Rapport de Projet en Base de Données

Réalisé Par : Lilian Gouble
Meriem Nouira
Will Souvannavong
Bilel Rahmouni

Unité d'enseignement : Base de Données

Professeur Encadreur : M. Bougueroua

Année : 2023-2024

Table de Matieres

Introduction

Le présent rapport constitue le compte-rendu du projet de Base de Données mené par notre équipe composée de quatre membres, Will Souvannavong, Meriem Nouira, Lilian Gouble et Bilel Rahmouni. Notre objectif principal était de concevoir et implémenter une base de données pour répondre aux besoins liés à la formation par apprentissage. Pour ce faire, nous avons exploité des données accessibles sur le site gouvernemental français dédié à l'éducation et à la formation professionnelle.

Description du Projet

Dans le cadre du projet, nous avons choisi de nous concentrer sur les données du fichier "Apprentissage Effectif détaillé 2009 – 2010" provenant du site gouvernemental. Ce fichier contient des informations précieuses sur la formation par apprentissage, que nous avons échantillonnées pour n'inclure que les 2048 premières lignes. Ces données ont été sélectionnées pour leur pertinence et leur représentativité dans le domaine de l'éducation et de la formation professionnelle.

1. Besoins auxquels le Projet Répond

Le projet s'attache à répondre aux besoins des élèves sur plusieurs aspects :

Orientation Géographique :
Nous visons à identifier les villes offrant une variété de formations par apprentissage afin de faciliter l'accès à l'éducation pour les apprenants résidant dans différentes régions. En comprenant la répartition géographique des centres de formation, nous pouvons aider les élèves à choisir des programmes proches de leur lieu de résidence, facilitant ainsi leur participation à la formation. Cette étude géographique permet aussi aux écoles de savoir vers quelle zone / villes se tourner pour y installer une nouvelle antenne. L'enjeux géographique est donc de taille pour les Étudiants mais aussi pour les Écoles.

Préférences en Matière de Formation :
Nous cherchons à comprendre les préférences des élèves en termes de type de formation recherchée. En analysant les types de diplômes proposés dans les différents centres de formation, nous pouvons fournir des recommandations personnalisées aux apprentis, les aidant ainsi à choisir des programmes qui correspondent à leurs intérêts et à leurs objectifs professionnels.

Accessibilité pour les Apprenants Handicapés :
Nous nous engageons à garantir l'accessibilité des formations par apprentissage pour les apprenants handicapés. En examinant les habilitations handicap des centres de formation et en analysant leur disponibilité géographique, nous pouvons identifier les opportunités d'éducation inclusives et soutenir l'égalité des chances pour tous les apprenants.

2. Données à Traiter

Les données à traiter comprennent les informations suivantes :

- Types de diplômes offerts dans les centres de formation.
- Villes proposant le plus grand nombre de formations par apprentissage.
- Proportion Hommes/Femmes inscrits dans les programmes.
- Répartition des apprenants en tant que demi-pensionnaires, externes et internes.
- Nombre moyen de villes desservies par les centres de formation.
- Classement des centres de formation en termes d'effectifs d'apprenants.
- Classement des centres de formation en termes de formation proposées. (Ressemble trop à celle du haut mais pas grave).
- Corrélation entre le genre des apprenants et le type de diplôme.
- Accessibilité des formations pour les apprenants handicapés.
- Répartition des niveaux de diplôme par Département.
- Répartition des villes par secteur d'activité.
- Handicap : Quelle type de diplôme/formation.
- Nombre d'apprentis accueillis par entreprise (Code INSEE).
- Proportion d'apprentis ayant changé de spécialité par rapport à l'année précédente.
- Proportion d'apprentis Handicapé ayant changé de spécialité par rapport à l'année précédente.
- Lien entre spécialité de formation et département de l'entreprise.
- Répartition géographique des apprentis selon leur âge et formation.
- Impact du niveau de diplôme des parents sur le choix de formation de l'apprenti.
- Etude mobilité interdépartementale des apprentis pour leur formation.

En analysant ces données, notre projet vise à fournir des informations pertinentes et précieuses pour aider les élèves à prendre des décisions éclairées concernant leur orientation professionnelle et éducative.

Modèle Entité-Association

1. Explication de nos choix

Nous avons structuré notre base de données en créant plusieurs tables distinctes à partir du fichier Excel initial, chacune représentant une entité principale avec ses attributs. Voici un aperçu des tables que nous avons créées :

Organisation (ref_og, id_og, libelle_og);

Entreprise (ref_entreprise, code_insee_entreprise, code_naf_entreprise, depart_entreprise);

Etablissement (ref_etab, id_etab, nom_complet_cfa, code_uai_site, nom_site_formation, adresse1_site, adresse2_site, adresse3_site, code_commune_site_insee, code_postal_site, libelle_ville_site, #ref_og);

Formation (ref_formation, id_siteformation, duree_formation_mois, annee_formation, num_section, #ref_etab, #ref_diplome);

Jeune (ref_jeune, age_jeune_decembre, handicap_oui_non_vide, code_sexe, code_qualite, code_nationalite, code_pcs, libelle_pcs_parent, code_statut_jeune, code_depart_jeune_insee, code_commune_jeune_insee, code_postal_jeune, libelle_ville_jeune, #ref_formation);

Diplomes (ref diplome, diplome, libelle_diplome, code_niveau, type_diplome, code_groupe_specialite, libelle_specialite, libelle_specialite_com, code_origine_prec_cfa, libelle_origine_prec_cfa, code_origine_annee_prec, libelle_origine_annee_prec);

Entreprise_Formation (#ref_entreprise, #ref_formation);

Nous avons également supprimé les attributs "sexe", "libelle_statut_jeune" et "libelle_qualite" de la table Jeune ainsi que "numero_section" de la table Formation car ils étaient redondants avec les attributs “code_qualite”, “code_statut_jeune”, "code_sexe" et "num_section" respectivement.

