Bases de données et language SQL
Situation d'Apprentissage et d'Evaluation (SAE) 2-01
Conception et implémentation d'une base de données

Nayl Saifoudine et Hassan Ajdahim BUT SD Galton

13/05/2024

Sommaire

1 Présentation de la SAÉ

Pour cette SAÉ, notre groupe est composé de 2 personnes : Hassan Ajdahim et Nayl Saifoudine (Auteur).

1.1 Contexte général

L'objectif de cette SAÉ est de concevoir puis alimenter une base de données permettant de stocker toutes les informations sur les festivals en France. Le travail principal consiste à récupérer des données officielles et à s'aider du langage SQL pour les stocker dans une base de données et de les explorer afin de produire des analyses.

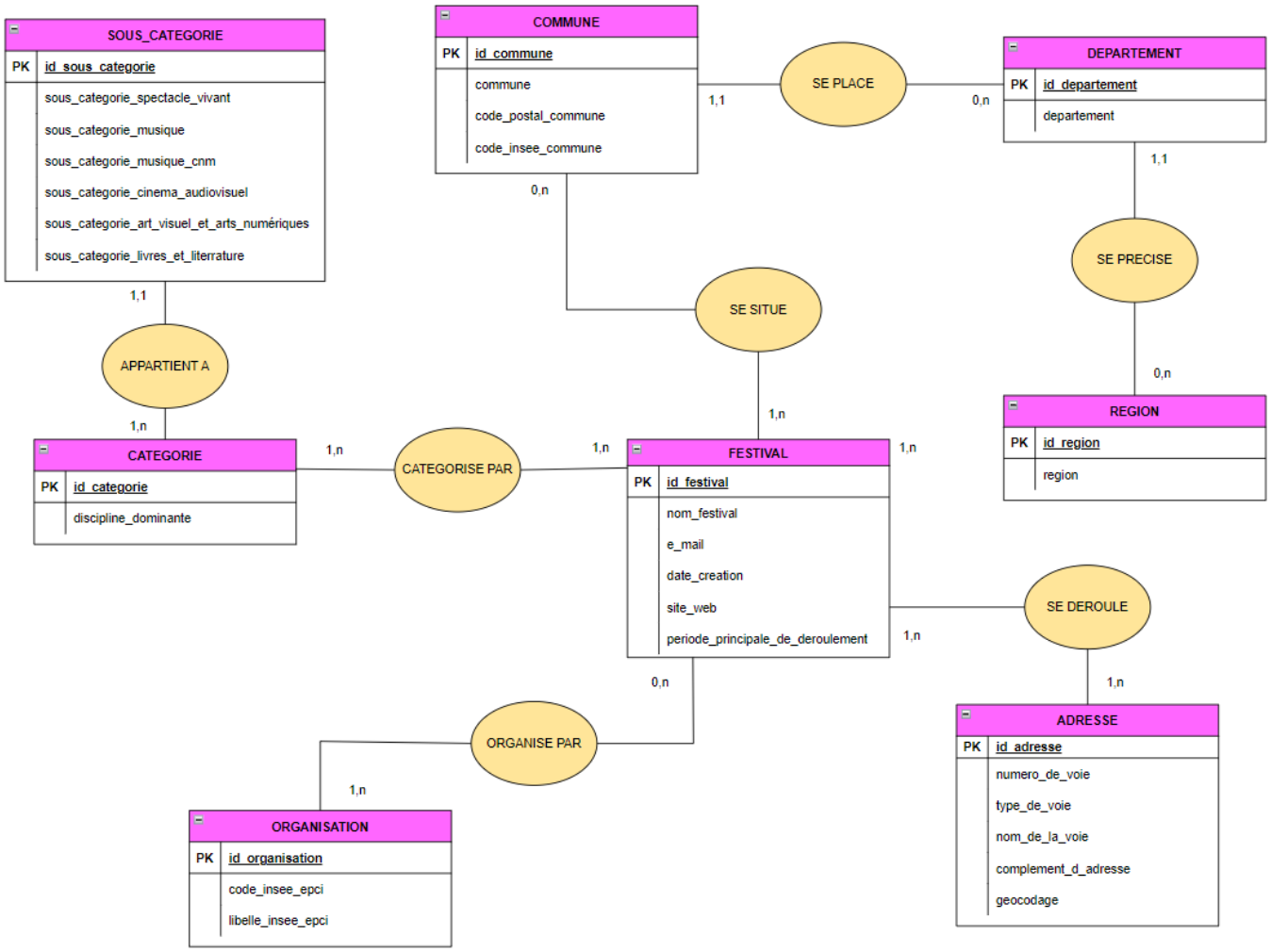
1.2 Description globale de la base de données

Pour cette SAÉ, nous avions comme consigne d'utiliser la base de données nommée "festivals-global-festivals-_-pl.csv" accessible depuis le site du gouvernement : data.gouv.
Cette base est composée de 7 283 lignes et 30 colonnes.
Elle est censée nous donner toutes les données concernant les festivals en france depuis 1900.
Cependant, cette base n'est pas utilisable telle qu'elle. Nous allons donc la transformer en base de données utilisable et pratique pour la recherche d'informations.

2 Modèles Entités-Associations et relationnel

Pour cela, nous allons tout d'abord créer un modèle Entités-Associations permettant de comprendre visuellement la base de données. Pour cela, nous avons utilisé le site internet "drawio.com" permettant de créer des graphiques et diagrammes spécifiques aux bases de données.

2.1 Modèle Entités-Associations



2.2 Description du modèle E-A et Normalisation

Dans cette partie de la description, nous allons nous concentrer sur la description des types associations et nous expliquerons plus en détail les types entités ultérieurement.

