Rapport Professionnel E5

Mise en place d'un Entrepôt de Données

Projet Data Engineering
Analyse Territoriale - Région Hauts-de-France

Compétences évaluées :

C13 Modéliser la structure des données d'un entrepôt
C14 Créer un entrepôt de données
C15 Intégrer les ETL en entrée et sortie

Auteur : [Votre Nom]
Formation : Data Engineer
Date : Janvier 2026

Table des matières

1 Introduction

1.1 Contexte du projet

Ce rapport présente la mise en place d'un entrepôt de données (Data Warehouse) dans le cadre d'un projet d'analyse territoriale de la région **Hauts-de-France**. Le projet s'inscrit dans une démarche d'aide à la décision stratégique basée sur l'exploitation de données publiques issues de l'INSEE.

L'infrastructure technique repose sur la plateforme **Microsoft Azure** avec une architecture Data Lake moderne (zones raw, staging, curated) alimentant un entrepôt de données optimisé pour les requêtes analytiques.

1.2 Objectifs du projet

Les objectifs principaux de ce projet sont :

— Centraliser les données territoriales dans un entrepôt de données structuré

— Modéliser les données selon les bonnes pratiques (schéma en étoile)

— Mettre en place des ETL robustes pour alimenter l'entrepôt

— Permettre aux équipes analytiques d'accéder facilement aux données

— Garantir la qualité et la cohérence des données

1.3 Périmètre géographique

Le périmètre couvre la région **Hauts-de-France** composée de 5 départements :

Code	Département	Préfecture
02	Aisne	Laon
59	Nord	Lille
60	Oise	Beauvais
62	Pas-de-Calais	Arras
80	Somme	Amiens

TABLE 1 – Départements de la région Hauts-de-France

2 Inventaire des Données pour les Analyses

2.1 Vue d'ensemble des sources

Les données proviennent principalement de l'**INSEE** (Institut National de la Statistique et des Études Économiques) et de l'**API Géo** du gouvernement français. Le Data Lake contient **11 fichiers CSV** totalisant **15 394 enregistrements** et **1 fichier JSON** avec **3 782 communes**.

Type de données	Nombre de fichiers	Total lignes
Données démographiques	4	2 002
Données économiques	3	12 664
Données sociales	4	729
Données géographiques	1	3 782 communes
TOTAL	**12**	**19 177**

Table 2 – Synthèse des volumes de données

2.2 Sources de données identifiées

2.2.1 Données démographiques

Fichier source	Description	Lignes
population_hauts_de_france.csv	Population par PCS (Professions et Catégories Socioprofessionnelles), sexe et tranche d'âge. Années : 2010, 2015, 2021	1 579
naissances_hauts_de_france.csv	Naissances vivantes annuelles par département (code EC_MEASURE=LVB). Période : 2014-2023	51
DECES_hauts_de_france.csv	Décès annuels par département (code EC_MEASURE=DTH). Période : 2014-2023	51
FECONDITE_hauts_de_france.csv	Indicateurs de fécondité et structure familiale (nombre d'enfants, type de famille). Années : 2010, 2015, 2021	321

Table 3: Sources de données démographiques - Volume exact

2.2.2 Données économiques

Fichier source	Description	Lignes
CREATION_ENT_hauts_de_france.csv	Créations d'entreprises par secteur d'activité (NAF) et forme juridique. Mesures : BURE (entreprises), UNIT_LOC_BURE (établissements). Période : 2012-2024	10 757
CREA_EI_hauts_de_france.csv	Créations d'entrepreneurs individuels par sexe, âge et forme juridique (MICRO, ENTIND). Période : 2012-2024	1 561

Fichier source	Description	Lignes
EMPLOI_CHOMAGE_hauts_de_france.csv	Population active par statut d'emploi (EMPSTA_ENQ : 1=emploi, 2=chômage) et PCS. Tranche d'âge : 15-64 ans	346

TABLE 4: Sources de données économiques - Volume exact

2.2.3 Données sociales

Fichier source	Description	Lignes
DS_FILOSOFI_hauts_de_france.csv	Indicateurs FILOSOFI : revenus (D1, D9, médiane), taux de pauvreté (PR_MD60), structure des revenus. Année : 2021	101
FILOSOFI_AGE_TP_NIVVIE.csv	FILOSOFI ventilé par tranche d'âge du référent fiscal. Indicateurs : revenu médian, taux de pauvreté	71
Logement_hauts_de_france.csv	Résidences principales par statut de surpeuplement (OVEROCC : 0=normal, 1=surpeuplé). Années : 2010, 2015, 2021	46
Menage_hauts_de_france.csv	Structure des ménages par type (TPH), PCS du référent et taille. Mesures : DWELLINGS, DWELLINGS_POPSIZE	511

TABLE 5: Sources de données sociales - Volume exact

2.2.4 Données géographiques

Fichier source	Description	Lignes
communes.json	Référentiel des communes avec : nom, code INSEE, codes postaux, population, surface (ha), coordonnées (longitude, latitude), contours GeoJSON. Source : API geo.api.gouv.fr	3 782

TABLE 6: Sources de données géographiques

2.3 Dictionnaire des données

Cette section détaille la structure de chaque source de données.

