Sae 1.04

Données démographiques mondiales

Elghomari--Jabeur Alaeddine
Scelles Etan
Jamain Simon
Lin Romaric
Nanthakumar Thenujan

Introduction

Dans le cadre de la SAE 1.04, le projet consiste à transformer un fichier CSV brut contenant des données démographiques mondiales issues du Département des Affaires Économiques et Sociales de l'ONU (Division Population) en une base de données relationnelle exploitable. Ces données couvrent la période de 1950 à 2023 et incluent des indicateurs tels que la population totale, le taux de natalité, le taux de mortalité, la migration, l'espérance de vie, et le ratio hommes/femmes.

L'objectif principal est de structurer ces données afin de faciliter leur visualisation, leur interrogation et leur analyse. Cela permet de mieux comprendre les dynamiques démographiques mondiales et régionales, et de produire des outils d'aide à la décision.

Pour ce projet, nous avons utilisé **SQLite** comme moteur de base de données, en l'intégrant via l'interface **DBeaver Community**. Ce choix s'explique par la légèreté de SQLite, sa simplicité d'installation, et son intégration directe dans des projets locaux sans besoin de serveur. DBeaver offre une interface graphique puissante pour administrer la base, importer les données, concevoir le modèle relationnel et exécuter les requêtes SQL.

La base a été construite selon un modèle hiérarchique structuré en quatre niveaux : **continent > région > sous-région > pays**, avec une table de faits centralisant les mesures démographiques annuelles. Des scripts ont été développés pour importer, nettoyer et structurer les données, puis pour valider leur intégrité à travers des requêtes d'analyse.

Ce travail s'inscrit dans une démarche complète de gestion de données : de l'importation brute à l'exploitation analytique, en passant par la modélisation relationnelle et la création de vues SQL.

Description de la problématique et des données traitées

Le projet s'inscrit dans une problématique d'exploitation de données démographiques mondiales issues du Département des Affaires Économiques et Sociales de l'ONU (Division Population). Ces données, considérées comme une référence internationale, couvrent la période de 1950 à 2023 et regroupent des indicateurs clés tels que :

- la population totale (aux dates du 1er janvier et du 1er juillet),
- le taux de natalité et de mortalité,
- la croissance démographique,
- le ratio hommes/femmes,
- l'espérance de vie,
- la migration nette,
- la densité de population,
- et d'autres mesures liées à la structure par âge et sexe.

La problématique centrale consiste à transformer ce fichier brut (au format CSV) en une base de données relationnelle cohérente, permettant d'effectuer des analyses ciblées, des comparaisons temporelles et géographiques, et de produire des visualisations synthétiques. L'objectif est de rendre ces données exploitables pour des applications d'aide à la décision, tout en garantissant leur intégrité, leur lisibilité et leur structuration.

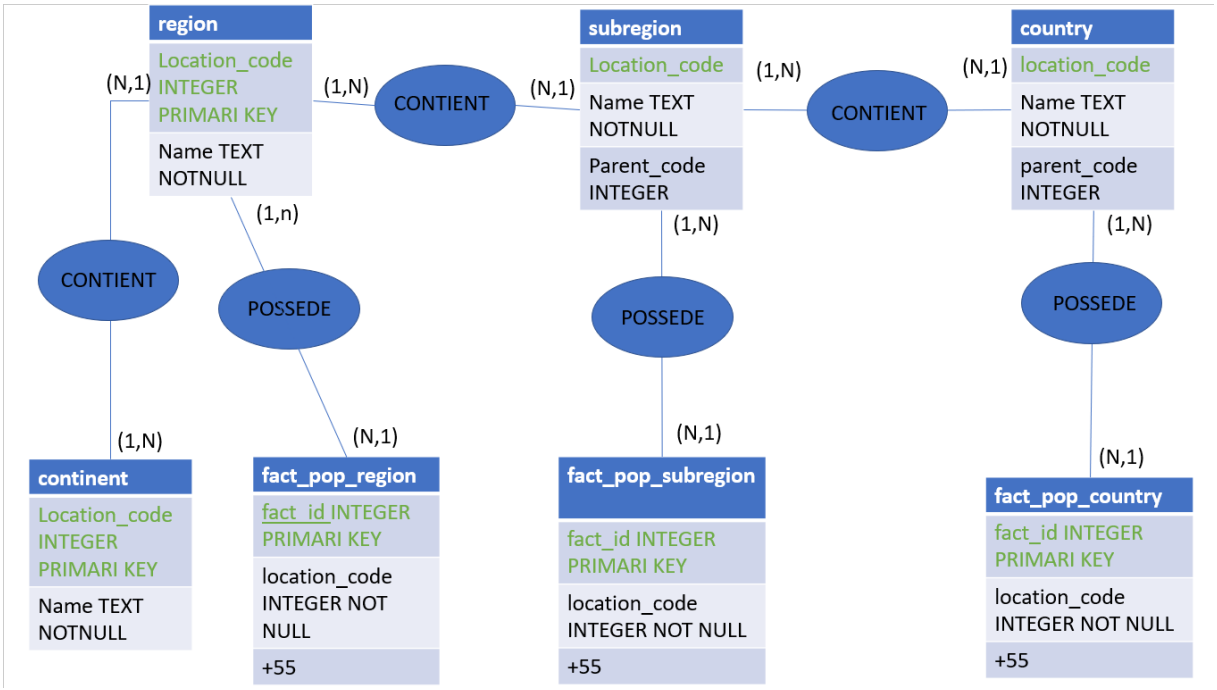
Pour cela, les données ont été modélisées selon une hiérarchie géographique à quatre niveaux : **continent > région > sous-région > pays**, et les indicateurs ont été centralisés dans une **table de faits**. Ce modèle permet de naviguer efficacement dans les données, de regrouper les mesures par zones géographiques, et de répondre à des requêtes complexes sur l'évolution démographique mondiale.

