



UNIVERSITE DES SIENCES ET TECHNOLOGIES HOUARI BOUMEDIENNE

## Rapport de réalisation du projet BI

Année 2025/2026

Étudiante : Boukhedouni Nada Raiham

Matricule : 232332239314

ING 3 Cyber Sécurité

Configuration

Années

1996 x 1997 x

1998 x 2006 x

Employés

1 x 2 x

3 x 4 x

5 x 6 x

7 x 8 x

Clients

Choose options

Registre complet

	orderid	customerid	employeeid	orderdate	requireddate	shippeddate	shipvia	freight	shipname	shipaddress	shipcity
0	10248	VINET	5	1996-07-04	1996-08-01	1996-07-16 00:00:00	3	32.38	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims
1	10248	VINET	5	1996-07-04	1996-08-01	1996-07-16 00:00:00	3	32.38	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims
2	10248	VINET	5	1996-07-04	1996-08-01	1996-07-16 00:00:00	3	32.38	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims
3	10249	TOMSP	6	1996-07-05	1996-08-16	1996-07-10 00:00:00	1	11.61	Toms Spezialit ten	Luisenstr. 48	M nster
4	10249	TOMSP	6	1996-07-05	1996-08-16	1996-07-10 00:00:00	1	11.61	Toms Spezialit ten	Luisenstr. 48	M nster
5	10250	HANAR	4	1996-07-08	1996-08-05	1996-07-12 00:00:00	2	65.83	Hanari Carnes	Rua do Pa o, 67	Rio de Jan
6	10250	HANAR	4	1996-07-08	1996-08-05	1996-07-12 00:00:00	2	65.83	Hanari Carnes	Rua do Pa o, 67	Rio de Jan
7	10250	HANAR	4	1996-07-08	1996-08-05	1996-07-12 00:00:00	2	65.83	Hanari Carnes	Rua do Pa o, 67	Rio de Jan
8	10251	VICTE	3	1996-07-08	1996-08-05	1996-07-15 00:00:00	1	41.34	Victuailles en stock	2, rue du Commerce	Lyon
9	10251	VICTE	3	1996-07-08	1996-08-05	1996-07-15 00:00:00	1	41.34	Victuailles en stock	2, rue du Commerce	Lyon

## Introduction:

Dans ce projet, l'objectif était de construire un **pipeline BI complet** permettant d'extraire des données depuis différentes sources, de les transformer, de les charger dans un format exploitable et de créer des **visualisations interactives** via un dashboard Streamlit.

Les principales étapes sont :

1. Extraction des données depuis SQLSERVER et ACCESS NORTHWIND 2012.
2. Transformation et nettoyage des données pour les rendre cohérentes et exploitables.
3. Chargement des données transformées dans des fichiers ou bases intermédiaires.
4. Création d'un dashboard interactif pour l'analyse visuelle des données.

Les livrables du projet incluent :