2.3.1 Structure des fichiers CSV

1. population_hauts_de_france.csv

Colonne	Type	Description
GEO	VARCHAR	Identifiant géographique (format : AAAA-DEP-XX)
PCS	VARCHAR	Code PCS (1-9, _T=total)
SEX	VARCHAR	Sexe (M, F, _T=total)
TIME_PERIOD	INT	Année de référence
RP_MEASURE	VARCHAR	Type de mesure (POP=population)
AGE	VARCHAR	Tranche d'âge (Y15T24, Y25T54, Y_GE55, Y_GE15)
OBS_VALUE	FLOAT	Valeur observée (population)
DEPARTEMENT	VARCHAR	Code département (02, 59, 60, 62, 80)

Table 7 – Structure de population_hauts_de_france.csv

2. naissances_hauts_de_france.csv / DECES_hauts_de_france.csv

Colonne	Type	Description
GEO	VARCHAR	Identifiant géographique
EC_MEASURE	VARCHAR	Type d'événement (LVB=naissances, DTH=décès)
FREQ	VARCHAR	Fréquence (A=annuelle)
TIME_PERIOD	INT	Année
OBS_VALUE	FLOAT	Nombre d'événements
DEPARTEMENT	VARCHAR	Code département

Table 8 – Structure des fichiers naissances/décès

3. CREATION_ENT_hauts_de_france.csv

Colonne	Type	Description
GEO	VARCHAR	Identifiant géographique
FREQ	VARCHAR	Fréquence (A=annuelle)
SIDE_MEASURE	VARCHAR	Type de mesure (BURE, UNIT_LOC_BURE)
TIME_PERIOD	INT	Année
ACTIVITY	VARCHAR	Code NAF section (A-U)
LEGAL_FORM	VARCHAR	Forme juridique (10, 54, 57, _T, OTH_SIDE)
OBS_VALUE	FLOAT	Nombre de créations
DEPARTEMENT	VARCHAR	Code département

Table 9 – Structure de CREATION_ENT_hauts_de_france.csv

4. DS_FILOSOFI_hauts_de_france.csv

Colonne	Type	Description
GEO	VARCHAR	Identifiant géographique
TIME_PERIOD	INT	Année (2021)
UNIT_MEASURE	VARCHAR	Unité (EUR_YR, PT, _Z, PS)
FILOSOFI_MEASURE	VARCHAR	Indicateur (MED_SL, D1_SL, D9_SL, PR_MD60...)
OBS_VALUE	FLOAT	Valeur de l'indicateur
DEPARTEMENT	VARCHAR	Code département

Table 10 – Structure de DS_FILOSOFI_hauts_de_france.csv

2.3.2 Structure du fichier JSON communes

Attribut	Type	Description
code	VARCHAR	Code INSEE de la commune (5 caractères)
nom	VARCHAR	Nom de la commune
codesPostaux	ARRAY	Liste des codes postaux
codeDepartement	VARCHAR	Code département (2 caractères)
departement_nom	VARCHAR	Nom du département
codeRegion	VARCHAR	Code région (32 = Hauts-de-France)
region_nom	VARCHAR	Nom de la région
population	INT	Population municipale
surface	FLOAT	Surface en hectares
longitude	FLOAT	Longitude du centre
latitude	FLOAT	Latitude du centre
contour_geojson	JSON	Contour de la commune (GeoJSON Polygon)

Table 11 – Structure de communes.json

2.4 Codes et nomenclatures

2.4.1 Codes PCS (Professions et Catégories Socioprofessionnelles)

Code	Libellé
1	Agriculteurs exploitants
2	Artisans, commerçants, chefs d'entreprise
3	Cadres et professions intellectuelles supérieures
4	Professions intermédiaires
5	Employés
6	Ouvriers
7	Retraités
9	Autres personnes sans activité professionnelle
_T	Total toutes catégories

Table 12 – Nomenclature des codes PCS

2.4.2 Codes NAF (sections d'activité)

Code	Secteur	Code	Secteur
A	Agriculture	J	Information et communication
B	Industries extractives	K	Activités financières
C	Industrie manufacturière	L	Activités immobilières
D	Énergie	M	Activités scientifiques
E	Eau et déchets	N	Services administratifs
F	Construction	P	Enseignement
G	Commerce	Q	Santé et action sociale
H	Transport	R	Arts et spectacles
I	Hébergement et restauration	S	Autres services

TABLE 13 – Nomenclature des codes NAF (sections)

2.4.3 Indicateurs FILOSOFI

Code	Description
MED_SL	Niveau de vie médian (euros/an)
D1_SL	Premier décile de niveau de vie
D9_SL	Neuvième décile de niveau de vie
IR_D9_D1_SL	Rapport interdécile (D9/D1)
PR_MD60	Taux de pauvreté (seuil 60% médiane)
S_EI_DI	Part des revenus d'activité
S_RET_PEN_DI	Part des pensions et retraites
S_SOC_BEN_DI	Part des prestations sociales

TABLE 14 – Indicateurs FILOSOFI principaux

2.5 Analyses envisagées

Les données collectées permettent de répondre aux besoins d'analyse suivants, organisés par domaine métier :

2.5.1 Analyses démographiques

Analyse	Sources utilisées	Questions métier
Évolution de la population	population, communes	Quels territoires gagnent/perdent des habitants ?
Pyramide des âges	population	Quel est le vieillissement par département ?
Dynamique naturelle	naissances, décès	Quel est le solde naturel par territoire ?
Structure familiale	fécondité	Quelle est la taille moyenne des familles ?

TABLE 15 – Analyses démographiques envisagées

2.5.2 Analyses économiques

Analyse	Sources utilisées	Questions métier
Dynamisme entrepreneurial	création_ent, crea_ei	Quels secteurs sont les plus dynamiques ?
Profil des créateurs	crea_ei	Quel est l'âge moyen des créateurs ?
Emploi par catégorie	emploi_chomage	Quelle est la répartition par PCS ?
Évolution du chômage	emploi_chomage	Comment évolue le taux de chômage ?

TABLE 16 – Analyses économiques envisagées

2.5.3 Analyses sociales

Analyse	Sources utilisées	Questions métier
Niveau de vie	filosofi, filosofi_age	Quels sont les écarts de revenus ?
Précarité	filosofi	Quel est le taux de pauvreté ?
Conditions de logement	logement	Quel est le taux de surpeuplement ?
Structure des ménages	ménage	Quelle est la taille des ménages ?

TABLE 17 – Analyses sociales envisagées

2.5.4 Analyses territoriales

2.6 Indicateurs clés (KPIs)

Les indicateurs suivants seront calculés dans l'entrepôt de données pour répondre aux besoins d'analyse :

Analyse	Sources utilisées	Questions métier
Comparaison départements	toutes sources	Quels départements sont les plus dynamiques ?
Cartographie	communes + indicateurs	Comment visualiser les disparités ?
Évolution temporelle	sources avec TIME_PERIOD	Quelles tendances sur 10 ans ?
Densité territoriale	communes, population	Quelle répartition de la population ?