Un schéma relationnel clair

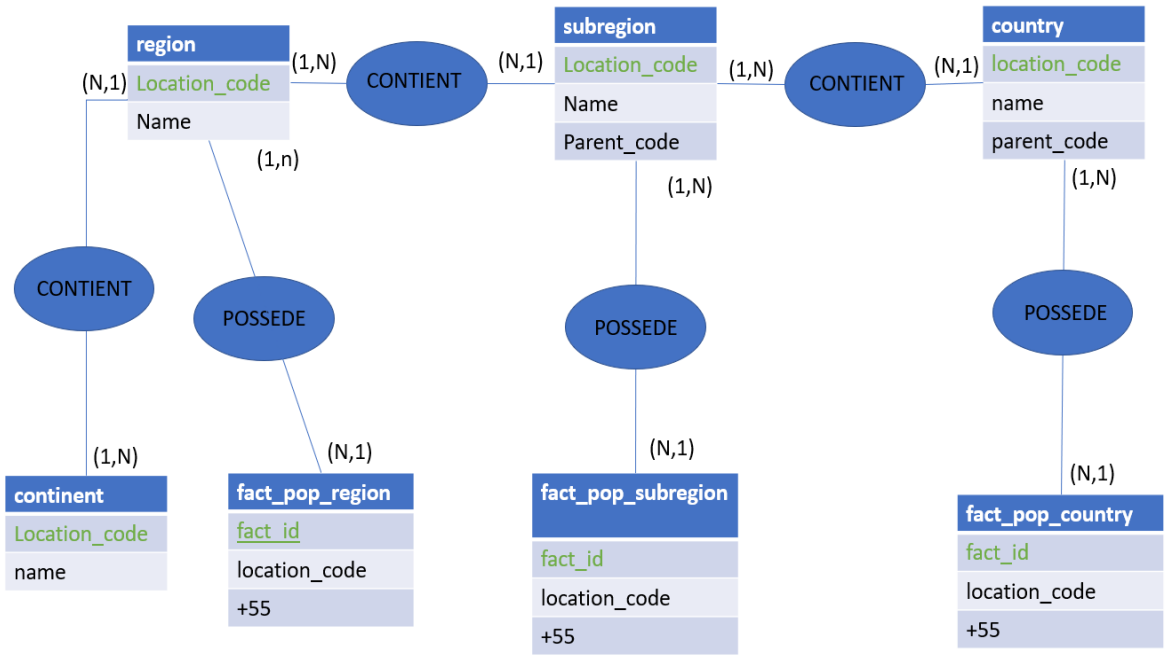
Les deux diagrammes suivants présentent la modélisation de la base de données : le MCD (Modèle Conceptuel de Données) permettant de représenter les entités et leurs associations, et le schéma Entité-Relation (ER) traduisant cette structure en tables relationnelles avec les clés primaires et étrangères.

Voici le MCD et le schéma ER correspondant à la base WorldPopulation.db :

ER

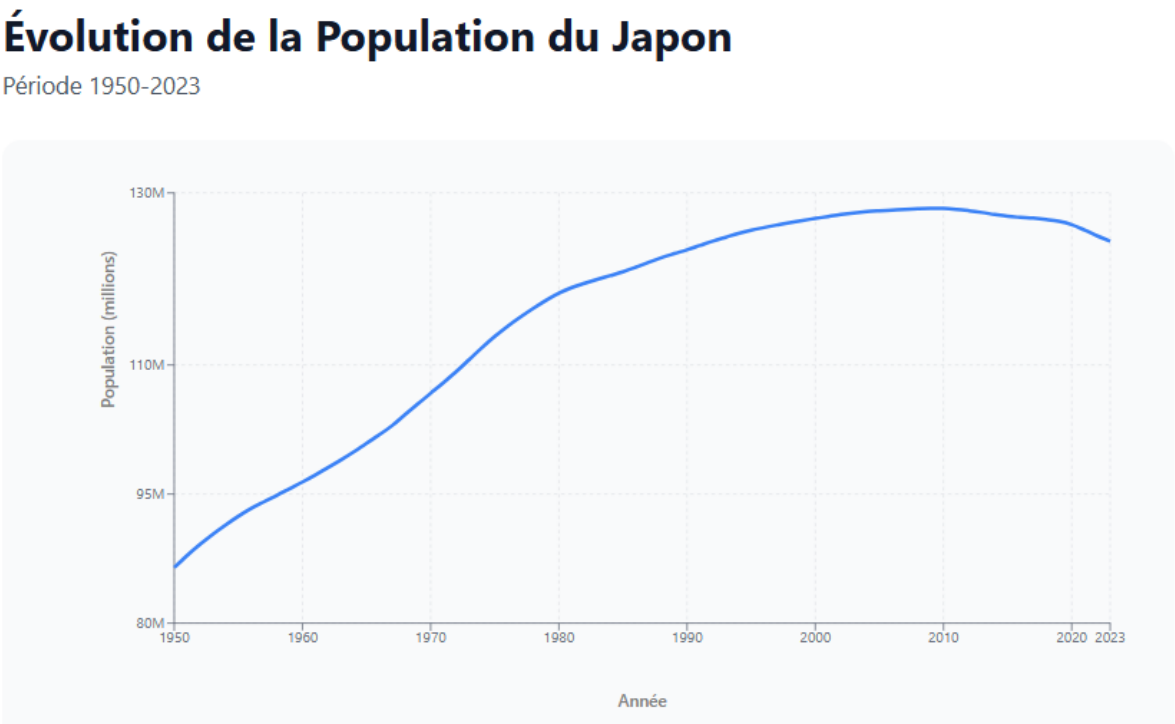


MCD



Visualisation

Nous avons choisi d'étudier l'évolution de la croissance démographique du Japon sur la période allant de 1950 à 2023, afin d'analyser les dynamiques de population à travers les phases de croissance, de stabilisation et de déclin observées dans les données officielles.