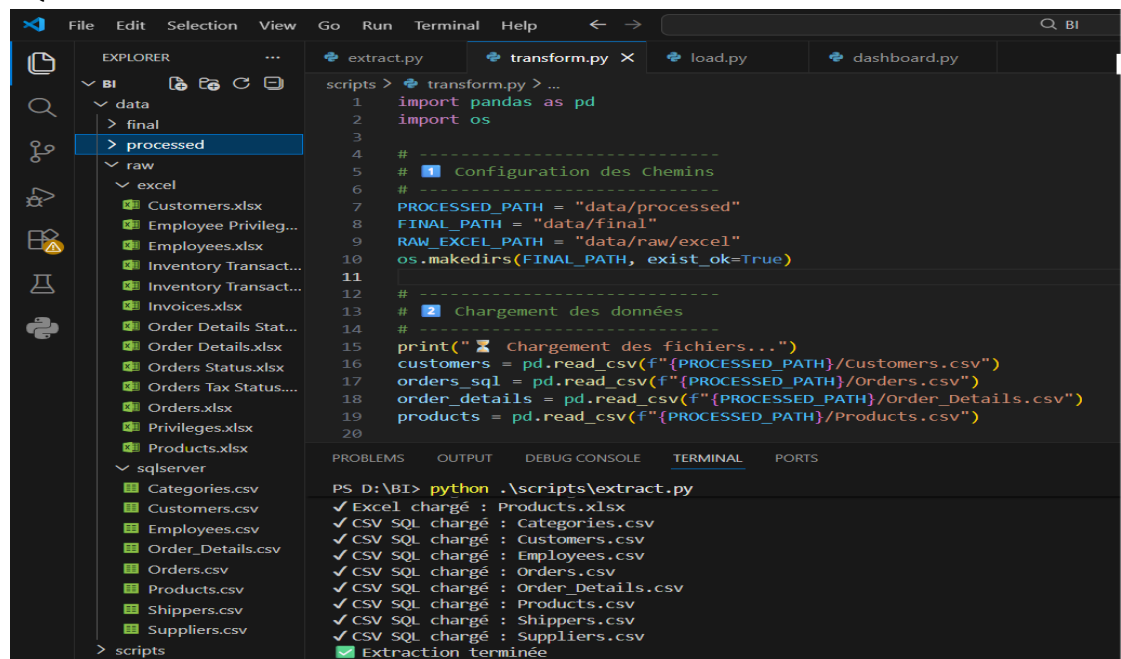
- Scripts Python (.py) pour ETL et visualisation.
- Dashboard interactif avec Streamlit.
- Rapport détaillé (ce document).
- Vidéo présentant le projet de A à Z.

## Arborescence du projet:

**Dossier data** : fichiers sources et données extraites

**/\_ dossier raw** : - **dossier excel** : contient les fichiers extraits de ACCESS NORTHWIND 2012

- **dossier sql** : contient les fichiers extraits de la base de données Northwind sous SQLSERVER



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the project structure. The main editor shows the `transform.py` file, and the terminal at the bottom shows the output of the `extract.py` script.

**Project Structure (Explorer):**

- BI
  - data
    - final
    - processed
    - raw
      - excel
        - Customers.xlsx
        - Employee Privileg...
        - Employees.xlsx
        - Inventory Transact...
        - Invoices.xlsx
        - Order Details.xlsx
        - Orders Status.xlsx
        - Orders Tax Status....
        - Orders.xlsx
        - Privileges.xlsx
        - Products.xlsx
      - sqlserver
        - Categories.csv
        - Customers.csv
        - Employees.csv
        - Order\_Details.csv
        - Orders.csv
        - Products.csv
        - Shippers.csv
        - Suppliers.csv

**transform.py (Main Editor):**

```
1 import pandas as pd
2 import os
3
4 # -----
5 # 1 Configuration des Chemins
6 # -----
7 PROCESSED_PATH = "data/processed"
8 FINAL_PATH = "data/final"
9 RAW_EXCEL_PATH = "data/raw/excel"
10 os.makedirs(FINAL_PATH, exist_ok=True)
11
12 # -----
13 # 2 Chargement des données
14 # -----
15 print("📄 Chargement des fichiers...")
16 customers = pd.read_csv(f"{PROCESSED_PATH}/Customers.csv")
17 orders_sql = pd.read_csv(f"{PROCESSED_PATH}/Orders.csv")
18 order_details = pd.read_csv(f"{PROCESSED_PATH}/Order_Details.csv")
19 products = pd.read_csv(f"{PROCESSED_PATH}/Products.csv")
20
```

