

Rapport Complet – TMDb Data Pipeline (Challenge)

Date début du challenge : 11/08/2025

Partie 1

Note exceptionnelle — Pour faciliter la correction de ce challenge, je laisse exceptionnellement mon jeton API TMDb (Bearer v4) accessible dans le projet. En contexte réel, cette clé serait gérée via variables d'environnement/secret manager et ne serait jamais commitée.

1) Résumé exécutif

Ce projet met en place un pipeline de données conteneurisé qui récupère des films depuis l'API TMDb, les transforme, puis les charge dans PostgreSQL selon un schéma en étoile. Les données sont exposées via des vues analytiques (dernier état, par genre, par continent, top) et consultables dans une application Streamlit. Le tout se lance en une commande Docker Compose.

Points clés :

- Sources : API TMDb + CSV pays→continent.
- ETL Python (requests, tenacity, SQLAlchemy) : extraction, nettoyage, UPSERT dims/ponts, fait quotidien.
- Modèle : fact_movie_daily (film × jour) + dimensions (movie, genre, company, country, language, date) + ponts M:N.
- Exposition : vues SQL optimisées + viz Streamlit (KPI, genres, top, continents).
- Exécution : Docker Compose (Postgres, pgAdmin, ETL, Viz) — démarrage à froid en 1 commande.
- Qualité : tests unitaires (pytest) et linting (ruff).

2) Contexte du projet

Le commanditaire (analyste/scientist/équipe métier) souhaite un dataset quotidien sur les films (popularité, votes) pour suivre les tendances, comparer les genres/pays et préparer de futurs cas ML (reco, forecasting). Le pipeline doit être simple à déployer et à maintenir.

- Public cible : analystes, data scientists, PM marketing.
- Fréquence : chargement quotidien avec « dernier état » par film.
- Contraintes : démarrage rapide, idempotence, documentation claire, pas de dépendances SaaS imposées.

- Données : API TMDb (discover + details) ; enrichissement CSV pays→continent.

3) Choix d'implémentation technique et d'architecture

Raisons de ces choix et compromis faits pour tenir un bon ratio simplicité/robustesse.

Domaine	Choix	Justification
Langage	Python 3.11	Écosystème data mature, écriture rapide, bibliothèques robustes.
HTTP	requests + tenacity	Client simple + retries/backoff pour résilience API.
DB	PostgreSQL 16	Sûr, performant, vues/indices avancés, large adoption.
ORM/driver	SQLAlchemy + psycopg2	Transactions, exécutions par lots, UPSERT natif.
Orchestration	Docker Compose	Reproductibilité locale, réseau isolé, runbook simple.
Admin	pgAdmin	UI web pour requêtes/diagnostics.
Viz	Streamlit	Dataviz rapide pour validation métier.
Modèle	Schéma en étoile	Requêtes analytiques simples et performantes, extensible.
Qualité	pytest + ruff	Prévenir régressions & garder un style homogène.

4) Démarche (processus de réalisation)

- Initialisation : dépôt, `.gitignore`, `.env.example` et Docker Compose minimal (Postgres + pgAdmin).
- Schéma DB : scripts d'init (`01_schema.sql`, `02_views.sql`, `03_indexes.sql`), vérification via pgAdmin.
- ETL API : `tmdb_client.py` (discover/details) + `cleaning.py` (mapping) + `db.py` (UPSERT).
- Orchestration : `load_movies.py` (extraction → transformation → chargement), logs de synthèse.
- Qualité : `pytest` (tests cleaning), `ruff` (lint/format).
- Analytics : vues SQL (`vw_movie_latest`, genres, top).
- Dataviz : app Streamlit (KPI, genres, top).
- Multi-source : CSV pays→continent + `load_country_continent.py` + vue continent + viz.

- Démarrage à froid : ``docker compose --profile etl up -d postgres pgadmin viz etl`` (ETL + loader).

5) Modèle de données & vues (synthèse)

- Fait : ``fact_movie_daily(date_id, movie_id, popularity, vote_average, vote_count)`` – 1 ligne par (film, jour).
- Dims : ``dim_movie``, ``dim_genre``, ``dim_company``, ``dim_country(continent)``, ``dim_language``, ``dim_date``.
- Ponts : ``bridge_movie_*`` pour M:N (genre, company, country, language).
- Vues : ``vw_movie_latest``, ``vw_genre_popularity``, ``vw_top_movies``, ``vw_continent_popularity``.
- Index : clés des ponts, ``fact(movie_id)``, ``fact(date_id)``, ``(movie_id, date_id DESC)``, ``dim_country(continent)``.