Résultats des vues

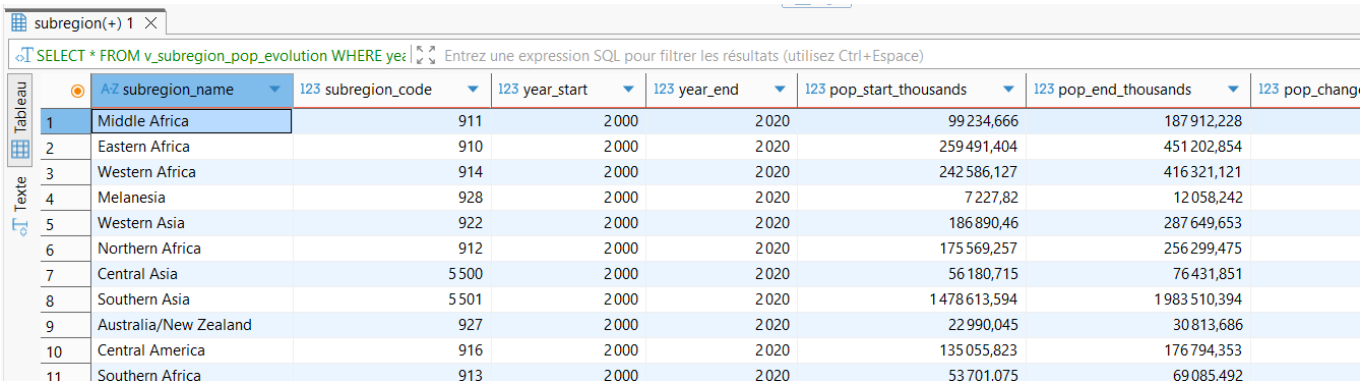
Les vues SQL qu'on a créées permettent de simplifier les analyses en regroupant des requêtes complexes sous un nom. Par exemple, au lieu de refaire à chaque fois les jointures pour calculer la population d'une région ou l'évolution d'une sous-région, on peut juste interroger la vue.
Ça rend le travail plus rapide, plus lisible et surtout plus réutilisable dans l'application web qu'on fera en SAE1.01.

Différentes vues de la SAE :

```
-- ========================================================================
-- THÈME 5 : CRÉATION DE VUES (REQUÊTES NOMMÉES)
-- ========================================================================

-- Question 1 : Population moyenne, minimale et maximale par région sur toute la
période--
CREATE VIEW v_pop_stats_by_region AS
SELECT
    r.name AS region_name,
    r.location_code AS region_code,
    ROUND(AVG(fr.Total_Population_as_of_1_July_thousands), 2) AS avg_popula-
tion_thousands,
    MIN(fr.Total_Population_as_of_1_July_thousands) AS min_population_thousands,
    MAX(fr.Total_Population_as_of_1_July_thousands) AS max_population_thousands,
    MIN(fr.years) AS first_year,
    MAX(fr.years) AS last_year,
    COUNT(DISTINCT fr.years) AS nb_years_covered
FROM fact_pop_region fr
JOIN region r ON fr.location_code = r.location_code
WHERE fr.Total_Population_as_of_1_July_thousands IS NOT NULL
GROUP BY r.location_code, r.name
ORDER BY avg_population_thousands DESC;

-- Exemple d'utilisation :
SELECT * FROM v_pop_stats_by_region;
```

	region_name	region_code	avg_population_thousands	min_population_thousands	max_population_thousands
1	Asia	935	3028059,93	1368075,415	4778(
2	Europe	908	687737,14	548867,473	749
3	Africa	903	672221,14	227776,419	1480
4	Latin America and the Cai	904	413720,62	167782,158	658
5	Northern America	905	275245,64	168009,338	382
6	Oceania	909	26790,27	12582,044	45

```sql
-- Question 2 : Évolution de la population par sous-région entre deux années données
CREATE VIEW v_subregion_pop_evolution AS
SELECT
    s.name AS subregion_name,
    s.location_code AS subregion_code,
    fs1.years AS year_start,
    fs2.years AS year_end,
    fs1.Total_Population_as_of_1_July_thousands AS pop_start_thousands,
    fs2.Total_Population_as_of_1_July_thousands AS pop_end_thousands,
    ROUND(fs2.Total_Population_as_of_1_July_thousands - fs1.Total_Population_as_of_1_July_thousands, 2) AS pop_change_thousands,
    ROUND(
        ((fs2.Total_Population_as_of_1_July_thousands - fs1.Total_Population_as_of_1_July_thousands) /
         fs1.Total_Population_as_of_1_July_thousands) * 100,
        2
    ) AS pop_change_percent,
    r.name AS parent_region
FROM fact_pop_subregion fs1
JOIN fact_pop_subregion fs2
    ON fs1.location_code = fs2.location_code
JOIN subregion s ON fs1.location_code = s.location_code
JOIN region r ON s.parent_code = r.location_code
WHERE fs1.Total_Population_as_of_1_July_thousands IS NOT NULL
  AND fs2.Total_Population_as_of_1_July_thousands IS NOT NULL
  AND fs1.years < fs2.years;

-- Exemple d'utilisation pour comparer 2000 et 2020 :
SELECT * FROM v_subregion_pop_evolution
WHERE year_start = 2000 AND year_end = 2020
ORDER BY pop_change_percent DESC;
```



subregion(+) 1

SELECT * FROM v_subregion_pop_evolution WHERE yea | Entrez une expression SQL pour filtrer les résultats (utilisez Ctrl+Espace)

Tableau | Texte

	subregion_name	subregion_code	year_start	year_end	pop_start_thousands	pop_end_thousands	pop_chang
1	Middle Africa	911	2000	2020	99234,666	187912,228	
2	Eastern Africa	910	2000	2020	259491,404	451202,854	
3	Western Africa	914	2000	2020	242586,127	416321,121	
4	Melanesia	928	2000	2020	7227,82	12058,242	
5	Western Asia	922	2000	2020	186890,46	287649,653	
6	Northern Africa	912	2000	2020	175569,257	256299,475	
7	Central Asia	5500	2000	2020	56180,715	76431,851	
8	Southern Asia	5501	2000	2020	1478613,594	1983510,394	
9	Australia/New Zealand	927	2000	2020	22990,045	30813,686	
10	Central America	916	2000	2020	135055,823	176794,353	
11	Southern Africa	913	2000	2020	53701,075	69085,492	

```sql
-- Question 3 : Quelles sont les régions avec un âge médian supérieur à 30 ans et
un ratio hommes/femmes déséquilibré (supérieur à 100) ?
CREATE VIEW v_regions_age_gender_imbalance AS
SELECT
    r.name AS region_name,
    r.location_code AS region_code,
    fr.years AS year,
    ROUND(fr.Median_Age_as_of_1_July_years, 1) AS median_age,
    ROUND(fr.Population_Sex_Ratio_as_of_1_July_males_per_100_females, 2) AS
sex_ratio_m_per_100f,
    ROUND(fr.Total_Population_as_of_1_July_thousands, 2) AS total_population_thou-
sands,
    c.name AS continent_name,
    CASE
        WHEN fr.Population_Sex_Ratio_as_of_1_July_males_per_100_females > 105 THEN
'Déséquilibre fort (>105)'
        WHEN fr.Population_Sex_Ratio_as_of_1_July_males_per_100_females > 102 THEN
'Déséquilibre modéré (102-105)'
        ELSE 'Déséquilibre léger (100-102)'
    END AS niveau_desequilibre
FROM fact_pop_region fr
JOIN region r ON fr.location_code = r.location_code
JOIN continent c ON r.parent_code = c.location_code
WHERE fr.Median_Age_as_of_1_July_years > 30
  AND fr.Population_Sex_Ratio_as_of_1_July_males_per_100_females > 100
  AND fr.Median_Age_as_of_1_July_years IS NOT NULL
  AND fr.Population_Sex_Ratio_as_of_1_July_males_per_100_females IS NOT NULL
ORDER BY fr.years DESC, fr.Population_Sex_Ratio_as_of_1_July_males_per_100_females
DESC;

-- Exemple d'utilisation pour l'année la plus récente :
SELECT * FROM v_regions_age_gender_imbalance
WHERE year = (SELECT MAX(years) FROM fact_pop_region)
ORDER BY sex_ratio_m_per_100f DESC;
```