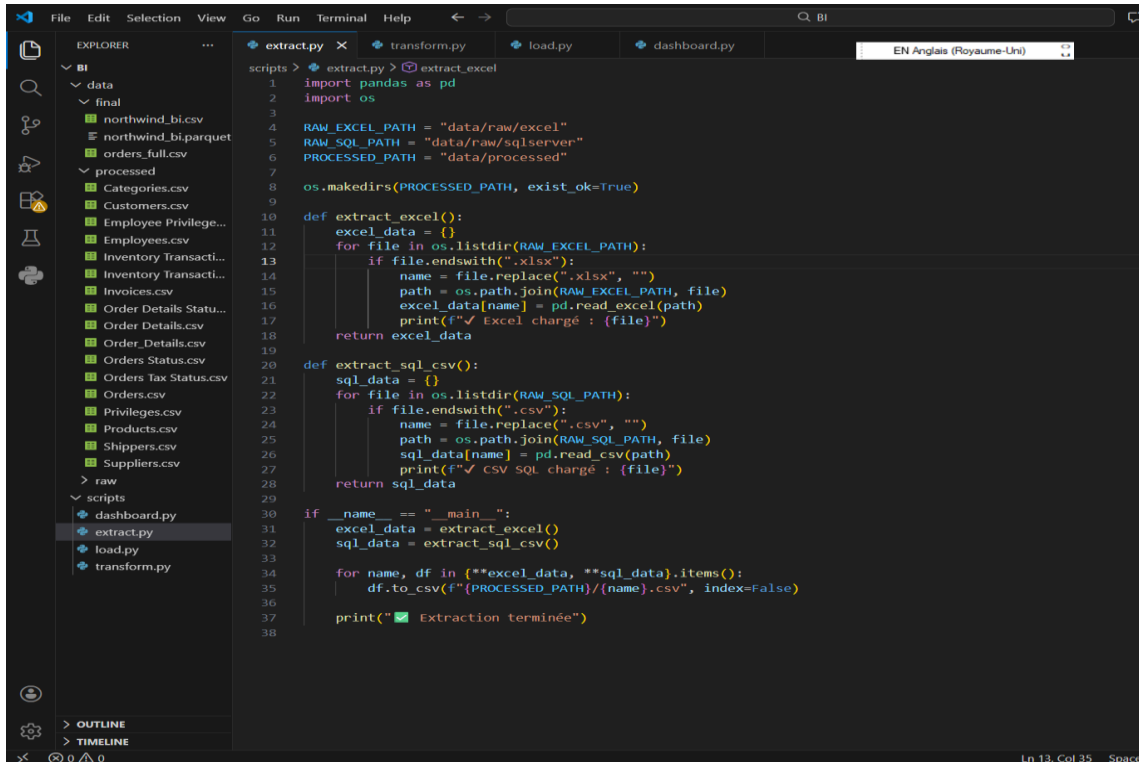
**Terminal (Output):**

```
PS D:\BI> python .\scripts\extract.py
✓ Excel chargé : Products.xlsx
✓ CSV SQL chargé : Categories.csv
✓ CSV SQL chargé : Customers.csv
✓ CSV SQL chargé : Employees.csv
✓ CSV SQL chargé : Orders.csv
✓ CSV SQL chargé : Order_Details.csv
✓ CSV SQL chargé : Products.csv
✓ CSV SQL chargé : Shippers.csv
✓ CSV SQL chargé : Suppliers.csv
✓ Extraction terminée
```

**/-dossier processed** : contient les fichiers transformés (nettoyés et normalisés) en format CSV

**/-dossier final** : contient les fichiers finaux qui vont être les fichiers source pour l'affichage du dashboard **northwind\_bi.csv** pour la Vérification des données et Partage avec utilisateurs métiers, **northwind\_bi.parquet** Performance et analytics aussi pour le Dashboard Streamlit

**Orders\_full.csv** qui contient les données des commandes



```
1 import pandas as pd
2 import os
3
4 RAW_EXCEL_PATH = "data/raw/excel"
5 RAW_SQL_PATH = "data/raw/sqlserver"
6 PROCESSED_PATH = "data/processed"
7
8 os.makedirs(PROCESSED_PATH, exist_ok=True)
9
10 def extract_excel():
11     excel_data = {}
12     for file in os.listdir(RAW_EXCEL_PATH):
13         if file.endswith(".xlsx"):
14             name = file.replace(".xlsx", "")
15             path = os.path.join(RAW_EXCEL_PATH, file)
16             excel_data[name] = pd.read_excel(path)
17             print(f"✓ Excel chargé : {file}")
18     return excel_data
19
20 def extract_sql_csv():
21     sql_data = {}
22     for file in os.listdir(RAW_SQL_PATH):
23         if file.endswith(".csv"):
24             name = file.replace(".csv", "")
25             path = os.path.join(RAW_SQL_PATH, file)
26             sql_data[name] = pd.read_csv(path)
27             print(f"✓ CSV SQL chargé : {file}")
28     return sql_data
29
30 if __name__ == "__main__":
31     excel_data = extract_excel()
32     sql_data = extract_sql_csv()
33
34     for name, df in [{"excel_data", **sql_data}.items():
35         df.to_csv(f"{PROCESSED_PATH}/{name}.csv", index=False)
36
37     print("✓ Extraction terminée")
38
```

**Dossier scripts** : contient tous les scripts python nécessaires pour réaliser le projet

**extract\_sqlserver.py** : Ce script Python permet d'extraire des données depuis une base de données SQL Server (Northwind) et de les sauvegarder sous forme de fichiers CSV.