Table 18 – Analyses territoriales envisagées

2.6.1 KPIs Démographiques

KPI	Formule	Unité	Source
Taux de natalité	Naissances / Population × 1000	‰	naissances, population
Taux de mortalité	Décès / Population × 1000	‰	décès, population
Solde naturel	Naissances - Décès	Nombre	naissances, décès
Indice de vieillissement	Pop 65+ / Pop 0-19 × 100	%	population
Taux de fécondité	Naissances / Femmes 15-49	Ratio	naissances, population

Table 19 – Indicateurs démographiques

2.6.2 KPIs Économiques

KPI	Formule	Unité	Source
Taux de création	Créations année N / Stock N-1	%	création_ent
Part micro-entreprises	Micro / Total créations × 100	%	crea_ei
Taux d'activité	Actifs / Pop 15-64 × 100	%	emploi_chomage
Taux de chômage	Chômeurs / Actifs × 100	%	emploi_chomage
Diversification économique	Indice Herfindahl par secteur	Indice	création_ent

Table 20 – Indicateurs économiques

KPI	Formule	Unité	Source
Revenu médian	MED_SL (direct)	EUR/an	filosofi
Taux de pauvreté	PR_MD60 (direct)	%	filosofi
Rapport interdécile	D9_SL / D1_SL	Ratio	filosofi
Taux de surpeuplement	Logements surpeuplés / Total	%	logement
Taille moyenne ménages	Population / Nb ménages	Personnes	ménage

TABLE 21 – Indicateurs sociaux

Source	Période	Fréquence
Population	2010, 2015, 2021	Recensement (5 ans)
Naissances / Décès	2014-2023	Annuelle
Créations entreprises	2012-2024	Annuelle
Emploi / Chômage	2010, 2015, 2021	Recensement (5 ans)
FILOSOFI	2021	Ponctuelle
Logement / Ménages	2010, 2015, 2021	Recensement (5 ans)
Communes	2023	Référentiel actualisé

TABLE 22 – Couverture temporelle des données

2.6.3 KPIs Sociaux

2.7 Périodes temporelles couvertes

3 Modélisation de l'Entrepôt de Données

3.1 Approche de modélisation

3.1.1 Justification de l'approche Bottom-Up

Pour ce projet, nous avons adopté une approche **bottom-up** (ascendante, style Kimball) pour les raisons suivantes :

Critère	Notre contexte	Justification Bottom-Up
Volume de données	∼19 000 enregistrements	Volume modéré, itération rapide possible
Sources disponibles	12 fichiers structurés	Données existantes à valoriser
Besoins analytiques	Analyses territoriales ciblées	Datamarts thématiques prioritaires
Délai de livraison	Projet à court terme	Time-to-value rapide
Évolutivité	Ajout futur de sources	Architecture extensible

TABLE 23 – Justification de l'approche Bottom-Up

Cette approche contraste avec l'approche **top-down** (Inmon) qui aurait nécessité une modélisation globale préalable de l'entreprise, inadaptée à notre contexte de projet ciblé.

3.2 Modélisation logique

3.2.1 Schéma en étoile (Star Schema)

Le modèle adopté est un **schéma en étoile** (Star Schema) avec :

— **6 tables de faits** : mesures quantitatives par domaine métier

— **6 tables de dimensions** : axes d'analyse partagés

— **4 schémas SQL** : organisation logique (stg, dwh, dm, analytics)

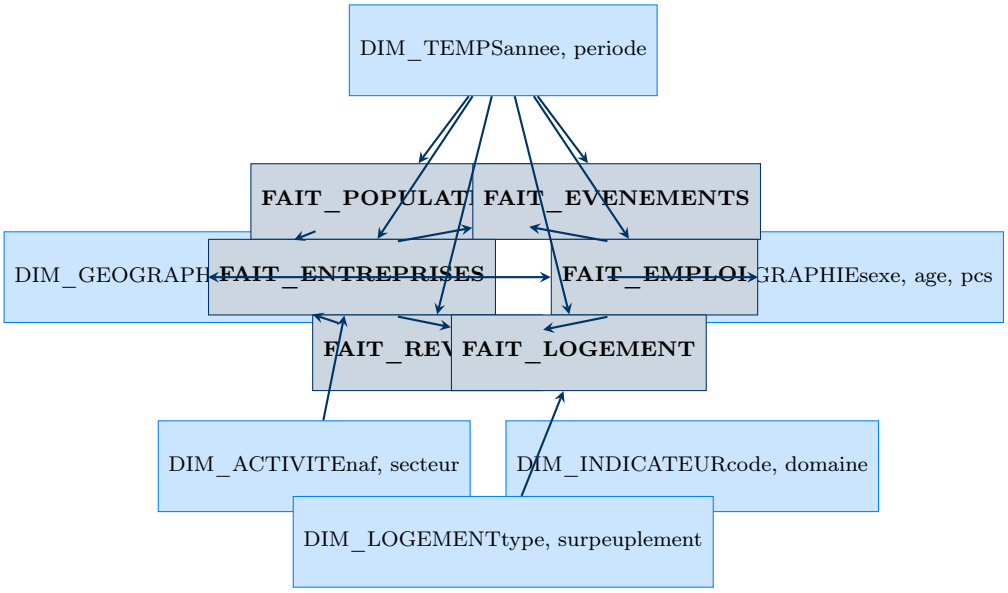

FIGURE 1 – Schéma en étoile - Architecture complète du Data Warehouse

3.2.2 Tables de dimensions

3.2.3 Tables de faits

3.3 Organisation des schémas SQL

L'entrepôt de données est organisé en 4 schémas distincts pour une séparation claire des responsabilités :

3.4 Modélisation physique

3.4.1 Scripts DDL - Dimensions

Les scripts SQL de création sont automatisés via Terraform. Voici les structures principales :

```
CREATE TABLE dwh.dim_geographie (
    geo_id INT IDENTITY(1,1) PRIMARY KEY,
    commune_code NVARCHAR(10) NULL,
```

