BUT SD 2

R3.02 Système d'information décisionnel – Power BI

DEVOIR 2

Modélisation, Traitement, Visualisation

01/11/2024

Groupe : Oikos

Diallo Thierno

Table des matières

I. Modélisation

1. Définition des sources

Nom de la source	Nom de la données consommée	Type de la source	Format de la source	Elément de base dans la source	Volumétrie de l'élément de base (en octet)	Fréquence de consommation	Nombre d'éléments consommés par une operation	Délais de retention (jour)	vol calculée
ville_spacial	ville spacial	structuré	csv	(id_departement, libelle_commune, longitute, latitude)	45	1 fois	10000	9125	450000
ville_niveau_vie	niveau de vie des villes	structuré	csv	(ville, mediane_niveau_vie, part_menage_imposable)	29	AN	10000	9125	7250000
ville_commerce2	commerce	structuré	CSV	(id_departement, libelle_commune, annee, population, hypermache, supermache [, ETC...])	113	AN	10000	9125	28250000

	EN OCTETS	EN MO
TOTAL	35950000	35.95

- **source ville_spacial** : ce fichier contient la liste des villes avec leur coordonnée géographique et le code du département au quelle elles appartiennent, ce fichier est donc charger une fois
- **Source ville_niveau_de_vie** : ce fichier contient les informations économiques de chaque ville (niveau de vie médian et part des ménages imposable). Il est censé être actualisé chaque année.
- **Source ville_commerce2** : ce fichier contient sur sa première ligne les catégories de commerce, sur la deuxième ligne les type de commerce (chaque type de commerce représente une colonne), nous avons aussi des colonnes pour le nom de la commune, code département, population et année). Il est actualisé chaque année.

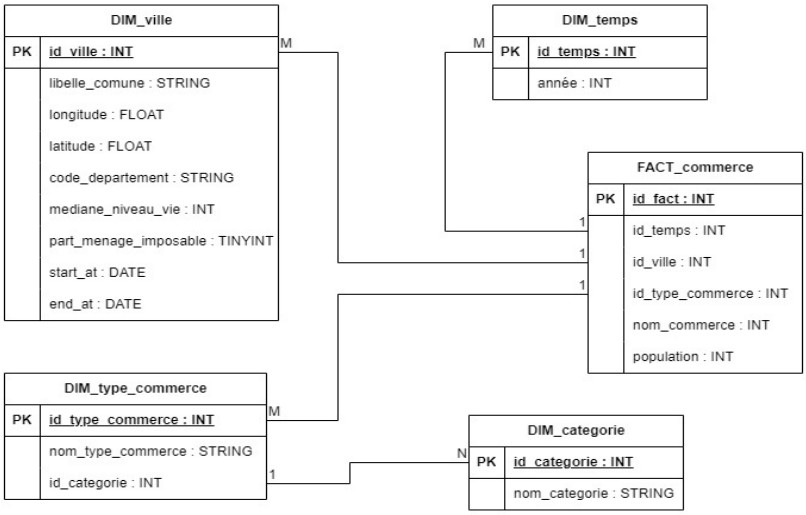
2. Définition des mesures

NOM DE LA MESURE	LISTE DES DIMENSION ASSOCIEES	TYPE DE LA MESURE	FORMULE DE CALCUL	FONCTION D'AGGREGATION	KPI ASSOCIES
nombre_commerce	ville, type de commerce, temps	additif	-	SOMME	densité de commerce par habitant
population	ville, type de commerce, temps	additif	-	SOMME	aucun

- **Mesure nombre_commerce** : La mesure nombre_commerce compte le nombre total de commerces dans une ville donnée pour un type de commerce spécifique et sur une période temporelle donnée, selon les dimensions appliquées dans notre rapport, cette mesure nous permet de répondre à des questions telles que « répartition des commerces par types de commerces »
- **Mesure population** : La mesure population représente le nombre total d'habitants dans chaque commune à une année donnée. Elle est essentielle pour évaluer la taille potentielle du marché dans chaque ville. Elle nous permet de répondre à des questions comme « le nombre d'habitant par département et par année »

3. Modélisation et explication

Pour réaliser ce modèle, j'ai opté pour un modèle flocon afin d'analyser les faits :



- **DIM_TEMPS** : la clé technique est id_temps. La clé fonctionnelle est l'attribut (années). Comme SCD, j'ai choisi le type 1, car les valeurs ne seront pas modifier. Cette dimension va me permettre de suivre mes faits par rapport à l'année.
- **DIM_CATEGORIE** : Les attributs de cette dimension sont (id_categorie, nom_categorie). La clé technique est id_categorie et le pour le SCD, j'ai choisi le type 1 car, elles ne sont pas censé changer, si jamais une catégorie change de nom, on modifie simplement le nom. La clé technique est nom_categorie
- **DIM_TYPE_COMMERCE** : Les attributs de cette dimension sont (id_type_commerce, nom_type_commerce, id_categorie). La clé technique de cette dimension est id_type_commerce, qui permet d'identifier de manière unique chaque type de commerce. La clé fonctionnelle est l'attribut nom_type_commerce, qui représente le nom du type de commerce (par exemple, poissonnerie, boulangerie, etc.). id_categorie est une clé étrangère qui fait référence id_categorie dans la table DIM_CATEGORIE, elle permet de savoir à quel catégorie est rattaché chaque type de commerce. Pour cette dimension, j'ai choisi le SCD de type 1, car les types de commerce ne sont pas censés changer de manière significative dans le temps. En cas de modification, les anciennes valeurs seront remplacées par les nouvelles.
- **DIM_VILLE** : Les attributs de cette table sont (id_ville, libelle_commune, longitude, latitude, code_departement, mediane_niveau_vie, part_menage_imposable, start_at et end_at). La clé technique est id_ville, elle permet d'identifier chaque ligne de la base de données. La clé technique est composée des attributs (longitude, latitude) car ce sont des valeurs qui ne changeront jamais. En termes de SCD, j'ai opté pour le SCD type 2. Car la mediane_niveau_vie et part_menage_imposable est susceptible de changer très fréquemment, comme l'aspect économiques est important pour nos analyse, nous voulons conserver l'historique. Pour pouvoir gérer les différentes versions de chaque ville, nous utilisons start_at et end_at pour exprimer la période de validité des occurrences.
- **FACT_COMMERCE** : La table FACT_commerce représente les faits liés aux commerces dans les différentes communes. Ses attributs sont (id_ville, id_temps, id_type_commerce, nombre_commerce, population). La clé technique est id_fact, qui permet d'identifier chaque ligne. La clé fonctionnelle est formée par les 3 attributs