1. **APPARTIENT A** Cette relation indique qu'une 'SOUS_CATEGORIE' appartient à une 'CATEGORIE'. La cardinalité est de 1,1 à 1,n, signifiant qu'une catégorie peut avoir plusieurs sous-catégories (au minimum 1), mais une sous-catégorie ne peut avoir qu'une seule et unique catégorie (au minimum 1).

2. **CATEGORISE PAR** Relation entre 'CATEGORIE' et 'FESTIVAL', indiquant qu'un festival est catégorisé par une catégorie. La cardinalité est de 1,n à 1,n, signifiant qu'un festival peut être associé à plusieurs catégories et vice versa.

3. **SE PLACE** Relation entre 'COMMUNE' et 'DEPARTEMENT', indiquant qu'une commune se situe dans un département. La cardinalité est de 1,n à 1,1, signifiant qu'un département peut avoir plusieurs communes mais une commune n'appartient qu'à un seul département.

4. **SE SITUE** Relation entre 'FESTIVAL' et 'COMMUNE', indiquant où se situe le festival. La cardinalité est de 1,n à 1,1, signifiant qu'un festival peut se dérouler dans une seule commune, mais une commune peut accueillir plusieurs festivals.

5. **SE DEROULE** Relation entre 'FESTIVAL' et 'ADRESSE', indiquant quel festival se déroule à quelle adresse. La cardinalité est de 1,n à 1,1, signifiant qu'un festival peut se dérouler dans une seule région, mais une région peut accueillir plusieurs festivals.

6. **SE PRECIS** Relation entre 'DEPARTEMENT' et 'REGION', indiquant qu'un département est situé dans une région. La cardinalité est de 1,n à 1,n, signifiant qu'une adresse peut recevoir plusieurs festivals (au moins 1), et vice-versa.

7. **ORGANISE PAR** Relation entre 'FESTIVAL' et 'ORGANISATION', indiquant qui organise le festival. La cardinalité est de 1,n à 0,n, signifiant qu'un festival peut être organisé par plusieurs organisations (ou aucune) et qu'une organisation peut organiser plusieurs festivals (au minimum 1).

Dans un but pratique, par la suite, on utilisera des noms plus simples pour les types associations tels que 'FESTIVAL_ORGANISATION' à la place de 'ORGANISE PAR'.

Abordons maintenant la question de la normalité du modèle.

Pour la Première Forme Normale (**1NF**), il est impératif que les attributs des tables soient atomiques et non-composites.
Dans notre cas, tous les attributs sont déjà atomiques et non-composites. le modèle est donc déjà en 1NF.

Pour la Deuxième Forme Normale (**2NF**), il est impératif que les parties d'une clé primaire ne permettent pas de déterminer un attribut non-clé de la table.
Dans notre cas, les seules tables où les clés primaires sont multiples sont les tables des types associations. Et dans ces tables, il n'y a aucune partie de la clé qui permette de déterminer un attribut non-clé (car il n' y a pas d'attribut non-clé dans les tables des types associations). Notre modèle est donc déjà en 2NF.

Pour la Troisième Forme Normale (**3NF**), il est impératif que les attributs des tables ne permettent pas de déterminer un autre attribut de cette même table.
Dans notre cas, il existait dans notre base de données des attributs qui ne respectaient pas cette règle. On peut notamment citer les 2 attributs suivants : la décennie de création et l'adresse postale. En effet, par exemple, la décennie de création pouvait être déterminée à partir d'un autre attribut de la même table qui est l'année de création (date_creation). Et l'adresse postale était la concaténation de plusieurs autres tables telles que le numéro de voie (numero_de_voie), le tyep de voie (type_de_voie) ainsi que le nom de la voie (nom_de_la_voie). On pouvait donc à partir de cet attribut déterminer 3 autres attributs, ce qui respecte pas la 3NF. Nous avons donc décidé d'enlever ces 2 attributs de notre base de données.
Après la suppression des attributs gênants, notre base de données est finalement en 3NF.

Pour la Forme Normale de Boyce-Codd (**BCNF**), il est impératif que les attributs des tables ne permettent pas de déterminer une partie de la clé primaire de cette même table.

Dans notre cas, les seules tables où les clés primaires sont multiples sont les tables des types associations. Et dans ces tables, il n'y a aucun attribut (qui permettent de déterminer une partie de la clé). Notre modèle est donc déjà en BCNF.

2.3 Modèle relationnel

A partir du modèle Entités-Associations, nous allons créer le modèle relationnel associé à cette base de données.

- REGION(id_region, region)
- DEPARTEMENT(id_departement, departement, id_region) Avec id_region faisant référence à la table REGION (id_region)
- COMMUNE(code_postal_commune, commune_principale_de_déroulement, code_insee_commune, id_departement) Avec id_departement faisant référence à la table DEPARTEMENT (id_departement)
- ADRESSE(id_adresse, numero_de_voie, type_de_voie, nom_de_la_voie, complement_d_adresse, geocodage)
- CATEGORIE(id_categorie, discipline_dominante)
- SOUS_CATEGORIE(id_sous_categorie, sous_categorie_spectacle_vivant, sous_categorie_musique, sous_categorie_musique_cnn, sous_categorie_cinema_audiovisuel, sous_categorie_art_visuel_et_arts_numeriques, sous_categorie_livres_et_litterature, id_categorie) Avec id_categorie faisant référence à la table CATEGORIE (id_categorie)
- ORGANISATION(id_organisation, code_insee_epci, libelle_insee_epci)
- FESTIVAL(id_festival, nom_festival, e_mail, site_web, decennie_de_creation, periode, id_adresse, id_commune, id_organisation, id_categorie) Avec id_adresse, id_commune, id_organisation et id_categorie faisant référence respectivement aux tables ADRESSE (id_adresse), COMMUNE (id_commune), ORGANISATION(id_organisation) et CATEGORIE (id_categorie)
- FESTIVAL_CATEGORIE(id_festival, id_categorie) Avec id_festival et id_categorie faisant référence respectivement aux tables FESTIVAL (id_festival) et CATEGORIE (id_categorie)
- FESTIVAL_ORGANISATION(id_festival, id_organisation) Avec id_festival et id_organisation faisant référence respectivement aux tables FESTIVAL (id_festival) et ORGANISATION (id_organisation)
- FESTIVAL_COMMUNE(id_festival, id_commune) Avec id_festival et id_commune faisant référence respectivement aux tables FESTIVAL (id_festival) et COMMUNE (id_commune)
- FESTIVAL_ADRESSE(id_festival, id_adresse) Avec id_festival et id_adresse faisant référence respectivement aux tables FESTIVAL (id_festival) et ADRESSE (id_adresse)

