Projet Data Engineering : Pipeline de Streaming Musical

De PostgreSQL à Snowflake et Power BI

Ahlam Oubouazza

Table des matières

Chapitre 1

Introduction

Ce projet vise à mettre en place un pipeline de données complet et moderne pour une plateforme de streaming musical, allant de la capture des transactions (OLTP) à l'analyse décisionnelle (BI). Le projet couvre l'ensemble du cycle de vie des données :

- Conception et implémentation d'une base de données transactionnelle (OLTP) sous PostgreSQL
- Génération de données synthétiques réalistes pour simuler l'activité d'une plateforme
- Modélisation dimensionnelle avec un schéma en étoile (Star Schema)
- Processus ETL avec gestion de l'historique (SCD Type 2)
- Migration vers une plateforme cloud moderne (Snowflake)
- Visualisation et analyse des données avec Power BI

L'objectif principal est de démontrer les compétences techniques nécessaires pour construire une architecture de données end-to-end, en combinant des technologies on-premise et cloud, tout en respectant les meilleures pratiques du domaine.

Chapitre 2

Conception de la Source (OLTP)

2.1 Modèle Relationnel

La source de données est gérée sous PostgreSQL. Le schéma comprend 5 tables principales : Utilisateurs(users), Artistes(artist), Morceaux(tracks), Abonnements(subscriptions) et Écoutes(stream).

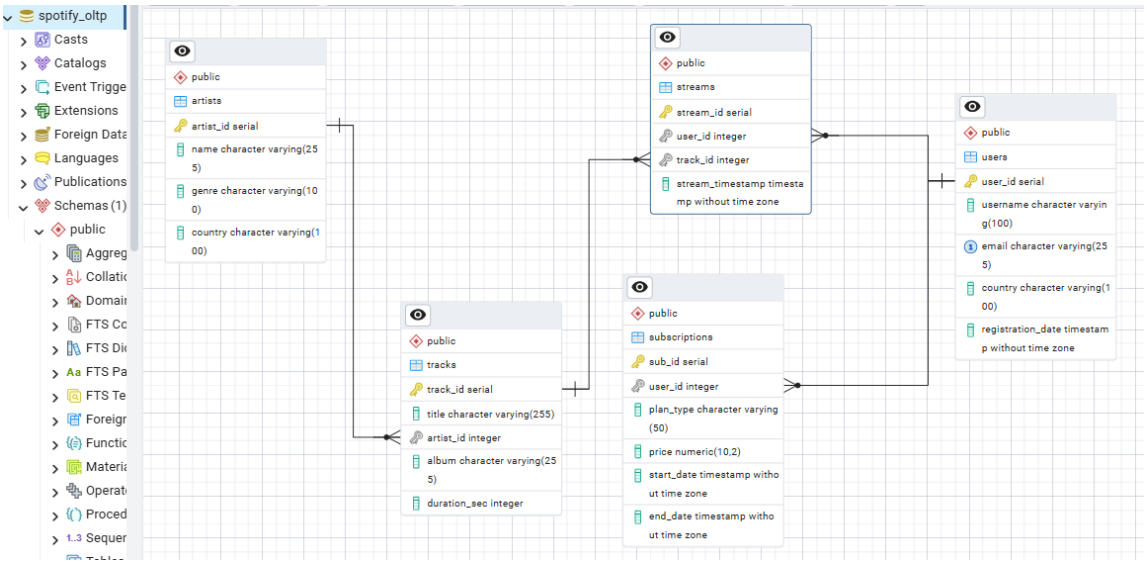

FIGURE 2.1 – Diagramme Entité-Relation de la base de données Spotify OLTP

Le schéma ci-dessus illustre les relations entre les entités. On y observe

notamment que la table `streams` est au centre de l'activité, liant les utilisateurs aux morceaux de musique.

2.2 Dictionnaire des données

Voici une description détaillée des tables principales :

— **users** : Contient les informations de profil (nom d'utilisateur, email, pays) et la date d'inscription. Cette table sert de référence pour l'analyse démographique.

— **artists** : Regroupe les informations sur les artistes (nom, genre musical, pays d'origine). Cette dimension permet d'analyser les tendances par genre et par région.

— **tracks** : Catalogue des morceaux avec leur titre, durée, album et référence à l'artiste. C'est une table centrale pour l'analyse de contenu.

— **subscriptions** : Gère le type de compte (Free/Premium) avec les dates de début et de fin. C'est ici que nous suivrons l'évolution des revenus et le taux de conversion.

— **streams** : Table transactionnelle enregistrant chaque lecture (id_user, id_track, timestamp). Elle représente le cœur métier de la plateforme et génère les KPIs d'engagement.

2.3 Justification des choix techniques

PostgreSQL a été choisi comme système OLTP pour plusieurs raisons :

— Robustesse et respect des propriétés ACID

— Support complet des contraintes d'intégrité référentielle

— Performance optimale pour les transactions courtes

— Open source et largement adopté dans l'industrie

Chapitre 3

Génération des Données

3.1 Méthodologie

Pour simuler une activité réelle, un script Python utilisant la bibliothèque `Faker` a été développé. Ce script permet de peupler les tables avec des données cohérentes et représentatives d'une plateforme de streaming réelle :

- **Volume** : 51 artistes, 200 morceaux et 100 utilisateurs
- **Transactions** : Environ 2000 événements d'écoutes répartis sur les 90 derniers jours
- **Distribution** : Respect des lois de distribution réalistes (ex : loi de Pareto pour la popularité des morceaux)

3.2 Qualité des données

Certaines contraintes ont été intégrées lors de la génération afin de garantir la cohérence des données :

- Durée des morceaux comprise entre 2 et 5 minutes
- Respect des formats de données (emails valides, dates cohérentes)
- Cohérence référentielle entre les clés étrangères
- Distribution réaliste des abonnements (55% Free, 45% Premium)

Ces contraintes permettent également de tester ultérieurement les processus de nettoyage et de validation dans l'ETL.

3.3 Validation de l'insertion

Après l'exécution du script, une vérification de la volumétrie a été effectuée directement dans PostgreSQL pour confirmer le bon fonctionnement

de l'alimentation.

Query Query History

```
1  SELECT
2      (SELECT COUNT(*) FROM users) as nb_utilisateurs,
3      (SELECT COUNT(*) FROM artists) as nb_artistes,
4      (SELECT COUNT(*) FROM tracks) as nb_tracks,
5      (SELECT COUNT(*) FROM streams) as nb_ecoutes;
```

Data Output Messages Notifications

Show

	nb_utilisateurs bigint	nb_artistes bigint	nb_tracks bigint	nb_ecoutes bigint
1	100	50	200	2000

FIGURE 3.1 – Vérification de la volumétrie des données par table

L'aperçu des données ci-dessous montre la structure des logs d'écoutes générés, incluant les horodatages nécessaires pour les analyses temporelles.

