Sellappah Shamini

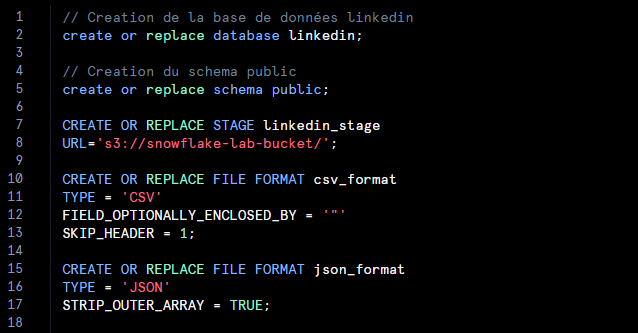
Gastinel Louis

Calimouttoupoulle Davin

Youssef Ben Jemaa

## 

## **PLAN GLOBAL DU PROJET LINKEDIN – SNOWFLAKE & STREAMLIT**



Ici montre l’étape de préparation de l’espace de travail dans Snowflake.

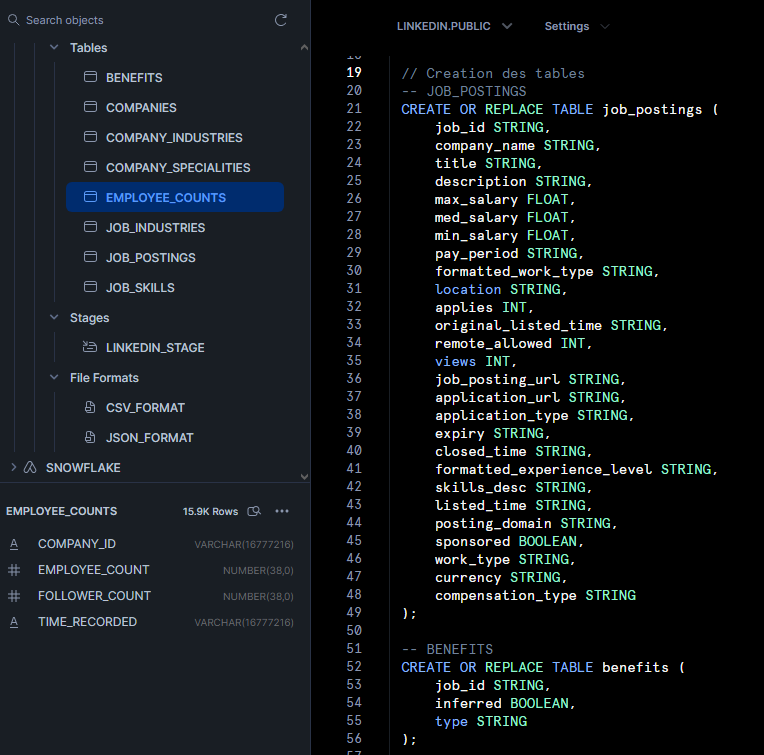
On y crée :