2.4 Description du schéma relationnel

Le schéma relationnel est composé de 12 tables :

REGION

L'entité "REGION" stocke des informations sur les régions et comprend la clé primaire 'id_region' et l'attribut 'region' (le nom de la région).

- id_region : Identifiant unique de la région ; INTEGER PRIMARY KEY
- region : Nom de la région ; VARCHAR

DEPARTEMENT

L'entité "DEPARTEMENT" représente les départements français et inclut la clé primaire 'id_departement' ainsi que l'attribut 'departement' (le nom du département). Elle est en relation avec l'entité 'REGION', indiquant le lien entre départements et régions.

- id_departement : Identifiant unique du département ; INTEGER PRIMARY KEY
- departement : Nom du département ; VARCHAR
- id_region : Identifiant unique de la région ; INTEGER

COMMUNE
L'entité 'COMMUNE' stocke des informations sur les communes où se déroulent les festivals. Elle comprend la clé primaire 'id_commune' et des attributs tels que 'commune' (le nom de la commune), 'code_postal_commune' (code postal) et 'code_insee_commune' (code INSEE, un identifiant unique pour les communes en France). Cette entité est en relation avec l'entité 'DEPARTEMENT', représentant le département dans lequel la commune est située.

- code_postal_commune : Identifiant unique de la commune ; INTEGER PRIMARY KEY
- commune_principale_de_deroulement : Nom de la commune ; VARCHAR
- code_insee_commune : Identifiant utilisé par l'INSEE pour chaque commune : INTEGER
- id_departement : Identifiant faisant référence au département ; INTEGER

ADRESSE
L'entité 'ADRESSE' contient des informations détaillées sur les adresses spécifiques des festivals, y compris la clé primaire 'id_adresse', 'numero_de_voie' (numéro de voie), 'type_de_voie' (type de voie), 'nom_de_la_voie' (nom de la voie), 'adresse_postale' (adresse postale), 'complement_adresse' (complément d'adresse), et 'geolocalisation' (coordonnées géographiques).

- id_adresse : Identifiant unique de l'adresse ; INTEGER PRIMARY KEY
- numero_de_voie : Numéro de la voie ; INTEGER
- type_de_voie : Type de la voie (Rue, Avenue, etc) ; VARCHAR
- nom_de_la_voie : Nom de la voie ; VARCHAR
- complement_d_adresse : Nom supplémentaire en cas de besoin de spécification ; VARCHAR

CATEGORIE
L'entité 'CATEGORIE' décrit les catégories principales des festivals avec la clé primaire 'id_categorie' et les attributs tels que 'discipline_dominante' (discipline dominante) et 'id_sous_categorie' (référence à la sous-catégorie). Cette entité est en relation avec 'SOUS_CATEGORIE' pour décrire plus en détail les types de festivals.

- id_categorie : Identifiant unique de la catégorie ; INTEGER PRIMARY KEY
- disipline_dominante : Nom de la discipline dominante ; VARCHAR

SOUS_CATEGORIE
L'entité 'SOUS_CATEGORIE' représente les différentes sous-catégories auxquelles un festival peut appartenir. Elle possède une clé primaire 'id_sous_categorie' qui identifie de manière unique chaque sous-catégorie. Les sous-catégories comprennent des attributs tels que 'sous_categorie_spectacle_vivant' (spectacles vivants), 'sous_categorie_musique' (musique), 'sous_categorie_cinema_audiovisuel' (cinéma et audiovisuel), 'sous_categorie_art_virtuel_et_arts_numeriques' (art virtuel et arts numériques), et 'sous_categorie_livres_et_litterature' (livres et littérature). Cette entité est reliée à l'entité 'CATEGORIE', indiquant que chaque catégorie peut avoir plusieurs sous-catégories.

- id_sous_categorie : Identifiant unique de la sous catégorie ; INTEGER PRIMARY KEY
- sous_categorie_spectacle_vivant : Sous catégorie composée des spectacles vivants ; VARCHAR
- sous_categorie_musique : Sous catégorie composée des catégories de musiques ; VARCHAR
- sous_categorie_musique_cnn : Sous catégorie composée des styles de musiques ; VARCHAR
- sous_categorie_cinema_audiovisuel : Sous catégorie composée des cinémas et de l'audiovisuel ; VARCHAR
- sous_categorie_art_visuel_et_arts_numériques : Sous catégorie composée de l'art visuel et de de l'art numérique ; VARCHAR
- sous_categorie_livres_et_littérature : Sous catégorie composée des livres et littérature ; VARCHAR
- id_categorie : Sous catégorie faisant référence aux catégories de musiques ; VARCHAR

FESTIVAL
L'entité 'FESTIVAL' représente les festivals eux-mêmes, avec des attributs comme 'id_festival' (clé primaire), 'nom_festival' (nom du festival), 'e_mail' (email de contact), 'date_creation' (date de création), 'site_web' (site web du festival), et 'periode_principale_de_deroulement' (période principale de déroulement). Cette entité est en relation avec 'ADRESSE' pour préciser où se déroulent les festivals et 'ORGANISATION' pour indiquer qui organise les festivals.