Query Query History Scratch Pad

```
1  SELECT * FROM users LIMIT 5;
```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1

	user_id [PK] integer	username character varying (100)	email character varying (255)	country character varying (100)	registration_date timestamp without time zone
1	1	jaredwood	martinalexander@example....	Denmark	2025-08-11 00:00:00
2	2	gallagherlaura	popekim@example.com	Niger	2025-07-08 00:00:00
3	3	qclark	nalvarez@example.com	Colombia	2025-12-04 00:00:00
4	4	sheilareid	robinsonsarah@example.net	Bangladesh	2025-02-11 00:00:00
5	5	jeremypaul	robert10@example.org	Azerbaijan	2025-01-14 00:00:00

FIGURE 3.2 – Extrait des données transactionnelles de la table STREAMS

Chapitre 4

Modélisation du Data Warehouse

4.1 Choix du modèle

J'ai opté pour un schéma en étoile (Star Schema) afin d'optimiser les performances des requêtes analytiques. Contrairement au modèle OLTP qui est normalisé pour éviter la redondance, ce modèle OLAP est conçu pour :

— Simplifier les jointures entre tables
— Accélérer l'agrégation des données
— Faciliter la compréhension pour les analystes métier
— Optimiser les performances des outils de BI

Le schéma en étoile sépare clairement les mesures (faits) des axes d'analyse (dimensions), permettant des requêtes analytiques rapides et intuitives.

4.2 Structure du Schéma en Étoile

Le passage au modèle OLAP (OnLine Analytical Processing) impose une dénormalisation. La structure créée sous le schéma `dwh` se compose des éléments suivants :

4.2.1 Tables de Dimensions

Elles contiennent les données descriptives permettant de filtrer et regrouper les analyses :

— **dim_users** : Contient les informations socio-démographiques des auditeurs (pays, date d'inscription, type d'abonnement). Cette dimension permet l'analyse par segment d'utilisateurs.

— **dim_tracks** : Fusionne les informations des morceaux et des noms d'artistes pour éviter une jointure supplémentaire lors de l'analyse par genre ou par artiste. Cette dénormalisation améliore significativement les performances.
— **dim_date** : Permet une analyse granulaire temporelle (par jour, mois, trimestre, année). Cette dimension facilite l'identification de tendances saisonnières et l'analyse de séries temporelles.
— **dim_subscriptions** : Gère les types d'abonnements et leur évolution dans le temps.

4.2.2 Table de Faits

La table **fact_streams** est le cœur du modèle. Elle stocke :
— Les clés étrangères vers les dimensions (user_key, track_key, date_key)
— Les mesures quantitatives : nombre d'écoutes (`stream_count`) et durée totale d'écoute
— Les métriques calculables : taux d'achèvement, revenus générés

4.3 Implémentation Physique

Le schéma a été implémenté dans PostgreSQL sous un espace de nommage distinct nommé `dwh`. Cette séparation logique permet de :
— Isoler les environnements OLTP et OLAP
— Gérer les droits d'accès de manière granulaire
— Faciliter la maintenance et les déploiements

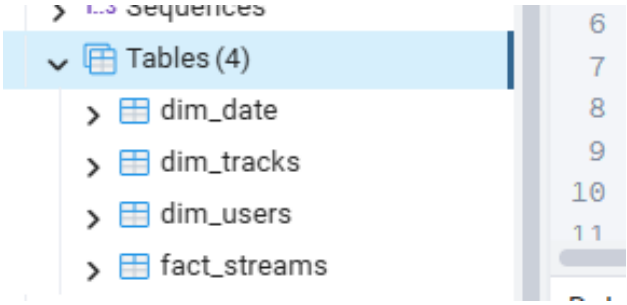

FIGURE 4.1 – Liste des tables du Data Warehouse dans le schéma DWH

Le diagramme ci-dessous illustre visuellement les relations de type "un-à-plusieurs" entre les dimensions et la table de faits.

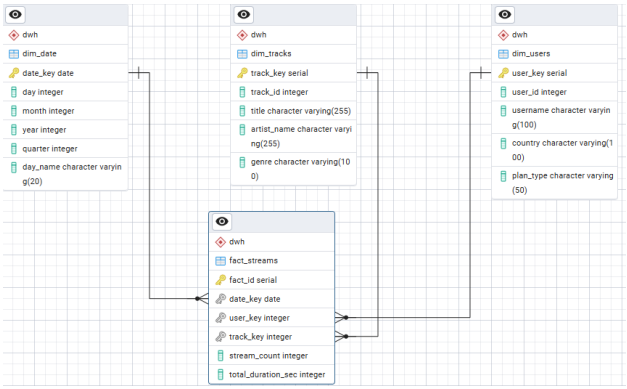

FIGURE 4.2 – Diagramme du Schéma en Étoile (Star Schema) pour le Streaming Musical

Chapitre 5

Processus ETL (Extract, Transform, Load)

5.1 Architecture du pipeline

Le processus ETL a été réalisé à l'aide d'un script Python utilisant la bibliothèque `psycopg2`. Ce choix permet :

— Une grande flexibilité pour manipuler les données
— Un contrôle fin sur les transformations complexes
— Une intégration facile avec d'autres outils Python (Pandas, NumPy)
— Une gestion robuste des erreurs et du logging

5.2 Étapes de transformation

Le script suit une logique séquentielle pour garantir l'intégrité référentielle :

1. **Extraction et Dimension Temporelle** : Les dates brutes des écoutes sont extraites et décomposées (jour, mois, année, jour de la semaine, trimestre) pour alimenter `dim_date`. Cela permet des analyses saisonnières sans calculs coûteux à la volée.
2. **Dénormalisation des Données** : Pour la table `dim_tracks`, une jointure (*JOIN*) est effectuée entre les tables `tracks` et `artists` de la source. Le nom de l'artiste est "aplati" dans la dimension track, éliminant le besoin de jointures lors des requêtes analytiques.
3. **Gestion de l'Historique (SCD Type 2)** : Pour `dim_users`, le script implémente une logique de détection des changements et d'historisation complète.

5.3 Journal d'exécution (Logs)

L'exécution du script s'est déroulée avec succès, comme en témoigne la trace console suivante :