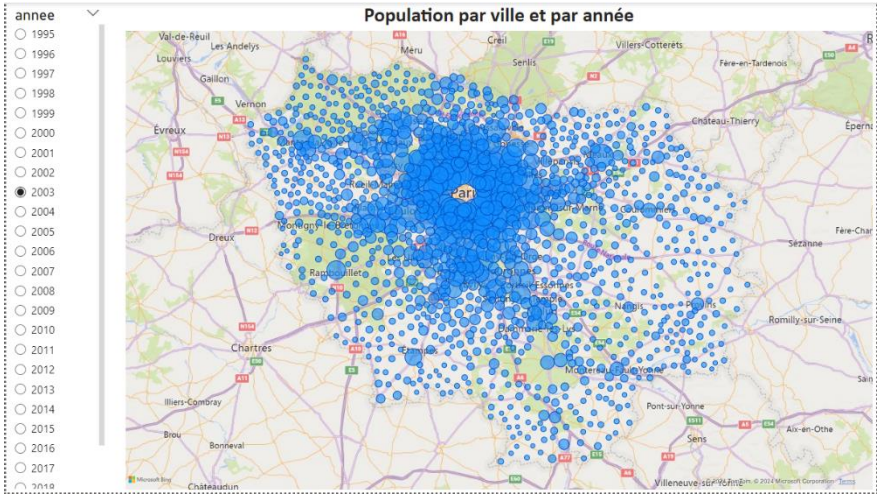
suivant : id_ville, id_temps, id_type_commerce. id_ville est une clé étrangère qui fait référence à id_ville dans la table fim_ville, id_type_commerce est une clé étrangère qui fait référence à id_type_commerce dans la table dim_type_commerce et id_temps est une clé étrangère qui fait référence à id_temps dans la table dim_temps. L'attribut nombre_commerce permet de comptabiliser le nombre de commerce d'un certain type dans une ville donnée et année donnée et population également. Grace au lien avec les autres dimensions, on peut faire de nombreuse analyse telles que : « ou pouvons-nous ouvrir des poissonneries ».

4. Explication de mes manipulation SSIS pour peupler mes tables

- DIM_TEMPS : afin de remplir la table DIM_temps, j'ai créé deux table (la table principale et une table temporaire [à vidé après chaque utilisation]), et avec une tache de flux de donnée, j'ai lu le fichier, supprimer les doublons et enregistrer dans la table temporaire. Puis avec un composant de tache d'exécution SQL, j'ai insérer dans la table principale juste les nouvelles valeurs puis vidé la table temporaire.
- DIM_VILLE : pour peupler la table DIM_VILLE, j'ai créé deux table (la table principale et une table temporaire [à vidé après chaque utilisation]), et avec une tache de flux de donnée, j'ai lu les deux fichiers contenant des informations sur les villes, supprimer les doublons et trier les éléments grâces au composant trier, puis effectuer une jointure entre les deux fichier(composant jointure et fusion) et enregistrer dans la table temporaire. Puis avec un composant de tache d'exécution SQL, j'ai insérer dans la table principale juste les nouvelles villes ou les villes dont les informations avait changé (SCD 2) puis vidé la table temporaire.
- DIM_CATEGORIE et DIM_TYPE_COMMERCE : pour récupérer les catégories et type de commerce dans les deux premières lignes du fichier commerce, j'ai créé un composant Script (qui a la fonction de source de donnée) qui lit mes deux premières ligne, les transpose et les transmet au composant suivant. Cependant pour la configuration de ce composant, lorsque nous créons une connexion avec notre fichier commerce, il faut préciser qu'il ne contient pas d'entête. Puis avec un procéder similaire a DIM_VILLE, j'ai créé des tables principales et temporaire pour catégories et type_categorie. J'ai remplis la table catégorie en premier, puis fait une jointure entre la table catégorie et la table temporaire de type_commerce afin de récupérer les id. Puis supprimer tous les éléments en double puis rempli la table type_commerce.
- FACT_COMMERCE : pour remplir la table fact_commerce, j'ai lu le fichier commerce2 en ignorant la première ligne (ligne de catégorie). Ce fichier étant comme un tableau croisé dynamique, avec un composant UNPIVOT, j'ai transformé les colonnes type_commerce en une seul colonne contenant les différentes modalités et créer une nouvelle colonne contenant les valeurs lier puis insérer dans une table temporaire. Puis avec une requête SQL, j'ai remplacé le texte qui se trouvait dans nombre population et nombre commerce par 0 et insérer les nouvelles lignes dans la table final. Enfin, pour gérer les valeurs aberrantes, j'ai aussi utilisé un scripte SQL avec une condition afin de remplacer les valeurs aberrantes par la médiane.

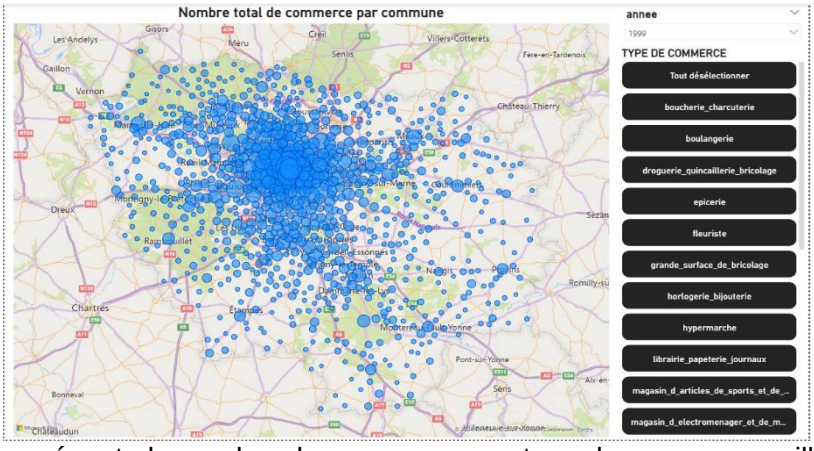
II. Visuel simple

1. Répartition de la population sur une carte et l'évolution de cette répartition dans le temps.



Cette carte nous présente le nombre d'habitant par ville et par année. Chaque cercle représente une ville, et la taille de ce cercle est proportionnelle à la population de cette ville pour une année donnée. On voit que les villes autours de Paris ont des cercle généralement grand et plus on s'éloigne de paris, plus les cercle rétrécisse. Ce qui suggère que Paris exerce un fort pouvoir d'attraction des populations notamment à cause des offres d'emploi, les universités, les service etc… . Ainsi les populations en s'installant en périphérie de paris, esquive la cherté de la vie à paris tous en profitant de ses avantage (cela est rendu possible grâce au système de transport très dense).

2. Répartition des commerces sur carte avec un filtre sur les types de commerces et l'évolution de cette répartition dans le temps



Cette carte nous présente le nombre de commerce par type de commerce, ville et année. La taille des cercles est proportionnelle au nombre de commerce. On constate que la taille des bulles est de plus en plus grande lorsqu'on se rapproche de paris. Cela peut s'expliqué par la forte densité de population à Paris et dans sa périphéries, ce qui créer une forte demande et donc cause l'augmentation des magasins afin de satisfaire les besoins de ses populations et constitue également un marché très attrayant pour les entreprises dans ces secteurs.

