**Dakar Institute of Technology**



**RepSEN/Ensup-priv/AP/376-2021 Domaine : Sciences et Technologies Département : Informatique Spécialité : Intelligence Artificielle**

**Projet Examen Big Data**

Présenté par :

Jean Raymond Mbad DIOUF

Mamadou Ciré Kader Baldé

Mbeni zimba

**Année Académique 2024 – 2025**

Table des matières

[1. Introduction 3](#_Toc195042572)

[2. Préparation des données sources 3](#_Toc195042573)

[2.1 Choix du SGBD local 3](#_Toc195042574)

[2.2 Création de la base locale 4](#_Toc195042575)

[3. Ingestion des données 4](#_Toc195042576)

[3.2 Ingestion dans Snowflake 4](#_Toc195042577)

[4. Traitement des données avec DBT 5](#_Toc195042578)

[4.1 Environnement DBT 5](#_Toc195042579)

[4.2 Transformation RAW -> STAGGING 5](#_Toc195042580)

[4.3 Transformation STAGGING -> WAREHOUSE 5](#_Toc195042581)

[4.4 Transformation WAREHOUSE -> MARTS 6](#_Toc195042582)

[5. Orchestration avec Airflow 6](#_Toc195042583)

[6. Visualisation des données 7](#_Toc195042584)

[6.1 Outil choisi : Power BI 7](#_Toc195042585)

[6.2 Dashboard réalisé 7](#_Toc195042586)

[7. Problèmes rencontrés et solutions apportées 7](#_Toc195042587)

[8. Conclusion 8](#_Toc195042588)

[Annexes 8](#_Toc195042589)

# 1. Introduction

Ce projet s'inscrit dans le cadre du cours d'Architecture Big Data en Master 2. L'objectif est de concevoir une architecture complète pour un système de gestion des ventes de livres en exploitant des outils modernes permettant de gérer efficacement de gros volumes de données.

L'approche adoptée consiste à simuler un système de ventes, ingérer les données dans un datawarehouse Snowflake, les transformer avec DBT, orchestrer les étapes avec Airflow et enfin visualiser les résultats à l'aide de Power BI. Ce projet vise à fournir une vision analytique complète permettant d’optimiser les décisions commerciales.

# 2. Préparation des données sources

# 2.1 Choix du SGBD local

Le choix de PostgreSQL s’est imposé en raison de sa stabilité, de sa compatibilité avec les outils modernes (Snowflake, DBT) et de son adoption généralisée dans les projets Big Data. Sa richesse fonctionnelle, notamment au niveau SQL, a été un atout important.

# 2.2 Création de la base locale

Le fichier `postgres\_diagram\_bigdata\_m2.sql` a été utilisé pour générer la base de données initiale, comprenant les tables : ventes, factures, category, books, customers. Nous avons enrichi ces tables avec des données réalistes simulées (plus de 1000 lignes) pour refléter un vrai système de vente. Les champs `date\_edit` ont été formatés strictement selon `YYYYMMDD`.

# 3. Ingestion des données

3.1 Mise en place du pipeline d’ingestion

Le pipeline d’ingestion a été développé en Python en utilisant le package `snowflake-connector-python`. Il se compose de plusieurs modules permettant l’extraction des données de PostgreSQL, leur conversion en DataFrame Pandas, puis leur chargement dans les tables RAW de Snowflake via des commandes SQL `PUT` et `COPY INTO`.

Ce pipeline est déclenché automatiquement via Airflow, où chaque table (ventes, factures, etc.) correspond à une tâche indépendante permettant une exécution parallélisée.

# 3.2 Ingestion dans Snowflake

La base de données `BOOKSHOP` est organisée en 4 schémas distincts :

- \*\*RAW\*\* : contient les données brutes issues de PostgreSQL, sans modification.

- \*\*STAGGING\*\* : permet une première standardisation des données.

- \*\*WAREHOUSE\*\* : structure en étoile avec des tables de faits et de dimensions nettoyées.

- \*\*MARTS\*\* : contient une table unique consolidée (OBT - One Big Table) pour les usages analytiques.

# 4. Traitement des données avec DBT

# 4.1 Environnement DBT

DBT a été installé via pip dans un environnement virtuel Python. Une configuration via le fichier `profiles.yml` a permis de connecter notre projet à Snowflake. Les modèles DBT ont été organisés en sous-dossiers logiques, et versionnés via Git.