- Bibliothèques : pandas et pyodbc
- Definition de chemin de sortie data/raw/sqlserver/
- Fonction `extract_from_sqlserver()` : connection a sql server
- Liste des tables a extraire
- Extraction et sauvegarde de données

**extract.py** : Le script d'extraction permet de collecter automatiquement les données provenant de fichiers Excel et de données issues d'une base SQL exportées au format CSV. Chaque fichier est chargé dans un DataFrame Pandas, puis sauvegardé dans un répertoire centralisé au format CSV. Cette étape garantit une uniformisation des sources de données et facilite les phases suivantes de transformation et de visualisation.

- Bibliothèques: pandas pour manipuler les fichiers

Os : pour naviguer dans les dossiers et fichiers système

- On commence par définir le chemin des fichiers sources et le dossier qui va contenir les fichiers générés dans cette phase
- Création de dossier de sortie
- Fonction `extract_excel()` : traite les fichiers xlsx, charge chaque fichier dans un data frame pandas
- Fonction `extract_sql_csv()` : traite les fichiers sql sous format csv, stocke les DataFrames dans un dictionnaire
- Fusionne les deux dictionnaires, puis sauvegarde chaque data frame dans le dossier processed

**Transform.py** : Ce script consolide toutes les tables Northwind (Access + SQL), nettoie les clés, corrige les problèmes de dates, et produit une table finale BI prête pour le dashboard.

-Bibliothèques: pandas pour manipuler les fichiers

Os : pour naviguer dans les dossiers et fichiers système

- Configuration des chemins (path) : lis les données déjà extraites, écrit une table finale propre, et crée le dossier final dans data pour sauvegarder les fichiers nécessaires pour le Dashboard
- Charge les données, renomme les colonnes pour éviter les conflits lors des jointures
- Fusionne les commandes sql et access pour avoir une seule table orders et crée un identifiant technique unique
- Nettoie les clés de jointures pour ne pas perdre des données
- Jointures des tables et normalisations des noms de colonnes
- Gestion des dates et donne un petit rapport

**Load.py** : L'étape de chargement consiste à vérifier l'intégrité des données finales, à valider les indicateurs clés (nombre de commandes, périodes couvertes) et à sauvegarder la table BI dans un format Parquet optimisé pour les performances du dashboard Streamlit.

- Définit les chemins des fichiers sources et des fichiers output
- Vérifie l'existence des fichiers sources
- Charge les données, convertit les dates et sauvegarde au format parquet

**Dashboard.py** : Le dashboard interactif développé avec Streamlit permet d'analyser les données Northwind à travers des indicateurs clés (nombre de commandes, taux de livraison, chiffre d'affaires) et des visualisations dynamiques. L'utilisation du format Parquet et du cache Streamlit garantit des performances optimales, tandis que les filtres interactifs offrent une analyse multi-dimensionnelle facilitant la prise de décision.

- Configuration de la page streamlit où il définit le titre du dashboard et la mise en page
- Création des dimensions temporelles et calcul des KPI globaux (total commandes, commandes livrées, commandes non livrées, taux de livraison)
- Les sliders (filtres interactifs) : années, employés, clients et affichage des KPI