- id_festival : Identifiant unique du festival ; INTEGER PRIMARY KEY
- nom_festival : Nom du festival ; VARCHAR
- e_mail : Adresse mail du festival ; VARCHAR
- date_de_creation : Année de création du festival ; INTEGER

- site_web : Site internet du festival ; VARCHAR
- e_mail : Adresse mail du festival ; VARCHAR
- periode : Période du festival ; VARCHAR
- id_adresse : Identifiant faisant référence à l'adresse ; INTEGER
- id_commune : Identifiant faisant référence à la commune ; INTEGER
- id_organisation : Identifiant faisant référence à la organisation ; INTEGER
- id_categorie : Identifiant faisant référence à la catégorie ; INTEGER

ORGANISATION

L'entité 'ORGANISATION' décrit les organisations qui gèrent les festivals, avec des attributs tels que 'id_organisation' (clé primaire), 'code_insee_epci' (code type d'organisation), et 'libelle_insee_epci' (libellé du type d'organisation).

- id_organisation : Identifiant unique de l'organisation ; INTEGER PRIMARY KEY
- code_insee_epci : Identifiant de l'INSEE de l'organisation ; INTEGER
- libelle_insee_epci : Nom de l'organisateur ; VARCHAR

FESTIVAL_CATEGORIE

L'entité 'FESTIVAL_CATEGORIE' relie les festivals aux catégories grâce à leurs identifiants tels que 'id_festival' et 'id_categorie' qui forment ensemble la clé primaire.

- id_festival : Identifiant faisant référence à la région ; INTEGER
- id_categorie : Identifiant faisant référence à la catégorie ; INTEGER
- PRIMARY KEY(id_festival, id_categorie)

FESTIVAL_ORGANISATION

L'entité 'FESTIVAL_ORGANISATION' relie les festivals aux catégories grâce à leurs identifiants tels que 'id_festival' et 'id_organisation' qui forment ensemble la clé primaire.

- id_festival : Identifiant faisant référence à la région ; INTEGER
- id_organisation : Identifiant faisant référence à la organisation ; INTEGER
- PRIMARY KEY(id_festival, id_organisation)

FESTIVAL_COMMUNE

L'entité 'FESTIVAL_COMMUNE' relie les festivals aux catégories grâce à leurs identifiants tels que 'id_festival' et 'id_commune' qui forment ensemble la clé primaire.

- id_festival : Identifiant faisant référence au festival : INTEGER
- id_commune : Identifiant faisant référence à la commune : INTEGER
- PRIMARY KEY(id_festival, id_commune)

FESTIVAL_ADRESSE

L'entité 'FESTIVAL_ADRESSE' relie les festivals aux catégories grâce à leurs identifiants tels que 'id_festival' et 'id_adresse' qui forment ensemble la clé primaire.

- id_festival : Identifiant faisant référence au festival : INTEGER
- id_adresse : Identifiant faisant référence à l'adresse : INTEGER
- PRIMARY KEY(id_festival, id_adresse)