	region_name	region_code	year	median_age	sex_ratio_m_per_100f	total_population_thousands	contin
1	Asia	935	2023	31,7	103,52	4778004,49	Afro_Eura
2	Northern America	905	2023	38,3	100,75	382902,74	Américair
3	Oceania	909	2023	32,9	100,36	45562,79	Océanien

```sql
-- ========================================================================
-- ETAPE 10 : Test de validation
-- ========================================================================

-- Vue 1 : Population par région pour un continent donné (ex. Afro-Eurasien en 2000)

CREATE VIEW Population_par_region_par_continent AS
SELECT
    Region.name,
    fact_pop_region.Total_Population_as_of_1_January_thousands
FROM Region
JOIN fact_pop_region ON Region.location_code = fact_pop_region.location_code
JOIN Continent ON Region.parent_code = Continent.location_code
WHERE Continent.Name = 'Afro_Eurasien'
  AND years = 2000;

-- Utilisation :
SELECT * FROM Population_par_region_par_continent;
```

region(+) 1

SELECT * FROM Population_par_region_par_continent | Entrez une expression SQL

Tableau

	A-Z name	123 Total_Population_as_of_1_January_thousands
1	Africa	820 180,248
2	Asia	3 721 978,166
3	Europe	728 105,163

```sql
-- Vue 2 :

DROP VIEW IF EXISTS Region_la_plus_peuple;

CREATE VIEW Region_la_plus_peuple AS
SELECT
    r.name AS region_name,
    fr."Total Population, as of 1 January (thousands)" AS population
FROM region r
JOIN fact_pop_region fr ON r.location_code = fr.location_code
WHERE fr."Year" = 2000
ORDER BY fr."Total Population, as of 1 January (thousands)" DESC
LIMIT 1;


SELECT * FROM Region_la_plus_peuple;
```

	A-Z name	123 Total_Population_as_of_1_January_thousands
1	Asia	3 721 978,166

Les requêtes principales utilisées

Afin de garantir l'intégrité des données et d'éviter toute ambiguïté dans les relations, nous avons choisi de séparer la base en plusieurs tables de faits distinctes. La table initiale pop regroupait à la fois des données pour les pays, les sous-régions et les régions, ce qui empêchait la mise en place de clés étrangères fiables. En restructurant la base, nous avons créé trois tables spécialisées : fact_pop_country, fact_pop_subregion et fact_pop_region. Chacune est liée directement à sa table de dimension (country, subregion, region) par une clé étrangère, ce qui assure une cohérence stricte et simplifie les requêtes. Cette organisation hiérarchique (continent → région → sous-région → pays) permet d'exploiter les données démographiques de manière claire, performante et évolutive, tout en facilitant les analyses comparatives entre différents niveaux géographiques.

```sql
/* Verifie si l'importation a fonctionné */
SELECT * FROM pop LIMIT 50;
SELECT * FROM pop LIMIT 50;
SELECT name FROM sqlite_master WHERE type='table';
```

```sql
/* Creation et peuplement des table region, subregion et country*/

CREATE TABLE continent (
 location_code INTEGER PRIMARY KEY,
 name TEXT NOT NULL
);

INSERT INTO continent(location_code, name)
VALUES
      (1, 'Afro_Eurasien'),
      (2, 'Américain'),
      (3, 'Océanien');
```

```sql
CREATE TABLE region (
 location_code INTEGER PRIMARY KEY,
 name TEXT NOT NULL,
 parent_code INTEGER,
 FOREIGN KEY (parent_code) REFERENCES continent(location_code)
);

INSERT INTO region (location_code, name, parent_code)
SELECT DISTINCT
    "Location code",
    "Region, subregion, country or area",
    CASE
        WHEN "Region, subregion, country or area" IN ('Africa', 'Europe', 'Asia')
            THEN 1
        WHEN "Region, subregion, country or area" IN ('Northern America', 'Latin
America and the Caribbean')
            THEN 2
        WHEN "Region, subregion, country or area" = 'Oceania'
            THEN 3
        ELSE NULL
    END
FROM pop
WHERE "Type" = 'Region';
```

```sql
CREATE TABLE subregion (
 location_code INTEGER PRIMARY KEY,
 name TEXT NOT NULL,
 parent_code INTEGER,
 FOREIGN KEY (parent_code) REFERENCES region(location_code)
);

INSERT INTO subregion
SELECT DISTINCT
 "Location code",
 "Region, subregion, country or area",
 "Parent code"
FROM pop
WHERE "Type" = 'Subregion';

INSERT INTO subregion (location_code, name, parent_code)
VALUES (905, 'Northern America (sub-region)', 905);
```

```sql
CREATE TABLE country (
 location_code INTEGER PRIMARY KEY,
 name TEXT NOT NULL,
 parent_code INTEGER,
 FOREIGN KEY (parent_code) REFERENCES subregion(location_code)
);

INSERT INTO country
SELECT DISTINCT
 "Location code",
 "Region, subregion, country or area",
 "Parent code"
FROM pop
WHERE "Type" = 'Country/Area';
```

```sql
/* Verification */

SELECT
       co.name,
       r.name,
       s.name,
       c.name
FROM continent co
JOIN region r ON co.location_code = r.parent_code
JOIN subregion s ON r.location_code = s.parent_code
JOIN country c    ON s.location_code = c.parent_code
WHERE c.name = 'Canada';
```