- Affichage des KPI et les cercles relatifs des commandes livrées et non livrées par région
- Analyse 3D avancée client , employé, période (x : mois, y : employé, z : client)
- Affichage du registre complet

```

1 import pandas as pd
2 import os
3
4 # Configuration des Chemins
5
6 PROCESSED_PATH = "data/processed"
7 FINAL_PATH = "data/final"
8 RAW_EXCEL_PATH = "data/raw/excel"
9 os.makedirs(FINAL_PATH, exist_ok=True)
10
11 # Chargement des données
12
13 print("🔍 Chargement des fichiers...")
14 customers = pd.read_csv(F"{PROCESSED_PATH}/Customers.csv")
15 orders_sql = pd.read_csv(F"{PROCESSED_PATH}/Orders.csv")
16 order_details = pd.read_csv(F"{PROCESSED_PATH}/Order_Details.csv")
17 products = pd.read_csv(F"{PROCESSED_PATH}/Products.csv")
18
19 # Chargement Excel avec renommage immédiat pour éviter les conflits
20 orders_excel = pd.read_excel(F"{RAW_EXCEL_PATH}/orders.xlsx")
21 orders_excel = orders_excel.rename(columns={
22     'Order ID': 'OrderID',
23     'Order Date': 'OrderDate',
24     'Shipped Date': 'ShippedDate',
25     'Customer': 'CustomerID',
26     'Employee': 'EmployeeID'
27 })
28
29 # Fusion des deux sources de commandes
30 orders = pd.concat([orders_sql, orders_excel], ignore_index=True)
31 orders['unique_row_id'] = range(len(orders)) # identifiant unique pour les 878 lignes
32 initial_count = len(orders)
33
34 # Nettoyage des clés (Jointures)
35
36 def clean_keys(column):
37     return column.astype(str).str.replace(r'\.0$', '', regex=True).str.strip().replace('nan', pd.NA)
38
39 for d in [orders, customers, order_details, products]:
40     for col in ['CustomerID', 'OrderID', 'ProductID']:
41         if col in d.columns:
42             d[col] = clean_keys(d[col])

```

**Schema.sql** : contient le script a exécuter dans sql management studio pour créer le datawarehouse (Northwind\_DWH)

```

1 CREATE DATABASE Northwind_DWH;
2 GO
3
4 USE Northwind_DWH;
5 GO
6
7 -- La table sera créée automatiquement par Python,

```

**to\_sql.py** : ce script fait le transfert des fichiers pandas vers sqlserver

- Bibliothèques : pandas , pyodbc pour se connecter a SQL SERVER via ODBC

- Numpy pour gérer les types numériques et les valeurs manquantes.
- Charge et nettoie le fichier parquet qui contient les données finales
- Normalisation des colonnes :
- Les colonnes numériques remplacent NAN par NONE
- Les colonnes datetime utilisent DATETIME2 et les dates hors limites sont remplacées par None.
- Les colonnes texte sont converties en str et les NaN en None
- Connection a sqlserver : Connexion à la base **Northwind\_DWH** sur SQL Server local
- Création dynamique de la table
- Insertion des données , vérification finale et fermeture de connexion a la base

**Dossier reports** : contient ce document un rapport qui explique étape par étape la conception et la réalisation du projet

**Dossier figures** : contient les captures des résultats finaux

**Dossier vidéo** : contient la vidéo explicative

**Justification des choix :**

**Streamlit** : - Streamlit est utilisé pour créer et déployer rapidement un dashboard BI interactif directement en Python, sans avoir besoin de technologies web complexes (HTML, CSS, JavaScript).

- Intégration naturelle avec Pandas :Streamlit manipule directement des DataFrame
- Facilite l'analyse exploratoire
- Cache intégré (@st.cache\_data)
- Chargement rapide des données Parquet