• une base de données nommée linkedin

• un schéma public pour organiser les tables

• un stage externe pointant vers le bucket S3 où se trouvent les fichiers

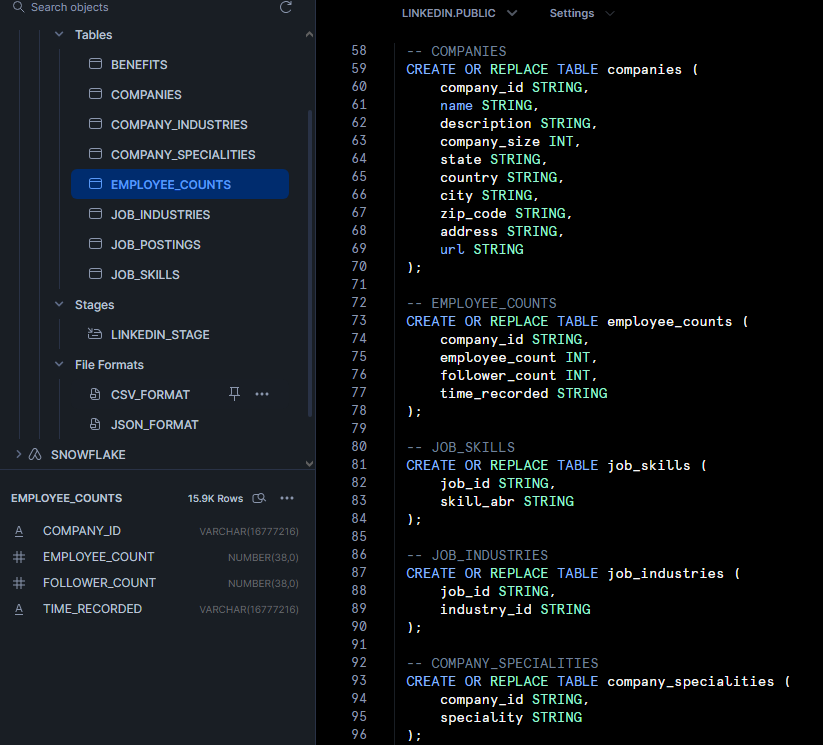
• deux formats de fichiers : un pour les fichiers CSV, un autre pour les JSON (avec option STRIP\_OUTER\_ARRAY = TRUE pour déplier les tableaux JSON).



Ici, on voit la création de la table principale job\_postings, qui contient toutes les informations liées aux offres d’emploi : titre, description, salaire, type de contrat, lieu, date de publication, etc.

Cette table est au cœur du projet car c’est celle qui regroupe les données des annonces publiées sur LinkedIn.

On crée également la table benefits, qui permet d’associer les avantages proposés à chaque offre via l’identifiant job\_id.



Ici on crée les autres tables complémentaires :

•companies : les infos sur chaque entreprise (taille, localisation, site, etc.)

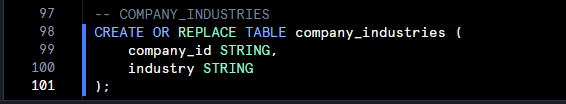
•employee\_counts : le nombre d’employés et de followers sur LinkedIn

•job\_skills : compétences associées à chaque offre

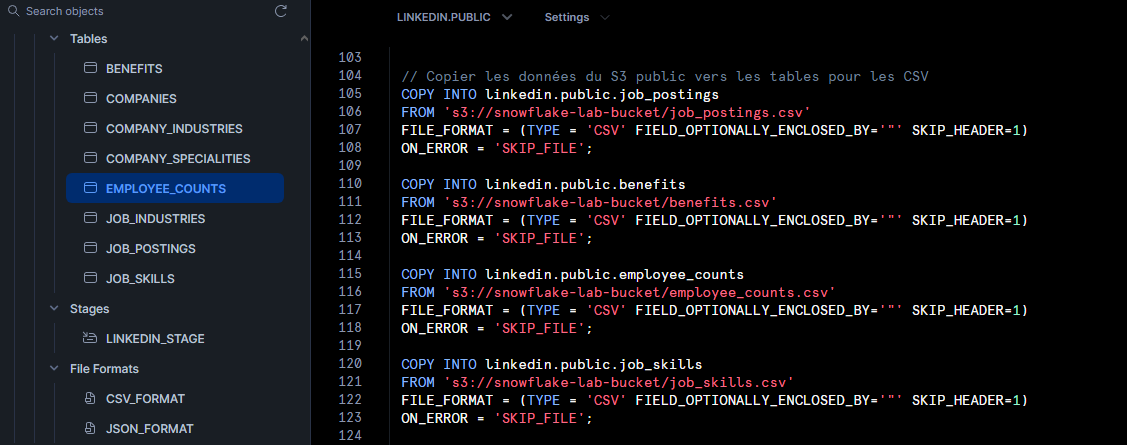
•job\_industries : secteurs d’activité liés aux offres

•company\_specialities et company\_industries : spécialités et industries associées aux entreprises

Ces tables seront ensuite jointes à job\_postings pour enrichir les analyses.