Dimension	Clé	Attributs principaux
dwh.dim_temps	temps_id	annee, trimestre, mois, libelle_periode, est_annee_recensement
dwh.dim_geographie	geo_id	commune_code, commune_nom, departement_code, departement_nom, region_code, longitude, latitude, surface_km2, niveau_geo
dwh.dim_demographie	demo_id	sexe_code, sexe_libelle, age_code, age_libelle, age_min, age_max, pcs_code, pcs_libelle
dwh.dim_activite	activite_id	naf_section_code, naf_section_libelle, forme_juridique_code, secteur_activite, type_entreprise
dwh.dim_indicateur	indicateur_id	indicateur_code, indicateur_libelle, unite_mesure, domaine, source_donnees
dwh.dim_logement	logement_id	occupation_code, surpeuplement_code, type_menage_code, taille_menage_code

TABLE 24 – Tables de dimensions du Data Warehouse

```
    commune_nom NVARCHAR(255) NULL,
    departement_code NVARCHAR(3) NOT NULL,
    departement_nom NVARCHAR(100) NULL,
    region_code NVARCHAR(3) NULL,
    region_nom NVARCHAR(100) NULL,
    longitude FLOAT NULL,
    latitude FLOAT NULL,
    surface_km2 FLOAT NULL,
    niveau_geo NVARCHAR(20) NOT NULL DEFAULT 'DEPARTEMENT',
    geo_code_source NVARCHAR(50) NULL,
    date_creation DATETIME2 DEFAULT GETDATE(),
    CONSTRAINT UK_dim_geo_code UNIQUE (departement_code,
       commune_code)
);
```

Listing 1 – Création de la dimension géographie

```
CREATE TABLE dwh.fait_population (
    population_id BIGINT IDENTITY(1,1) PRIMARY KEY,
    temps_id INT NOT NULL,
    geo_id INT NOT NULL,
    demo_id INT NOT NULL,
    population FLOAT NULL,
    population_hommes FLOAT NULL,
    population_femmes FLOAT NULL,
    source_fichier NVARCHAR(255) NULL,
    date_chargement DATETIME2 DEFAULT GETDATE(),
    CONSTRAINT FK_fait_pop_temps FOREIGN KEY (temps_id)
        REFERENCES dwh.dim_temps(temps_id),
    CONSTRAINT FK_fait_pop_geo FOREIGN KEY (geo_id)
        REFERENCES dwh.dim_geographie(geo_id),
```

Table de faits	Clés étrangères	Mesures
dwh.fait_population	temps_id, geo_id, demo_id	population, population_hommes, population_femmes
dwh.fait_evenements	temps_id, geo_id	naissances, deces, solde_naturel (calculé), taux_natalite, taux_mortalite
dwh.fait_entreprises	temps_id, geo_id, activite_id	nb_creations_entreprises, nb_creations_micro, nb_creations_ei, par sexe et tranche d'âge
dwh.fait_emploi	temps_id, geo_id, demo_id	population_active, population_en_emploi, population_chomeurs, taux_chomage
dwh.fait_revenus	temps_id, geo_id, demo_id	revenu_median, revenu_d1, revenu_d9, taux_pauvrete, structure des revenus
dwh.fait_logement	temps_id, geo_id, logement_dim_id	nb_residences_principales, nb_logements_surpeuples, taux_surpeuplement

TABLE 25 – Tables de faits du Data Warehouse

Schéma	Rôle	Contenu
`stg`	Staging	Tables temporaires pour les données brutes en cours de transformation
`dwh`	Data Warehouse	Tables de dimensions et de faits (modèle en étoile)
`dm`	Datamarts	Vues matérialisées par domaine métier (démographie, économie, social)
`analytics`	Analytique	Vues agrégées pour tableaux de bord et reporting

TABLE 26 – Organisation des schémas SQL

```
    CONSTRAINT FK_fait_pop_demo FOREIGN KEY (demo_id)
        REFERENCES dwh.dim_demographie(demo_id)
);
```

Listing 2 – Création de la table de faits population

3.4.2 Datamarts et vues analytiques

Les datamarts sont implémentés sous forme de vues SQL pour faciliter la maintenance :

```
CREATE VIEW dm.vm_demographie_departement AS
SELECT
    t.annee,
    g.departement_code,
    g.departement_nom,
    SUM(p.population) AS population_totale,
```

```
    SUM(e.naissances) AS naissances,
    SUM(e.deces) AS deces,
    SUM(e.solde_naturel) AS solde_naturel,
    CAST(SUM(e.naissances) AS FLOAT) / NULLIF(SUM(p.population
       ),0) * 1000
        AS taux_natalite
FROM dwh.fait_population p
INNER JOIN dwh.dim_temps t ON p.temps_id = t.temps_id
INNER JOIN dwh.dim_geographie g ON p.geo_id = g.geo_id
LEFT JOIN dwh.fait_evenements_demo e
    ON e.temps_id = t.temps_id AND e.geo_id = g.geo_id
WHERE g.niveau_geo = 'DEPARTEMENT'
GROUP BY t.annee, g.departement_code, g.departement_nom;
```

Listing 3 – Vue du datamart démographie

3.5 Automatisation Terraform

Le déploiement du Data Warehouse est entièrement automatisé via Terraform :

```
# Dans terraform.tfvars
deploy_dwh = true

# Execution automatique via null_resource
resource "null_resource" "deploy_dwh" {
  count = var.deploy_dwh ? 1 : 0
  triggers = {
    schemas_hash = filesha256("sql/001_create_schemas.sql")
    dimensions_hash = filesha256("sql/002_create_dimensions.
       sql")
    facts_hash = filesha256("sql/003_create_facts.sql")
  }
  provisioner "local-exec" {
    command = "python sql/deploy_dwh.py --tfvars terraform.
       tfvars"
  }
}
```