Nous avons établi des relations entre ces tables :

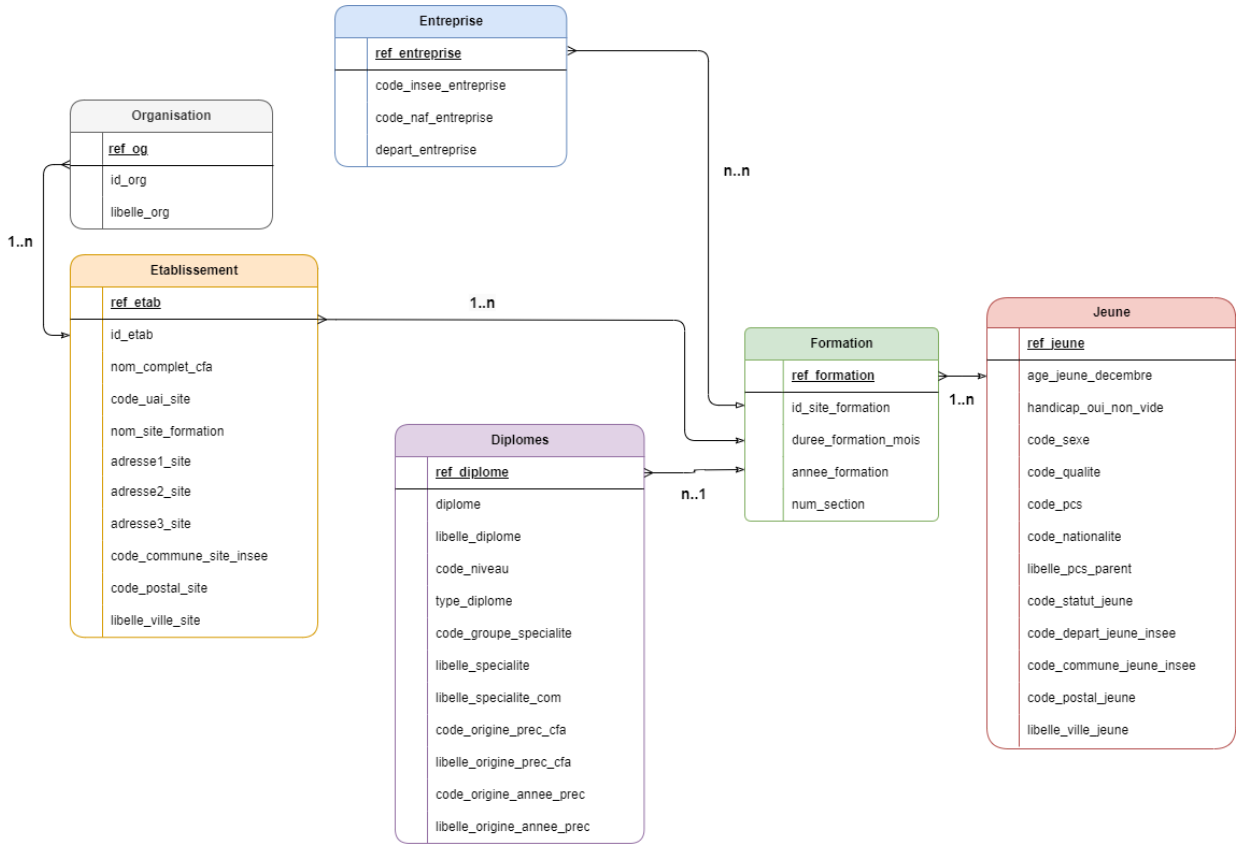
Dans la table Etablissement, la clé étrangère ref_og fait référence à la clé primaire ref_og de l'Organisation pour établir une relation avec la table Organisation.

La table Formation utilise les clés étrangères ref_etab (de Etablissement), ref_entreprise (de Entreprise) et ref_diplome (de Diplomes) pour établir des relations avec les tables Etablissement, Entreprise et Diplomes respectivement.

La table Jeune utilise la clé étrangère ref_formation pour établir une relation avec la table Formation.

Ce modèle de base de données offre une vue organisée des différentes entités et de leurs relations, facilitant ainsi la gestion et l'analyse des données.