Pour la création et le peuplement des tables de faits, la table fact_pop_country est en exemple. On a fait la même chose pour les autres tables de faits (Subregion et Region). Les seules choses qui changent sont la condition WHERE à la fin de l'insertion et les FOREIGN KEY à la fin des CREATE TABLE. Les nombres présents dans la table de données comportait des virgules, on a donc utilisé REPLACE pour les remplacer par des points pour que DBeaver les comprennent comme des réels plutôt que comme des chaînes de caractères.

```sql
-- ======================================================================
-- CRÉATION DE LA TABLE fact_pop_country
-- ======================================================================

CREATE TABLE fact_pop_country (
    fact_id INTEGER PRIMARY KEY AUTOINCREMENT,
    location_code INTEGER NOT NULL,
    years INTEGER NOT NULL,
    -- Population totale et démographie de base
    Total_Population_as_of_1_January_thousands REAL,
    Total_Population_as_of_1_July_thousands REAL,
    Male_Population_as_of_1_July_thousands REAL,
    Female_Population_as_of_1_July_thousands REAL,
    Population_Density_as_of_1_July_persons_per_square_km REAL,
    Population_Sex_Ratio_as_of_1_July_males_per_100_females REAL,
    Median_Age_as_of_1_July_years REAL,
    -- Croissance et changements de population
    Natural_Change_Births_minus_Deaths_thousands REAL,
    Rate_of_Natural_Change_per_1000_population REAL,
    Population_Change_thousands REAL,
    Population_Growth_Rate_percentage REAL,
    Population_Annual_Doubling_Time_years REAL,
    -- Naissances et fertilité
    Births_thousands REAL,
    Births_by_women_aged_15_to_19_thousands REAL,
    Crude_Birth_Rate_births_per_1000_population REAL,
    Total_Fertility_Rate_live_births_per_woman REAL,
    Net_Reproduction_Rate_surviving_daughters_per_woman REAL,
    Mean_Age_Childbearing_years REAL,
    Sex_Ratio_at_Birth_males_per_100_female_births REAL,
    -- Décès
    Total_Deaths_thousands REAL,
    Male_Deaths_thousands REAL,
    Female_Deaths_thousands REAL,
    Crude_Death_Rate_deaths_per_1000_population REAL,
    -- Espérance de vie
    Life_Expectancy_at_Birth_both_sexes_years REAL,
    Male_Life_Expectancy_at_Birth_years REAL,
    Female_Life_Expectancy_at_Birth_years REAL,
    Life_Expectancy_at_Age_15_both_sexes_years REAL,
    Male_Life_Expectancy_at_Age_15_years REAL,
    Female_Life_Expectancy_at_Age_15_years REAL,
    Life_Expectancy_at_Age_65_both_sexes_years REAL,
    Male_Life_Expectancy_at_Age_65_years REAL,
    Female_Life_Expectancy_at_Age_65_years REAL,
    Life_Expectancy_at_Age_80_both_sexes_years REAL,
    Male_Life_Expectancy_at_Age_80_years REAL,
    Female_Life_Expectancy_at_Age_80_years REAL,
    -- Mortalité infantile et juvénile
    Infant_Deaths_under_age_1_thousands REAL,
    Infant_Mortality_Rate_infant_deaths_per_1000_live_births REAL,
    Live_Births_Surviving_to_Age_1_thousands REAL,
    Under_Five_Deaths_under_age_5_thousands REAL,
    Under_Five_Mortality_deaths_under_age_5_per_1000_live_births REAL,
    -- Mortalité par âge
    Mortality_before_Age_40_both_sexes_deaths_under_age_40_per_1000_live_births
REAL,
    Male_Mortality_before_Age_40_deaths_under_age_40_per_1000_male_live_births
REAL,
```

```
    Female_Mortality_before_Age_40_deaths_under_age_40_per_1000_female_live_births
REAL,
    Mortality_before_Age_60_both_sexes_deaths_under_age_60_per_1000_live_births
REAL,
    Male_Mortality_before_Age_60_deaths_under_age_60_per_1000_male_live_births
REAL,
    Female_Mortality_before_Age_60_deaths_under_age_60_per_1000_female_live_births
REAL,
    Mortality_between_Age_15_and_50_both_sexes_deaths_un-
der_age_50_per_1000_alive_at_age_15 REAL,
    Male_Mortality_between_Age_15_and_50_deaths_un-
der_age_50_per_1000_males_alive_at_age_15 REAL,
    Female_Mortality_between_Age_15_and_50_deaths_under_age_50_per_1000_fe-
males_alive_at_age_15 REAL,
    Mortality_between_Age_15_and_60_both_sexes_deaths_un-
der_age_60_per_1000_alive_at_age_15 REAL,
    Male_Mortality_between_Age_15_and_60_deaths_un-
der_age_60_per_1000_males_alive_at_age_15 REAL,
    Female_Mortality_between_Age_15_and_60_deaths_under_age_60_per_1000_fe-
males_alive_at_age_15 REAL,
    -- Migration
    Net_Number_of_Migrants_thousands REAL,
    Net_Migration_Rate_per_1000_population REAL,
    -- Clé étrangère vers country
    FOREIGN KEY (location_code) REFERENCES country(location_code)
);

-- ======================================================================
-- PEUPLEMENT DE fact_pop_country
-- ======================================================================

INSERT INTO fact_pop_country (
    location_code,
    years,
    Total_Population_as_of_1_January_thousands,
    Total_Population_as_of_1_July_thousands,
    Male_Population_as_of_1_July_thousands,
    Female_Population_as_of_1_July_thousands,
    Population_Density_as_of_1_July_persons_per_square_km,
    Population_Sex_Ratio_as_of_1_July_males_per_100_females,
    Median_Age_as_of_1_July_years,
    Natural_Change_Births_minus_Deaths_thousands,
    Rate_of_Natural_Change_per_1000_population,
    Population_Change_thousands,
    Population_Growth_Rate_percentage,
    Population_Annual_Doubling_Time_years,
    Births_thousands,
    Births_by_women_aged_15_to_19_thousands,
    Crude_Birth_Rate_births_per_1000_population,
    Total_Fertility_Rate_live_births_per_woman,
    Net_Reproduction_Rate_surviving_daughters_per_woman,
    Mean_Age_Childbearing_years,
    Sex_Ratio_at_Birth_males_per_100_female_births,
    Total_Deaths_thousands,
    Male_Deaths_thousands,
    Female_Deaths_thousands,
    Crude_Death_Rate_deaths_per_1000_population,
    Life_Expectancy_at_Birth_both_sexes_years,
    Male_Life_Expectancy_at_Birth_years,
```

```
    Female_Life_Expectancy_at_Birth_years,
    Life_Expectancy_at_Age_15_both_sexes_years,
    Male_Life_Expectancy_at_Age_15_years,
    Female_Life_Expectancy_at_Age_15_years,
    Life_Expectancy_at_Age_65_both_sexes_years,