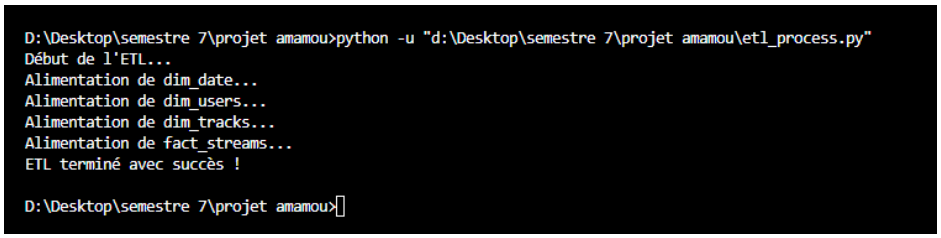

FIGURE 5.1 – Journal d'exécution du script ETL Python

5.4 Vérification dans le Data Warehouse

Après l'exécution, nous avons vérifié le remplissage de la table de faits. La figure 5.2 montre que les clés étrangères pointent correctement vers les dimensions et que les mesures sont prêtes pour l'analyse.

```
1   SELECT * FROM dwh.fact_streams LIMIT 10;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1

	fact_id [PK] integer	date_key date	user_key integer	track_key integer	stream_count integer	total_duration_sec integer
1	1	2025-12-...	82	58	1	159
2	2	2025-12-...	71	182	1	295
3	3	2025-12-...	24	22	1	146
4	4	2025-12-...	79	155	1	130
5	5	2025-10-...	7	84	1	231
6	6	2025-11-...	93	77	1	298
7	7	2025-12-...	50	62	1	202
8	8	2025-12-...	69	8	1	143
9	9	2025-12-...	28	196	1	269
10	10	2025-10-...	47	120	1	192

FIGURE 5.2 – Aperçu des données chargées dans la table de faits

5.5 Logique d'historisation (SCD Type 2)

5.5.1 Principe et Justification

L'un des points critiques de l'ETL est la gestion des changements d'état des utilisateurs. Contrairement à un simple *Update* qui écraserait l'historique, notre script implémente une logique de **Slowly Changing Dimension Type 2 (SCD Type 2)**.

Cette approche est essentielle pour :

— Conserver l'intégrité des analyses historiques
— Calculer précisément les revenus passés selon les abonnements de l'époque
— Mesurer les taux de conversion et de rétention
— Analyser l'impact des changements d'abonnement sur le comportement d'écoute

5.5.2 Implémentation Technique

Pour répondre aux exigences de gestion de l'historique, la table `dim_users` a été enrichie de trois colonnes :

— `start_date` : Date de début de validité du profil
— `end_date` : Date de fin de validité (NULL si le profil est actuel)
— `is_active` : Indicateur booléen permettant d'identifier rapidement la version courante d'un utilisateur

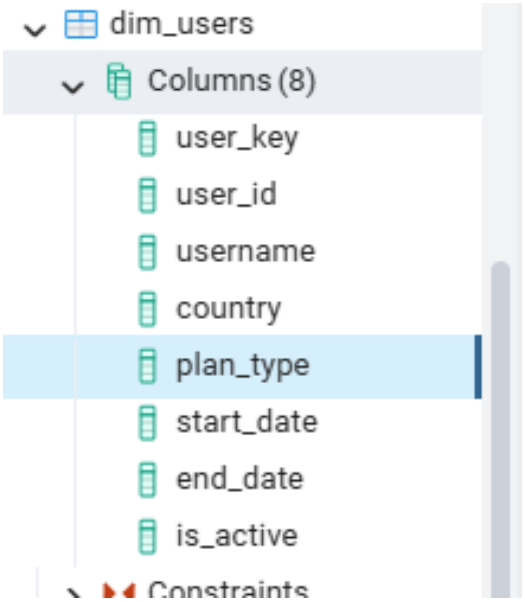

FIGURE 5.3 – Structure de la table dim_users avec support SCD Type 2

5.5.3 Algorithme d'Historisation

Le processus suit ces étapes :

1. Comparaison du `plan_type` de la source avec celui du DWH
2. Si un changement est détecté :
   — La ligne active est marquée comme expirée (`is_active = false`)
   — Une `end_date` est ajoutée (date du changement)
   — Une nouvelle version du profil utilisateur est créée
   — La nouvelle ligne devient la ligne de référence actuelle
3. Les clés de substitution (surrogate keys) garantissent l'unicité de chaque version

5.5.4 Validation de l'Historisation

Pour valider le bon fonctionnement du SCD Type 2, nous avons effectué un test en conditions réelles :

1. Modification du plan d'abonnement d'un utilisateur dans la source (passage de 'Free' à 'Premium')
2. Relance du script ETL
3. Vérification de la présence de deux enregistrements pour le même utilisateur

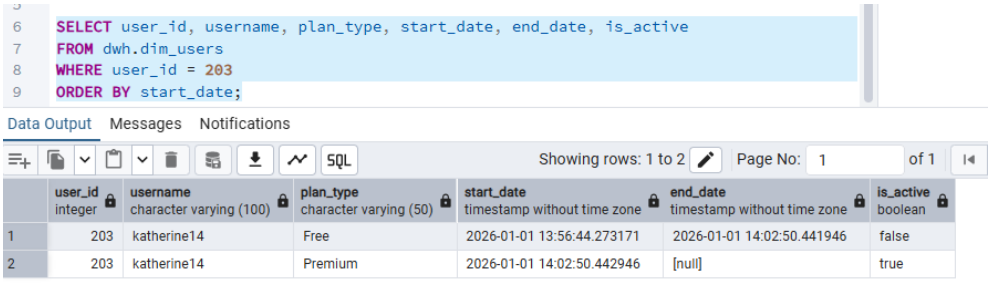

FIGURE 5.4 – Preuve de l'historisation : Deux lignes pour l'utilisateur 3 (une inactive, une active)

Comme illustré ci-dessus, le système gère parfaitement l'historique :
— La ligne initiale (user_key=3, plan_type='Free') est marquée comme inactive avec une date de fin
— Une nouvelle ligne (user_key=103, plan_type='Premium') est créée pour refléter le statut actuel

— Cela permet de conserver l'intégrité des analyses historiques tout en reflétant l'état actuel