# 4.2 Transformation RAW -> STAGGING

Les transformations suivantes ont été appliquées :

- `stg\_ventes` : conversion de `date\_edit` en format DATE, renommage des colonnes pour cohérence.

- `stg\_factures` : même traitement que pour ventes.

- `stg\_category`, `stg\_books`, `stg\_customers` : renommage standardisé, suppression des valeurs nulles.

# 4.3 Transformation STAGGING -> WAREHOUSE

- \*\*Dimensions\*\* :

- `dim\_customers` : ajout d’un champ `nom` (concaténation de `first\_name` et `last\_name`), filtrage des doublons.

- `dim\_books` : enrichissement avec des indicateurs de disponibilité.

- `dim\_category` : nettoyage des labels.

- \*\*Faits\*\* :

- `fact\_ventes`, `fact\_factures` : enrichissement par date (`annees`, `mois`, `jour`), conversion de types.

- `fact\_books\_\*` : calculs d’agrégations pour l’analyse temporelle des ventes.

# 4.4 Transformation WAREHOUSE -> MARTS

La table `obt\_sales` a été construite comme une jointure multi-table entre :

- `fact\_ventes`, `fact\_factures`

- `dim\_books`, `dim\_category`, `dim\_customers`

Elle contient toutes les informations nécessaires à l’analyse détaillée des ventes (prix, client, catégorie, date, etc.).

# 5. Orchestration avec Airflow

Airflow a été configuré localement via Docker, avec une architecture modulaire :

- Un DAG principal pour l’ingestion des données

- Trois DAGs secondaires pour les transformations DBT (RAW → STAGGING, STAGGING → WAREHOUSE, WAREHOUSE → MARTS)

Chaque DAG contient des tâches `BashOperator` exécutant les commandes DBT via des scripts shell. Des dépendances strictes assurent l’ordre d’exécution correct. Des notifications par e-mail sont envoyées en cas d’échec de tâche.

# 6. Visualisation des données

# 6.1 Outil choisi : Power BI

Dans le cadre de la visualisation des ventes de livres, notre choix s’est porté sur Power BI en raison de sa puissance, de sa simplicité d’utilisation et de ses capacités d’analyse avancées. Cet outil permet de créer des tableaux de bord interactifs, de se connecter à diverses sources de données (notamment Snowflake), et de modéliser efficacement les données grâce à Power Query et DAX. Son intégration avec l’écosystème Microsoft et ses fonctionnalités de partage en font une solution adaptée aux besoins d’un projet Big Data

# 6.2 Dashboard réalisé

Le dashboard connecté à Snowflake propose :

- Une \*\*vue mensuelle\*\* des ventes avec possibilité de filtrer par client ou catégorie

- Une \*\*carte de chaleur\*\* des jours les plus vendeurs

- Des \*\*graphes top 10 livres\*\* et \*\*top clients\*\*

- Un \*\*tableau interactif\*\* avec téléchargement CSV possible

# 7. Problèmes rencontrés et solutions apportées

- \*\*Connexion Snowflake\*\* : problèmes de rôle résolus avec des `GRANT` explicites.

- \*\*Problèmes d’encodage\*\* : traités par une standardisation UTF-8 dès l’ingestion.

- \*\*Tests DBT échoués\*\* : ajout de `schema.yml` avec des tests de `unique`, `not\_null` pour validation.

- \*\*Montée en charge Airflow\*\* : optimisée via parallélisation et exécution planifiée par intervalles.

# 8. Conclusion

Ce projet nous a permis d’acquérir une vision end-to-end d’un système Big Data. Nous avons mis en pratique les étapes critiques d’un pipeline moderne de traitement de données : ingestion, transformation, orchestration et visualisation. Ce travail nous a permis de comprendre les enjeux du reporting, du stockage cloud, et de l’automatisation des processus de données.

# Annexes

- Scripts Python (extraction, chargement Snowflake)

- Modèles DBT (transformations SQL et configuration)

- DAGs Airflow (.py)

- Captures d’écrans : Airflow UI, Snowflake, DBT CLI, Dashboard Power BI

- Vidéo démo :

- Repository GitHub :