3 Peuplement de la base de données

3.1 Script SQL de création des tables

Voici le script SQL permettant la création des tables nécessaires au bon fonctionnement de notre base de donneés.

```sql
DROP TABLE IF EXISTS festival_adresse;
DROP TABLE IF EXISTS festival_organisation;
DROP TABLE IF EXISTS festival_commune;
DROP TABLE IF EXISTS festival_categorie;

DROP TABLE IF EXISTS organisation CASCADE;
DROP TABLE IF EXISTS festival CASCADE;
DROP TABLE IF EXISTS sous_categorie CASCADE;
DROP TABLE IF EXISTS categorie CASCADE;
DROP TABLE IF EXISTS adresse CASCADE;
DROP TABLE IF EXISTS commune CASCADE;
```

```
DROP TABLE IF EXISTS departement CASCADE;
DROP TABLE IF EXISTS region CASCADE;




CREATE TABLE region (
id_region INT PRIMARY KEY,
region VARCHAR
);

CREATE TABLE departement (
id_departement INT PRIMARY KEY,
departement VARCHAR,
id_region INT,
FOREIGN KEY (id_region) REFERENCES region(id_region)
);

CREATE TABLE commune (
id_commune INT PRIMARY KEY,
commune VARCHAR,
code_postal_commune VARCHAR,
code_insee_commune VARCHAR,
id_departement INT,
FOREIGN KEY (id_departement) REFERENCES departement(id_departement)
);

CREATE TABLE adresse (
id_adresse INT PRIMARY KEY,
numero_de_voie VARCHAR,
type_de_voie VARCHAR,
nom_de_la_voie VARCHAR,
complement_d_adresse VARCHAR,
geocodage VARCHAR,
id_commune INT,
FOREIGN KEY (id_commune) REFERENCES commune(id_commune)
);

CREATE TABLE categorie (
id_categorie INT PRIMARY KEY,
discipline_dominante VARCHAR
);

CREATE TABLE sous_categorie (
id_sous_categorie INT PRIMARY KEY,
sous_categorie_spectacle_vivant VARCHAR,
sous_categorie_musique VARCHAR,
sous_categorie_musique_cnm VARCHAR,
sous_categorie_cinema_audiovisuel VARCHAR,
sous_categorie_art_visuel_et_arts_numériques VARCHAR,
sous_categorie_livres_et_litterature VARCHAR,
id_categorie INT,
FOREIGN KEY (id_categorie) REFERENCES categorie(id_categorie)
);


CREATE TABLE organisation (
id_organisation INT PRIMARY KEY,
code_insee_epci VARCHAR,
libelle_insee_epci VARCHAR
);

CREATE TABLE festival (
id_festival INT PRIMARY KEY,
nom_festival VARCHAR,
e_mail VARCHAR,
date_creation VARCHAR,
site_web VARCHAR,
periode_principale_de_deroulement_du_festival VARCHAR,
id_commune INT,
id_adresse INT,
id_organisation INT,
FOREIGN KEY (id_adresse) REFERENCES adresse(id_adresse),
FOREIGN KEY (id_commune) REFERENCES commune(id_commune),
```

```
88        FOREIGN KEY (id_organisation) REFERENCES organisation(id_organisation)
89        );
90
91        CREATE TABLE festival_categorie (
92        id_festival INT,
93        id_categorie INT,
94        PRIMARY KEY (id_festival, id_categorie),
95        FOREIGN KEY (id_festival) REFERENCES festival(id_festival),
96        FOREIGN KEY (id_categorie) REFERENCES categorie(id_categorie)
97        );
98
99        CREATE TABLE festival_organisation (
100       id_festival INT,
101       id_organisation INT,
102       PRIMARY KEY (id_festival , id_organisation),
103       FOREIGN KEY (id_festival) REFERENCES festival(id_festival),
104       FOREIGN KEY (id_organisation) REFERENCES organisation(id_organisation)
105       );
106
107       CREATE TABLE festival_commune (
108       id_festival INTEGER NOT NULL,
109       id_commune INTEGER NOT NULL,
110       PRIMARY KEY (id_festival, id_commune),
111       FOREIGN KEY (id_festival) REFERENCES FESTIVAL(id_festival),
112       FOREIGN KEY (id_commune) REFERENCES COMMUNE(id_commune)
113       );
114
115       CREATE TABLE festival_adresse (
116       id_festival INTEGER NOT NULL,
117       id_adresse INTEGER NOT NULL,
118       PRIMARY KEY (id_festival, id_adresse),
119       FOREIGN KEY (id_festival) REFERENCES festival (id_festival),
120       FOREIGN KEY (id_adresse) REFERENCES adresse (id_adresse)
121       );
122
```

Script SQL de création des tables

3.2 Script SQL de peuplement des tables

Le script Python contient des chemins d'accès spécifiques. Si vous voulez l'utiliser, il faudra les changer pour que le programme fonctionne.

Script Python de création des fichiers textes

```
1         import pandas as pd
2         import os
3
4         # Définir le chemin vers le fichier CSV téléchargé
5         csv_file_path = r'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\festivals
      -global-festivals-_-pl.csv'
6
7         # Lire le fichier CSV en utilisant le séparateur de point-virgule
8         data = pd.read_csv(csv_file_path, sep=';')
9
10        # Remplacer les valeurs manquantes par 'null'
11        data = data.fillna('null')
12
13        # Dictionnaire des tables et des colonnes correspondantes
14        tables_columns = {
15            'region': ['region_principale_de_deroulement'],
16            'departement': ['departement_principal_de_deroulement', '
      region_principale_de_deroulement'],
17            'commune': ['commune_principale_de_deroulement', '
      code_postal_de_la_commune_principale_de_deroulement', 'code_insee_commune', '
      departement_principal_de_deroulement'],
18            'adresse': ['numero_de_voie', 'type_de_voie_rue_avenue_boulevard_etc', '
      nom_de_la_voie', 'complement_d_adresse_facultatif', 'geocodage_xy', '
      commune_principale_de_deroulement'],
19            'categorie': ['discipline_dominante'],
20            'sous_categorie': ['sous_categorie_spectacle_vivant', 'sous_categorie_musique', '
      sous_categorie_musique_cnm', 'sous_categorie_cinema_et_audiovisuel', '
```

```
        sous_categorie_arts_visuels_et_arts_numeriques', 'sous_categorie_livre_et_litterature', '
        discipline_dominante'],
21          'festival': ['nom_du_festival', 'adresse_e_mail', 'annee_de_creation_du_festival', '
        site_internet_du_festival', 'commune_principale_de_deroulement', '
        periode_principale_de_deroulement_du_festival', 'adresse_postale', '
        code_insee_epci_collage_en_valeur'],
22          'organisation': ['code_insee_epci_collage_en_valeur', 'libelle_epci_collage_en_valeur
        '],
23          'festival_categorie': ['identifiant', 'discipline_dominante'],
24          'festival_organisation': ['identifiant', 'code_insee_epci_collage_en_valeur'],
25          'festival_commune': ['identifiant', 'commune_principale_de_deroulement'],
26          'festival_adresse': ['identifiant', 'adresse_postale']
27      }
28
29      # Dictionnaire des clés étrangères
30      foreign_keys = {
31          'departement': {'id_region': 'region_principale_de_deroulement'},
32          'commune': {'id_departement': 'departement_principal_de_deroulement'},
33          'adresse': {'id_commune': 'commune_principale_de_deroulement'},
34          'sous_categorie': {'id_categorie': 'discipline_dominante'},
35          'festival': {'id_commune': 'commune_principale_de_deroulement', 'id_adresse': '
        adresse_postale', 'id_organisation': 'code_insee_epci_collage_en_valeur'},
36          'festival_categorie': {'id_festival': 'identifiant', 'id_categorie': '
        discipline_dominante'},
37          'festival_organisation': {'id_festival': 'identifiant', 'id_organisation': '
        code_insee_epci_collage_en_valeur'},
38          'festival_commune': {'id_festival': 'identifiant', 'id_commune': '
        commune_principale_de_deroulement'},
39          'festival_adresse': {'id_festival': 'identifiant', 'id_adresse': 'adresse_postale'}
40      }
41
42      # Définir un répertoire de sortie temporaire
43      output_dir = r"C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes"
44      if not os.path.exists(output_dir):
45          os.makedirs(output_dir)
46
47      # Fonction pour générer des identifiants
48      def generate_ids(df, table):
49          id_column_name = f'id_{table}'
50          df.insert(0, id_column_name, range(1, len(df) + 1))
51          return df
52
53      # Fonction pour valider et formater les colonnes d'entiers
54      def validate_and_format_integers(df, columns):
55          for col in columns:
56              df[col] = pd.to_numeric(df[col], errors='coerce').fillna('null').astype(str)
57          return df
58
59      # Créer les fichiers TXT avec les clés étrangères
60      for table, columns in tables_columns.items():
61          # Vérifier si les colonnes existent dans le fichier CSV
62          existing_columns = [col for col in columns if col in data.columns]
63          if not existing_columns:
64              continue
65
66          if table in ['festival_categorie', 'festival_organisation', 'festival_commune', '
        festival_adresse']:
67              # Pour ces tables, ne conserver que les clés étrangères
68              df = data[existing_columns].copy()
69          else:
70              # Ajouter la colonne 'id_nom_table' qui s'incrémente automatiquement
71              df = data[existing_columns].copy()
72              df = generate_ids(df, table)
73
74              # Valider et formater les colonnes d'entiers
75              integer_columns = [col for col in df.columns if 'id_' in col]
76              df = validate_and_format_integers(df, integer_columns)
77
78          # Ajouter les clés étrangères
79              if table in foreign_keys:
80                  for fk, ref_col in foreign_keys[table].items():
81                      if ref_col in data.columns:
82                          ref_data = data[[ref_col]].drop_duplicates().reset_index(drop=True)
83                          ref_data = generate_ids(ref_data, fk.replace('id_', ''))
84                          try:
85                              df = df.merge(ref_data, how='left', left_on=ref_col, right_on=ref_col)
```

```
                        df = df.rename(columns={f'id_{fk.replace("id_", "")}': fk})
                        df = df.drop(columns=[ref_col])
                    except KeyError:
                        print(f"KeyError: '{ref_col}' not found in DataFrame. Skipping this merge.")

        # Écrire dans un fichier TXT avec les données complètes
        output_file = os.path.join(output_dir, f'{table}.txt')
        df.to_csv(output_file, sep='\t', index=False)

    # Lister les fichiers créés pour vérification
    output_files = os.listdir(output_dir)
    output_files