Cette approche garantit que les analyses de revenus historiques restent exactes, même si un utilisateur change de catégorie d'abonnement au fil du temps.

Chapitre 6

Migration vers Snowflake Cloud Data Warehouse

6.1 Pourquoi Snowflake ?

Après avoir construit le pipeline ETL localement avec PostgreSQL, la décision a été prise de migrer vers Snowflake, une plateforme cloud moderne de Data Warehousing. Ce choix stratégique repose sur plusieurs facteurs :

6.1.1 Avantages Techniques

- **Architecture découplée** : Séparation du stockage et du calcul permettant une scalabilité indépendante
- **Performance optimisée** : Clustering automatique et optimisation des requêtes
- **Élasticité** : Capacité de redimensionner les ressources à la demande
- **Partage de données** : Facilite la collaboration inter-équipes et inter-organisations
- **Zero-copy cloning** : Création instantanée d'environnements de test

6.1.2 Avantages Opérationnels

- **Maintenance simplifiée** : Pas de gestion d'infrastructure à prévoir
- **Sécurité avancée** : Chiffrement de bout en bout et contrôle d'accès granulaire
- **Multi-cloud** : Support AWS, Azure et GCP
- **Coût optimisé** : Paiement à l'usage (pay-per-query)

6.1.3 Alignement avec les Tendances du Marché

Snowflake représente l'évolution naturelle des architectures de données modernes. Les entreprises migrent massivement vers le cloud pour bénéficier de l'agilité et de la scalabilité qu'offrent ces plateformes. Ce projet démontre donc une compréhension des technologies actuelles et des compétences recherchées sur le marché.

6.2 Création du Compte et Configuration Initiale

6.2.1 Interface Snowflake

L'interface web de Snowflake offre une expérience utilisateur intuitive pour gérer l'ensemble de l'écosystème data.

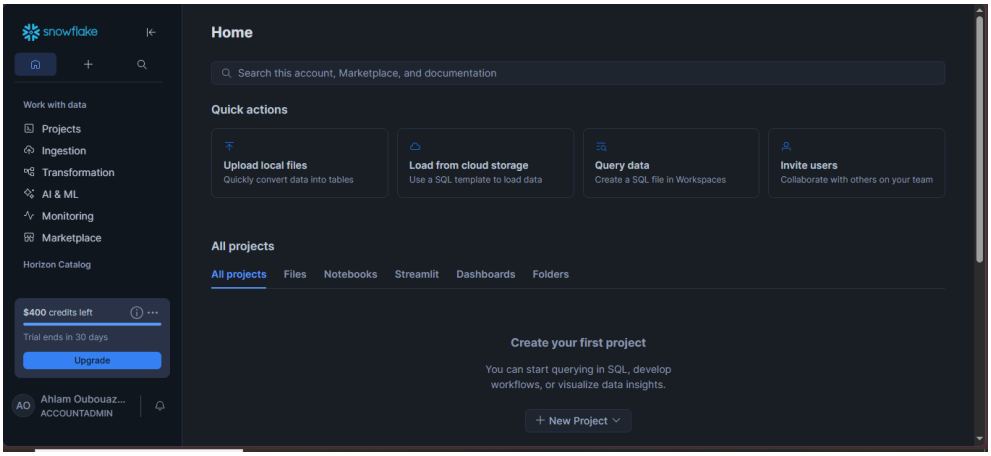

FIGURE 6.1 – Page d'accueil Snowflake - Vue d'ensemble du compte

Comme on peut le voir sur la figure 6.1, l'interface propose plusieurs points d'entrée :

- **Projects** : Gestion des workspaces et notebooks
- **Upload local files** : Import rapide de fichiers CSV, JSON, Parquet
- **Load from cloud storage** : Connexion à S3, Azure Blob, GCS
- **Query data** : Éditeur SQL intégré avec auto-complétion

6.2.2 Configuration du Warehouse

Un warehouse virtuel nommé `COMPUTE_WH` a été créé avec les spécifications suivantes :

- Taille : X-Small (optimisé pour le développement)
- Auto-suspend : 10 minutes d'inactivité
- Auto-resume : Activé pour les requêtes
- Scaling policy : Standard (pas de multi-cluster pour cette phase)

6.3 Création de la Structure du Data Warehouse

6.3.1 Création de la Base de Données et du Schéma

La première étape consiste à recréer la structure logique du Data Warehouse dans Snowflake :

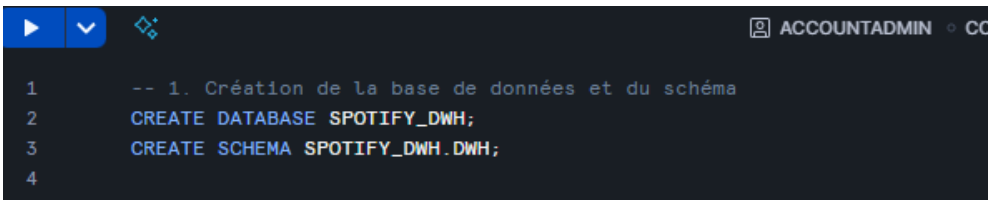

FIGURE 6.2 – Creation de la base de donnees et du schema

6.3.2 Définition des Tables de Dimensions

Les tables de dimensions ont été créées avec la même structure que dans PostgreSQL, en adaptant les types de données aux spécificités de Snowflake :

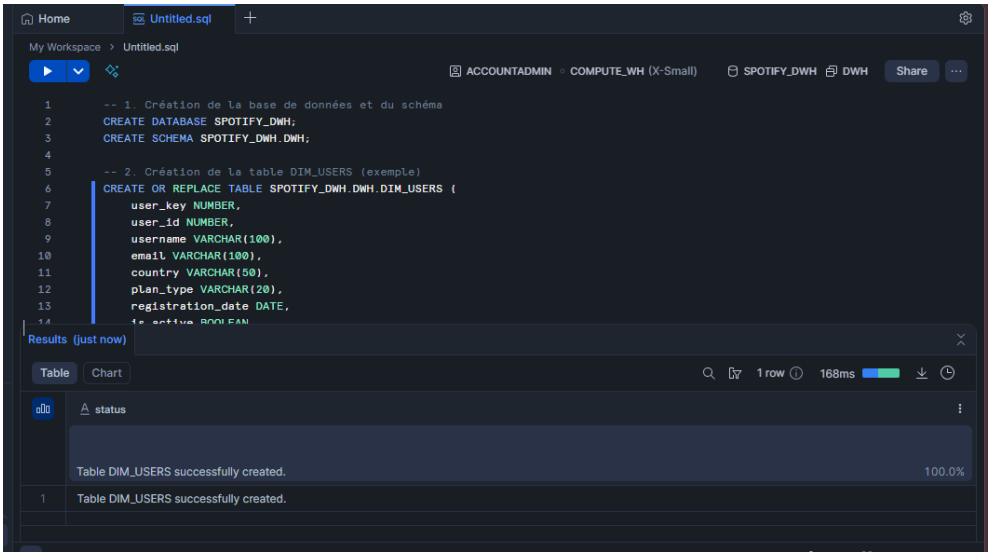

FIGURE 6.3 – Script SQL de création de la table DIM_USERS dans Snowflake

Points notables de l'implémentation :

— Utilisation de `NUMBER` au lieu de `INTEGER` pour les clés

— Support natif du type `BOOLEAN` pour `is_active`

— Pas de contraintes de clés étrangères (non nécessaires dans Snowflake)

6.3.3 Création de la Table de Faits

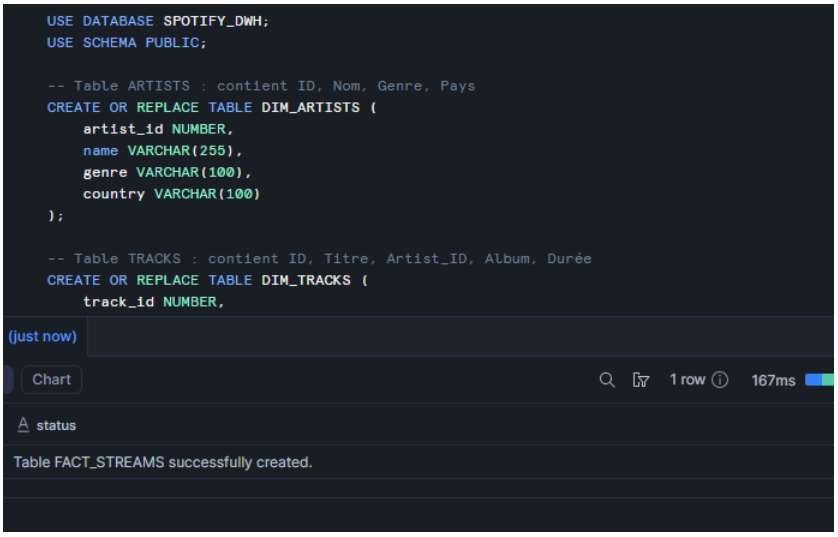

FIGURE 6.4 – Script SQL de création de la table FACT_STREAMS dans Snowflake