3. Un visuel en mode tableau qui regroupe sur les départements et le nombre d'habitant par année

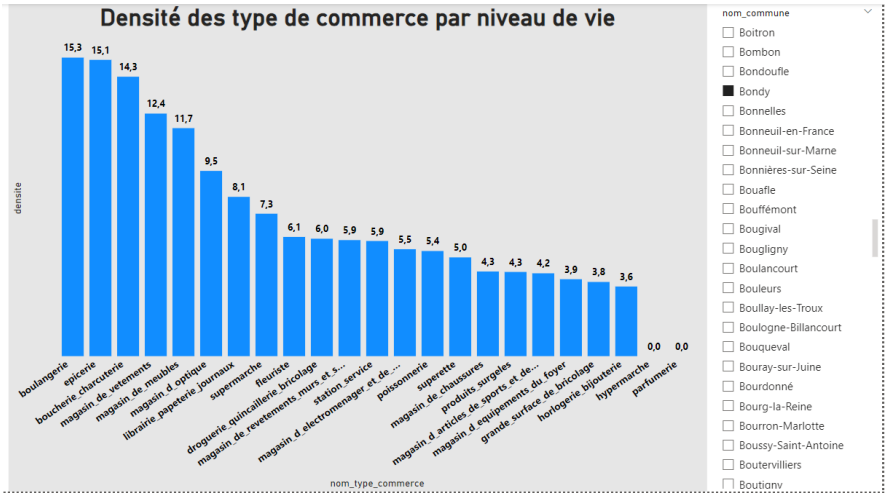
Population par année et par département

annee	code_depatement	population total
1995	95	1144436
1996	95	1171976
1997	95	1182647
1998	95	1177919
1999	95	1179373
2000	95	1174991
2001	95	1180777
2002	95	1198162
2003	95	1203901
2004	95	1219384
2005	95	1218254
2006	95	1209313

Ce tableau nous présente l'évolution du nombre d'habitant par département et par année. Pour le département du 95, on constate une augmentation entre 1995 et 1997 puis une légère baisse de la population jusqu'aux années 2000. Puis elle commence a augmenté à partir de 2001. Cette baisse est assez faible, elle peut être due aux méthodes de recensement ou à un changement d'habitude chez les populations.

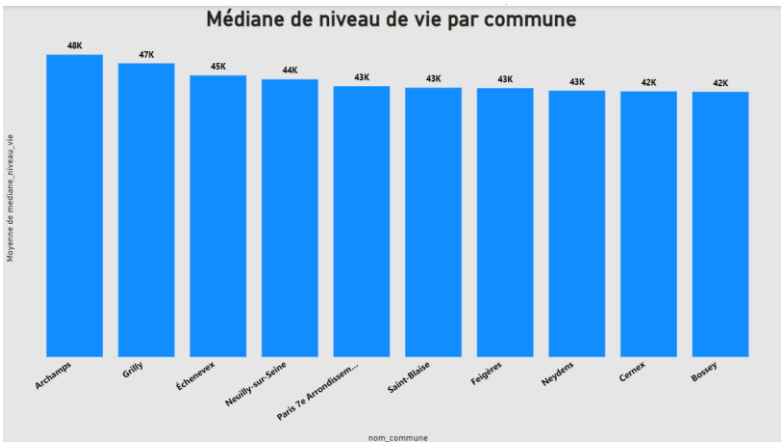
III. Visuel permettant de répondre aux questions suivantes

1. Quel est la densité des commerces par rapport aux nombres de ménages imposés et cela par type de commerce ?



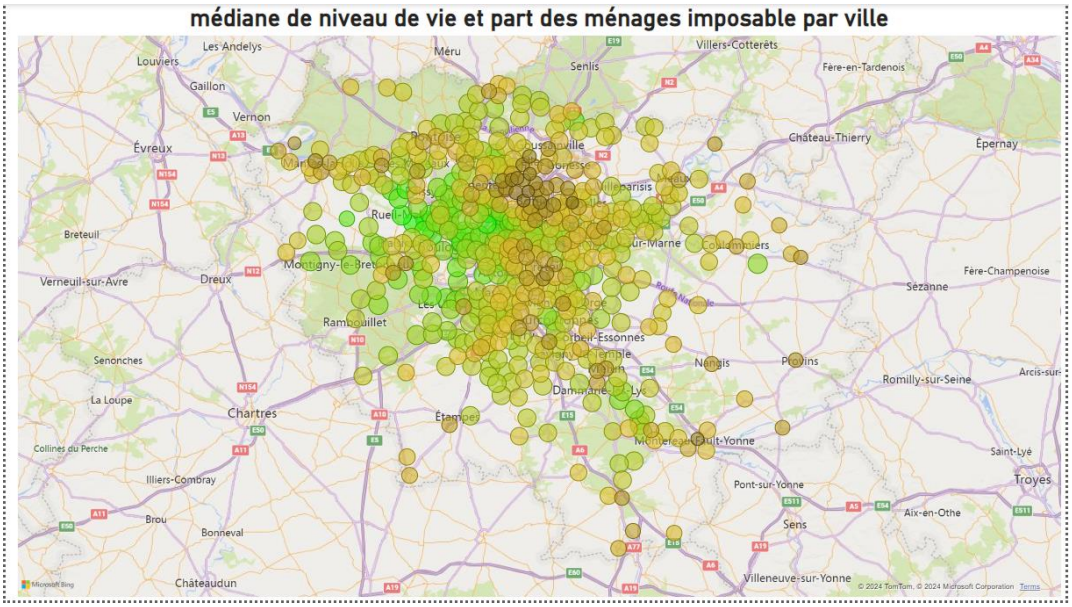
Pour répondre à cette question, j'ai dû calculé la densité des commerce par rapport à la part des ménages imposable (densité = total_commerce / part_menage_imposable) car nous ne disposons pas du nombre de ménages imposables (et du nombre total de ménage).Il est important de noté que ce filtre est très important car faire une somme de densité n'a aucun sens .Ainsi si on prend comme exemple la commune de Bondy, on voit que le type de commerce avec le plus de densité sont les boulangerie, et il n'y a pas de parfumerie. Il également une forte densité d'épicerie et pas mal de supermarché.