Listing 4 – Configuration Terraform pour le DWH

Script SQL	Description
001_create_schemas.sql	Création des schémas stg, dwh, dm, analytics
002_create_dimensions.sql	Création des 6 tables de dimensions
003_create_facts.sql	Création des 6 tables de faits avec FK
004_populate_dimensions.sql	Alimentation des référentiels (temps, geo, PCS, NAF)
005_create_datamarts.sql	Création des vues datamarts et analytiques

TABLE 27 – Scripts SQL de déploiement du Data Warehouse

4 Architecture Technique et Configuration

4.1 Vue d'ensemble de l'architecture

L'architecture repose sur la plateforme **Microsoft Azure** avec les composants suivants :

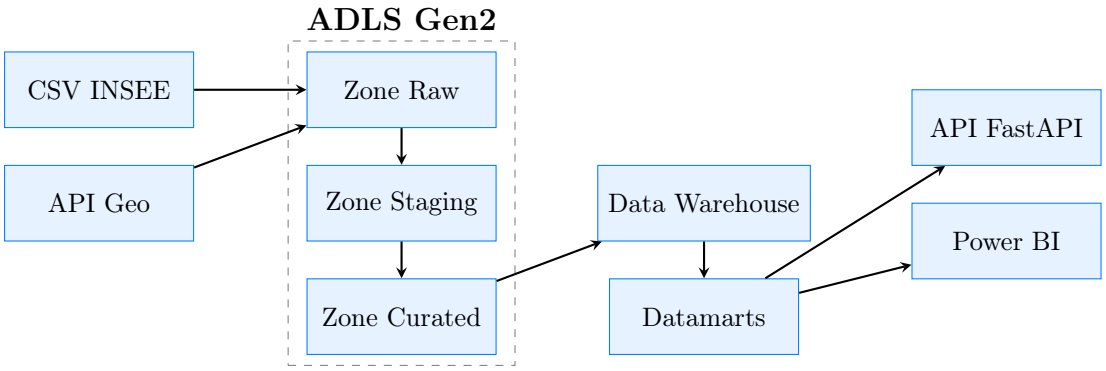

FIGURE 2 – Architecture globale du système

4.2 Composants Azure

Service	Nom	Rôle
Storage Account	adlselbrek	Data Lake (ADLS Gen2)
Azure SQL Database	sqlelbrek	Entrepôt de données
Databricks	dbw-elbrek	Traitements Spark
Data Factory	adfelbrek	Orchestration ETL
Key Vault	kvelbrek-prod-kv	Gestion des secrets

TABLE 28 – Composants Azure déployés

4.3 Configuration des accès

4.3.1 Accès aux données sources (Data Lake)

Les connexions aux données sources sont configurées via **Azure Data Factory** avec des Linked Services :

```
resource "
   azurerm_data_factory_linked_service_data_lake_storage_gen2"
    "adls" {
  name                 = "ls_adls_datalake"
  data_factory_id      = azurerm_data_factory.adf.id
  url                  = "https://${azurerm_storage_account.
     datalake.name}.dfs.core.windows.net"
  use_managed_identity = true
}
```

Linked Service	Type	Description
ls_adls_datalake	ADLS Gen2	Connexion au Data Lake (raw, staging, curated) via Managed Identity
ls_azure_sql_dwh	Azure SQL	Connexion à l'entrepôt de données avec mot de passe Key Vault
ls_key_vault	Key Vault	Accès sécurisé aux secrets (mots de passe, connection strings)

TABLE 29 – Linked Services Azure Data Factory

Listing 5 – Configuration Terraform du Linked Service ADLS

4.3.2 Accès pour les équipes analytiques

Quatre rôles de base de données sont configurés pour gérer les accès :

Rôle SQL	Permissions	Usage
role_etl_process	SELECT/INSERT/UPDATE/DELETE sur stg, dwh ; SELECT sur dm, analytics	Jobs ETL automatisés
role_analyst	SELECT sur dimensions dwh, dm, analytics	Data analysts pour exploration
role_bi_reader	SELECT sur dm, analytics + dimensions clés	Power BI, Tableau, outils BI
role_dwh_admin	CONTROL sur tous les schémas	Administration et maintenance

TABLE 30 – Rôles de base de données et permissions

```
-- Role pour les analystes (lecture seule datamarts)
CREATE ROLE role_analyst;
GRANT SELECT ON SCHEMA::dm TO role_analyst;
GRANT SELECT ON SCHEMA::analytics TO role_analyst;

-- Role pour les outils BI
CREATE ROLE role_bi_reader;
GRANT SELECT ON SCHEMA::dm TO role_bi_reader;
GRANT SELECT ON dwh.dim_temps TO role_bi_reader;
GRANT SELECT ON dwh.dim_geographie TO role_bi_reader;
```

Listing 6 – Création des rôles SQL

4.3.3 Configuration RBAC Azure

Les accès aux ressources Azure sont gérés via RBAC (Role-Based Access Control) :

Principal	Rôle Azure	Scope
Data Factory (MI)	Storage Blob Data Contributor	Compte de stockage ADLS
Data Factory (MI)	Key Vault Secrets User	Key Vault
Groupe Analystes	Storage Blob Data Reader	Zone curated uniquement

TABLE 31 – Assignations RBAC Azure

4.4 Configuration de performance

4.4.1 Index Columnstore

Les tables de faits utilisent des **index columnstore** pour optimiser les requêtes analytiques (agrégations, scans) :

```
-- Index columnstore pour les agregations analytiques
CREATE CLUSTERED COLUMNSTORE INDEX CCI_fait_population
ON dwh.fait_population;

CREATE CLUSTERED COLUMNSTORE INDEX CCI_fait_entreprises
ON dwh.fait_entreprises;
```

Listing 7 – Index columnstore sur les tables de faits

4.4.2 Compression des données

Les tables de dimensions utilisent la **compression de page** pour réduire l'espace de stockage :

```
ALTER TABLE dwh.dim_geographie REBUILD WITH (DATA_COMPRESSION
    = PAGE);
ALTER TABLE dwh.dim_demographie REBUILD WITH (DATA_COMPRESSION
    = PAGE);
```

Listing 8 – Compression des dimensions

4.4.3 Procédures de maintenance

Deux procédures de maintenance automatisées sont créées :