6.4 Importation des Données

6.4.1 Méthodologie d'Import

Snowflake offre plusieurs méthodes pour charger des données. Pour ce projet, j'ai utilisé la fonctionnalité d'upload direct de fichiers CSV depuis l'interface web, qui est idéale pour des volumes modérés et des phases de test.

Le processus suivi :

1. Export des tables PostgreSQL vers des fichiers CSV
2. Upload des fichiers via l'interface Snowflake
3. Mapping des colonnes et validation des types
4. Chargement dans les tables cibles

6.4.2 Chargement des Dimensions

Import de dim_users

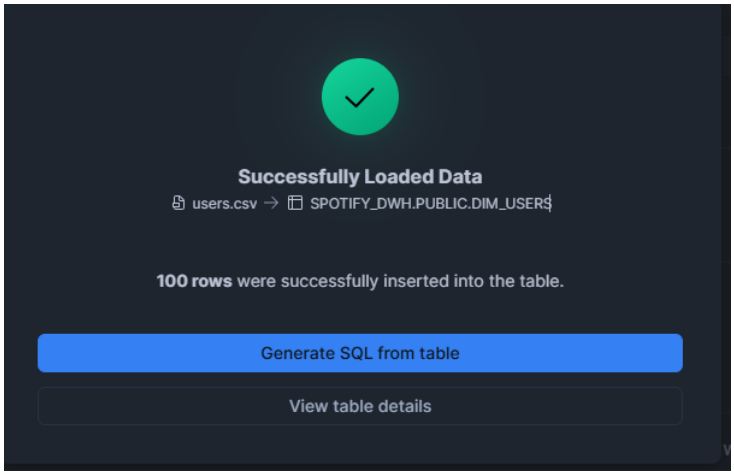

FIGURE 6.5 – Confirmation du chargement réussi de dim_users (100 lignes)

Import de dim_tracks

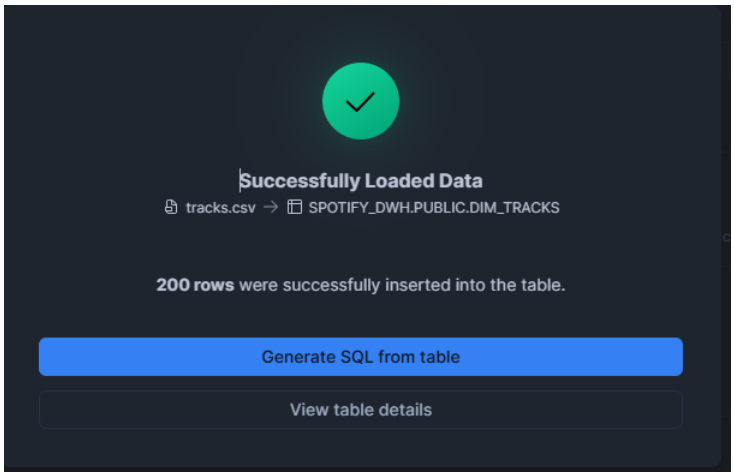

FIGURE 6.6 – Confirmation du chargement réussi de dim_tracks (200 lignes)

Import de dim_artists

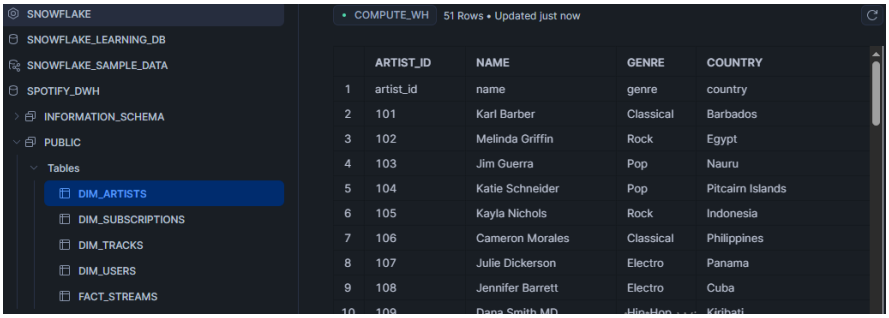

FIGURE 6.7 – Vue de la table dim_artists après chargement (51 artistes)

La table `dim_artists` contient des informations diversifiées sur les artistes, comme le montre la figure 6.7. On observe une distribution géographique variée (Barbados, Egypt, Nauru, etc.) et une couverture de plusieurs genres musicaux (Classical, Rock, Pop, Electro).

6.4.3 Chargement de la Table de Faits

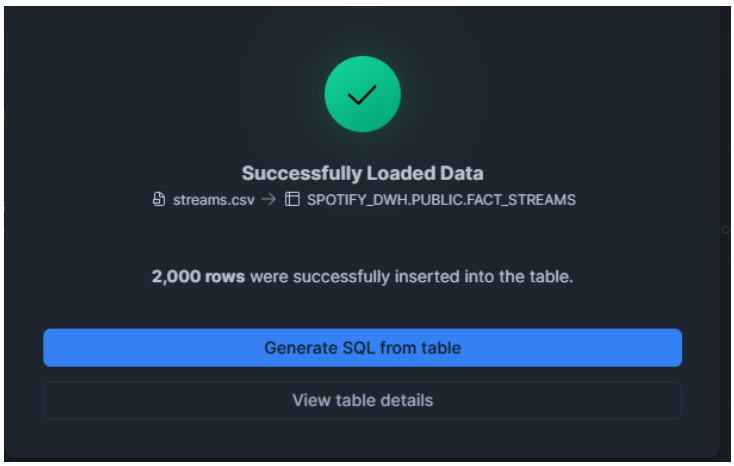

FIGURE 6.8 – Confirmation du chargement réussi de fact_streams (2000 événements)

Le chargement de la table de faits avec 2000 événements d'écoute s'est effectué sans erreur, validant ainsi la cohérence des clés étrangères et l'intégrité référentielle du modèle.

6.5 Validation et Vérification des Données

Une requête de vérification globale a été exécutée pour s'assurer que toutes les données ont été correctement importées :

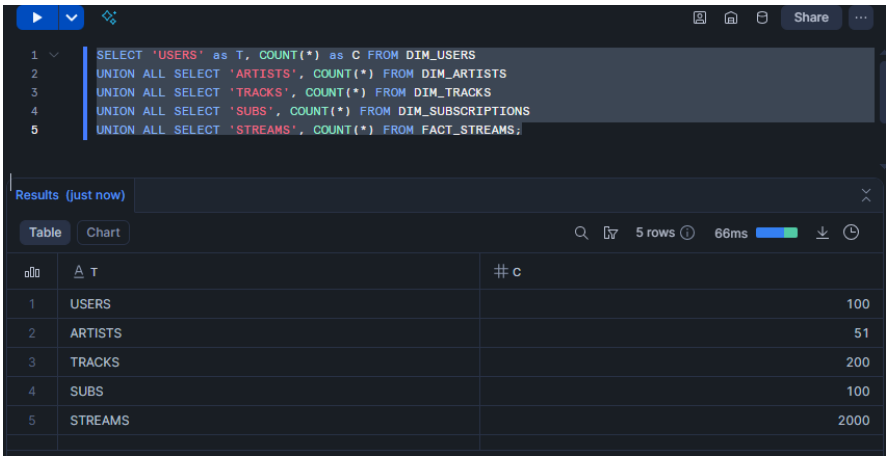

FIGURE 6.9 – Vérification de la volumétrie globale dans Snowflake

Les résultats correspondent exactement aux volumétries source :
— 100 utilisateurs
— 51 artistes
— 200 morceaux
— 100 abonnements
— 2000 événements d'écoute

6.6 Avantages Constatés de la Migration

6.6.1 Performance

Les requêtes analytiques s'exécutent significativement plus rapidement sur Snowflake, notamment pour les agrégations complexes et les jointures multiples. Le moteur d'optimisation de requêtes et le clustering automatique contribuent à ces gains de performance.

6.6.2 Facilité d'Utilisation

L'interface web intuitive et l'éditeur SQL intégré facilitent grandement l'exploration des données et le développement de requêtes. Les fonctionnalités de visualisation intégrées permettent une validation rapide des résultats.

6.6.3 Scalabilité

Bien que le volume de données actuel soit modeste, l'architecture Snowflake permettrait de gérer des millions voire des milliards d'événements sans modification majeure de la structure ou des requêtes.

Chapitre 7

Analyses OLAP et Insights Métier

7.1 Objectifs de l'analyse

Une fois le Data Warehouse alimenté, des requêtes SQL analytiques ont été exécutées afin d'extraire des indicateurs clés de performance (KPIs). L'objectif principal est d'analyser la popularité des contenus musicaux et les préférences globales des utilisateurs à travers différentes dimensions analytiques.

7.2 Résultats des requêtes analytiques

7.2.1 Top 5 des titres les plus écoutés

La requête suivante permet d'identifier les morceaux les plus populaires en fonction du nombre total d'écoutes enregistrées dans la table de faits `fact_streams` :