2. Quel sont les villes en Ile de France qui ont le niveau de vie le plus élevés (utiliser la médiane du niveau de vie)



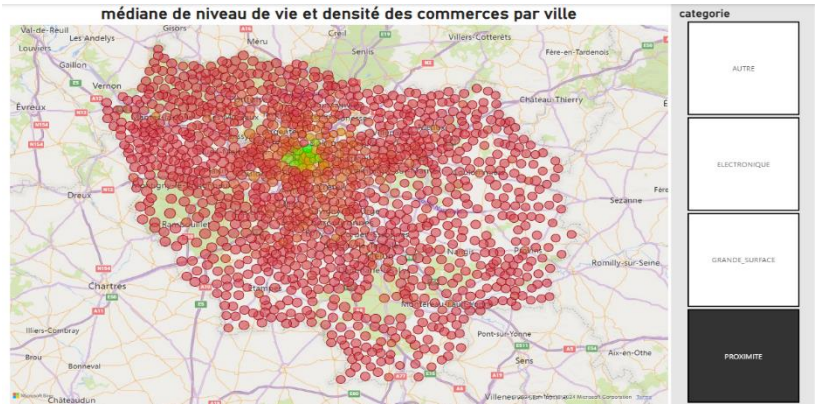
Ce graphique nous présente le top 10 des communes avec le niveau de médian le plus élevé. On constate que la commune avec le niveau de vie le plus élevé est Archamps (48k) et la deuxième est Grilly (47K). On voit qu'il y'a un grand écart (environ 6k) entre le premier et dixième. Ce qui témoigne d'une grande inégalité entre les différentes villes.

3. Est-ce qu'il y'a une corrélation géographique entre le niveau de vie et le nombre de foyer soumis à l'impôt ?



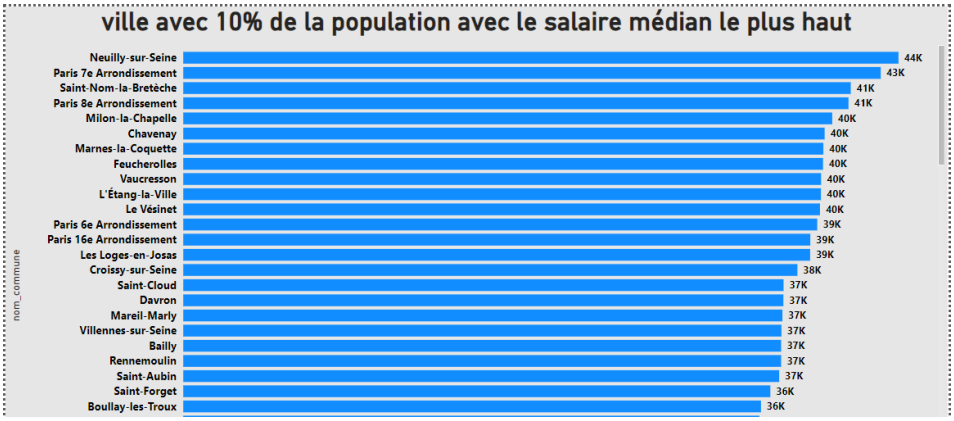
Cette carte nous présente la part des ménages imposable par ville et par niveau de vie. La taille des bulles est proportionnelle à la part des ménages imposable et la couleur est lier au niveau de vie (du noir [du plus faible] au vert [au plus élevé].On constate que les villes à l'ouest (gauche) ont un niveau de vie plus élevé que celle de l'est (droite). Mais également les bulles du côté ouest sont légèrement plus grandes. On peut donc affirmé qu'il existe une corrélation géographique entre le niveau de vie et la part des ménages imposable.

4. Est-ce qu'il y'a une corrélation géographique entre le niveau de vie et la densité des commerces de proximités (par type de commerce) ?



Cette carte nous présente la densité des commerces par ville et par niveau de vie. La taille des bulles est proportionnelle au niveau de vie et la couleur est lier (du noir [du plus faible] au vert [au plus élevé) à la densité des commerces. On constate que la densité de commerce est très élevé à Paris par rapport aux autres villes .on peut donc dire qu'il y'a un lien entre la densité de commerce et la situation géographique. Ce pendant on constate aussi que la taille des bulle est a peu près égal pour toute les villes. Il n'y y'a donc pas de lien a première vue entre la position géographique d'une ville et le niveau de vie

5. Quelle sont les villes ou vivent les 10% de population ayant un niveau de vie élevé ?



Ce graphique nous présente les villes où vivent les 10% de la population ayant un niveau de vie élevé. Pour le réalisé, j'ai créé une vue SQL avec la requête ci-dessous puis avec Power Bi créer une nouvelle colonne appeler fréquence qui contient la fréquence cumulé (pop_cumul / population total). Ainsi on obtient la liste des villes où vivent les 10% avec un niveau de vie élevé. Ainsi le niveau de vie médian se trouve entre 44K et 31k.
(NB : il y'a des villes qui n'apparaissent pas car nous n'avons pas leurs populations)

```sql
CREATE VIEW view_graph_2_e AS
SELECT dc.nom_commune, dc.mediane_niveau_vie, fc.population,
        SUM(fc.population) OVER (ORDER BY  dc.mediane_niveau_vie desc) AS cumul_pop
FROM fact_commerce fc
        JOIN dim_commune dc ON fc.id_ville = dc.id_ville
        JOIN dim_type_commerce dtc ON fc.id_type_commerce = dtc.id_type_commerce
        JOIN dim_temps dt ON fc.id_temps = dt.id_temps
WHERE dt.annee = (SELECT MAX(annee) FROM dim_temps)
  AND dtc.nom_type_commerce = (SELECT MAX(nom_type_commerce) FROM dim_type_commerce);
```

6. Quelle sont les villes ou vivent les 10% de population ayant un niveau de vie bas ?

Ville avec 10% de la population ayant le salaire médian le plus faible

nom_commune	mediane_niveau_vie
Méricourt	16810
Noisy-le-Sec	16660
Grigny	16590
Saint-Ouen	16540
Valenton	16390
Le Bourget	16380
Bondy	16260
Épinay-sur-Seine	16040
Le Blanc-Mesnil	16030
Mantes-la-Jolie	15990
Sevran	15970
L'Île-Saint-Denis	15920
Villeneuve-Saint-Georges	15890
Dugny	15840
Sarcelles	15320

Ce graphique nous présente les villes où vivent les 10% de la population ayant un niveau de vie le plus faible. Pour le réalisé, j'ai créé une vue SQL avec la requête ci-dessous puis avec Power Bi créer une nouvelle colonne appeler fréquence qui contient la fréquence cumulé (pop_cumul / population total). Ainsi on obtient la liste des villes où vivent les 10% avec un niveau de vie le plus faible. On constate que dans cette liste, il y'a de nombreuse ville qui sont issue du département de Seine-Saint-Denis(93), on peut donc émettre comme hypothèse que ce département est aussi celui avec le niveau de vie plus faible. **(NB : le tableau peut être déroulé [scroller])**

(NB : il y'a des villes qui n'apparaissent pas car nous n'avons pas leurs populations)

```sql
CREATE VIEW view_graph_2_f AS
SELECT dc.nom_commune, dc.mediane_niveau_vie, fc.population,
       SUM(fc.population) OVER (ORDER BY  dc.mediane_niveau_vie desc) AS cumul_pop
FROM fact_commerce fc
       JOIN dim_commune dc ON fc.id_ville = dc.id_ville
       JOIN dim_type_commerce dtc ON fc.id_type_commerce = dtc.id_type_commerce
       JOIN dim_temps dt ON fc.id_temps = dt.id_temps
WHERE dt.annee = (SELECT MAX(annee) FROM dim_temps)
  AND dtc.nom_type_commerce = (SELECT MAX(nom_type_commerce) FROM dim_type_commerce);
```