    Male_Life_Expectancy_at_Age_65_years,
    Female_Life_Expectancy_at_Age_65_years,
    Life_Expectancy_at_Age_80_both_sexes_years,
    Male_Life_Expectancy_at_Age_80_years,
    Female_Life_Expectancy_at_Age_80_years,
    Infant_Deaths_under_age_1_thousands,
    Infant_Mortality_Rate_infant_deaths_per_1000_live_births,
    Live_Births_Surviving_to_Age_1_thousands,
    Under_Five_Deaths_under_age_5_thousands,
    Under_Five_Mortality_deaths_under_age_5_per_1000_live_births,
    Mortality_before_Age_40_both_sexes_deaths_under_age_40_per_1000_live_births,
    Male_Mortality_before_Age_40_deaths_under_age_40_per_1000_male_live_births,
    Female_Mortality_before_Age_40_deaths_under_age_40_per_1000_fe-
male_live_births,
    Mortality_before_Age_60_both_sexes_deaths_under_age_60_per_1000_live_births,
    Male_Mortality_before_Age_60_deaths_under_age_60_per_1000_male_live_births,
    Female_Mortality_before_Age_60_deaths_under_age_60_per_1000_fe-
male_live_births,
    Mortality_between_Age_15_and_50_both_sexes_deaths_un-
der_age_50_per_1000_alive_at_age_15,
    Male_Mortality_between_Age_15_and_50_deaths_un-
der_age_50_per_1000_males_alive_at_age_15,
    Female_Mortality_between_Age_15_and_50_deaths_under_age_50_per_1000_fe-
males_alive_at_age_15,
    Mortality_between_Age_15_and_60_both_sexes_deaths_un-
der_age_60_per_1000_alive_at_age_15,
    Male_Mortality_between_Age_15_and_60_deaths_un-
der_age_60_per_1000_males_alive_at_age_15,
    Female_Mortality_between_Age_15_and_60_deaths_under_age_60_per_1000_fe-
males_alive_at_age_15,
    Net_Number_of_Migrants_thousands,
    Net_Migration_Rate_per_1000_population
)
SELECT
    "Location code",
    "Year",
    REPLACE("Total Population, as of 1 January (thousands)", ',', '.'),
    REPLACE("Total Population, as of 1 July (thousands)", ',', '.'),
    REPLACE("Male Population, as of 1 July (thousands)", ',', '.'),
    REPLACE("Female Population, as of 1 July (thousands)", ',', '.'),
    REPLACE("Population Density, as of 1 July (persons per square km)", ',', '.'),
    REPLACE("Population Sex Ratio, as of 1 July (males per 100 females)", ',',
'.'),
    REPLACE("Median Age, as of 1 July (years)", ',', '.'),
    REPLACE("Natural Change, Births minus Deaths (thousands)", ',', '.'),
    REPLACE("Rate of Natural Change (per 1,000 population)", ',', '.'),
    REPLACE("Population Change (thousands)", ',', '.'),
    REPLACE("Population Growth Rate (percentage)", ',', '.'),
    REPLACE("Population Annual Doubling Time (years)", ',', '.'),
    REPLACE("Births (thousands)", ',', '.'),
    REPLACE("Births by women aged 15 to 19 (thousands)", ',', '.'),
    REPLACE("Crude Birth Rate (births per 1,000 population)", ',', '.'),
    REPLACE("Total Fertility Rate (live births per woman)", ',', '.'),
    REPLACE("Net Reproduction Rate (surviving daughters per woman)", ',', '.'),
```

```
    REPLACE("Mean Age Childbearing (years)", ',', '.'),
    REPLACE("Sex Ratio at Birth (males per 100 female births)", ',', '.'),
    REPLACE("Total Deaths (thousands)", ',', '.'),
    REPLACE("Male Deaths (thousands)", ',', '.'),
    REPLACE("Female Deaths (thousands)", ',', '.'),
    REPLACE("Crude Death Rate (deaths per 1,000 population)", ',', '.'),
    REPLACE("Life Expectancy at Birth, both sexes (years)", ',', '.'),
    REPLACE("Male Life Expectancy at Birth (years)", ',', '.'),
    REPLACE("Female Life Expectancy at Birth (years)", ',', '.'),
    REPLACE("Life Expectancy at Age 15, both sexes (years)", ',', '.'),
    REPLACE("Male Life Expectancy at Age 15 (years)", ',', '.'),
    REPLACE("Female Life Expectancy at Age 15 (years)", ',', '.'),
    REPLACE("Life Expectancy at Age 65, both sexes (years)", ',', '.'),
    REPLACE("Male Life Expectancy at Age 65 (years)", ',', '.'),
    REPLACE("Female Life Expectancy at Age 65 (years)", ',', '.'),
    REPLACE("Life Expectancy at Age 80, both sexes (years)", ',', '.'),
    REPLACE("Male Life Expectancy at Age 80 (years)", ',', '.'),
    REPLACE("Female Life Expectancy at Age 80 (years)", ',', '.'),
    REPLACE("Infant Deaths, under age 1 (thousands)", ',', '.'),
    REPLACE("Infant Mortality Rate (infant deaths per 1,000 live births)", ',',
'.'),
    REPLACE("Live Births Surviving to Age 1 (thousands)", ',', '.'),
    REPLACE("Under-Five Deaths, under age 5 (thousands)", ',', '.'),
    REPLACE("Under-Five Mortality (deaths under age 5 per 1,000 live births)",
',', '.'),
    REPLACE("Mortality before Age 40, both sexes (deaths under age 40 per 1,000
live births)", ',', '.'),
    REPLACE("Male Mortality before Age 40 (deaths under age 40 per 1,000 male live
births)", ',', '.'),
    REPLACE("Female Mortality before Age 40 (deaths under age 40 per 1,000 female
live births)", ',', '.'),
    REPLACE("Mortality before Age 60, both sexes (deaths under age 60 per 1,000
live births)", ',', '.'),
    REPLACE("Male Mortality before Age 60 (deaths under age 60 per 1,000 male live
births)", ',', '.'),
    REPLACE("Female Mortality before Age 60 (deaths under age 60 per 1,000 female
live births)", ',', '.'),
    REPLACE("Mortality between Age 15 and 50, both sexes (deaths under age 50 per
1,000 alive at age 15)", ',', '.'),
    REPLACE("Male Mortality between Age 15 and 50 (deaths under age 50 per 1,000
males alive at age 15)", ',', '.'),
    REPLACE("Female Mortality between Age 15 and 50 (deaths under age 50 per 1,000
females alive at age 15)", ',', '.'),
    REPLACE("Mortality between Age 15 and 60, both sexes (deaths under age 60 per
1,000 alive at age 15)", ',', '.'),
    REPLACE("Male Mortality between Age 15 and 60 (deaths under age 60 per 1,000
males alive at age 15)", ',', '.'),
    REPLACE("Female Mortality between Age 15 and 60 (deaths under age 60 per 1,000
females alive at age 15)", ',', '.'),
    REPLACE("Net Number of Migrants (thousands)", ',', '.'),
    REPLACE("Net Migration Rate (per 1,000 population)", ',', '.')
FROM pop
WHERE "Year" IS NOT NULL
  AND "Location code" IS NOT NULL
  AND "Location code" IN (SELECT location_code FROM country);
```

