



UNIVERSITE DES SCIENCES ET TECHNOLOGIES HOUARI BOUMEDIENNE

Rapport de réalisation du projet BI

Année 2025/2026

Étudiante : Boukhedouni Nada Raiham

Matricule : 232332239314

ING 3 Cyber Sécurité

The screenshot shows a user interface for a Business Intelligence application. On the left, there is a dark sidebar with several sections:

- Configuration**: A section for selecting years, with buttons for 1996, 1997, 1998, 2006, and a dropdown arrow.
- Années**: Buttons for 1996, 1997, 1998, 2006, and a dropdown arrow.
- Employés**: Buttons for 1x, 2x, 3x, 4x, 5x, 6x, 7x, 8x, and a dropdown arrow.
- Clients**: A "Choose options" dropdown.

The main area is titled **Registre complet** (Complete Register) and displays a table with 10 rows of data. The table has the following columns:

	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 :

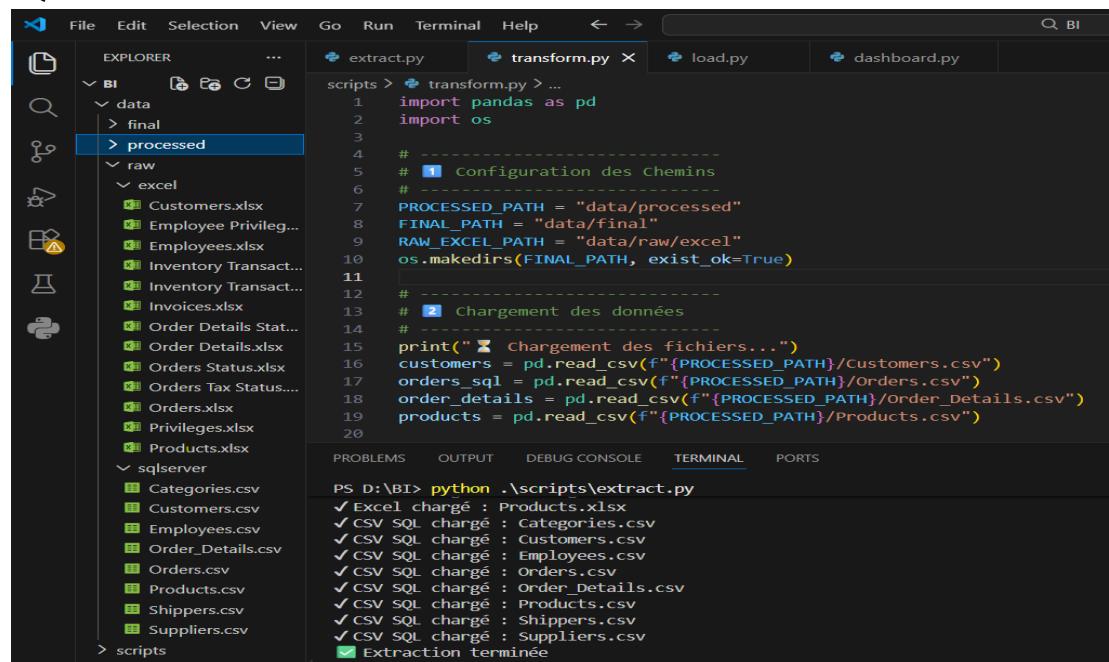
- Scripts Python (.py) pour ETL et visualisation.
- Dashboard interactif avec Streamlit.
- Rapport détaillé (ce document).
- Vidéo présentant le projet de A à Z.

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 a code editor interface with several tabs open. The left sidebar displays a file tree for a project named 'BI'. The 'scripts' folder contains four files: 'extract.py', 'transform.py', 'load.py', and 'dashboard.py'. The 'transform.py' file is currently selected and its content is visible in the main editor area:

```
scripts > extract.py transform.py X load.py dashboard.py
scripts > transform.py ...
1 import pandas as pd
2 import os
3
4 # -----
5 # 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 # 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")
```

