Création des Tables :

```
CREATE TABLE Adresse (
    id INT PRIMARY KEY AUTO_INCREMENT,
    numero_rue INT NOT NULL,
    code_postal INT NOT NULL,
    nom_rue VARCHAR(20) NOT NULL) :

CREATE TABLE Etablissement (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nom VARCHAR(20) NOT NULL,
    description VARCHAR(20) NOT NULL,
    superficie INT NOT NULL,
    adresse_id INT NOT NULL,
    FOREIGN KEY (adresse_id) REFERENCES adresse(id)) ;

CREATE TABLE Personne (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nom VARCHAR(20) NOT NULL,
    prenom VARCHAR(20) NOT NULL,q
    situation VARCHAR(20) NOT NULL,
    sexe BOOLEAN NOT NULL,
    ville_naissance VARCHAR(20) NOT NULL,
    date_naissance DATE NOT NULL,
    nationalite VARCHAR(20) NOT NULL) ;

CREATE TABLE Projet (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nom VARCHAR(20) NOT NULL,
    description VARCHAR(20) NOT NULL,
    date_debut DATE NOT NULL,
    date_fin_prevue DATE NOT NULL,
    montant_budget INT NOT NULL,
    budget_id INT NOT NULL, (devenu obsolète)
    FOREIGN KEY (budget_id) REFERENCES budget(id) (devenu obsolète)) ;

CREATE TABLE Employe (
    id INT AUTO_INCREMENT,
    personne_id INT NOT NULL,
    fonction VARCHAR(20) NOT NULL,
    date_embauche Date NOT NULL,
    date_fin Date,
    domaine VARCHAR(20) NOT NULL,
    PRIMARY KEY (id, personne_id),
    FOREIGN KEY (personne_id) REFERENCES Personne(id)) ;
```

Devenu (obsolète)

```
CREATE TABLE Budget (
    id INT PRIMARY KEY AUTO_INCREMENT,
    montant INT NOT NULL,
    annee INT NOT NULL) ;
```

```
CREATE TABLE Rattacher (
    id INT AUTO_INCREMENT,
    adresse_id INT NOT NULL,
    personne_id INT NOT NULL,
    PRIMARY KEY (id, adresse_id, personne_id),
    FOREIGN KEY (personne_id) REFERENCES Personne(id),
    FOREIGN KEY (adresse_id) REFERENCES Adresse(id)
);

CREATE TABLE Travailler (
    id INT AUTO_INCREMENT,
    employe_id INT NOT NULL,
    etablissement_id INT NOT NULL,
    PRIMARY KEY (id, employe_id, etablissement_id),
    FOREIGN KEY (employe_id) REFERENCES Employe(id),
    FOREIGN KEY (etablissement_id) REFERENCES Etablissement(id)
);

CREATE TABLE Comporter (
    id INT AUTO_INCREMENT,
    projet_id INT NOT NULL,
    etablissement_id INT NOT NULL,
    PRIMARY KEY (id, projet_id, etablissement_id),
    FOREIGN KEY (projet_id) REFERENCES Projet(id),
    FOREIGN KEY (etablissement_id) REFERENCES Etablissement(id)
);
```

Insertion des données :

```
INSERT INTO adresse (numero_rue, code_postal, nom_rue)
VALUES (12,94600,"Rue des Robino")

INSERT INTO projet (nom, description, date_debut, date_fin_prevue, montant_budget)
VALUES ("Rénovation Gymnase", "Rénovation du matériels et des locaux", "2025-02-14", "2025-08-12", 11000)

INSERT INTO etablissement (nom, description, superficie, adresse_id)
VALUES ("Ecole Primaire", "Ecole accueillant des élèves de CP au CM2", 2300, 3)

INSERT INTO personne (nom, prenom, situation, sexe, ville_naissance, date_naissance, nationalite)
VALUES ("Ouakkouche", "Hadil", "célibataire", 0, "Bejaia", "2001-08-29", "Algérienne")

INSERT INTO employe (personne_id, fonction, date_embauche, domaine)
VALUES (2, "Institutrice", "2022-03-14", "Ecole Primaire");
```

REQUÊTES SQL :

Répertorier tous les employés en CDI :

```
SELECT Personne.nom, Personne.prenom, Employe.fonction
FROM Personne
JOIN Employe ON Personne.id = Employe.personne_id
WHERE Employe.date_fin IS NULL;
```