7. Quelle sont les villes ou vivent les 10% de population ayant un niveau de vie Moyen (en non médian) ?

Ville avec 10% de la population ayant le salaire médian moyen

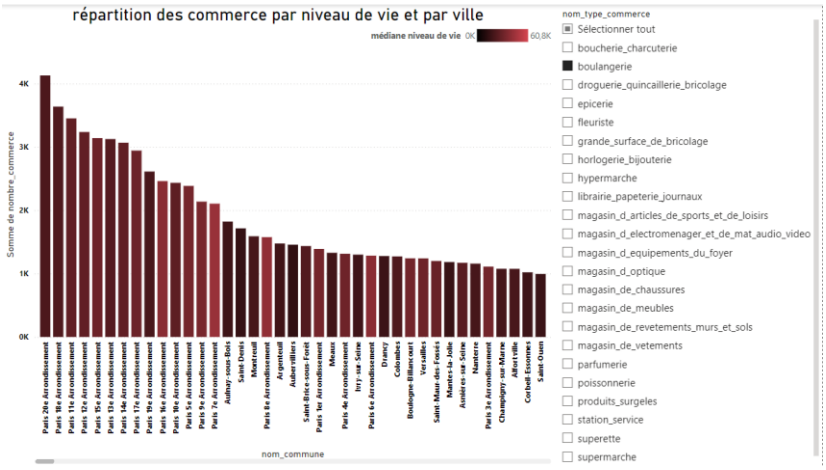
nom_commune	Moyenne de mediane_niveau_vie
Barcy	24100,00
Chambry	24100,00
Bannost-Villegagnon	24110,00
Chauconin-Neufmontiers	24110,00
Étrépilly	24110,00
Limetz-Villez	24110,00
Paris 13e Arrondissement	24110,00
Champlan	24120,00
Draveil	24120,00
Frétoy	24140,00
Les Clayes-sous-Bois	24160,00
Vaux-sur-Lunain	24160,00

Ce graphique nous présente les villes où vivent les 10% de la population ayant un niveau de vie médian moyen. Pour le réalisé, j'ai créé une vue SQL avec la requête ci-dessous puis avec Power Bi créer une nouvelle colonne appeler fréquence qui contient la fréquence cumulé

(pop_cumul / population total). Ainsi on obtient la liste des villes où vivent les 10% avec un niveau de vie médian moyen. **(NB : le tableau peut être déroulé [scroller])**
(NB : il y'a des villes qui n'apparaissent pas car nous n'avons pas leurs populations)

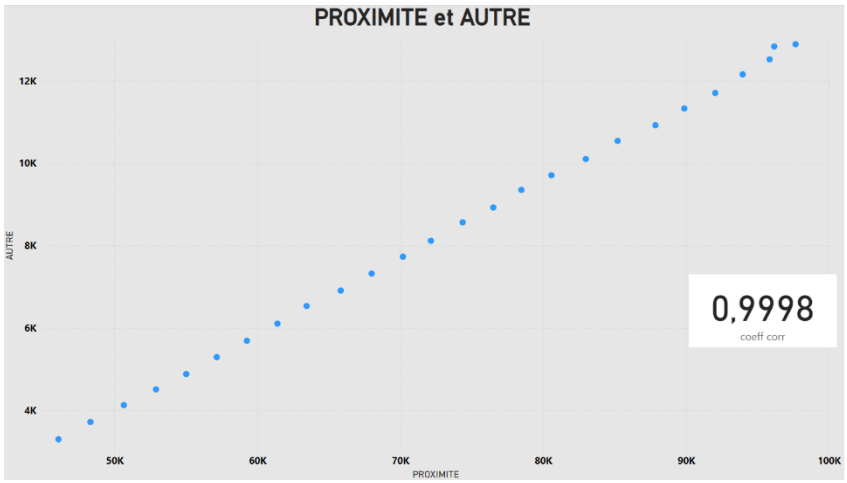
```sql
CREATE VIEW view_graph_2_g AS
SELECT
    dc.nom_commune,
    dc.mediane_niveau_vie,
    ABS(dc.mediane_niveau_vie - (SELECT AVG(dc2.mediane_niveau_vie) FROM dim_commune dc2)) AS
median_moins_moyenne,
    fc.population,
    SUM(fc.population) OVER (ORDER BY ABS(dc.mediane_niveau_vie - (SELECT
AVG(dc2.mediane_niveau_vie) FROM dim_commune dc2))) AS cumul_pop
FROM
    fact_commerce fc
    JOIN dim_commune dc ON fc.id_ville = dc.id_ville
    JOIN dim_type_commerce dtc ON fc.id_type_commerce = dtc.id_type_commerce
    JOIN dim_temps dt ON fc.id_temps = dt.id_temps
WHERE dt.annee = (SELECT MAX(annee) FROM dim_temps)
  AND dtc.nom_type_commerce = (SELECT MAX(nom_type_commerce) FROM dim_type_commerce);
```

8. Quelle est la répartition des types de commerce par niveau de vie ?



Ce graphique illustre la répartition des commerces en fonction du niveau de vie médian par commune en France. Chaque barre représente une commune, avec sa hauteur indiquant le nombre de commerces et sa couleur représentant le niveau de vie médian (plus foncé pour un niveau de vie plus faible). Les communes sont classées de gauche à droite par nombre de commerces décroissant, allant d'environ 4 000 pour les plus élevées à beaucoup moins pour les dernières. On observe que les communes avec le plus grand nombre de commerces de boucherie-charcuterie tendent à avoir un niveau de vie plus faible.

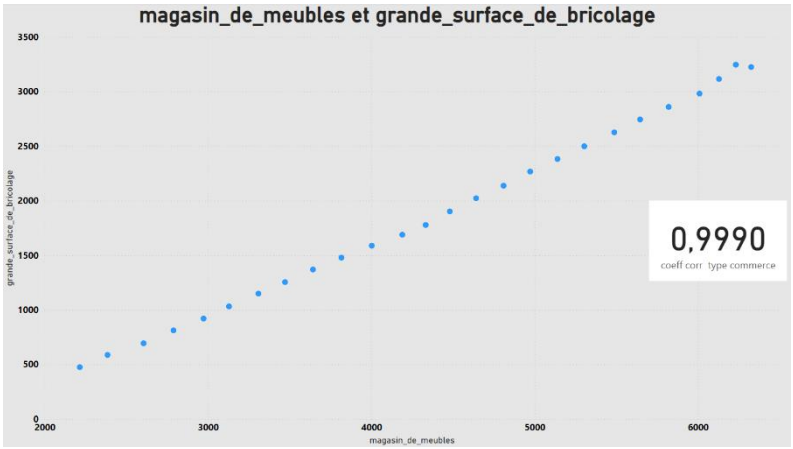
9. Pourriez-vous identifier une corrélation entre deux catégories de commerces ?