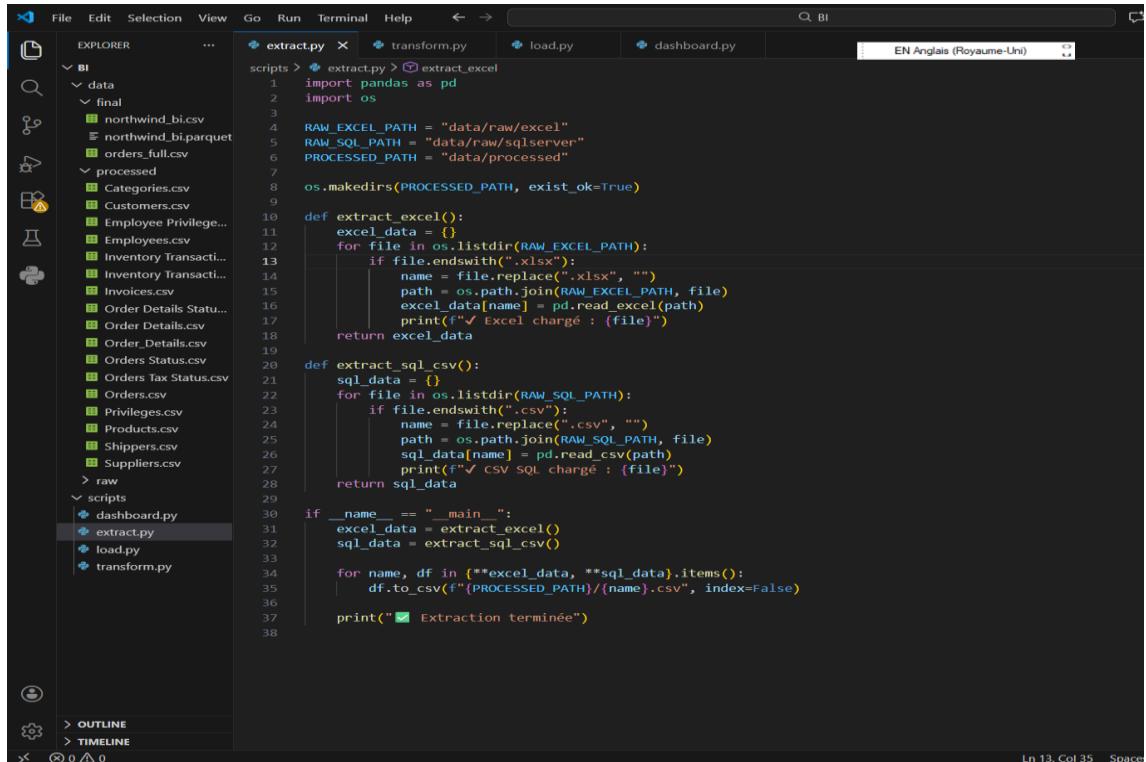
Below the editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab shows the command 'python .\scripts\extract.py' being run, followed by a series of checkmarks indicating successful file processing:

```
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



The screenshot shows a Jupyter Notebook interface with several open files:

- extract.py**: Contains code for extracting data from Excel and SQL Server.
- transform.py**: Contains code for transforming the extracted data.
- load.py**: Contains code for loading the transformed data.
- dashboard.py**: Contains code for generating a dashboard.

The **extract.py** file contains the following code:

```
scripts > extract.py > extract.py > extract_excel
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
- Définition de chemin de sortie data/raw/sqlserver/
- Fonction extract_from_sqlserver() : connection à sql server
- Liste des tables à 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 fichier sources et le dossier qui va contenir les fichiers générés dans cette phase
- Création de dossier de sortie
- Fonction extract_excel() : traite les fichier xlsx , charge chaque fichier dans un data frame pandas
- Fonction extract_sql_csv() : traite les fichier sql sous format csv, stocke les DataFrames dans un dictionnaire
- Fusionne les deux dictionnaire , 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 créé 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.

- Définie les chemins des fichier sources et des fichiers output
- Vérifie l'existence des fichier sources
- Charge les donnes , convertie 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 ou il définit le titre du dashboard et la mise en page
- Crédit des dimensions temporelles et calcul des KPI globaux (total commandes, commandes livrées , commandes non livrées, taux de livraison)
- Les slidebars(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

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Back, Forward, Search (Q), Refresh, Help (BI).
- EXPLORER:** Shows a tree view of files and folders:
 - BI
 - data
 - figures
 - notebooks
 - reports
 - scripts
 - dashboard.py
 - extract.py
 - load.py
 - transform.py
 - video
- Code Cell:** The current cell contains Python code for data processing, specifically for extracting, transforming, and loading data from CSV and Excel files. The code includes imports for pandas and os, configuration of paths, data loading, merging of datasets, and cleaning of keys.
- Status Bar:** EN Anglais (Royaume-Uni), Ln 11, Col 1, Spc.