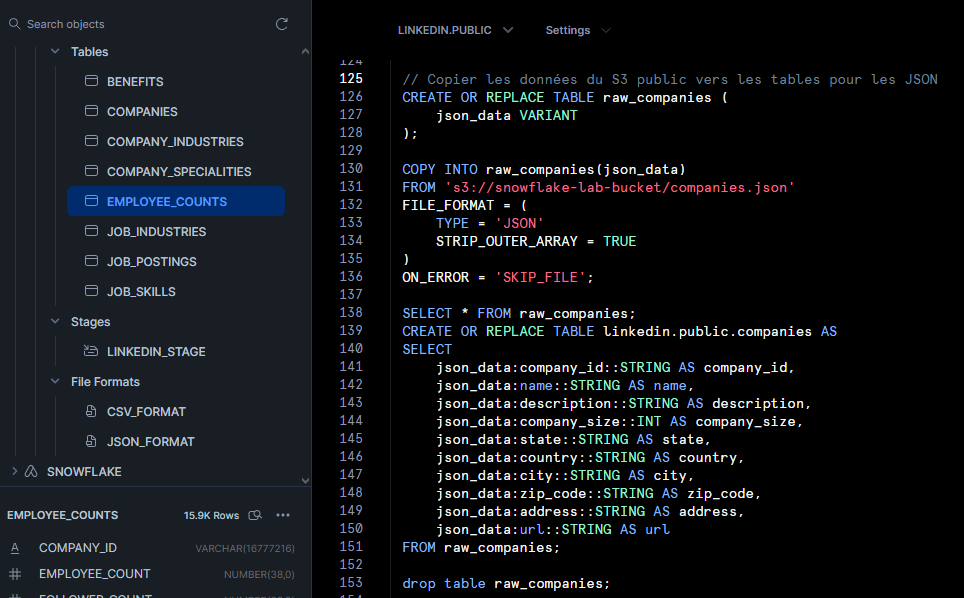
Création de la table company\_industries



L’image ici illustre le chargement des fichiers CSV depuis le bucket S3 vers les tables créées précédemment.

On utilise ici la commande COPY INTO avec le format CSV défini plus tôt.

Chaque fichier est envoyé dans sa table correspondante (job\_postings, benefits, employee\_counts, job\_skills), avec l’option SKIP\_HEADER = 1 pour ignorer la ligne d’en-tête.



-Chargement et traitement des fichiers JSON

Ce screen montre comment on gère les fichiers JSON, qui contiennent des objets encodés dans des tableaux [ {...}, {...} ].