Réponses argumentées aux différentes questions posées dans le document

Question de réflexion #1 : Comment pourrait-on séparer logiquement ces deux types d'informations dans une base relationnelle ?

On sépare les tableaux pour distinguer les données descriptives (régions, sous-régions, pays) des données chiffrées (population, natalité, mortalité), ce qui évite les redondances. Cette organisation garantit la cohérence des relations et facilite les analyses hiérarchiques avec des requêtes SQL claires et efficaces.

Question de réflexion #2 : Pourquoi choisit-on SQLite pour ce projet ? Quels sont ses avantages et limites par rapport à d'autres bases de données telles que PostgreSQL ou MySQL par exemple ?

On choisit SQLite car c'est une base légère, simple à installer et adaptée aux projets académiques ou de petite taille. Ses avantages sont la portabilité et l'absence de serveur, mais ses limites apparaissent lorsqu'on a besoin de forte concurrence d'accès ou de gestion de très gros volumes de données, où PostgreSQL ou MySQL sont plus performants

Question de vérification #3 : Que remarquez-vous dans les colonnes ? Quelles informations semblent redondantes ou répétées ?

Le CSV mélange des colonnes descriptives (Type, Region/Subregion/Country, Location code, Parent code) et des colonnes de mesures annuelles (population, natalité, mortalité, espérance de vie, etc.). En effet, chaque ligne du fichier correspond à un pays (ou région) donné et une année donnée. Par exemple, le code pays « 250 » apparaît sur toutes les lignes de la France de 1950 à 2023. Autrement dit, les informations de dimension sont dupliquées pour chaque année. Le tutoriel relève déjà cette redondance : « Location code apparaît plusieurs fois pour un même pays… ». Ces colonnes répétitives justifient la création ultérieure de tables de dimension, on évitera ainsi le stockage multiple des mêmes informations géographiques

Question de réflexion #4 : Pourquoi séparer les régions, sous-régions et pays dans des tables différentes ?

On sépare les régions, sous-régions et pays dans des tables différentes afin de respecter la hiérarchie géographique et éviter la duplication des données. En faisant çaon réduit les risques d'erreurs ou d'incohérences (orthographe différente, codes incorrects, données dupliquées). Cette organisation assure la cohérence des relations et facilite les requêtes SQL.

<u>Question de compréhension #5</u> : Que signifient les termes « clé primaire » et « clé étrangère » dans ce contexte ? Comment garantissent-elles la cohérence des relations entre tables ?

Une **clé primaire** est un attribut (ou un ensemble d'attributs) qui identifie de manière unique chaque ligne d'une table. Dans ce projet, par exemple, location_code est la clé primaire des tables region, subregion et country, garantissant qu'aucune région ou pays n'a le même identifiant.
Une **clé étrangère** est un attribut qui établit un lien entre deux tables en faisant référence à la clé primaire d'une autre table. Ici, parent_code dans la table subregion est une clé étrangère pointant vers location_code de la table region.
Ces clés assurent la **cohérence des relations** : elles empêchent l'insertion de valeurs invalides et garantissent que chaque sous-région ou pays est rattaché à une région existante, ce qui maintient l'intégrité des données dans la base

<u>Question de réflexion #6</u> : Pourquoi crée-t-on un champ fact_id auto-incrémenté alors que nous avons déjà location_code et year ? Ces deux derniers champs peuvent-ils servir de clés ? Quelle pourrait être la clé primaire "logique" de cette table ?

Dans la table fact_population, le champ fact_id auto-incrémenté joue le rôle de clé primaire. Il garantit qu'il existe toujours un identifiant unique pour chaque ligne, ce qui facilite les requêtes et la gestion interne, même si d'autres colonnes venaient à être modifiées. Toutefois, la véritable clé primaire logique de cette table est le couple (location_code, year), car pour un pays ou une région donnée, il ne devrait exister qu'une seule observation par année. Cependant pour simplifier les jointures et gérer aisément les références on crée et utilise fact_id.

<u>Question #7a</u> :

Quelle région du monde est la plus peuplée en 2000 ?