2. Schéma du modèle entité relationnel

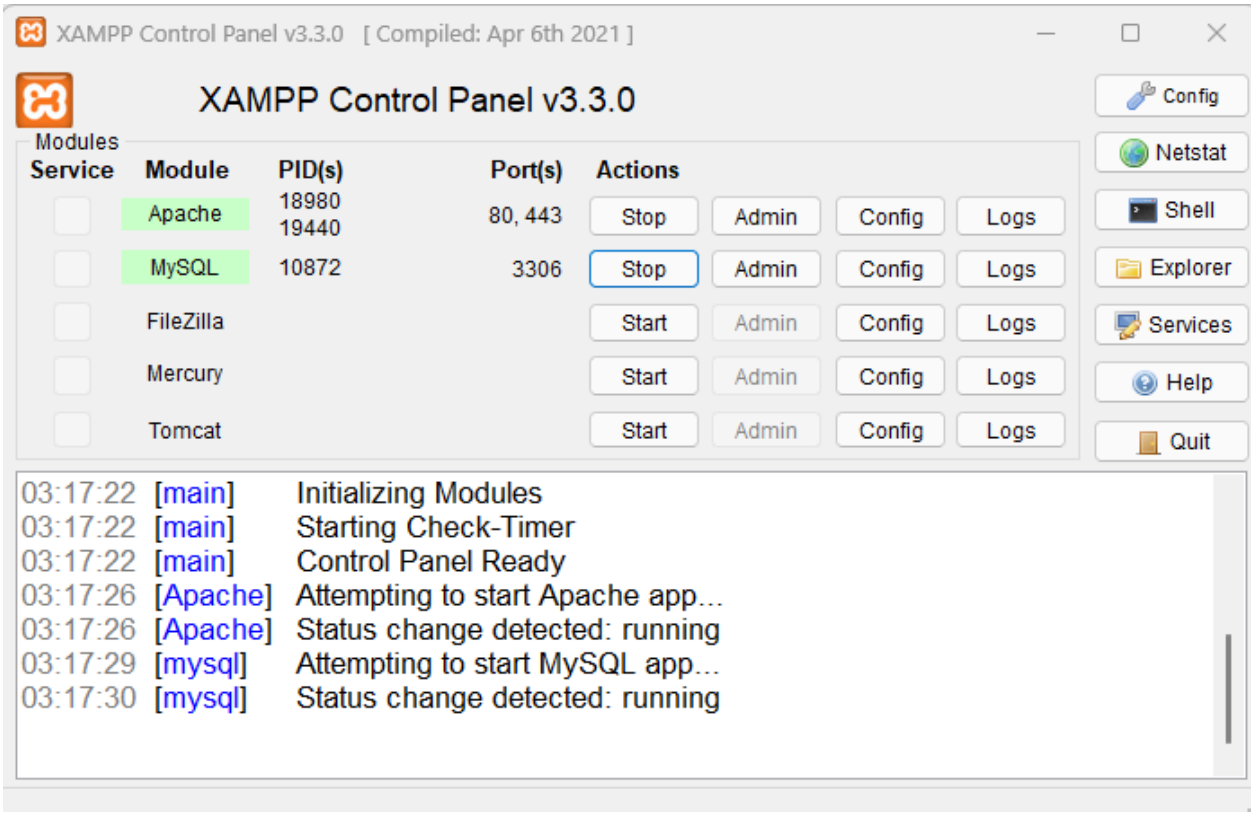


Mise en place de la base de données

1. Mise en place de l'environnement

Pour créer notre base de données, nous avons utilisé l'environnement de développement XAMPP, qui fournit un serveur Apache, MySQL et PHP. Voici les étapes que nous avons suivies pour mettre en place notre base de données :

Installation de XAMPP :
Nous avons téléchargé et installé XAMPP à partir du site officiel, en choisissant la version compatible avec notre système d'exploitation. Une fois l'installation terminée, nous avons lancé XAMPP et démarré les services Apache et MySQL.

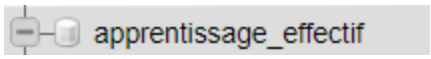


Accès à PHPMyAdmin :

Après avoir démarré les services Apache et MySQL, nous avons accédé à l'interface PHPMyAdmin en ouvrant un navigateur Web et en naviguant vers l'URL http://localhost/phpmyadmin/. Cela nous a permis d'accéder à une interface conviviale pour gérer notre base de données MySQL.

Création de la Base de Données :

Dans PHPMyAdmin, nous avons créé une nouvelle base de données en utilisant l'interface graphique fournie. Nous avons nommé notre base de données selon nos besoins, par exemple "apprentissage_effectif".



2. Création des Tables

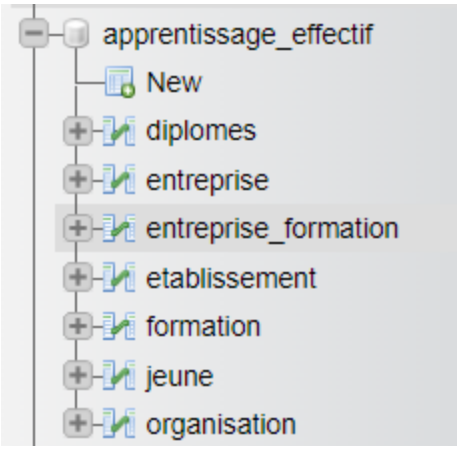
À l'aide de l'onglet "SQL" de PHPMyAdmin, nous avons exécuté des requêtes SQL pour créer les différentes tables de notre base de données. Voici les requêtes que nous avons utilisées pour créer les tables :

```sql
-- Création de la table Organisation
CREATE TABLE Organisation (
    ref_og VARCHAR(36) PRIMARY KEY,
    id_og INT,
    libelle_og VARCHAR(255)
);

-- Création de la table Entreprise
CREATE TABLE Entreprise (
    ref_entreprise VARCHAR(36) PRIMARY KEY,
    code_insee_entreprise INT,
    code_naf_entreprise VARCHAR(50),
    depart_entreprise INT
);

-- Création de la table Etablissement
CREATE TABLE Etablissement (
```

```
    ref_etab VARCHAR(36) PRIMARY KEY,
    id_etab INT,
    nom_complet_cfa VARCHAR(255),
    code_uai_site VARCHAR(16),
    nom_site_formation VARCHAR(255),
    adressel_site VARCHAR(255),
    adresse2_site VARCHAR(255),
    adresse3_site VARCHAR(255),
    code_commune_site_insee INT,
    code_postal_site INT,
    libelle_ville_site VARCHAR(255),
    ref_og VARCHAR(36),
    ref_entreprise VARCHAR(36),
    FOREIGN KEY (ref_og) REFERENCES Organisation(ref_og),
    FOREIGN KEY (ref_entreprise) REFERENCES Entreprise(ref_entreprise)
);

-- Création de la table Diplomes
CREATE TABLE Diplomes (
    ref_diplome VARCHAR(36) PRIMARY KEY,
    diplome VARCHAR(16),
    libelle_diplome VARCHAR(255),
    code_niveau INT,
    type_diplome VARCHAR(50),
    code_groupe_specialite INT,
    libelle_specialite VARCHAR(255),
    libelle_specialite_com VARCHAR(255),
    code_origine_prec_cfa INT,
    libelle_origine_prec_cfa VARCHAR(255),
    code_origine_annee_prec INT,
    libelle_origine_annee_prec VARCHAR(255)
);

-- Création de la table Formation
CREATE TABLE Formation (
    ref_formation VARCHAR(36) PRIMARY KEY,
```

```
    id_siteformation INT,
    duree_formation_mois INT,
    annee_formation INT,
    num_section INT,
    ref_etab VARCHAR(36),
    ref_entreprise VARCHAR(36),
    ref_diplome VARCHAR(36),
    FOREIGN KEY (ref_etab) REFERENCES Etablissement(ref_etab),
    FOREIGN KEY (ref_diplome) REFERENCES Diplomes(ref_diplome)
);

-- Création de la table Entreprise_Formation
CREATE TABLE Entreprise_Formation (
    ref_formation VARCHAR(36) PRIMARY KEY,
    ref_entreprise VARCHAR(36),
    FOREIGN KEY (ref_entreprise) REFERENCES Entreprise(ref_entreprise),
    FOREIGN KEY (ref_formation) REFERENCES Formation(ref_formation),
);

-- Création de la table Jeune
CREATE TABLE Jeune (
    ref_jeune VARCHAR(36) PRIMARY KEY,
    age_jeune_decembre INT,
    handicap_oui_non_vide VARCHAR(50),
    code_sexe ENUM('M','F'),
    code_qualite VARCHAR(255),
    code_nationalite INT,
    code_pcs INT,
    libelle_pcs_parent VARCHAR(255),
    code_statut_jeune ENUM('A','X'),
    code_depart_jeune_insee INT,
    code_commune_jeune_insee INT,
    code_postal_jeune INT,
    libelle_ville_jeune VARCHAR(255),
    ref_formation VARCHAR(36),
    FOREIGN KEY (ref_formation) REFERENCES Formation(ref_formation)
);
```