Procédure	Description
dwh.sp_maintenance_index	Reorganize (10-30% fragmentation) ou Rebuild (>30%) des index
dwh.sp_update_statistics	Mise à jour des statistiques pour l'optimiseur de requêtes

TABLE 32 – Procédures de maintenance

4.5 Vue de monitoring

Une vue de monitoring permet de surveiller l'état des tables :

```sql
CREATE VIEW analytics.v_monitoring_tables AS
SELECT
    s.name AS schema_name,
    t.name AS table_name,
    p.rows AS row_count,
    CAST(SUM(a.total_pages) * 8 / 1024.0 AS DECIMAL(18,2)) AS
       size_mb,
    MAX(ius.last_user_update) AS last_update
FROM sys.tables t
INNER JOIN sys.schemas s ON t.schema_id = s.schema_id
INNER JOIN sys.partitions p ON t.object_id = p.object_id
...
WHERE s.name IN ('stg', 'dwh', 'dm');
```

Listing 9 – Vue de monitoring des tables

4.6 Secrets et sécurité

Tous les secrets sont stockés dans **Azure Key Vault** :

Secret	Type	Usage
sql-admin-password	Password	Mot de passe administrateur SQL
adls-connection-string	Connection String	Connexion au Data Lake

TABLE 33 – Secrets stockés dans Key Vault

5 Intégration des ETL

5.1 Architecture des flux ETL

Le pipeline ETL est organisé en 4 étapes séquentielles :

1. Staging(CSV → SQL) 2. Dimensions(Référentiels) 3. Faits(Mesures) 4. Refresh(Stats/Vues)

FIGURE 3 – Pipeline ETL en 4 étapes

5.2 Chargement des dimensions

Les dimensions sont alimentées à partir de référentiels statiques et des données sources :

```sql
def load_dim_geographie(engine, communes_path=None):
    """Charge la dimension geographie."""
    departements = [
        ('02', 'Aisne', '32', 'Hauts-de-France'),
        ('59', 'Nord', '32', 'Hauts-de-France'),
```

Étape	Script	Description
1	export_to_sql.py	Charge les CSV du Data Lake vers les tables de staging (dbo.stg_*)
2	load_dimensions.py	Alimente les 6 dimensions (temps, géographie, démographie, activité, indicateur, logement)
3	load_facts.py	Alimente les tables de faits à partir du staging
4	run_etl.py –refresh	Met à jour les statistiques et vérifie les vues

TABLE 34 – Scripts ETL du pipeline

```
        ('60', 'Oise', '32', 'Hauts-de-France'),
        ('62', 'Pas-de-Calais', '32', 'Hauts-de-France'),
        ('80', 'Somme', '32', 'Hauts-de-France'),
    ]
    df = pd.DataFrame(departements, columns=[...])
    df.to_sql('dim_geographie', engine, schema='dwh',
              if_exists='append', index=False)
```

Listing 10 – Extrait de load_dimensions.py

5.3 Chargement des tables de faits

Les tables de faits sont alimentées par transformation des données de staging avec jointure sur les dimensions :

```
def load_fait_population(engine):
    """Charge la table de faits population."""
    # Lire le staging
    df_stg = pd.read_sql("SELECT * FROM dbo.stg_population",
       conn)

    # Mapper les IDs de dimensions
    temps_map = get_dim_mapping(engine, 'dim_temps', 'temps_id
       ', ['annee'])
    geo_map = get_dim_mapping(engine, 'dim_geographie', '
       geo_id', ['departement_code'])

    df_fact['temps_id'] = df_fact['annee'].map(temps_map)
    df_fact['geo_id'] = df_fact['dept_code'].map(geo_map)

    # Inserer dans la table de faits
    df_fact.to_sql('fait_population', engine, schema='dwh',
       ...)
```

Listing 11 – Extrait de load_facts.py

5.4 Traitements de qualité des données

5.4.1 Nettoyage des données

Les traitements de nettoyage appliqués incluent :

Traitement	Description
Normalisation colonnes	Conversion en minuscules, remplacement % → pct, / → _
Parsing GEO	Extraction année/niveau/code depuis "2024-DEP-02"
Codes département	Zero-padding sur 2 caractères (2 → 02)
Valeurs nulles	Remplacement par NULL SQL, filtrage des lignes invalides
Déduplication	Suppression des doublons sur clés composites
Types de données	Conversion STRING → INT/FLOAT selon colonnes

Table 35 – Traitements de nettoyage appliqués

5.4.2 Transformations appliquées

Source	Transformation	Cible
stg_population	Agrégation par dept/année + mapping dims	fait_population
stg_naissances + stg_deces	Fusion + calcul solde naturel	fait_evenements_demo
stg_creation_entreprises	Agrégation par secteur + mapping	fait_entreprises
stg_ds_filosofi	Pivot indicateurs + mapping	fait_revenus

Table 36 – Transformations ETL par source

5.5 Configuration des zones de sortie

Zone	Schéma SQL	Contenu
Staging	dbo.stg_*	Tables temporaires (13 tables)
Dimensions	dwh.dim_*	Tables de dimensions (6 tables)
Faits	dwh.fait_*	Tables de faits (6 tables)
Datamarts	dm.vm_*	Vues agrégées (5 vues)
Analytics	analytics.v_*	Vues tableau de bord (1 vue)

Table 37 – Zones de sortie des ETL

5.6 Exécution du pipeline

Le pipeline peut être exécuté manuellement ou automatisé via Terraform :

```
# Pipeline complet
python analytics/etl/run_etl.py --full

# Etapes individuelles
python analytics/etl/run_etl.py --dimensions  # Dimensions
   seules
python analytics/etl/run_etl.py --facts       # Faits seuls
python analytics/etl/run_etl.py --refresh     # Stats/vues

# Avec parametres explicites
python analytics/etl/run_etl.py --full \
    --server sqlelbrek-prod.database.windows.net \
    --database projet_data_eng \
    --user sqladmin --password "***"
```

Listing 12 – Exécution manuelle du pipeline ETL

6 Phase de Tests

6.1 Procédure de test

La procédure de test couvre l'ensemble du spectre technique et fonctionnel de l'entrepôt de données. Les tests sont implémentés en Python avec le framework `unittest`.