Ce graphique nous présente un nuage de point entre la catégorie « proximite » sur l'axe des abscisses et la catégorie « autre » sur l'axe des ordonnées. La disposition des points forment une ligne approximativement une ligne droite. Cela signifie qu'il existe une corrélation positive entre les deux variables : plus la valeur de « PROXIMITE » augmente, plus la valeur de « AUTRE » augmente. Le coefficient de corrélation étant très proche de 1 (arrondi) viens soutenir notre interprétation graphique. Cependant il faut faire attention à cette interprétation, en effet cette augmentation peut être dû à l'évolution démographique, c'est-à-dire comme la population a augmenté (augmentation de la demande), l'offre a aussi augmenté (augmentation des commerce).
Pour réaliser ce graphique, j'ai d'abord utilisé cette requête SQL pour créer une vue contenant les informations dont j'ai besoin groupé, puis utiliser power query pour créer un TCD.

```sql
CREATE VIEW graph_2_i AS
SELECT dt.annee, dcat.nom_categorie, SUM(fc.nombre_commerce) AS total_commerce
FROM
    fact_commerce fc JOIN dim_temps dt ON fc.id_temps = dt.id_temps
       JOIN dim_type_commerce dtc ON fc.id_type_commerce = dtc.id_type_commerce
       JOIN dim_categorie dcat ON dtc.id_categorie = dcat.id_categorie
GROUP BY dt.annee, dcat.nom_categorie
```

10. Pourriez-vous identifier une corrélation entre deux types de commerces ?



Ce graphique nous présente un nuage de point entre le type de commerce « magasin de meuble » sur l'axe des abscisses et le type de commerce « grande surface de bricolage » sur l'axe des ordonnées. La disposition des points forment une ligne approximativement

une ligne droite. Cela signifie qu'il existe une corrélation positive entre les deux variables : plus la valeur de « magasin de meuble » augmente, plus la valeur de « grande surface de bricolage » augmente. Le coefficient de corrélation étant très proche de 1 viens soutenir notre interprétation graphique. Cependant il faut faire attention à cette interprétation, en effet cette augmentation peut être dû à l'évolution démographique, c'est-à-dire comme la population a augmenté (augmentation de la demande), l'offre a aussi augmenté (augmentation des commerce). Pour réaliser ce graphique, j'ai d'abord utilisé cette requête SQL pour créer une vue contenant les informations dont j'ai besoin groupé, puis utiliser power query pour créer un TCD

```sql
CREATE VIEW view_graph_2_j AS
SELECT dt.annee, dtc.nom_type_commerce, SUM(fc.nombre_commerce) AS total_commerce
FROM fact_commerce fc JOIN dim_temps dt ON fc.id_temps = dt.id_temps
JOIN dim_type_commerce dtc ON fc.id_type_commerce = dtc.id_type_commerce
GROUP BY dt.annee, dtc.nom_type_commerce
```

IV. Lieux idéals pour ouvrir une poissonnerie

Afin de répondre cette question, nous avons dû définir qu'elles sont les caractéristiques du lieu idéal et leur niveau d'importance.
Le lieu idéal doit :

- Avoir un minimum de concurrent
- Les habitants doivent avoir un niveau de vie suffisant pour acheter nos produits
- Il doit également avoir une grande population pour pouvoir avoir des clients facilement.

Puis après cela, nous avons calculé un score pour chaque ville, les villes avec le score le plus grand sont les meilleurs lieux pour installer notre commerce.

1. Calcul pour nombre de commerce.

Nous voulons les villes avec le minimum de concurrent, donc pour chaque ville, nous avons calculé l'inverse du nombre commerce, ainsi les villes avec un faible nombre de poissonnerie auront un grand score et les autre un faible score.
Apres cette étape, nous devons normaliser nos données pour forcer les valeurs à être comprise entre 0 et 1. Pour cela, nous avons utilisé les formules suivantes :

$$inv\_nb\_com = \frac{1}{nombre\_commerce}$$

$$X\_normalise = \frac{X_i - X_{min}}{X_{max} - X_{min}} => X\_normalise \in [0,1]$$

Ce qui donne en DAX :

```dax
inverse_commerce = divide(1, [nombre_commerce], 1)

normalise_nb_commerce = DIVIDE([inverse_commerce] -
MIN(fact_commerce[inverse_commerce]), MAX(fact_commerce[inverse_commerce]) -
MIN(fact_commerce[inverse_commerce]))
```

2. Calcul pour niveau de vie et population.

Pour le niveau de vie et de population, nous ne voulons pas changer l'ordre des valeurs, donc nous n'avons pas besoin de les inversés, ce pendant nous devons les normalisé en utilisant la formule ci-dessus. Ce qui donne en DAX :

```
normalise_niveau_vie = DIVIDE([mediane_niveau_vie] - MIN(fact_commerce[mediane_niveau_vie]),
MAX(fact_commerce[mediane_niveau_vie]) - MIN(fact_commerce[mediane_niveau_vie]))

normalise_population = DIVIDE([population] - MIN(fact_commerce[population]),
MAX(fact_commerce[population]) - MIN(fact_commerce[population]))
```

3. Calcul du score.

Apres ces étapes, nous pouvons calculer le score de chaque ville. Pour cela il faut qu'on attribue un niveau d'importance à chacune de nos variables.
De mon côté, voici la répartition que j'ai choisi :

- **Nombre de commerce : 50%**
  Ce critère est le plus important, car il indique la dynamique commerciale de la ville et la probabilité de succès pour un nouveau commerce
- **Niveau de vie : 30%**
  Un niveau de vie plus élevé indique un pouvoir d'achat plus important, Ce critère est pondéré à 30 % pour refléter son impact et sans le surévaluer par rapport à l'activité commerciale
- **Population : 20%**
  Bien que la population représente la base de clients potentiels, elle a un poids moindre, car le succès dépendra aussi de leur pouvoir d'achat et de la dynamique commerciale de la ville.