Ce qui nous génère les tables correspondantes :



3. Transfert de Données depuis Excel vers la Base de Données

Par la suite, il est observé qu'aucune colonne ne possède des données uniques. C'est pourquoi les valeurs ref_og, ref_etab, ref_diplome, ref_formation, ref_entreprise sont créées avec des valeurs UUID4. Ensuite, un script est élaboré pour ajouter ces colonnes avec des valeurs uniques afin de faciliter l'établissement de relations entre les tables :

```
import pandas as pd
import uuid

# Load the CSV file into a DataFrame
csv_file = 'bulk_data_2009_2010.csv'  # Replace with your CSV file path
df = pd.read_csv(csv_file)

# Generate unique values for each column
df['ref_og'] = [str(uuid.uuid4()) for _ in range(len(df))]
df['ref_etab'] = [str(uuid.uuid4()) for _ in range(len(df))]
df['ref_formation'] = [str(uuid.uuid4()) for _ in range(len(df))]
df['ref_diplome'] = [str(uuid.uuid4()) for _ in range(len(df))]
df['ref_entreprise'] = [str(uuid.uuid4()) for _ in range(len(df))]
df['ref_jeune'] = [str(uuid.uuid4()) for _ in range(len(df))]

# Append the new columns to the CSV file
df.to_csv('bulk_with_primary.csv', index=False)
```