6) Scripts Python – rôles

- ``config.py`` : variables d'environnement et DSN (SETTINGS).
- ``db.py`` : ``get_engine()``, ``upsert()`` générique (INSERT ... ON CONFLICT).
- ``tmdb_client.py`` : ``discover_movies_pages()``, ``fetch_movie_details()`` + timeouts/retries.
- ``cleaning.py`` : fonctions pures de mapping (dims, ponts, fait).
- ``load_movies.py`` : orchestrateur ETL principal (log de synthèse, lots).
- ``load_country_continent.py`` : enrichissement ``dim_country.continent`` à partir du CSV.
- ``viz/app.py`` : tableaux & bar charts (KPI, genres, top, continent).
- ``tests/*`` : tests unitaires (cleaning) ; linting via ruff.

7) Difficultés rencontrées & résolutions

- ``COPY src ./src`` (build ETL) : chemin absent au début → créer l'arborescence ``etl/src`` puis rebuild.
- Imports dans le conteneur : ``ModuleNotFoundError`` → s'assurer que ``/app`` est dans ``PYTHONPATH`` (compose setup).
- UPSERT ponts : ``ON CONFLICT DO UPDATE SET`` vide → utiliser ``DO NOTHING`` (clé composite).
- pgAdmin « serveur » non persisté : normal sans volume dédié → reconfigurer l'entrée si recréé.
- Docker networking 'not found' : ``down --remove-orphans``, ``network prune``, (sous WSL : ``wsl --shutdown``).

8) Possibilités d'amélioration

- Planification quotidienne : service boucle 24h ou scheduler dockerisé (Ofelia/Supercronic), ou CronJob K8s.

- Qualité de données avancée : Great Expectations (completude, bornes, schémas), rapports HTML.
- Performance : vue matérialisée pour `vw_movie_latest`, partitionnement du fait par mois si volumétrie.
- Sécurité : compte read-only pour la viz, rotation de secrets, variables chiffrées en CI.
- Couverture fonctionnelle : endpoint TMDb `credits` → `dim_person` + pont cast/crew + viz associée.
- CI/CD : GitHub Actions (ruff + pytest + tests d'intégration DB), build images, scan vulnérabilités.

9) Exploitation (runbook)

Démarrage à froid (tout-en-un) :

```
docker compose --profile etl up -d postgres pgadmin viz etl
```

Vérifications :

- `docker compose logs etl --tail 200`
- `SELECT COUNT(*) FROM dim_movie;`
- `SELECT * FROM vw_continent_popularity;`