Puis avec nous calculons le score avec la formule suivante, le score est entre 0 et 1.

$$score\_ville = 0.5 * normalise\_commerce + 0.3 * normalise\_niveau\_vie + 0.2 * normalise\_poputon$$

Ce qui donne en DAX :

```
score_besoin = 0.5*[normalise_nb_commerce] + 0.3*[normalise_niveau_vie] +
0.2*[normalise_population]
```

4. Classement des villes pour l'année 2020.

Ainsi, pour répondre à la question initiale à savoir quel sont les meilleurs villes pour installées des poissonneries, nous pouvons répondre selon ce tableau, par ordre décroissant de score : Saint-Cloud, Saint-Nom-la-Bretèche, Asnières-sur-Seine (top 3)

V. Devoir facultatif (bonus)

1. Les entités qu'on peut identifier dans ce fichier

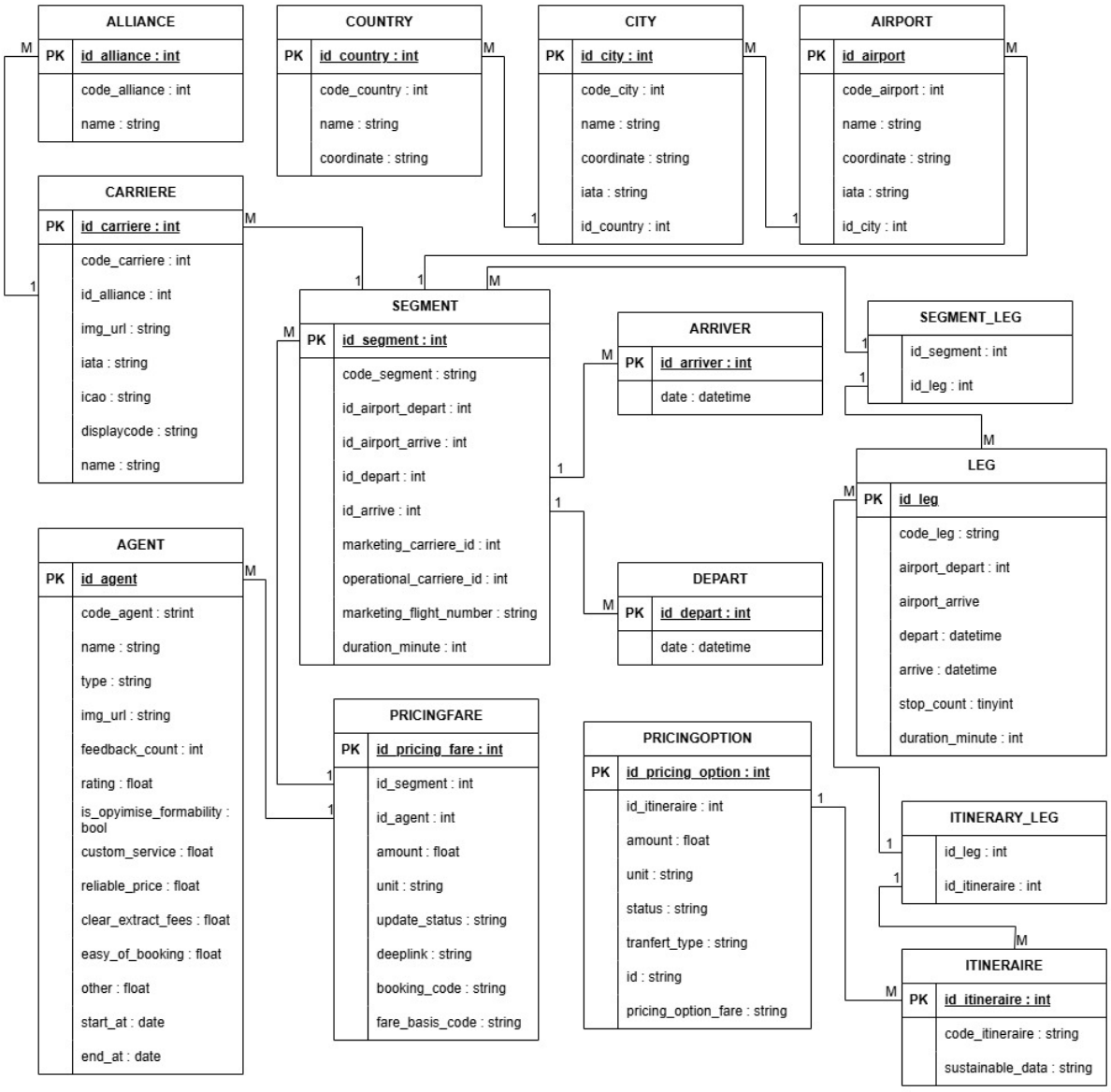
Les entités qu'on peut identifier dans ces fichiers sont : Alliance, Country, City, Airport, Carriere, Segment, Temp_Arriver, Temp_Depart, Agent, Leg, Pricing_Fare, Pricing_Option et Itineraire

2. Attribut de chaque entité.

- Alliance : code_alliance, name
- Country : code_country, name, coordinate
- City : code_city, name, coordinate, code_country
- Airport : code_airport, name, coordinate, code_city
- Carriere : code_carriere, img_url, iata, icao, displaycode, name, code_alliance
- Segment : code_segment, code_airport_depart, code_airport_arrive, date_depart, date_arrive, marketing_carriere_id, operational_carriere_id, marketing_flight_number, duration_minute
- Arriver : date
- Depart : date
- Leg : code_leg, airport_depart, airport_arrive, depart, arrive, duration_minute, stop_count
- Agent : code_agent, name, type, img_url, feedback_count, rating, is_opyimise_formability, custom_service, reliable_price, clear_extract_fees, easy_of_booking, other
- Pricing_fare : code_segment, code_agent, amount, unit, update_status, deeplink, booking_code, fare_basis_code

- Pricing_option : code_itineraire, amount, unit, status, tranfert_type, pricing_option_fare)
- Itineraire : code_itineraire, sustainable_data

3. Modèle



- Alliance : Les attributs de cette table sont id_alliance, code_alliance, et name. La clé technique est id_alliance, et la clé fonctionnelle est code_alliance, elle permet d'identifier de manière unique chaque alliance aérienne. Étant donné que nom d'une alliance change rarement, un SCD de type 1 est utilisé ici.
- Country : Cette table contient les attributs id_country, code_country, name et coordinate. La clé primaire est id_country, et la clé fonctionnelle est code_country. les informations sur les pays sont peu susceptibles de changer fréquemment, un SCD de type 1 est appliqué ici.
- City : La table CITY contient les attributs id_city, code_city, name, coordinate, iata, et id_country. La clé technique est id_city, et la clé fonctionnelle est code_city. Les changements dans les attributs comme le code iata ou les coordonnées sont rares,

donc un SCD de type 1 est appliqué. Id_country est une clé étrangère qui fait reference à id_country dans la table country.