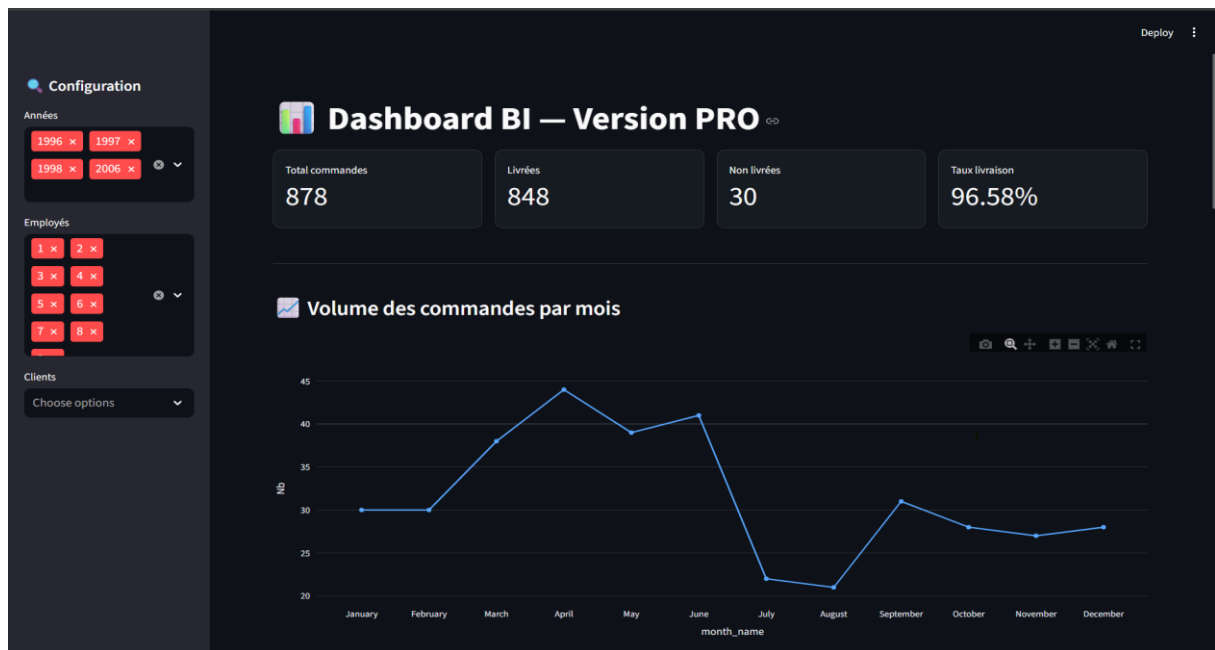
**Plotly.express** : Simplicité d'utilisation

- Une seule ligne pour créer un graphique complexe
- Syntaxe claire et lisible
- Visualisations riches
- Graphiques linéaires
- Barres
- Cercles relatifs
- Scatter 3D
- Intégration parfaite avec Streamlit ( Pas de configuration complexe)

Streamlit a été choisi pour sa simplicité, sa rapidité de développement et son intégration native avec Pandas, permettant de créer des dashboards interactifs en Python sans complexité web. Plotly Express a été utilisé pour ses capacités de visualisation interactive, sa

facilité d'utilisation et son excellente compatibilité avec Streamlit, offrant des graphiques dynamiques adaptés à l'analyse décisionnelle.

Quelques captures du résultats final :



## Conclusion :

Les étapes clés réalisées sont :

## Extraction des données

- Extract\_sqlserver.py automatise l'extraction des données depuis une base SQL Server vers des fichiers CSV.  
Il constitue la première étape d'un pipeline de données (ETL), facilitant l'analyse, le traitement ou le chargement des données dans un autre système
- Chargement des fichiers Access Northwind et CSV SQL
- Uniformisation et stockage dans un répertoire `processed/`

## Transformation

- Nettoyage des clés et normalisation des colonnes
- Fusion des sources et gestion des doublons
- Calcul des KPI principaux : nombre de commandes, taux de livraison
- Gestion des dates et des anomalies

## Chargement et optimisation

- Validation des données finales
- Export en formats CSV et Parquet pour un accès rapide et performant
- Garantir l'intégrité des données pour le dashboard

## Visualisation et dashboard BI

- Développement d'un dashboard interactif avec **Streamlit**
- Visualisations dynamiques avec **Plotly Express** (linéaire, donuts, 3D)
- Filtres par année, employé et client pour l'analyse décisionnelle

## Transfert et préparation des données dans SQL Server

- Nettoyage et normalisation des données avec Pandas (gestion des doublons, valeurs manquantes et dates hors limites)
- Création dynamique de la table Fact\_Orders avec types SQL adaptés
- Insertion des 878 commandes avec pyodbc et fast\_executemany pour optimiser les performances
- Vérification finale du nombre de lignes pour garantir l'intégrité des données