Après cela, on execute le script “insert_elements.py” qui permet de remplir les tables de la base de données respectivement selon le nom des colonnes :

```
import pandas as pd
import mysql.connector

#Lecture du fichier CSV
df = pd.read_csv('bulk_with_primary.csv')

#Remplacer les NaN par None dans le DataFrame
df = df.where(pd.notnull(df), None)

#Connexion à la base de données MySQL en spécifiant le plugin d'authentification
conn = mysql.connector.connect(
    host='localhost',
    port='3306',
    user='root',
    password='',
    database='apprentissage_effectif',
    auth_plugin='mysql_native_password'
)

cursor = conn.cursor()

try:
    # Insertion dans les tables
    for index, row in df.iterrows():
        # Insertion dans la table Organisation
        sql = "INSERT INTO Organisation (id_og, libelle_og) VALUES (%s, %s)"
        val = (row.get('id_og'), row.get('libelle_og'))
        cursor.execute(sql, val)

        # Insertion dans la table Entreprise
        sql = "INSERT INTO Entreprise (code_insee_entreprise, code_naf_entreprise, depart_entreprise)
VALUES (%s, %s, %s)"
```

```
        val = (row.get('code_insee_entreprise'), row.get('code_naf_entreprise'),
row.get('depart_entreprise'))
        cursor.execute(sql, val)

        # Insertion dans la table Etablissement
        sql = "INSERT INTO Etablissement (id_etab, nom_complet_cfa, code_uai_site,
nom_site_formation, adresse1_site, adresse2_site, adresse3_site, code_commune_site_insee,
code_postal_site, libelle_ville_site, ref_og, ref_entreprise) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s)"
        val = (row.get('id_etab'), row.get('nom_complet_cfa'), row.get('code_uai_site'),
row.get('nom_site_formation'), row.get('adresse1_site'), row.get('adresse2_site'),
row.get('adresse3_site'), row.get('code_commune_site_insee'), row.get('code_postal_site'),
row.get('libelle_ville_site'), row.get('ref_og'), row.get('ref_entreprise'))
        cursor.execute(sql, val)

        # Insertion dans la table Diplomes
        sql = "INSERT INTO Diplomes (diplome, libelle_diplome, code_niveau, type_diplome,
code_groupe_specialite, libelle_specialite, libelle_specialite_com, code_origine_prec_cfa,
libelle_origine_prec_cfa, code_origine_annee_prec, libelle_origine_annee_prec) VALUES (%s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s)"
        val = (row.get('diplome'), row.get('libelle_diplome'), row.get('code_niveau'),
row.get('type_diplome'), row.get('code_groupe_specialite'), row.get('libelle_specialite'),
row.get('libelle_specialite_com'), row.get('code_origine_prec_cfa'), row.get('libelle_origine_prec_cfa'),
row.get('code_origine_annee_prec'), row.get('libelle_origine_annee_prec'))
        cursor.execute(sql, val)

        # Insertion dans la table Formation
        sql = "INSERT INTO Formation (duree_formation_mois, annee_formation, num_section, ref_etab,
ref_entreprise, ref_diplome) VALUES (%s, %s, %s, %s, %s, %s)"
        val = (row.get('duree_formation_mois'), row.get('annee_formation'), row.get('num_section'),
row.get('ref_etab'), row.get('ref_entreprise'), row.get('ref_diplome'))
        cursor.execute(sql, val)

        # Insertion dans la table Entreprise_Formation
        sql = "INSERT INTO Entreprise_formation (ref_entreprise, ref_formation) VALUES (%s, %s)"
        val = (row.get('ref_entreprise'), row.get('ref_formation'))
```

```
        cursor.execute(sql, val)



        # Insertion dans la table Jeune
        sql = "INSERT INTO Jeune (age_jeune_decembre, handicap_oui_non_vide, code_sexe,
code_qualite, code_nationalite, code_pcs, libelle_pcs_parent, code_statut_jeune,
code_depart_jeune_insee, code_commune_jeune_insee, code_postal_jeune, libelle_ville_jeune,
ref_formation) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
        val = (row.get('age_jeune_decembre'), row.get('handicap_oui_non_vide'), row.get('code_sexe'),
row.get('code_qualite'), row.get('code_nationalite'), row.get('code_pcs'), row.get('libelle_pcs_parent'),
row.get('code_statut_jeune'), row.get('code_depart_jeune_insee'),
row.get('code_commune_jeune_insee'), row.get('code_postal_jeune'), row.get('libelle_ville_jeune'),
row.get('ref_formation'))
        cursor.execute(sql, val)

    # Valider les changements et fermer la connexion
    conn.commit()
    print("Data inserted successfully.")
except mysql.connector.Error as err:
    print("An error occurred:", err)

#Fermer la connexion
conn.close()
```

Après l'exécution de ce script, les tables de la base de données "apprentissage_effectif" sont peuplées avec les données correspondantes extraites du fichier Excel. En outre, grâce à l'ajout des nouvelles valeurs ref, nous avons pu mettre en place des clés primaires et étrangères uniques, facilitant ainsi l'établissement de relations significatives entre les différentes tables de la base de données.

Requêtes

Dans cette section, nous présentons les services proposés aux utilisateurs à travers une série de requêtes SQL avancées. Ces requêtes sont conçues pour explorer en profondeur les données de notre base "apprentissage_effectif" et répondre à des questions clés. En utilisant des techniques telles que les jointures et les agrégations, nos requêtes fournissent des insights pertinents sur les types de diplômes, les effectifs d'apprenants, les tendances démographiques, et bien plus encore. En analysant ces données, notre projet vise à offrir des informations précieuses pour guider les choix d'orientation professionnelle et éducative des utilisateurs.

1. **Analyse des Tendances de Formation :** Offrir une vue d'ensemble des tendances en matière de formations les plus populaires et émergentes.

```
SELECT libelle_diplome, COUNT(*) AS nombre_apprenants
FROM Formation
JOIN Diplomes ON Formation.ref_diplome = Diplomes.ref_diplome
GROUP BY libelle_diplome
ORDER BY nombre_apprenants DESC;
```

2. **Analyse de l'Accessibilité :** Évaluer l'accessibilité des formations pour les apprenants handicapés.

```
SELECT
    IFNULL(handicap_oui_non_vide, 'Non précisé') AS handicap_oui_non_vide,
    COUNT(*) AS nombre_apprenants_handicapes
FROM Jeune
GROUP BY handicap_oui_non_vide;
```

3. **Analyse de la Répartition Géographique :** Identifier les régions géographiques offrant le plus grand nombre de formations par apprentissage.

```
SELECT libelle_ville_site, COUNT(*) AS nombre_formations
FROM Etablissement
GROUP BY libelle_ville_site
ORDER BY nombre_formations DESC;
```

4. **Évaluation de la Diversité des Apprenants :** Examiner la diversité démographique des apprenants en termes de sexe, âge, et handicap.

```
SELECT code_sexe, COUNT(*) AS nombre_apprenants
FROM Jeune
GROUP BY code_sexe;
```

5. **Suivi de la Mobilité Interdépartementale :** Examiner les déplacements des apprenants entre les départements pour leurs formations.

```
SELECT code_depart_jeune_insee, COUNT(*) AS nombre_apprenants
FROM Jeune
GROUP BY code_depart_jeune_insee;
```

6. **Analyse de la Durée des Formations :** Examiner la distribution des durées de formation pour chaque type de diplôme.

```
SELECT libelle_diplome, AVG(duree_formation_mois) AS duree_formation_moyenne
FROM Formation
JOIN Diplomes ON Formation.ref_diplome = Diplomes.ref_diplome
GROUP BY libelle_diplome;
```