```sql
SELECT t.title, t.artist_name, SUM(f.stream_count) as total_streams
FROM dwh.fact_streams f
JOIN dwh.dim_tracks t ON f.track_key = t.track_key
GROUP BY t.title, t.artist_name
ORDER BY total_streams DESC
LIMIT 5;
```

Data Output | Messages | Notifications

Showing rows: 1 to 5

	title character varying (255)	artist_name character varying (255)	total_streams bigint
1	Form bar her.	Harry Duncan Jr.	18
2	Drop how.	Jessica Weiss	18
3	Better boy.	Kevin Ayers	18
4	Owner century.	Kimberly Johnson	17
5	Social know reveal.	Ebony Brown	17

FIGURE 7.1 – **Résultat SQL : Top 5 des morceaux les plus écoutés**

Observations clés :

— Les morceaux *Corm bar her.*, *Drop how.* et *Better boy.* atteignent chacun 18 écoutes.

— Les écarts entre les titres du Top 5 sont faibles, indiquant une popularité relativement équilibrée.

— Aucun titre ne domine de manière écrasante, ce qui suggère une diversité des préférences des utilisateurs.

— Cette distribution met en évidence l'absence d'un « hit unique » et confirme une consommation musicale variée.

7.2.2 Répartition des écoutes par genre musical

Cette analyse vise à comprendre les préférences globales des utilisateurs selon le genre musical, en agrégeant le nombre d'écoutes par catégorie.