```

Script SQL de peuplement

```
\COPY region FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\region.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY departement FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\departement.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY commune FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\commune.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY adresse FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\adresse.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY categorie FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\categorie.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY sous_categorie FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\sous_categorie.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY organisation FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\organisation.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY festival FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\festival.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

```

Tous les codes Python et SQL ci dessus sont fonctionnels cependant la petite partie concernant les types associations provoque des erreurs. Elle est donc mise de côté en dessous.

```
\COPY festival_categorie FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\festival_categorie.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY festival_organisation FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\festival_organisation.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY festival_commune FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\festival_commune.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

\COPY festival_adresse FROM 'C:\Users\nayls\OneDrive\Bureau\BUT\Cours\SQL\Di_Gregorio\SAE\textes_new\festival_adresse.txt' delimiter E'\t' csv HEADER encoding 'UTF8';

```

3.3 Description commentée des différentes étapes de votre script de peuplement

Tout d'abord, nous avons commencé, à partir du modèle relationnel, à créer en SQL les tables nécessaires au bon fonctionnement de la base de données, c'est à dire les tables des types entités et les tables des types associations. Ensuite, grâce à un script Python, nous avons créé les fichiers textes permettant plus tard d'utiliser la fonction "COPY" en SQL. L'une des étapes les plus difficiles a été de créer ce programme impliquant

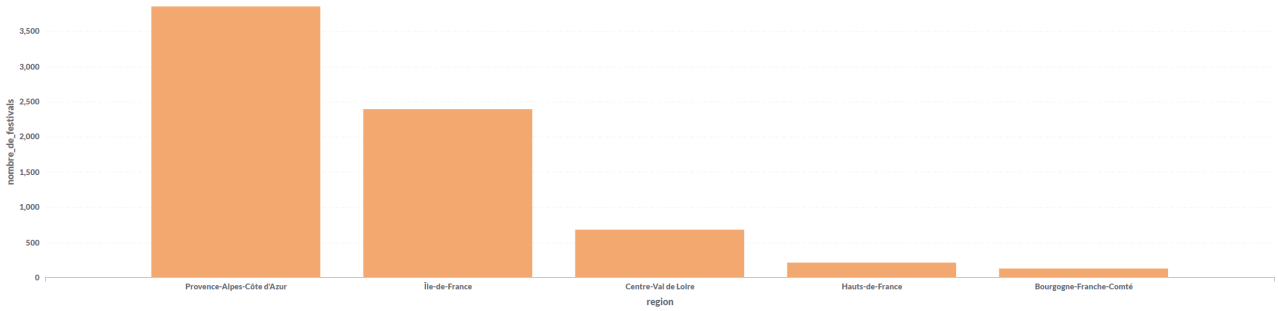
l'implémentation des attributs classiques (partie plus facile) mais aussi des clés étrangères (partie difficile) dans les fichiers textes. C'est notamment cette étape qui a manqué de perfection car, par la suite, au niveau du peuplement, les erreurs ont commencé à bondir de tous les sens. En effet, pour finir la conception de cette base, il ne nous manquait plus qu'à l'alimenter. Cependant, seulement une partie n'a pas été alimenté dû à des erreurs lors de l'éxécution du script de peuplement (l'erreur était qu'il y avait encore des doublons lors de la création des tables des types associations). Nous avons donc peuplé la partie des tables des types entités mais pas les tables des types associations.