7. **Analyse de la Répartition des Formations par Année :** Identifier les tendances annuelles en matière de nombre de formations offertes.

```
SELECT annee_formation, COUNT(*) AS nombre_formations
FROM Formation
GROUP BY annee_formation;
```

8. **Analyse de la Répartition des Formations par Secteur d'Activité :** Identifier les secteurs d'activité offrant le plus grand nombre de formations.

```
SELECT code_naf_entreprise, COUNT(*) AS nombre_formations
FROM Formation
JOIN Entreprise ON Formation.ref_entreprise = Entreprise.ref_entreprise
WHERE code_naf_entreprise IS NOT NULL
GROUP BY code_naf_entreprise;
```

9. **Analyse de la Répartition des Diplômes par Niveau :** Identifier la répartition des types de diplômes offerts par niveau de formation.

```
SELECT code_niveau, GROUP_CONCAT(DISTINCT type_diplome SEPARATOR '/') AS types_diplomes,
COUNT(*) AS nombre_diplomes
FROM Diplomes
GROUP BY code_niveau;
```

10. **Analyse de la Répartition des Diplômes par Type :** Identifier la répartition des types de diplômes offerts dans les centres de formation.

```
SELECT diplome, libelle_diplome, COUNT(*) AS nombre_diplomes
FROM Diplomes
GROUP BY diplome
ORDER BY nombre_diplomes DESC;
```

11. **Proportion Hommes/Femmes inscrits dans les programmes :** Examiner la répartition des apprenants par sexe.

```
SELECT code_sexe, COUNT(*) AS nombre_apprenants
FROM Jeune
GROUP BY code_sexe;
```

12. **Répartition des apprenants selon leur statut (demi-pensionnaires, externes, internes) :** Examiner la répartition des apprenants selon leur statut d'hébergement.

```
SELECT code_statut_jeune, COUNT(*) AS nombre_apprenants
FROM Jeune
GROUP BY code_statut_jeune;
```

13. **Classement des centres de formation en termes de types de diplômes offerts :** Classer les centres de formation en fonction du nombre de types de diplômes offerts.

```
SELECT E.nom_complet_cfa, COUNT(DISTINCT D.diplome) AS nombre_types_diplomes
FROM Formation F
JOIN Etablissement E ON F.ref_etab = E.ref_etab
JOIN Diplomes D ON F.ref_diplome = D.ref_diplome
GROUP BY E.nom_complet_cfa
ORDER BY nombre_types_diplomes DESC;
```

14. **Corrélation entre le genre des apprenants et le type de diplôme :** Examiner la corrélation entre le genre des apprenants et le type de diplôme qu'ils poursuivent.

```
SELECT libelle_diplome,
    ROUND((SUM(CASE WHEN code_sexe = 'F' THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2) AS
pourcentage_femmes
FROM Jeune J
JOIN Formation F ON J.ref_formation = F.ref_formation
JOIN Diplomes D ON F.ref_diplome = D.ref_diplome
GROUP BY libelle_diplome
ORDER BY libelle_diplome;
```

15. **Nombre d'apprentis accueillis par entreprise (Code INSEE) :** Calculer le nombre d'apprentis accueillis par entreprise selon le code INSEE.

```
SELECT E.code_insee_entreprise, COUNT(*) AS nombre_apprentis
FROM Formation F
JOIN Entreprise E ON F.ref_entreprise = E.ref_entreprise
WHERE code_insee_entreprise IS NOT NULL
GROUP BY E.code_insee_entreprise;
ORDER BY nombre_apprentis DESC;
```

16. **Analyse des Types de Diplômes par Niveau** : Examiner la répartition des types de diplômes offerts par niveau de formation en distinguant les différentes filières (ex: "Technologique", "Professionnel").

```
SELECT code_niveau, filiere, GROUP_CONCAT(DISTINCT type_diplome SEPARATOR '/') AS
types_diplomes, COUNT(*) AS nombre_diplomes
FROM Diplomes
GROUP BY code_niveau, filiere;
```

17. **Classement des Établissements par Nombre d'Apprenants :** Classer les établissements en fonction du nombre total d'apprenants qu'ils accueillent.

```
SELECT nom_complet_etablissement, COUNT(*) AS nombre_apprenants
FROM Jeune
JOIN Formation ON Jeune.ref_formation = Formation.ref_formation
JOIN Etablissement ON Formation.ref_etab = Etablissement.ref_etab
GROUP BY nom_complet_etablissement
```

```
ORDER BY nombre_apprenants DESC;
```

18. **Proportion d'Apprenants Handicapés par Formation :** Calculer la proportion d'apprenants handicapés pour chaque formation.

```
SELECT libelle_formation,
    COUNT(*) AS nombre_apprenants,
    COUNT(CASE WHEN handicap_oui_non_vide = 'Oui' THEN 1 ELSE NULL END) AS
nombre_apprenants_handicapes,
    ROUND((COUNT(CASE WHEN handicap_oui_non_vide = 'Oui' THEN 1 ELSE NULL END) * 100.0) /
COUNT(*), 2) AS pourcentage_handicapes
FROM Jeune
JOIN Formation ON Jeune.ref_formation = Formation.ref_formation
GROUP BY libelle_formation
ORDER BY pourcentage_handicapes DESC;
```

19. **Répartition des Diplômes par Type :** Examiner la répartition des types de diplômes offerts dans les centres de formation en fonction du niveau.

```
SELECT code_niveau, GROUP_CONCAT(DISTINCT type_diplome SEPARATOR '/') AS types_diplomes,
COUNT(*) AS nombre_diplomes
FROM Diplomes
GROUP BY code_niveau;
```