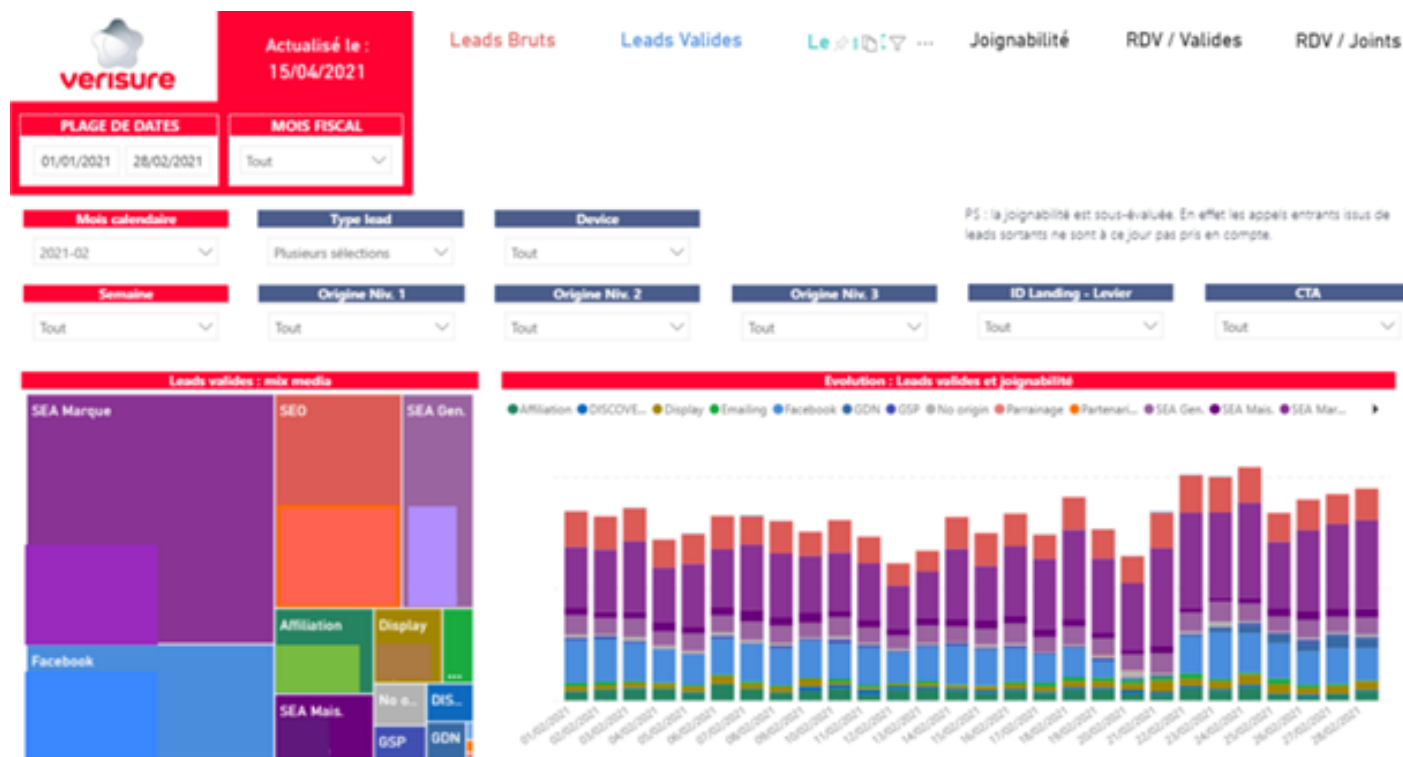
Arrêt/Reset :

- `docker compose down --remove-orphans` (conserve les données)
- `docker compose down -v --remove-orphans` (reset complet)

10) Conclusion

La solution livrée couvre les attentes du challenge : ingestion robuste (TMDb), schéma analytique clair, vues prêtes à l'emploi, dataviz simple et exécution reproductible. Les pistes proposées permettent une montée en maturité (planification, QA, perf, CI/CD, nouvelles entités).

Partie 2



1. Objectif du rapport

Ce tableau de bord Power BI présente la répartition et l'évolution des **leads validés** issus des différents canaux marketing, permettant de suivre la performance des actions digitales et d'orienter les investissements publicitaires.

2. Répartition des leads par canal (Mix médias)

Le treemap met en évidence les principaux contributeurs au volume de leads validés :

- **SEA Marque** : canal dominant, représentant la plus grande part des leads.
- **SEO et SEA Générique** : autres sources majeures.

- **Affiliation** et **Display** : volumes significatifs.
 - **Facebook** : présence notable, mais moins importante que le SEA.
 - Autres canaux : GSP, GDN, Display secondaire.
-

3. Évolution temporelle

Le graphique en barres empilées montre :

- **Variation régulière** du volume de leads sur la période, avec quelques pics hebdomadaires.
 - La structure des canaux reste relativement stable, avec **SEA Marque** et **SEO** occupant constamment une part importante.
 - Pas de baisse marquée, signe d'une stabilité dans la génération de leads.
-

4. Observations spécifiques

- La **joignabilité** indiquée est sous-évaluée (les appels externes sur leads sortants ne sont pas inclus).
 - Les canaux payants (SEA, Display) restent dominants, mais le SEO conserve une bonne part, signe d'une performance organique solide.
-

5. Conclusion

La performance globale des canaux marketing est stable avec une dominance claire du **SEA Marque**. Les données permettent d'orienter les budgets vers les leviers les plus performants tout en surveillant la contribution des canaux organiques et sociaux.