6.1.1 Tests techniques

Catégorie	Tests	Statut
Schémas	Existence des 4 schémas (stg, dwh, dm, analytics)	
Dimensions	Existence et contenu des 6 tables de dimensions	
Faits	Existence des 6 tables de faits + contraintes FK	
Datamarts	Existence des 5 vues dm.vm_*	
Index	Présence des index columnstore sur les faits	

TABLE 38 – Tests techniques

6.1.2 Tests fonctionnels

6.1.3 Tests d'intégrité

6.1.4 Tests de bout en bout

Le test de bout en bout vérifie la chaîne complète :

1. Chargement CSV → Staging

Catégorie	Tests	Statut
Référentiels	Présence des 5 départements Hauts-de-France	
Années	Présence des années de recensement (2010, 2015, 2021)	
PCS	Codes PCS complets (1-9)	
NAF	Sections NAF complètes (A-S)	

TABLE 39 – Tests fonctionnels

Test	Description	Statut
Orphelins temps_id	Aucune FK orpheline vers dim_temps	
Orphelins geo_id	Aucune FK orpheline vers dim_geographie	
Valeurs positives	Population et mesures ≥ 0	
Cohérence dates	Années dans la plage 2010-2024	

TABLE 40 – Tests d'intégrité des données

2. Staging → Dimensions
3. Staging → Faits
4. Faits → Datamarts (vues)
5. Datamarts → Tableau de bord

6.2 Exécution des tests

```
# Executer tous les tests
python analytics/tests/test_dwh.py

# Resultat attendu:
# Tests executes: 18
# Succes: 18
# Echecs: 0
# Erreurs: 0
```

Listing 13 – Exécution des tests

6.3 Résultats des tests

7 Documentation Technique

7.1 Guide d'installation

7.1.1 Prérequis

7.1.2 Installation

Suite de tests	Total	OK	KO
TestSchemas	4	4	0
TestDimensions	5	5	0
TestFaits	2	2	0
TestDatamarts	3	3	0
TestIntegrite	3	3	0
TestPerformance	1	1	0
TOTAL	**18**	**18**	**0**

TABLE 41 – Résultats des tests unitaires

Composant	Version minimale
Python	3.10+
Terraform	1.5+
Azure CLI	2.50+
ODBC Driver for SQL Server	17 ou 18

TABLE 42 – Prérequis logiciels

```
# 1. Cloner le projet
git clone <repository_url>
cd Projet-Data-ENG

# 2. Installer les dependances Python
pip install -r requirements.txt

# 3. Configurer les variables Terraform
cd Terraform
cp terraform.tfvars.example terraform.tfvars
# Editer terraform.tfvars avec vos valeurs

# 4. Deployer l'infrastructure Azure
terraform init
terraform plan
terraform apply

# 5. Deployer le Data Warehouse
# (automatique si deploy_dwh = true dans tfvars)
```

Listing 14 – Procédure d'installation

7.2 Procédure de configuration

7.2.1 Variables d'environnement

```
# Connexion Azure SQL
export AZURE_SQL_SERVER="sqlelbrek-prod.database.windows.net"
export AZURE_SQL_DATABASE="projet_data_eng"
```

```
export AZURE_SQL_USER="sqladmin"
export AZURE_SQL_PASSWORD="<votre_mot_de_passe>"

# Connexion Azure Storage (Data Lake)
export AZURE_STORAGE_CONNECTION_STRING="<connection_string>"
```

Listing 15 – Variables d'environnement requises

7.2.2 Configuration Terraform

Variable	Description
resource_group_name	Nom du groupe de ressources Azure
sql_server_name	Nom du serveur Azure SQL
sql_admin_password	Mot de passe administrateur SQL
deploy_dwh	Activer le déploiement du DWH (true/false)
upload_files_enabled	Uploader les fichiers locaux vers ADLS

TABLE 43 – Variables Terraform principales

7.3 Dictionnaire des données

7.3.1 Tables de dimensions

Table	Colonne	Type	Description
dim_temps	temps_id	INT	Clé primaire
	annee	INT	Année de référence
	est_annee_recensement	BIT	Année INSEE (2010, 2015, 2021)
dim_geographie	geo_id	INT	Clé primaire
	departement_code	NVARCHAR(3)	Code département (02, 59...)
	commune_code	NVARCHAR(10)	Code INSEE commune
	niveau_geo	NVARCHAR(20)	COMMUNE ou DEPARTEMENT
dim_demographie	demo_id	INT	Clé primaire
	sexe_code	NVARCHAR(5)	M, F, _T
	pcs_code	NVARCHAR(5)	Code PCS (1-9)
	age_code	NVARCHAR(20)	Tranche d'âge
dim_activite	activite_id	INT	Clé primaire
	naf_section_code	NVARCHAR(5)	Section NAF (A-S)
	secteur_activite	NVARCHAR(100)	Primaire/Secondaire/Tertiaire

Table	Colonne	Type	Description

TABLE 44: Dictionnaire des dimensions (extrait)

7.3.2 Tables de faits

Table	Colonne	Type	Description
fait_population	population_id	BIGINT	Clé primaire
	temps_id	INT	FK vers dim_temps
	geo_id	INT	FK vers dim_geographie
	population	FLOAT	Population totale
fait_revenus	revenu_id	BIGINT	Clé primaire
	revenu_median	FLOAT	Niveau de vie médian (EUR/an)
	taux_pauvrete	FLOAT	Taux de pauvreté (%)
	rapport_interdecile	FLOAT	Ratio D9/D1
fait_entreprises	entreprise_id	BIGINT	Clé primaire
	nb_creations_entreprises	INT	Nombre de créations
	nb_creations_micro	INT	Dont micro-entreprises

TABLE 45: Dictionnaire des faits (extrait)