20. **Évaluation de la Mobilité Interdépartementale par Formation :** Examiner les déplacements des apprenants entre les départements pour chaque formation.

```
SELECT libelle_formation, code_depart_jeune_insee, COUNT(*) AS nombre_apprenants
FROM Jeune
JOIN Formation ON Jeune.ref_formation = Formation.ref_formation
GROUP BY libelle_formation, code_depart_jeune_insee;
```

Visualisation de Données

Dans le cadre de notre analyse des données sur l'apprentissage effectif, nous avons utilisé des requêtes SQL pour extraire des informations pertinentes de notre base de données. Ensuite, nous avons visualisé ces données à l'aide de bibliothèques Python telles que Pandas et Plotly Express. Voici un résumé des visualisations réalisées :

Répartition des apprenants par type de diplôme :
Nous avons utilisé un graphique en violon pour visualiser la répartition des apprenants selon différents types de diplômes. Chaque violon représente un type de diplôme, montrant la distribution des apprenants en termes de nombre.

Répartition géographique des centres de formation :
Une barre horizontale a été utilisée pour représenter le nombre de formations offertes dans différentes villes. Cette visualisation donne un aperçu de la répartition géographique des centres de formation.

Accessibilité des formations pour les apprenants handicapés :
Nous avons créé un graphique circulaire (camembert) pour montrer la proportion d'apprenants handicapés par rapport à ceux qui ne le sont pas. Cela nous permet de comprendre l'accessibilité des formations pour les personnes handicapées.

Répartition des niveaux de diplôme par département :
Enfin, nous avons utilisé un graphique à barres pour illustrer la répartition des différents niveaux de diplômes dans chaque département. Cela nous donne un aperçu de la diversité des niveaux de diplômes offerts à travers les départements.

Ces visualisations nous ont permis de mieux comprendre les données sur l'apprentissage effectif et d'identifier des tendances ou des modèles significatifs.

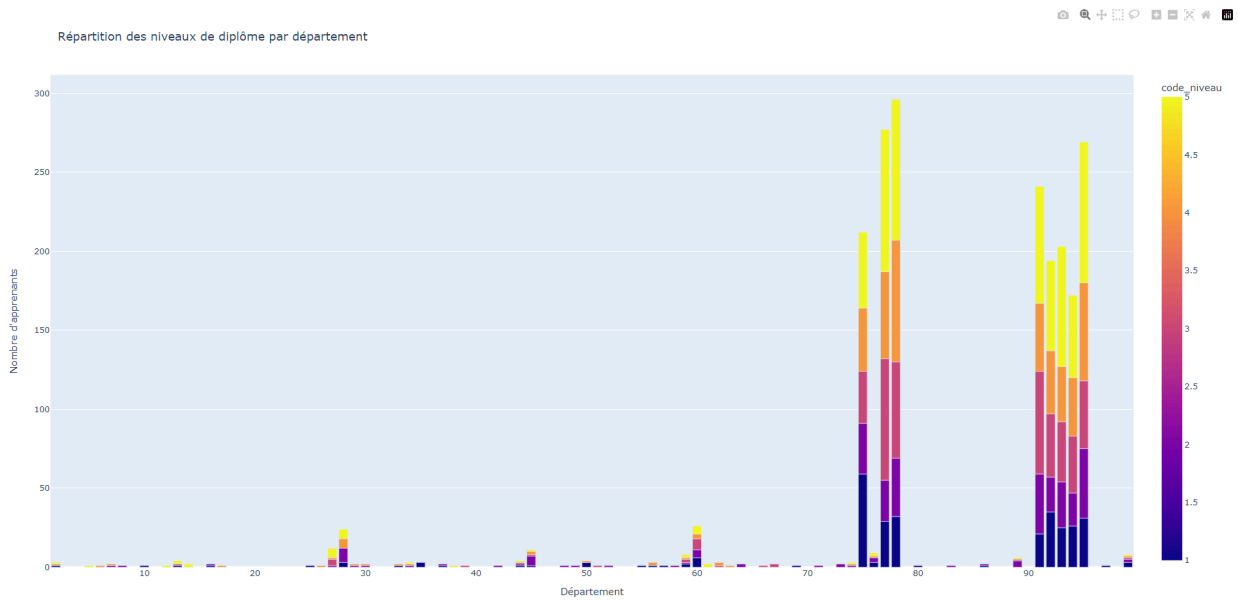
Voici le script nommé “data_visualisation.py” que nous avons développé pour visualiser les données de notre base de données sur l'apprentissage effectif :

```
import mysql.connector
import pandas as pd
import plotly.express as px

# Se connecter à la base de données
conn = mysql.connector.connect(
    host='localhost',
    port='3306',
    user='root',
    password='',
    database='apprentissage_effectif',
    auth_plugin='mysql_native_password'
)

# Exécuter une requête SQL pour récupérer les données
query = """
    SELECT libelle_diplome, COUNT(*) AS nombre_apprenants
    FROM Formation
    JOIN Diplomes ON Formation.ref_diplome = Diplomes.ref_diplome
    GROUP BY libelle_diplome
    ORDER BY nombre_apprenants DESC;
"""
df = pd.read_sql(query, conn)

# Créer une visualisation en violon avec un titre ajusté
fig = px.violin(df, y='libelle_diplome', x='nombre_apprenants', orientation='h', title='Répartition des apprenants par type de diplôme',
                labels={'nombre_apprenants': 'Nombre d\'apprenants', 'libelle_diplome': 'Type de diplôme'},
                points='all', box=True, hover_name='libelle_diplome', color='nombre_apprenants')

# Personnaliser la mise en page
fig.update_layout(xaxis_title='Nombre d\'apprenants', yaxis_title='Type de diplôme')
fig.update_traces(meanline_visible=True, jitter=0.05, scalemode='count')
```

```
# Ajuster la taille du texte du titre des diplômes
fig.update_layout(title_font=dict(size=18))
# Afficher la visualisation
fig.show()

#-------------------------------------------------------------------------------------
# Exécuter une requête SQL pour récupérer les données
query_geo = """
    SELECT libelle_ville_site, COUNT(*) AS nombre_formations
    FROM Etablissement
    GROUP BY libelle_ville_site
    ORDER BY nombre_formations DESC;
"""
df_geo = pd.read_sql(query_geo, conn)

# Créer une visualisation de la répartition géographique
fig_geo = px.bar(df_geo, x='libelle_ville_site', y='nombre_formations', title='Répartition géographique
des centres de formation')
# Afficher la visualisation
fig_geo.show()
#-------------------------------------------------------------------------------------
# Exécuter une requête SQL pour récupérer les données
query_handicap = """
    SELECT handicap_oui_non_vide, COUNT(*) AS nombre_apprenants
    FROM Jeune
    GROUP BY handicap_oui_non_vide;
"""
df_handicap = pd.read_sql(query_handicap, conn)
# Créer une visualisation de l'accessibilité pour les handicapés
fig_handicap = px.pie(df_handicap, names='handicap_oui_non_vide', values='nombre_apprenants',
title='Accessibilité des formations pour les apprenants handicapés')
# Afficher la visualisation
fig_handicap.show()
#-------------------------------------------------------------------------------------
# Exécuter une requête SQL pour récupérer les données
```

```
query = """
    SELECT code_depart_jeune_insee, code_niveau, COUNT(*) AS nombre_apprenants
    FROM Jeune
    JOIN Formation ON Jeune.ref_formation = Formation.ref_formation
    JOIN Diplomes ON Formation.ref_diplome = Diplomes.ref_diplome
    GROUP BY code_depart_jeune_insee, code_niveau;
"""
df = pd.read_sql(query, conn)
# Créer une visualisation de la répartition des niveaux de diplôme par département
fig = px.bar(df, x='code_depart_jeune_insee', y='nombre_apprenants', color='code_niveau',
             title='Répartition des niveaux de diplôme par département',
             labels={'code_depart_jeune_insee': 'Département', 'nombre_apprenants': 'Nombre
d\'apprenants'})
# Afficher la visualisation
fig.show()
# Fermer la connexion
conn.close()
```