DISTINCT + JOIN

Récupérer tous les employés et l'établissement dans lesquels ils travaillent :

```
SELECT DISTINCT(P.nom), ET.nom AS nom_etablissement
FROM personne P
JOIN employe E ON P.id = E.personne_id
JOIN Travailler T ON E.id = T.employe_id
JOIN Etablissement ET ON T.etablissement_id = ET.id
```

DELETE

Une personne est décédée, il faudrait ne plus la renseigner dans la base de donnée

```
DELETE FROM Personne
WHERE id = 5;
```

JOIN + ORDER BY + LIMIT

Lister les 5 premiers établissements avec leurs adresses complète triés par code postal :

```
SELECT Etablissement.nom, Adresse.numero_rue, Adresse.nom_rue, Adresse.code_postal
FROM Etablissement
JOIN Adresse ON Etablissement.adresse_id = Adresse.id
ORDER BY Adresse.code_postal
LIMIT 5;
```

LIKE

Trouver les projets qui ont 'Rénovation' dans leurs description :

```
SELECT *
FROM Projet
WHERE description LIKE '%Rénovation%';
```

BETWEEN

Afficher les projets dont le budget est compris entre 20000 et 50000 euros :

```
SELECT *
FROM Projet
WHERE montant_budget BETWEEN 20000 AND 50000;
```

UPDATE

Sophie Martin vient de se mariée, mettre à jour sa situation :

```
UPDATE Personne
SET situation = 'mariée'
WHERE id = 4
```

TRUNCATE TABLE

Les projets sont terminés, il faudrait vider complètement la table projet :

```
TRUNCATE TABLE projet ;
```

EXPLAIN

Obtenir des informations sur l'exécution d'une requête pour lister les employés et leurs fonctions :

```
EXPLAIN SELECT Personne.nom, Personne.prenom, Employe.fonction
FROM Personne
JOIN Employe ON Personne.id = Employe.personne_id;
```

CASE + ORDER BY + LIKE

Pour catégoriser les employés selon leur fonction :

```
SELECT personne_id, fonction,
CASE
  WHEN fonction LIKE '%Responsable%' THEN 'Niveau 1'
  WHEN fonction LIKE '%Directeur%' THEN 'Niveau 2'
  ELSE 'Autre Niveau'
END AS niveau_hierarchique
FROM Employe
ORDER BY niveau_hierarchique ASC
```

JOIN + GROUP BY + HAVING

Retourne le nom des personnes ayant plus d'un emploi et son nombre :

```
SELECT Personne.id, Personne.nom, Personne.prenom, COUNT(Employe.id) AS nombre_emplois
FROM Personne
JOIN Employe ON Personne.id = Employe.personne_id
GROUP BY Personne.id, Personne.nom, Personne.prenom
HAVING COUNT(Employe.id) > 1;
```

IN

Sélectionner les établissements appartenant à la liste de code postaux 01330 et 01340 :

```
SELECT *
FROM Etablissement
WHERE adresse_id IN (SELECT id FROM Adresse WHERE code_postal IN (01330, 01340));
```

UNION

Obtenir une liste complète des projets ainsi que le nom des personnes qui y travaillent en tant qu'employés, dans une seule liste :

```
SELECT 'Projet' AS type, id AS identifiant, nom
FROM Projet

UNION

SELECT 'Employe' AS type, Employe.id AS identifiant, CONCAT(Personne.nom, ' ',
Personne.prenom) AS nom
FROM Employe
JOIN Personne ON Employe.personne_id = Personne.id
JOIN Projet ON Employe.id = Projet.id;
```

ALTER TABLE

On supprime l'année dans budget et on rajoute l'attribut dans Projet car plus cohérent avec notre contexte (vu avec le client)

```
ALTER TABLE BUDGET
DROP annee

ALTER TABLE Projet
ADD montant_budget INT ;
```

DROP TABLE

On décide de supprimer également la table Budget (évolution de notre MCD + MLD)

```
DROP TABLE Budget
```

SOUS-REQUÊTE

Obtenir une liste des personnes travaillant sur des projets nécessitant un budget supérieur à la moyenne des budgets de tous les projets.

```
-- Calculer la moyenne des budgets de tous les projets
WITH MoyenneBudget AS (
  SELECT AVG(montant_budget) AS budget_moyen
  FROM Projet
)

-- Obtenir les personnes travaillant sur des projets avec un budget supérieur à la moyenne
SELECT Personne.nom, Personne.prenom, Projet.nom AS nom_projet, Projet.montant_budget
FROM Personne
JOIN Employe ON Personne.id = Employe.personne_id
JOIN Projet ON Employe.id = Projet.id
CROSS JOIN MoyenneBudget -- Utilisation de la moyenne des budgets
WHERE Projet.montant_budget > (SELECT budget_moyen FROM MoyenneBudget);
```