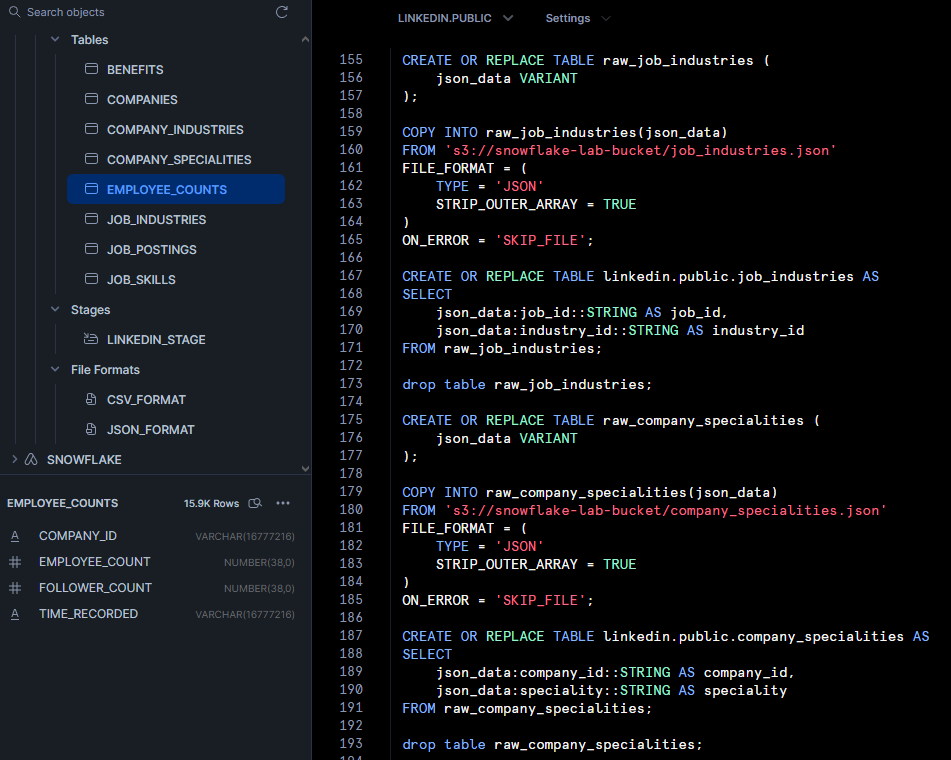
1. On commence par créer une table intermédiaire raw\_companies avec une seule colonne de type VARIANT (pour stocker du JSON brut).

2. On utilise COPY INTO ... STRIP\_OUTER\_ARRAY = TRUE pour déplier le tableau JSON.

3. On transforme ensuite le contenu JSON en vraies colonnes SQL dans une table compagnies, à l’aide d’un SELECT.

4. On supprime la table intermédiaire

Cette méthode est répétée pour les autres fichiers JSON dans les screens suivants

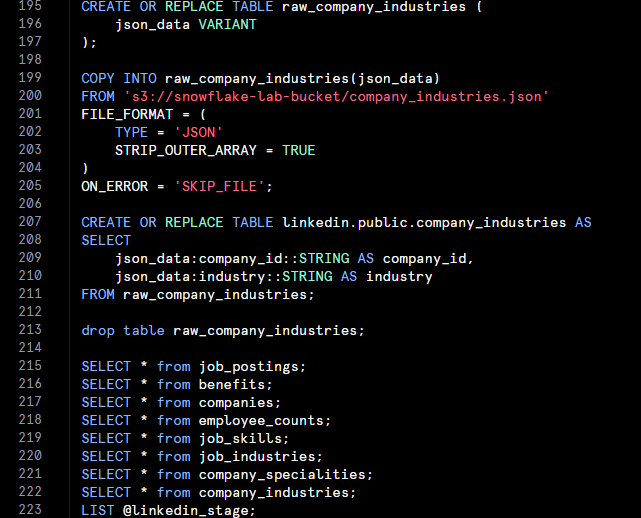


Même logique que précédemment, mais cette fois pour les fichiers job\_industries et company\_specialities.

• On crée d’abord des tables raw\_... pour importer les données JSON.

• Ensuite, on extrait les champs nécessaires (job\_id, industry\_id, company\_id, speciality) vers de vraies tables relationnelles exploitables.

• À la fin, on supprime les tables intermédiaires pour garder un environnement propre.

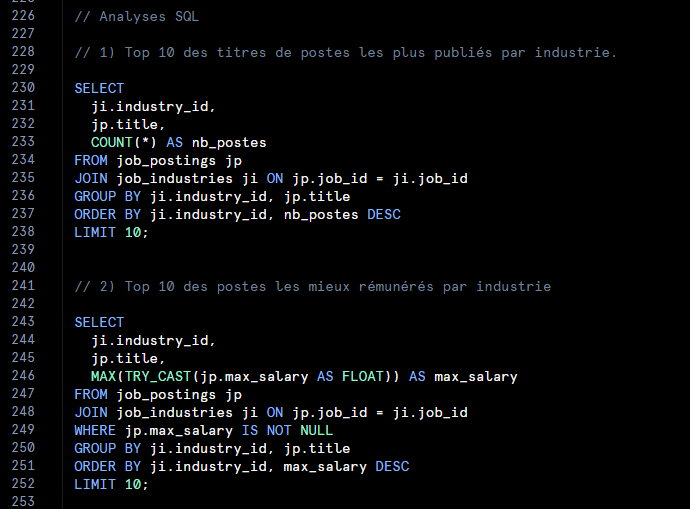


Ici, on importe company\_industries.json comme précédemment puis on crée la table finale company\_industries.