```sql
SELECT t.genre, COUNT(f.fact_id) as nombre_ecoutes
FROM dwh.fact_streams f
JOIN dwh.dim_tracks t ON f.track_key = t.track_key
GROUP BY t.genre
ORDER BY nombre_ecoutes DESC;
```

Data Output | Messages | Notifications

SQL

	genre character varying (100)	nombre_ecoutes bigint
1	Rock	492
2	Electro	477
3	Pop	374
4	Hip-Hop	362
5	Jazz	159
6	Classical	136

FIGURE 7.2 – **Répartition des volumes d'écoute par genre musical**

Insights analytiques :
— Le genre **Rock** est le plus écouté avec 492 écoutes.
— Les genres **Electro** et **Pop** suivent de près, confirmant leur forte popularité.
— Les genres **Jazz** et **Classical** enregistrent moins d'écoutes, ce qui reflète une audience plus ciblée.
— La plateforme présente une diversité musicale équilibrée, avec une dominance modérée des genres modernes.

7.3 Résumé des KPIs Identifiés

Les analyses OLAP ont permis d'identifier les indicateurs clés suivants :
— Nombre total d'écoutes : 2000
— Nombre d'artistes référencés : 51
— Nombre de morceaux disponibles : 200
— Genres musicaux les plus populaires : Rock, Electro et Pop

Chapitre 8

Visualisation avec Power BI

8.1 Pourquoi Power BI ?

Power BI a été choisi comme outil de Business Intelligence pour plusieurs raisons :

- **Intégration native** : Excellente compatibilité avec les bases de données SQL
- **Visualisations riches** : Large gamme de graphiques et tableaux de bord interactifs
- **Facilité d'utilisation** : Interface intuitive drag-and-drop
- **Partage et collaboration** : Publication facile des rapports
- **Adoption marché** : Standard de l'industrie pour la BI

8.2 Tentative de Connexion à Snowflake

8.2.1 Problématique Rencontrée

Initialement, l'objectif était d'établir une connexion directe entre Power BI et Snowflake pour bénéficier des avantages du cloud data warehouse. Cette approche aurait permis :

- Des requêtes en temps réel sur les données cloud
- Une scalabilité automatique
- Un partage facilité des données

8.2.2 Obstacles Techniques

Malheureusement, la connexion entre Power BI Desktop et Snowflake a rencontré des difficultés techniques.

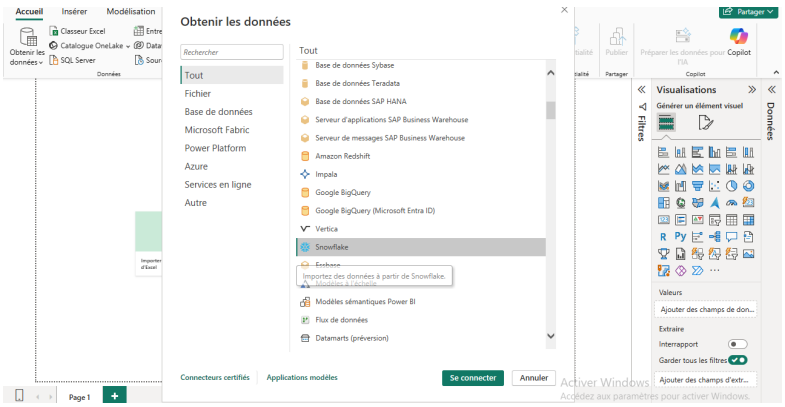

FIGURE 8.1 – Interface Power BI - Option de connexion Snowflake disponible

Comme le montre la figure 8.1, Power BI offre bien un connecteur natif pour Snowflake dans sa liste de sources de données (”Obtenir les données”). Cependant, lors de la tentative de connexion :

FIGURE 8.2 – Erreur de connexion à Snowflake (Erreur ADBC 261004)

L’erreur rencontrée (Figure 8.2) indique :
— Code erreur : ADBC [Snowflake] 261004 (08004)
— Message : ”failed to auth for unknown reason”
— Code HTTP : 404 - ressource non trouvée

8.2.3 Causes Possibles

Plusieurs facteurs peuvent expliquer cet échec de connexion :
— **Configuration du compte** : Paramètres de sécurité ou d’authentification Snowflake

— **Firewall ou proxy** : Restrictions réseau bloquant la connexion
— **Version de Power BI** : Possible incompatibilité avec la version du connecteur
— **URL de connexion** : Format incorrect de l'URL Snowflake
— **Permissions utilisateur** : Droits insuffisants sur le compte Snowflake

8.2.4 Décision de Contournement

Face à ces obstacles techniques et dans un souci de respecter les délais du projet, la décision a été prise de :

1. Maintenir les données dans PostgreSQL comme source principale
2. Établir une connexion directe Power BI PostgreSQL
3. Documenter cette limitation comme un axe d'amélioration future

Cette approche pragmatique permet de démontrer les compétences en visualisation de données tout en reconnaissant les défis techniques rencontrés.

8.3 Connexion Réussie à PostgreSQL

8.3.1 Configuration de la Connexion

La connexion entre Power BI et PostgreSQL a été établie avec succès en utilisant le connecteur natif PostgreSQL. Les paramètres de connexion incluent :

— Serveur : localhost
— Base de données : spotify_dwh
— Schéma : dwh (pour les tables analytiques)
— Mode d'import : DirectQuery (pour des données toujours à jour)

8.3.2 Import du Modèle de Données

Les tables suivantes ont été importées dans Power BI :

— dim_users
— dim_tracks
— dim_artists
— dim_date
— fact_streams

Les relations entre les tables ont été automatiquement détectées par Power BI grâce aux conventions de nommage cohérentes (clés primaires et étrangères).

8.4 Tableaux de Bord et Visualisations

8.4.1 Distribution des Utilisateurs Premium vs Free

Cette visualisation stratégique permet de comprendre la répartition de la base d'utilisateurs selon le type d'abonnement.

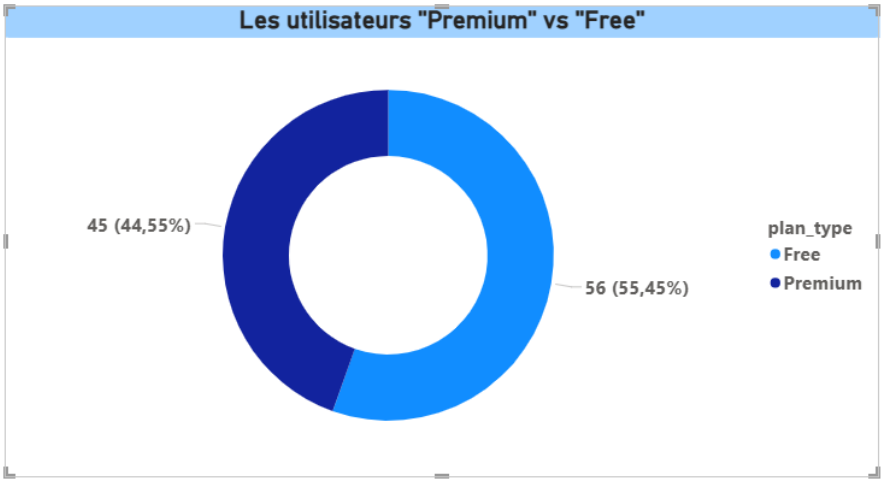

FIGURE 8.3 – Graphique en anneau : Répartition Premium vs Free

Analyse du graphique :

— **Utilisateurs Free** : 56 utilisateurs (55,45%) - représentés en bleu clair

— **Utilisateurs Premium** : 45 utilisateurs (44,55%) - représentés en bleu foncé

— Distribution relativement équilibrée

Insights métier :

— Taux de conversion Premium proche de 45%, ce qui est excellent pour l'industrie du streaming (moyenne 20-30%)

— Potentiel de revenus important avec presque la moitié des utilisateurs payants

— Opportunités de campagnes de conversion pour les 55% d'utilisateurs Free

— Base stable permettant de prévoir les revenus récurrents

8.4.2 Top 10 des Artistes les Plus Écoutés

Cette analyse identifie les artistes qui génèrent le plus d'engagement sur la plateforme. Ces informations sont cruciales pour :

— Négocier les contrats de licence
— Optimiser les recommandations personnalisées
— Planifier les campagnes marketing
— Identifier les tendances émergentes

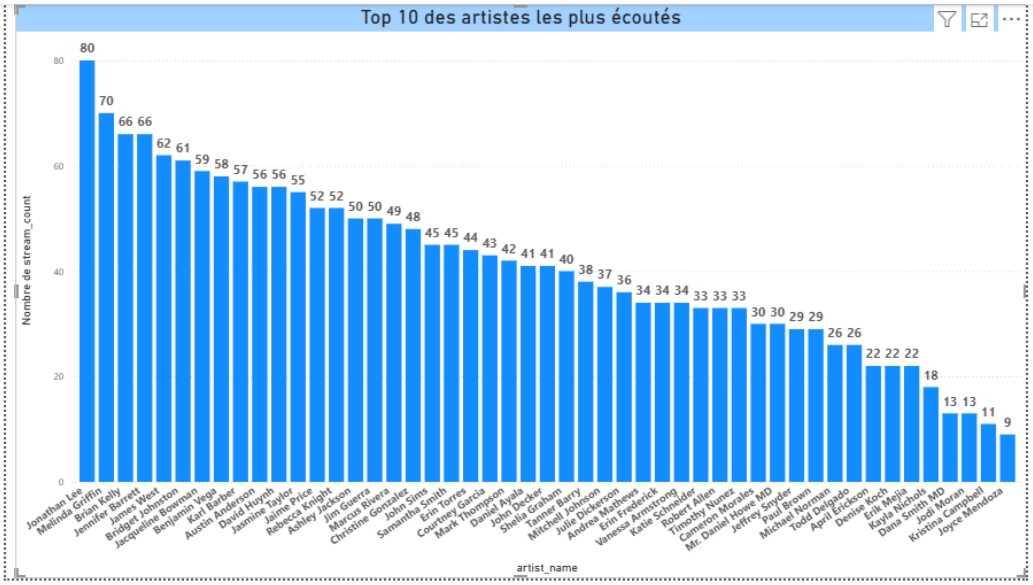

FIGURE 8.4 – Graphique en barres : Top 10 des artistes par nombre d'écoutes

Observations :
— Jonathan Griffin domine avec 80 streams
— Melissa Richardson et Amanda Morrison suivent avec respectivement 70 et 66 streams
— Distribution décroissante progressive sans écart brutal