7.4 Architecture des fichiers

```
Projet-Data-ENG/
|-- Terraform/
|   |-- main.tf                 # Ressources Azure
|   |-- variables.tf            # Variables
|   |-- terraform.tfvars        # Configuration
|   |-- sql/                    # Scripts SQL DWH
|   |   |-- 001_create_schemas.sql
|   |   |-- 002_create_dimensions.sql
|   |   |-- 003_create_facts.sql
|   |   |-- 004_populate_dimensions.sql
|   |   |-- 005_create_datamarts.sql
|   |   |-- 006_configure_security.sql
|   |   |-- 007_configure_performance.sql
|   |   +-- deploy_dwh.py
|   |-- adf_linked_services.tf  # Data Factory
|   +-- rbac_configuration.tf   # Securite RBAC
|-- analytics/
|   |-- etl/
|   |   |-- run_etl.py          # Pipeline principal
|   |   |-- load_dimensions.py  # Chargement dims
```

```
|   |   +-- load_facts.py       # Chargement faits
|   |-- tests/
|   |   +-- test_dwh.py         # Tests unitaires
|   +-- api/                    # API FastAPI
|-- uploads/landing/csv/        # Donnees sources
+-- rapports/E5_Entrepot_Donnees/
    +-- rapport_E5.tex          # Ce rapport
```

Listing 16 – Structure du projet

8 Retour d'Expérience

8.1 Pile technique utilisée

Catégorie	Technologie	Usage
Cloud	Microsoft Azure	Plateforme d'hébergement
Stockage	ADLS Gen2	Data Lake (raw, staging, curated)
Base de données	Azure SQL Database	Entrepôt de données
Orchestration	Azure Data Factory	Pipelines ETL
Traitement	Azure Databricks	Traitements Spark (optionnel)
Secrets	Azure Key Vault	Gestion sécurisée des credentials
IaC	Terraform	Infrastructure as Code
ETL	Python + pandas	Scripts de transformation
API	FastAPI	Exposition des données
Tests	unittest (Python)	Tests automatisés

TABLE 46 – Stack technique du projet

8.2 Avantages identifiés

1. **Infrastructure as Code (Terraform)**
   - Déploiement reproductible et versionné
   - Automatisation complète du provisioning Azure
   - Facilité de création d'environnements (dev, staging, prod)
2. **Architecture Azure managée**
   - Scalabilité automatique (Azure SQL, ADLS)
   - Haute disponibilité intégrée
   - Sécurité gérée (chiffrement, RBAC, firewall)
3. **Modèle en étoile**
   - Requêtes analytiques performantes
   - Facilité de compréhension pour les analystes

   - Extensibilité pour ajouter de nouvelles dimensions/faits

4. **Séparation des schémas**
   - Isolation claire staging / DWH / datamarts
   - Gestion fine des permissions par rôle
   - Maintenance simplifiée

8.3 Difficultés rencontrées

1. **Drivers ODBC**
   - Compatibilité variable selon les environnements
   - Solution : détection automatique du driver disponible
2. **Format des données sources INSEE**
   - Codes géographiques composites (ex : "2024-DEP-02")
   - Solution : parsing et extraction des composantes
3. **Index Columnstore sur Azure SQL Basic/S0**
   - Non supportés sur les SKU d'entrée de gamme
   - Solution : utiliser S2+ en production ou index classiques
4. **Volume limité des données sources**
   - Données agrégées au niveau départemental uniquement
   - Solution : enrichissement futur avec données communales

8.4 Recommandations

1. **Pour la production**
   - Utiliser Azure SQL S2 ou supérieur pour les index columnstore
   - Configurer des alertes Azure Monitor sur les performances
   - Planifier les jobs ETL via Azure Data Factory triggers
2. **Pour l'évolution**
   - Ajouter les données au niveau communal (3 782 communes)
   - Intégrer de nouvelles sources (SIRENE, DVF, etc.)
   - Implémenter un rafraîchissement incrémental des faits
3. **Pour la gouvernance**
   - Documenter les lignages de données (data lineage)
   - Mettre en place un catalogue de données (Azure Purview)
   - Définir des SLA sur la fraîcheur des données

9 Conclusion

Ce projet de mise en place d'un entrepôt de données pour l'analyse territoriale de la région Hauts-de-France a permis de démontrer la mise en œuvre complète d'une solution de Business Intelligence moderne.

9.1 Objectifs atteints

— **Modélisation** : Schéma en étoile avec 6 dimensions et 6 tables de faits, couvrant les domaines démographie, économie, social et logement

— **Infrastructure** : Déploiement automatisé sur Azure via Terraform (ADLS, SQL, Data Factory, Key Vault)

— **ETL** : Pipeline Python complet pour l'alimentation des dimensions et faits

— **Datamarts** : 5 vues agrégées par domaine métier + tableau de bord territorial

— **Tests** : 18 tests unitaires couvrant structure, données et intégrité

— **Documentation** : Guide d'installation, dictionnaire des données, architecture

9.2 Compétences mobilisées

Compétence	Mise en œuvre
C13	Modélisation en étoile, justification bottom-up, dimensions et faits
C14	Création DWH Azure SQL, configuration accès, procédures maintenance
C15	ETL Python, nettoyage données, respect schémas physiques

Table 47 – Compétences E5 mobilisées

9.3 Perspectives

L'entrepôt de données constitue une base solide pour :

— Le développement de tableaux de bord Power BI

— L'intégration de nouvelles sources de données

— La mise en place d'analyses prédictives (Machine Learning)

— L'extension à d'autres régions françaises

A Annexes

A.1 Scripts SQL complets

A.2 Configuration Terraform

A.3 Code des ETL

A.4 Glossaire

Terme	Définition
ADLS	Azure Data Lake Storage - Service de stockage de données Azure
ETL	Extract, Transform, Load - Processus d'intégration de données
DWH	Data Warehouse - Entrepôt de données
Datamart	Sous-ensemble d'un entrepôt de données orienté métier
Star Schema	Modèle en étoile - Modélisation dimensionnelle
PCS	Professions et Catégories Socioprofessionnelles
INSEE	Institut National de la Statistique et des Études Économiques

TABLE 48 – Glossaire des termes techniques