4 Visualisation des résultats

4.1 Interrogation variée de la base de données, Script SQL concernant les requêtes et Visualisation

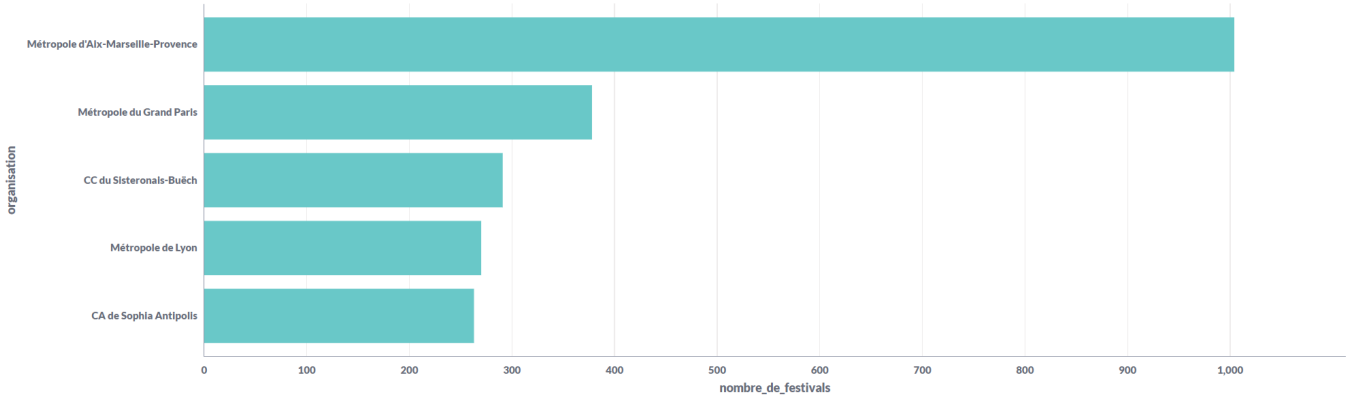
Pour l'interrogation de notre base de données, nous avons décidé de prendre 5 problématiques/thématiques différentes avec chacun sa propre datavisualisation.
Problématique 1 : Quelle est la répartition des festivals en France ?
Objectif : Déterminer comment les festivals sont répartis dans différentes régions.

```sql
SELECT r.region, COUNT(f.id_festival) AS nombre_de_festivals
FROM festival f
JOIN commune c ON f.id_commune = c.id_commune
JOIN departement d ON c.id_departement = d.id_departement
JOIN region r ON d.id_region = r.id_region
GROUP BY r.region
ORDER BY nombre_de_festivals DESC;

```



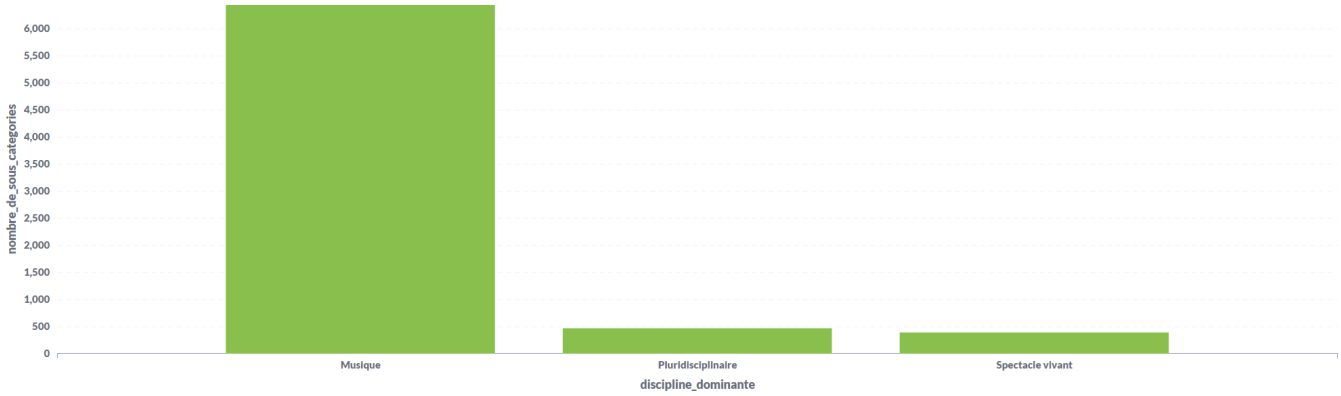
Problématique 2 : Quelle entité organisation est la plus active ?
Objectif : Identifier les entités organisatrices les plus actives en fonction du nombre de festivals qu'elles organisent.

```sql
SELECT o.libelle_insee_epci AS organisation, COUNT(f.id_festival) AS nombre_de_festivals
FROM festival f
JOIN organisation o ON f.id_organisation = o.id_organisation
GROUP BY o.libelle_insee_epci
ORDER BY nombre_de_festivals DESC
LIMIT 5;