- Airport : Cette table comporte les attributs id_airport, code_airport, name, coordinate, iata, et id_city. La clé technique est id_airport et la clé fonctionnelle est code_airport. Comme les aéroports changent rarement leur attribut, un SCD de type 1 est utilisé. Id_city est une clé étrangère qui fait reference à id_city dans la table city.
- Carriere : Les attributs de CARRIERE incluent id_carriere, code_carriere, id_alliance, img_url, iata, icao, displaycode et name. La clé technique est id_carriere, et la clé fonctionnelle est code_carriere. Puisque les informations de carriere change rarement, un SCD de type 1 est choisi ici. Id alliance est une clé etrangere qui fait reference à id_alliance dans la table alliance.
- Segment : Les attributs de cette table incluent id_segment, code_segment, id_airport_depart, id_airport_arrive, id_depart, id_arrive, marketing_carriere_id, operational_carriere_id, marketing_flight_number et duration_minute. La clé technique est id_segment, et la clé fonctionnelle est code_airport. Les détails de vol peuvent changer mais peu fréquemment, donc un SCD de type 1 s'applique ici. Id_airport_depart et id_airport_arrive sont des clés étrangères qui font référence à id_airport dans la table airport. Id_depart et id_arriver font référence respectivement aux tables depart et arriver.operational_carriere_id et marketing_carriere_id font référence à la table carriere.
- Arriver : Cette table contient les attributs id_arriver et date, avec la clé primaire id_arriver utilisée pour identifier chaque enregistrement d'arrivée. Étant donné que les dates d'arrivée sont spécifiques aux occurrences, un SCD de type 1 est approprié.
- Depart : Les attributs de cette table sont id_depart et date. La clé technique est id_depart, identifiant de manière unique chaque enregistrement de départ. Comme les départs sont généralement des enregistrements fixes, un SCD de type 1 est utilisé.
- Leg : Cette table inclut id_leg, code_leg, airport_depart, airport_arrive, depart, arrive, stop_count et duration_minute. La clé technique est id_leget la clé fonctionnel est code_leg. Puisque les étapes de vol peuvent subir des modifications opérationnellesque rarement, un SCD de type 1 est appliqué.
- Segment_Leg : Cette table associative contient id_segment et id_leg, reliant les segments aux étapes. En tant que table de liaison, elle ne nécessite pas de SCD.
- Pricing_fare : Les attributs ici incluent id_pricing_fare, id_segment, id_agent, amount, unit, update_status, deeplink, booking_code et fare_basis_code. La clé primaire id_pricing_fare identifie de manière unique chaque enregistrement de tarif. C’est une table de fait, elle n’a donc pas besoin de SCD.
- Pricing_option : Cette table inclut id_pricing_option, id_itineraire, amount, unit, status, tranfert_type, id, et pricing_option_fare. La clé primaire est id_pricing_option, qui identifie de manière unique chaque option tarifaire. C’est une table de fait
- Itineraire : Les attributs incluent id_itineraire, code_itineraire et sustainable_data. La clé primaire id_itineraire identifie de manière unique chaque itinéraire. Puisque que la structure d'un itinéraire reste généralement stable un SCD de type 1 est utilisé ici. La clé fonctionnelle est code_itineraire
- Itineraire_Leg : Cette table associative contient id_leg et id_itineraire, reliant les étapes aux itinéraires. En tant que table de liaison, elle ne nécessite pas de SCD.
- Agent : Les attributs incluent id_agent, code_agent, name, type, img_url, feedback_count, rating, is_optimise_formability, custom_service, reliable_price,

clear_extract_fees, easy_of_booking, other, start_at et end_at. La clé primaire est id_agent et la clé fonctionnelle est code_agent, qui identifie de manière unique chaque agent. Comme les données sur les agents (notation) peuvent évoluer avec le temps, un SCD de type 2 est appliqué avec start_at et end_at pour suivre l'historique des changements.

4. Explication détaillé

a. Comment faire pour trouver les entités.

Afin de trouver les entités, j'ai d'abord formaté le JSON pour qu'il soit lisible grâce à un site web (https://jsonformatter.curiousconcept.com/). Puis avec Visual Studio Code, j'ai ouvert le fichier et analysé chaque section, ce qui m'a permis de déterminé les entités et leurs attributs. Puis en faisant plusieurs recherches dans le fichier (ctrl + f), j'ai pu identifier les relations entre les entités. J'ai fait des recherche sur internet afin de savoir que représente chacune des entités ou (ex : segment) attribut (ex : iata) que je ne comprenais pas.

b. Explication de code réalise.

- Importation module : J'ai d'abord commencé par importer les modules que j'utiliserai. Nous avons json (qui est un module python qui permet de transformé les ficher json en dictionnaire python), psycopg2 (un module qui permet d'interagir avec la base de données postgreSQL) et datetime (qui permet de gérer les date en python).
- Connexion à la base de données : je me suis connecter à ma base de données postgreSQL, puis créer une fonction nommée execute_query() qui va me permettre d'exécuter mes requête futur.
- Création de table temporaire : dans un fichier SQL, j'ai écrit un script qui vite toute mes table temporaire. Donc je lis le fichier, et exécute les requêtes.
- Peuplement : pour peupler mes table, j'ai commencé par récupérer l'entité qu'i m'intéressais dans une variable, puis créer une boucle qui va récupérer chaque occurrence et écrire une requête d'insertion. Puis cette requête est exécuté, ce qui va remplir ma table temporaire, puis j'ai cris une autre requête qui insert dans ma table final que les valeurs qui n'existe pas encore, si il y'a des jointures à faire (ex : récupérer id) je les fais directement avec cette requête.
- Fermeture connexion : après tous les traitements, je ferme la connexion avec la base de données.

NB : le code étant long, je ne peux le mettre dans ce PDF, je vous le mets le lien vers mon GitHub pour y avoir accès.

Lien :
https://github.com/diallothierno0223/peuplement_sid/blob/main/traitement.ipynb

Exemple table carriere :

```
carriere = data['carriers']

query = "INSERT INTO WORK_CARRIERE (id_carriere, code_carriere, code_alliance,img_url, iata,
icao, displaycode, name) VALUES "

i = 1
for key, item in carriere.items():
    code_carriere = int(key)
    name = item['name']
    try:
        alliance_code = int(item['allianceId'])
    except:
        alliance_code = "NULL"
    img_url = item['imageUrl']
    iata = item['iata']
    icao = item['icao']
    displaycode = item['displayCode']

    query += f"({i}, {code_carriere}, {alliance_code}, '{img_url}', '{iata}', '{icao}', '{displaycode}', '{name}'),"
    i += 1

query = query[:-1] + ";"
execute_query(query)

query = "INSERT INTO WORK_CARRIERE2 (id_carriere, code_carriere, id_alliance,img_url, iata, icao,
displaycode, name) SELECT id_carriere, code_carriere, id_alliance, img_url, iata, icao, displaycode,
wc.name FROM ALLIANCE wa RIGHT JOIN WORK_CARRIERE wc on wa.code_alliance =
wc.code_alliance;"

execute_query(query)

query = "INSERT INTO CARRIERE (code_carriere, id_alliance,img_url, iata, icao, displaycode, name)
SELECT code_carriere, id_alliance,img_url, iata, icao, displaycode, name FROM WORK_CARRIERE2
WHERE NOT EXISTS (SELECT 1 FROM CARRIERE WHERE CARRIERE.code_carriere =
WORK_CARRIERE2.code_carriere);"
execute_query(query)
```