Schema.sql : contient le script à exécuter dans SQL Management Studio pour créer le datawarehouse (Northwind_DWH)

The screenshot shows the SSMS interface with the following details:

- Toolbar:** Fichier, Edition, Afficher, Requête, Git, Projet, Outils, Extensions, Fenêtre, Aide, Rechercher.
- Object Explorer:** Shows the connection to "localhost\SQLExpress (SQL Server 17.0.1000 - NADA-DRAGONFLY\hp)". The tree view includes nodes for Bases de données, Bases de données système, Instantanés de base de données, Northwind, and Northwind_DWH. Under Northwind_DWH, there are Tables, Views, Ressources externes, Synonymes, Programmabilité, Magasin des requêtes, Service Broker, Stockage, and Sécurité.
- SQL Query Editor:** The current query is "SQLQuery1.s...LYIhp (70)*". The code is as follows:

```
CREATE DATABASE Northwind_DWH;
GO

USE Northwind_DWH;
GO

-- La table sera créée automatiquement par Python,
```
- Status Bar:** Aucun problème détecté.
- Message Bar:** Messages: Commandes réussies.
- Bottom Status:** Heure de fin : 2025-12-19T11:13:18.5900747+01:00

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

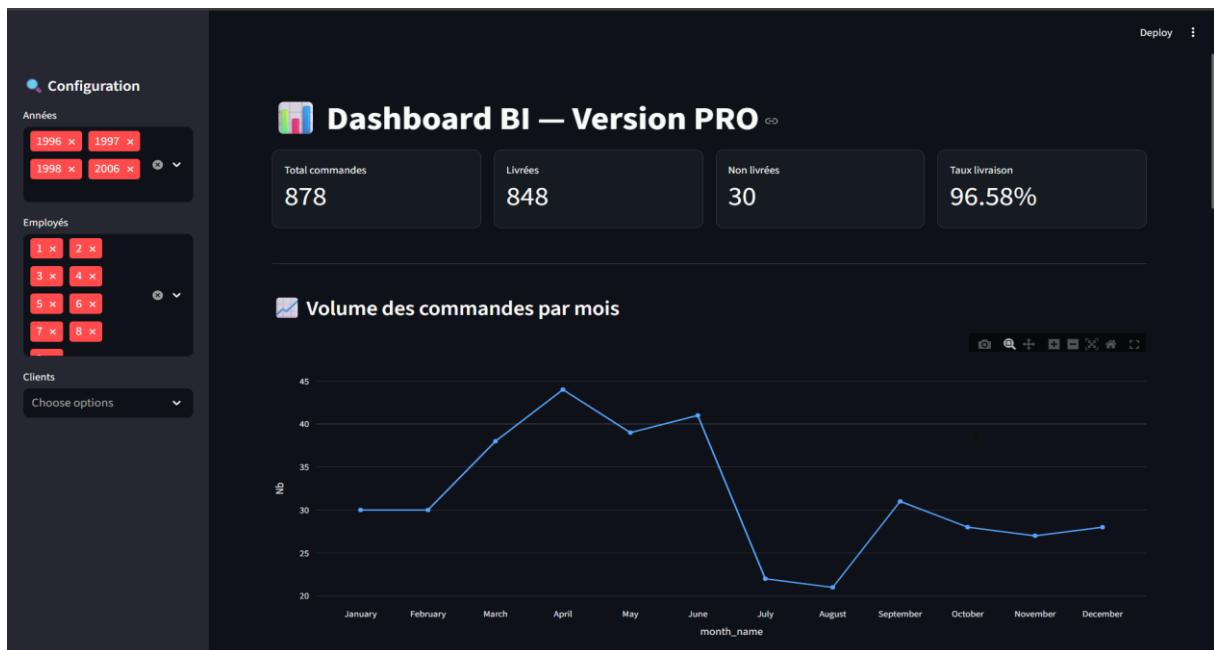
Plotly.express : Simplicité d'utilisation

- 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