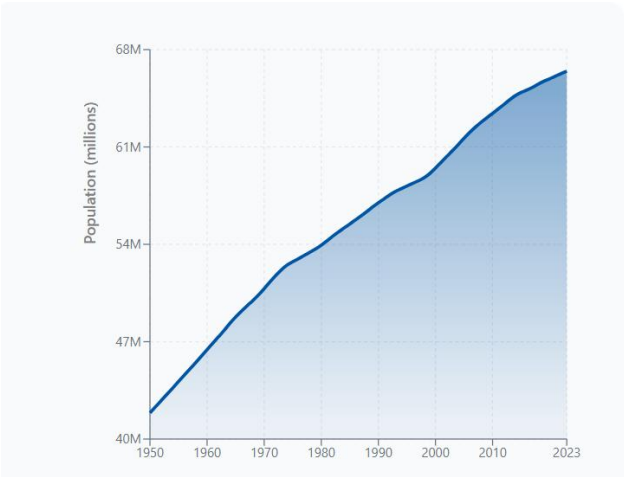
La région la plus peuplée en 2000 est l'Asie.

Quelles hypothèses pouvez-vous formuler sur son évolution depuis 1950 ?

Depuis 1950, la population asiatique augmente régulièrement et de manière continue, sans baisse apparente. On peut supposer que l'Asie a connu une croissance démographique soutenue sur toute cette période.

Question #7b

Essayez de tracer le graphique de l'évolution de la population française (avec Excel par exemple). Quelle tendance observez-vous ?



En traçant le graphique de l'évolution de la population française de 1950 à 2023, on observe une **tendance à la hausse continue et régulière**.

Plus précisément :

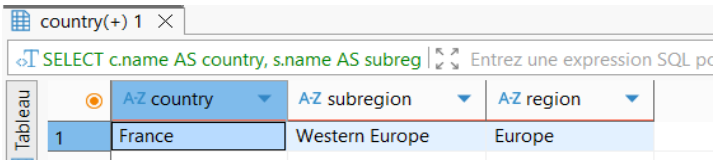
- La population française est passée de **41,9 millions en 1950** à **66,4 millions en 2023**, soit une **croissance de +58,6%** sur 73 ans.
- La courbe présente une **pente positive constante** sans période de déclin, contrairement à certains pays comme le Japon.
- La croissance est particulièrement marquée entre **1950 et 1975** puis se poursuit de manière plus modérée mais stable jusqu'en 2023.
- On observe une **croissance démographique soutenue et ininterrompue**, ce qui témoigne d'un renouvellement de la population grâce à un taux de natalité relativement élevé et à l'immigration.

Conclusion : La tendance générale est une croissance démographique linéaire et positive sur toute la période étudiée.

Question #7c

Donnez un exemple d'une relation pays → sous-région → région observée dans les résultats.

```
SELECT
  c.name AS country,
  s.name AS subregion,
  r.name AS region
FROM country c
JOIN subregion s ON c.parent_code = s.location_code
JOIN region r ON s.parent_code = r.location_code
WHERE c.name = 'France';
```



Est-ce cohérent avec vos connaissances géographiques ?

C'est plutôt cohérent avec nos connaissances géographiques.

Question #8a

Pourquoi la clause GROUP BY est-elle nécessaire pour construire cette vue ?

La vue doit **regrouper les données par région et par année** afin de calculer la **population totale par région**. Sans GROUP BY, la fonction d'agrégation SUM() ne saurait pas comment regrouper les lignes et additionner les populations.

Que permet d'ajouter la clause ORDER BY à la fin de la requête nommée ?

La clause ORDER BY définit **l'ordre d'affichage des résultats** dans la vue. Dans cet exemple : ORDER BY f.year, r.name trie d'abord par **année croissante**, puis par **nom de région**. Cela rend la vue plus lisible et facilite les analyses chronologiques.

Question #8b

Cas 1 : Pourquoi utilise-t-on la fonction d'agrégation SUM() dans la vue1 ?

On utilise la fonction d'agrégation **SUM()** dans cette vue pour **additionner les populations de tous les pays appartenant à une même région pour une année donnée**.

Sans cette fonction, la requête retournerait une ligne par pays, alors que l'objectif est d'obtenir **une seule valeur totale par région et par année**.

En résumé, SUM() permet de regrouper les données et de calculer la **population totale régionale**, ce qui est indispensable pour analyser l'évolution démographique par région.

Cas 2 : Pourquoi utilise-t-on la fonction d'agrégation ROUND() dans cette vue (numéro 2) ?

La fonction **ROUND()** est utilisée pour arrondir le résultat du calcul du pourcentage à deux décimales, ce qui améliore la lisibilité et la présentation des données. Sans cette fonction, les valeurs pourraient comporter de nombreuses décimales, rendant l'affichage peu pratique.

Comment modifier la vue pour afficher la part moyenne des 10 dernières années au lieu d'une seule année ?

Pour calculer la part moyenne des 10 dernières années, il faut d'abord **filtrer les données sur la période souhaitée**, par exemple avec WHERE f.year BETWEEN 2011 AND 2020. Ensuite, on **regroupe les résultats par région et par pays**, sans inclure l'année dans le regroupement, afin d'obtenir une seule ligne par pays et par région. Enfin, on applique la fonction d'agrégation **AVG()** sur le calcul du pourcentage pour obtenir la moyenne des parts sur ces dix années.

Conclusion

Cette SAE nous a permis de transformer un fichier Excel/CSV de l'ONU en une base de données relationnelle claire et exploitable. Le fichier contenait des données démographiques mondiales de 1950 à 2023, organisées par niveaux géographiques (région → sous-région → pays). On a commencé par analyser la structure du fichier afin d'identifier les entités et leurs relations. Cela nous a permis de créer un MCD, un schéma entité-relation et les tables de dimensions (region, subregion, country). On a également séparé les données en trois tables de faits différentes, ce qui simplifie les requêtes. On a ensuite réalisé plusieurs requêtes SQL pour vérifier l'importation des données et les exploiter : calcul de population par région, évolution démographique d'un pays, vérification de la hiérarchie géographique, création de vues, etc. En conclusion, cette SAE nous a permis de mieux comprendre comment organiser et exploiter efficacement des données à partir d'un fichier brut.