Requête utile dans notre contexte (BONUS)
Obtenir la liste des projets en cours de réalisation, avec les détails sur les établissements associés, les employés impliqués, et les informations sur le budget alloué.

```
SELECT
        PE.nom AS NomEmploye,
    EM.fonction AS fonctionEmploye,
    P.nom AS nomProjet,
    P.description AS descriptionProjet,
    E.nom AS nomEtablissement,
    P.montant_budget AS montantBudget
FROM Projet P
JOIN Comporter C ON P.id = C.projet_id
JOIN Etablissement E ON C.etablissement_id = E.id
JOIN Travailler T ON E.id = T.etablissement_id
JOIN Employe EM ON T.employe_id = EM.id
JOIN personne PE ON EM.personne_id = PE.id
WHERE P.date_debut <= CURDATE() AND P.date_fin_prevue >= CURDATE()
ORDER BY P.date_debut ASC;
```

INDEXATION :

```
-- Création d'un index sur la colonne 'nom' de la table 'Projet'
CREATE INDEX idx_nom_projet ON Projet(nom);

-- Création d'un index sur la colonne 'personne_id' de la table 'Employe'
CREATE INDEX idx_personne_id_employe ON Employe(personne_id);

-- Création d'un index sur la colonne 'adresse_id' de la table 'Etablissement'
CREATE INDEX idx_adresse_id_etablissement ON Etablissement(adresse_id);

-- Création d'un index sur la colonne 'ville_naissance' de la table 'Personne'
CREATE INDEX idx_ville_naissance_personne ON Personne(ville_naissance);
```

PROCEDURE STOCKEE :

Actualiser la date de fin prévu du projet à la date du jour si le seul employé affecté au projet quitte le projet

```
DELIMITER //

CREATE PROCEDURE MiseAJourDateFinProjet(
  IN employe_id INT
)
BEGIN
  DECLARE projet_id INT;
  DECLARE nb_employes INT;
```

```
  -- Récupération de l'ID du projet lié à l'employé
  SELECT projet_id INTO projet_id
  FROM employe
  JOIN travailler ON employe.id = employe_id
  JOIN etablissement ON etablissement.id = travailler.etablissement_id
  JOIN comporter ON etablissement.id = comporter.etablissement_id
  JOIN projet ON projet.id = projet_id ;

  -- Nombre total d'employés associés à ce projet
  SELECT COUNT(*) INTO nb_employes  FROM employe JOIN travailler ON employe.id =
employe_id
  JOIN etablissement ON etablissement.id = travailler.etablissement_id
  JOIN comporter ON etablissement.id = comporter.etablissement_id
  JOIN projet ON projet.id = projet_id ;

  -- Si l'employé supprimé est le seul employé associé à ce projet, mise à jour de la date de fin du
projet
  IF nb_employes = 0 THEN
    UPDATE Projet
    SET date_fin_prevue = CURRENT_DATE
    WHERE id = projet_id;
  END IF;
END //

DELIMITER ;
```

TRIGGER :

```
DELIMITER //

CREATE TRIGGER ApresSuppressionEmploye
AFTER DELETE ON travailler
FOR EACH ROW
BEGIN
  -- Appel de la procédure stockée pour mettre à jour la date de fin du projet
  CALL MiseAJourDateFinProjet(OLD.employe_id);
END //

DELIMITER ;
```

VUES :

Lister l’ensemble des personnes rattachés à une ou plusieurs adresses

```
CREATE VIEW VuePersonneAdresse AS
SELECT DISTINCT R.id AS rattacher_id, P.nom, P.prenom, P.sexe, P.date_naissance,
P.ville_naissance, P.nationalite,
      A.numero_rue, A.nom_rue, A.code_postal
```

```
FROM Rattacher R
JOIN Personne P ON R.personne_id = P.id
JOIN Adresse A ON R.adresse_id = A.id;
```

Lister les employés et les établissements auxquels ils sont liés.

```
CREATE VIEW VueEmployeEtablissement AS
SELECT T.id AS travailler_id, P.nom, P.prenom, P.sexe, P.date_naissance, P.ville_naissance,
P.nationalite,
    E.fonction, E.date_embauche, E.date_fin, E.domaine,
    T.etablissement_id, ET.nom AS etablissement_nom, ET.description AS
etablissement_description
FROM Travailler T
JOIN Employe E ON T.employe_id = E.id
JOIN Personne P ON E.personne_id = P.id
JOIN Etablissement ET ON T.etablissement_id = ET.id;
```