```



Problématique 3 : Quelle est la répartition du nombre de sous-catégories par catégories de festivals ?
Objectif : Identifier les catégories les plus tendances en fonction du nombre de sous-catégories qu'elles possèdent.

```sql
SELECT c.discipline_dominante, COUNT(sc.id_sous_categorie) AS nombre_de_sous_categories
FROM sous_categorie sc
JOIN categorie c ON sc.id_categorie = c.id_categorie
GROUP BY c.discipline_dominante
ORDER BY nombre_de_sous_categories DESC;

```



4.2 Commentaire des résultats

Pour la première visualisation, on voit clairement que la Provence-Alpes-Côtes-d'Azur domine ce classement avec plus de 3500 festivals se déroulant dans cette région. Cependant, elle est suivie de près par l'Île de France qui comptabilise plus de 2000 festivals à son actif. Et pour le reste, à part le Centre Val de Loire, aucune autre région ne compte beaucoup de festivals. On peut supposer que c'est parce la Provence-Alpes-Côtes-d'Azur profite d'un climat plus enclin à organiser un festival et aussi que l'Île de France a l'une densité de population les plus élevée de France.

Pour la deuxième visualisation, la Métropole d'Aix-Marseille-Provence domine largement ces concurrents avec plus de 1 000 festivals organisés. Bien plus bas, en deuxième position du classement, il y a la Métropole du Grand Paris avec seulement environ 370 festivals organisés. Cette tête du classement correspond notamment à notre conclusion du premier graphique. On se rend donc compte que le fait que la Métropole d'Aix-Marseille-Provence fait participer autant de festivals a joué sur le classement du graphique précédent. Et ce phénomène est répercuté sur les autres membres du classement. On peut notamment citer la Métropole du Grand Paris qui

est situé en Île de France.

Pour la troisième visualisation, on observe la répartition des sous-catégories de disciplines dominantes des festivals. Il est clair que la catégorie "Musique" est prédominante avec plus de 6 000 occurrences. Les catégories "Pluridisciplinaire" et "Spectacle vivant" sont beaucoup moins représentées, avec chacune un nombre significativement inférieur d'occurrences.

Cette prédominance des festivals de musique peut être attribuée à leur large attractivité et à leur diversité, qui attirent un public plus vaste. De plus, les festivals de musique bénéficient souvent de subventions et de soutiens logistiques plus importants, ce qui facilite leur organisation. En revanche, les festivals pluridisciplinaires et de spectacle vivant, bien qu'ils aient un public dédié, sont souvent plus spécialisés et peuvent nécessiter des infrastructures spécifiques moins courantes.

5 Organisation du travail

5.1 Répartition du travail

Pour la première partie concernant la partie modélisation, Hassan et moi avons travaillé ensemble jusqu'à l'élaboration du modèle entités-Associations. Ensuite, nous nous sommes répartis les tâches. Tout d'abord, j'ai personnellement commencé à rédiger le modèle relationnel pendant que Hassan commençait à coder le programme permettant de produire les fichiers servant au script de création SQL. Après avoir terminé d'écrire le modèle relationnel, j'ai commencé la rédaction du rapport final à rendre. Pendant que Hassan essayait de régler les soucis liés au code Python pour la création des fichiers, j'ai commencé à écrire le script SQL de création de table. Hassan a ensuite pris l'initiative de finir ce script ainsi que le script de peuplement. Ensuite, j'ai écrit l'ensemble des descriptions et explications des modèles E-A et Relationnel et ait procédé à l'organisation du rapport final. Après que nos script SQL de création de tables et de peuplement soient terminés, nous avons procédé à l'écriture des requêtes SQL permettant la visualisation des données présentes dans la base. J'ai ensuite écrit les interprétations et le texte manquants nécessaires au rapport final.

Il faut cependant noter que les étapes que j'ai citées ne reflètent pas vraiment le déroulé exact du projet. En effet, chaque étape du projet a nécessité beaucoup de temps et chacune de ces étapes supplémentaires nous faisaient réfléchir sur les précédentes étapes. Nous devions donc constamment retourner en arrière et changer quelques parties ou potentiellement tout les parties. A noter aussi le fait que chacun des membres du groupe a grandement contribué au travail de l'autre. C'est à dire que j'ai aidé Hassan sur les scripts de création et Hassan m'a aidé à la conception des modèles. Cela a donc été un travail complémentaire du début jusqu'à la fin.

5.2 Remarques et ressentis

Tout en sachant les risques liées à l'utilisation de Latex pour la mise en forme, j'ai décidé de l'utiliser car Latex permet de styliser énormément de choses à notre manière. Cependant, l'insertion de code extractible a été nouveau pour moi. J'ai donc dû faire des recherches mais aucune de ces recherches n'a abouti à une conclusion permettant d'avoir le code proprement tout en pouvant l'extraire du fichier en format PDF. Les codes ne sont donc pas extractibles du fichier. C'est pourquoi dans l'envoi du mail, j'ai aussi inclus des fichiers originaux de code pour pouvoir extraire les lignes de code. J'espère que vous comprendrez mon essai ainsi que nos efforts apporté au projet.

Cette SAÉ a été très intense pour nous. En effet, nous avons dû travailler des dizaines et des dizaines d'heures pour parfaire toutes les parties du projet. De l'exploration de la base de données jusqu'à notre dernière visualisation, nous avons travaillé d'arrache-pied tout en sachant que ce n'était pas notre seule SAÉ. Nous devions donc jongler entre les SAÉs et gérer nos emplois du temps pour pouvoir travailler sur les mêmes créneaux horaires. Cela n'a pas été de tout repos, loin de là mais nous savions que ce type d'exercice était typiquement une mission confiée à un alternant dans une entreprise. Nous avons donc donné notre maximum pour pouvoir fournir un résultat à la hauteur de nos efforts.
Nous espérons donc que ce rapport vous ait plu et que notre travail sera récompensé.