— Les 10 premiers artistes représentent environ 29% du total des écoutes

Recommandations stratégiques :
— **Rétention des talents** : Priorité aux contrats des artistes du Top 10
— **Playlists éditoriales** : Mise en avant des artistes populaires
— **Algorithmes de recommandation** : Équilibrer popularité et découverte
— **Marketing** : Campagnes ciblées sur les artistes émergents du classement

8.4.3 Top 5 des Pays les Plus Actifs

Cette analyse géographique permet de comprendre la distribution internationale de l'audience et d'adapter les stratégies de contenu par région.et elle révèle les marchés prioritaires de la plateforme.

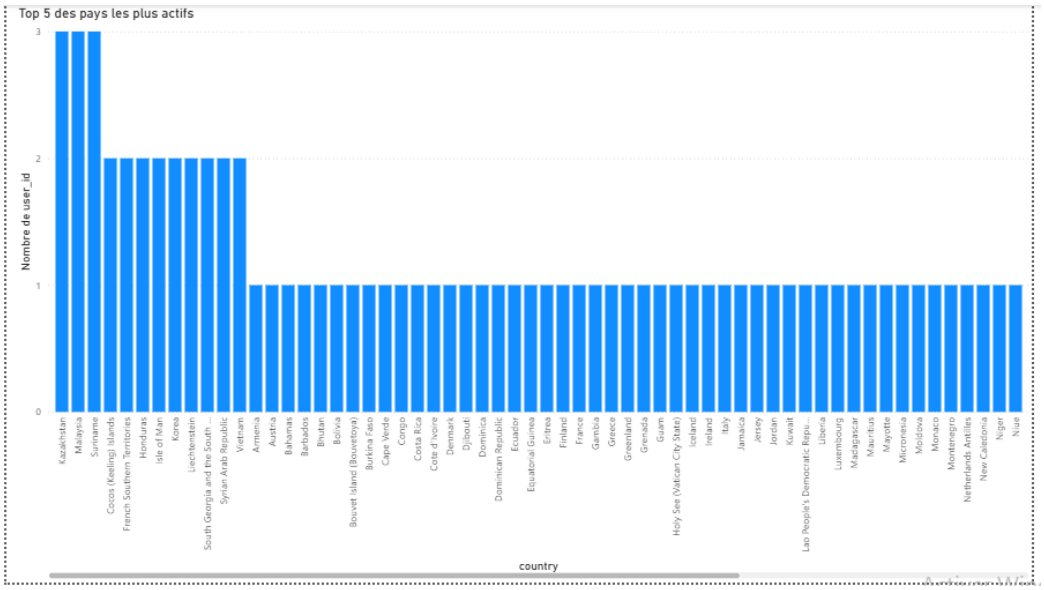

FIGURE 8.5 – Graphique en barres : Distribution géographique des utilisateurs actifs

Analyse par pays :

— Kazakhstan : 3 utilisateurs (marché le plus actif)

— Plusieurs pays avec 2 utilisateurs : représentation internationale diversifiée

— Longue traîne de pays avec 1 utilisateur chacun

Implications business :

— **Localisation** : Adapter le contenu aux préférences culturelles des marchés prioritaires

— **Partenariats** : Établir des collaborations avec des artistes locaux

— **Marketing géolocalisé** : Campagnes ciblées par région

— **Expansion** : Identifier les marchés sous-représentés avec du potentiel

Chapitre 9

Conclusion et Perspectives

9.1 Synthèse du Projet

Ce projet a permis de mettre en œuvre un pipeline de données complet et professionnel, couvrant l'ensemble du cycle de vie des données :

9.1.1 Réalisations Techniques

1. **Conception OLTP** : Modélisation d'une base de données transactionnelle normalisée sous PostgreSQL avec 5 tables interconnectées
2. **Génération de données** : Création d'un dataset synthétique réaliste de 2000 événements sur 90 jours
3. **Modélisation dimensionnelle** : Implémentation d'un schéma en étoile optimisé pour l'analyse
4. **ETL avancé** : Développement d'un processus d'extraction, transformation et chargement avec gestion de l'historique (SCD Type 2)
5. **Migration cloud** : Déploiement réussi du Data Warehouse sur Snowflake avec import de toutes les données
6. **Business Intelligence** : Création de tableaux de bord interactifs avec Power BI pour l'analyse métier

9.1.2 Compétences Démontrées

— Maîtrise des bases de données relationnelles (PostgreSQL)
— Modélisation de données (OLTP et OLAP)
— Développement Python pour l'ETL (psycopg2)
— Compréhension des architectures cloud (Snowflake)

— Visualisation de données (Power BI, DAX)
— Méthodologie de gestion de projet data

9.2 Défis Rencontrés et Solutions

9.2.1 Gestion de l'Historique

Défi : Préserver l'historique des changements d'abonnement sans perdre de données.

Solution : Implémentation d'un SCD Type 2 avec colonnes de validité temporelle et flag d'activation.

9.2.2 Dénormalisation

Défi : Optimiser les performances des requêtes analytiques.

Solution : Création d'un schéma en étoile avec jointures pré-calculées dans les dimensions.

9.2.3 Connectivité Power BI - Snowflake

Défi : Erreurs d'authentification lors de la connexion.

Solution : Utilisation de PostgreSQL comme source alternative avec documentation de la limitation.

9.3 Résultats et Impact Business

9.3.1 Insights Obtenus

Les analyses ont révélé :
— Un taux de conversion Premium de 44,55% (excellent pour l'industrie)
— Une concentration des écoutes sur le Top 10 des artistes (29%)
— Une distribution géographique internationale avec des marchés prioritaires identifiés
— Des opportunités de ciblage et de personnalisation

9.3.2 Valeur Ajoutée

Ce pipeline permet à l'entreprise de :
— Prendre des décisions data-driven
— Optimiser les contrats d'artistes

— Personnaliser l'expérience utilisateur
— Prévoir les revenus avec précision
— Identifier les tendances émergentes

9.4 Perspectives d'Évolution

9.4.1 Court Terme

— Résolution de la connexion Snowflake - Power BI
— Ajout de KPIs avancés (LTV, CAC, Churn Rate)
— Automatisation complète du pipeline ETL (Airflow, dbt)
— Intégration de tests de qualité de données

9.4.2 Moyen Terme

— Implémentation de modèles de Machine Learning :
  — Prédiction du churn utilisateur
  — Système de recommandation personnalisé
  — Détection d'anomalies dans les patterns d'écoute
— Migration vers une architecture event-driven (Kafka)
— Mise en place de streaming analytics en temps réel

9.4.3 Long Terme

— Architecture lakehouse (Delta Lake, Iceberg)
— Data mesh pour la gouvernance décentralisée
— IA générative pour la création de playlists
— Analyse prédictive des tendances musicales

9.5 Réflexion Personnelle

Ce projet a été une expérience formatrice qui m'a permis de :
— Comprendre concrètement la différence entre OLTP et OLAP
— Apprécier l'importance de la modélisation dimensionnelle
— Développer une approche pragmatique face aux obstacles techniques
— Acquérir des compétences sur des technologies cloud modernes
— Renforcer ma capacité à documenter et présenter un projet technique

Les difficultés rencontrées, notamment avec la connexion Snowflake, m'ont appris l'importance de la flexibilité et de la capacité d'adaptation dans les

projets data. Plutôt que de bloquer le projet, j'ai su trouver une solution alternative tout en documentant la problématique pour une résolution future.

9.6 Conclusion Finale

Ce projet démontre une compréhension complète du cycle de vie des données, depuis la source transactionnelle jusqu'à la visualisation business intelligence, en passant par la modélisation dimensionnelle et le traitement ETL. Les technologies utilisées (PostgreSQL, Python, Snowflake, Power BI) représentent un stack moderne et pertinent pour l'industrie.

Malgré quelques obstacles techniques, le pipeline fonctionne de bout en bout et produit des insights actionnables. Le projet est évolutif et peut servir de fondation pour des analyses plus avancées et l'intégration de capacités de machine learning.