Ce code nous permet de générer des pages web contenant différents graphiques basés sur les données de notre base de données sur l'apprentissage effectif. Parmi ces graphiques, l'un d'eux représente la répartition des niveaux de diplômes par département :



Interface pour l'exploitation des données

Dans le cadre de notre projet sur l'apprentissage effectif, notre équipe a développé une interface d'exploitation des données pour permettre aux utilisateurs d'interagir facilement avec les informations contenues dans notre base de données. Cette interface vise à fournir une expérience utilisateur intuitive et conviviale pour l'exploration et l'analyse des données sur les formations, les apprenants et les centres de formation.

1. Conception et développement

Notre équipe a opté pour le développement d'une interface web en utilisant le framework Flask en Python. Ce choix a été motivé par sa simplicité et sa flexibilité, ainsi que par notre familiarité avec cet outil. Nous avons également utilisé des technologies complémentaires telles que MySQL pour la gestion de la base de données et HTML/CSS pour la conception de l'interface utilisateur.

```
from flask import Flask, render_template, request
import mysql.connector

app = Flask(__name__)
# Connexion à la base de données
conn = mysql.connector.connect(
    host='localhost',
    port='3306',
    user='root',
    password='',
    database='apprentissage_effectif',
    auth_plugin='mysql_native_password'
)
# Route pour la page d'accueil
@app.route('/')
def index():
    return render_template('index.html')

# Route pour exécuter une requête SQL
@app.route('/query', methods=['POST'])
```

```
def execute_query():
    query = request.form['query']
    cursor = conn.cursor()
    cursor.execute(query)
    data = cursor.fetchall()
    cursor.close()
    return render_template('result.html', data=data)

if __name__ == '__main__':
    app.run(debug=True)
```

2. Fonctionnalités principales

L'interface d'exploitation des données permet aux utilisateurs de saisir des requêtes SQL via un formulaire web et d'afficher les résultats correspondants. Nous avons implémenté deux fonctionnalités principales :

- Exécution de requêtes SQL : Les utilisateurs peuvent saisir des requêtes SQL dans le formulaire prévu à cet effet sur la page d'accueil de l'interface. Une fois soumise, la requête est envoyée au serveur et exécutée sur la base de données.

- Affichage des résultats : Les résultats de la requête sont ensuite affichés sur une page dédiée, présentant les données sous forme de tableau HTML pour une visualisation claire et concise.
- Cette fonctionnalité offre aux utilisateurs un accès rapide à des analyses spécifiques sans avoir à rédiger de requêtes SQL personnalisées. Chaque bouton correspond à une analyse préconfigurée, permettant aux utilisateurs d'obtenir des insights pertinents en un seul clic.

3. Exécution

Pour exécuter l'interface d'exploitation des données, assurez-vous d'avoir installé les dépendances nécessaires. Les dépendances peuvent être installées en exécutant la commande suivante dans votre terminal :

```
pip install flask mysql-connector-python pandas
```

Une fois les dépendances installées, vous pouvez exécuter le script interface.py. Assurez-vous d'être dans le répertoire contenant le script interface.py avant d'exécuter la commande suivante :

```
python interface.py
```

Cette commande lancera un serveur local sur votre machine. Vous pouvez accéder à l'interface en ouvrant votre navigateur Web préféré et en accédant à l'URL suivante : http://127.0.0.1:5000.