On exécute ensuite une série de SELECT \* FROM ... pour s’assurer que toutes les données ont bien été chargées et que les tables sont opérationnelles et enfin, la commande LIST @linkedin\_stage; permet de lister tous les fichiers disponibles dans le stage S3.

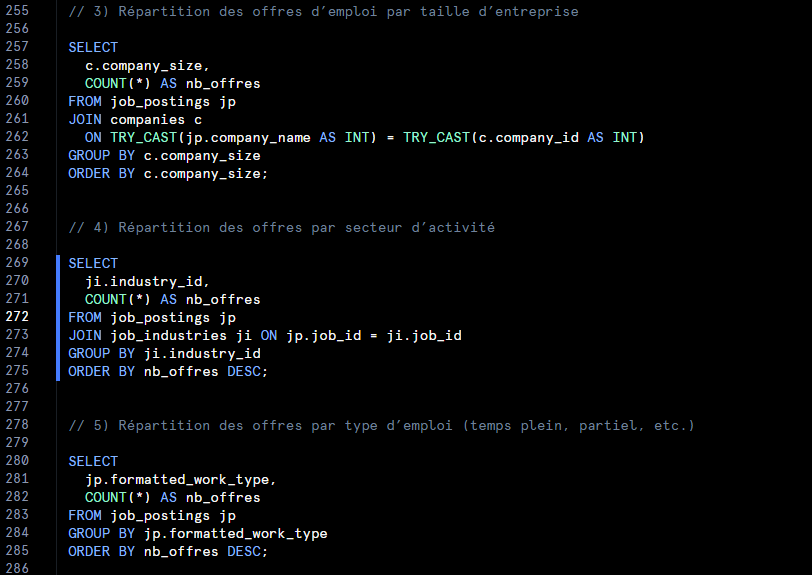
Analyse des données

1. Top 10 des titres de postes les plus publiés par industrie.
2. Top 10 des postes les mieux rémunérés par industrie.



Ici, pour faire un top 10, on utilise GROUP BY pour agréger les données (ex. par produit ou client), puis ORDER BY pour les trier selon un critère (comme une somme ou un nombre), et enfin LIMIT 10 pour ne garder que les 10 premières lignes du classement. De même pour la deuxième question.

1. Répartition des offres d’emploi par taille d’entreprise
2. Répartition des offres par secteur d’activité
3. Répartition des offres par type d’emploi



3. Ici, on relie chaque offre d’emploi à la taille de l’entreprise qui la publie. Comme les identifiants sont parfois sous forme de texte ou de nombre avec décimales, on utilise TRY\_CAST pour forcer la comparaison entre les deux.

On agrège ensuite les offres (COUNT(\*)) selon la taille d’entreprise (company\_size), puis on trie les résultats par ordre croissant de taille. Cela permet de voir quelles tailles d’entreprise recrutent le plus.

4. On relie ici chaque offre à son secteur d’activité via la table de jointure job\_industries.

Puis on compte combien d’offres sont présentes dans chaque secteur (industry\_id).

Enfin, on trie les résultats du secteur qui publie le plus d’offres à celui qui en publie le moins.

5. Cette requête nous montre le type d’emploi proposé dans les offres (temps plein, temps partiel, contrat, etc.).

On regroupe par formatted\_work\_type et on compte les occurrences pour voir les formes de contrat les plus représentées dans la base.