PHP

HTML

```
<input id="nom">
<button id="btn">Envoyer</button>
<div id="res"></div>
```

jQuery

```
$("#btn").click(function () {
    let nom = $("#nom").val();

    $.post("test.php", { nom: nom }, function (data) {
        $("#res").html(data);
    });
});
```

PHP (test.php)

```
<?php
echo "Bonjour " . $_POST['nom'];
```

□ CE QU'ILS ATTENDENT EN TEST

- ✓ Sélection DOM
 - ✓ Événements
 - ✓ AJAX simple
 - ✓ Lien jQuery ↔ PHP
 - ✓ Code clair
-

✗ Ce qu'ils n'attendent PAS

- ✗ Plugins complexes
 - ✗ Animations lourdes
 - ✗ jQuery UI avancé
-

🔗 COMPARAISON RAPIDE

Action	JS pur	jQuery
Sélection	document.querySelector	\$("#id")
Event	addEventListener	.click()
AJAX	fetch()	\$.ajax()

☐ À RÉVISER EN PRIORITÉ (CHECKLIST)

- ☒ CDN jQuery
 - ☒ \$(document).ready()
 - ☒ Sélecteurs
 - ☒ Events
 - ☒ .val(), .text(), .html()
 - ☒ AJAX vers PHP
-

Si tu veux :

- ☐ **sujet de test jQuery + PHP**
- 📁 **exercice CodeIgniter + AJAX**
- 📁 **anti-erreurs fréquentes**
- 🔄 **comparaison jQuery vs JS pur**

👉 Dis-moi 🗨️

c quoi asynchrone

Très bonne question 👍

Le **concept d'asynchrone** est **essentiel en JavaScript**, surtout avec **AJAX, jQuery, fetch, API**. Je t'explique **simplement**, avec des exemples clairs.

☐ C'est quoi "asynchrone" ?

👉 **Asynchrone = le programme n'attend pas qu'une tâche longue finisse pour continuer.**

Pendant qu'une action longue se fait (requête serveur, chargement de données...),
☞ **le reste du code continue à s'exécuter.**

✗ Synchrones (bloquant)

```
console.log("Début");  
  
alert("Chargement..."); // bloque tout  
  
console.log("Fin");
```

➡ Tant que l'alerte n'est pas fermée → rien d'autre ne s'exécute.

✓ Asynchrone (non bloquant)

```
console.log("Début");  
  
setTimeout(() => {  
  console.log("Tâche longue");  
}, 2000);  
  
console.log("Fin");
```

✦ Résultat :

Début
Fin
Tâche longue

☞ Le programme **ne bloque pas**.

🔄 Exemple concret (vie réelle)

☐🕒 Synchrones

→ Tu attends que le riz cuise avant de faire le plat ✗

☐🕒 Asynchrone

→ Tu mets le riz à cuire

→ Pendant ce temps, tu prépares la sauce ✓

❑ Pourquoi l'asynchrone est important ?

- ✓ Charger des données sans bloquer la page
 - ✓ Appels API
 - ✓ AJAX / jQuery
 - ✓ Meilleure expérience utilisateur
-

❑ Exemples asynchrones courants en JS

Fonction	Type
setTimeout	asynchrone
setInterval	asynchrone
fetch()	asynchrone
\$.ajax()	asynchrone
Events (click)	asynchrone

❑ AJAX (exemple simple)

```
$.post("test.php", function (data) {  
    console.log(data);  
});
```

☞ La page **continue de fonctionner** pendant l'appel PHP.

❑ Problème classique des débutants

✗ Mauvais :

```
let data;
```

```
$.get("test.php", function (res) {  
    data = res;  
});
```

```
console.log(data); // undefined ✗
```

☞ Le code s'exécute **AVANT** la réponse.

✓ **Bonne façon (callback)**

```
$.get("test.php", function (res) {  
  console.log(res); // OK  
});
```

☐ Autres façons de gérer l'asynchrone

◆ Callback

```
setTimeout(() => console.log("OK"), 1000);
```

◆ Promise

```
fetch("test.php")  
  .then(res => res.text())  
  .then(data => console.log(data));
```

◆ async / await (moderne)

```
async function charger() {  
  let res = await fetch("test.php");  
  let data = await res.text();  
  console.log(data);  
}
```

☐ Résumé ultra simple

☐ Asynchrone = le code continue pendant qu'une tâche se fait

● Synchron = le code attend et bloque

🔗 Pour ton test

Ils veulent voir que tu comprends :

- pourquoi console.log peut afficher undefined
 - pourquoi AJAX n'est pas immédiat
 - pourquoi on utilise success, then, await
-

Si tu veux :

- ☐ exercices corrigés
- 🔁 comparaison jQuery AJAX vs fetch
- ? pièges fréquents en test
- ☐ schéma visuel simple

