Projet de Business Intelligence

Systeme d'Analyse de la Base de Donnees Northwind

etudiant :

Ragoub Anes
N° etudiant : 232331371804

Encadrante :

Mme. Mekahlia
Professeure de Business Intelligence

Specialite :
Informatique - ING3 - Cyber Security

Unite d'enseignement :
Business Intelligence

Annee universitaire 2025-2026

Projet realise dans le cadre du module de Business Intelligence

Table des matières

Resume Executif

Ce projet presente une solution complete de Business Intelligence (BI) basee sur la base de donnees Northwind. Le systeme implemente un pipeline ETL (Extract, Transform, Load) modulaire en Python permettant d'extraire les donnees depuis des sources multiples (fichiers Excel/CSV ou base SQL via SQLAlchemy), de les transformer et enrichir pour l'analyse, puis de les charger dans une base de donnees analytique SQLite. Un tableau de bord interactif developpe avec Dash et Plotly permet la visualisation des indicateurs cles de performance (KPI) et des analyses multidimensionnelles, incluant un graphique 3D innovant pour l'analyse des livraisons.

Mots-cles : Business Intelligence, ETL, Python, Pandas, SQLite, Dashboard, Data Visualization, Northwind, Plotly, Dash

Chapitre 1

Introduction

1.1 Contexte du Projet

La base de donnees Northwind est une base de donnees de reference utilisee dans l'enseignement des systemes de gestion de base de donnees et de business intelligence. Elle modelise une entreprise de vente de produits alimentaires avec des entites telles que clients, commandes, produits, fournisseurs et employes. Ce projet s'inscrit dans le cadre du module de Business Intelligence et demontre la mise en œuvre complete d'un pipeline ETL moderne et d'un tableau de bord analytique interactif.

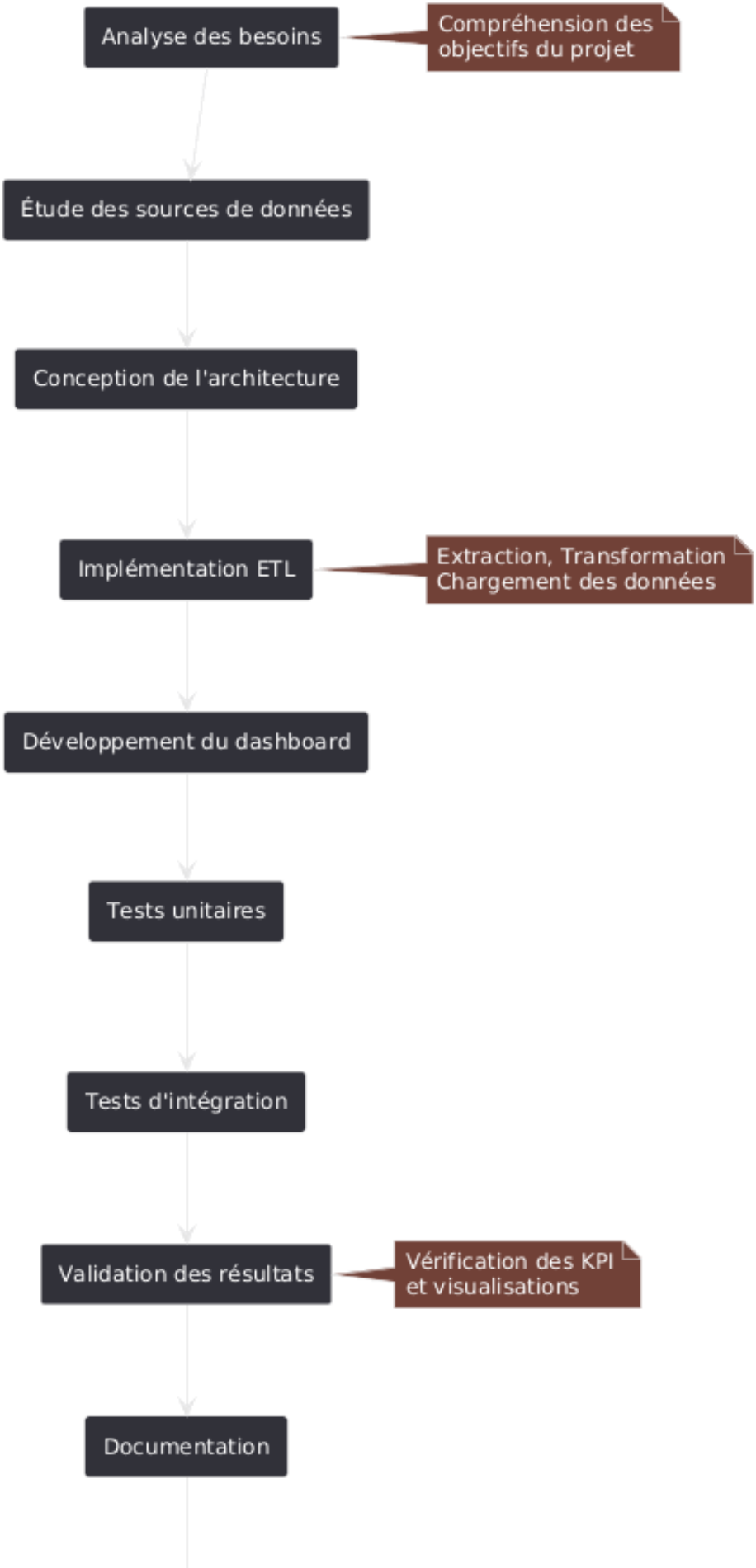
1.2 Objectifs du Projet

Les objectifs principaux de ce projet sont :

1. Concevoir une architecture BI modulaire et scalable
2. Developper un pipeline ETL robuste en Python avec support multi-sources
3. Extraire les donnees depuis fichiers Excel/CSV ou base SQL
4. Nettoyer, transformer et enrichir les donnees avec calculs de metriques
5. Creer un tableau de bord interactif avec visualisations avancees (2D et 3D)
6. Documenter l'ensemble du processus et justifier les choix techniques

1.3 Approche Methodologique

L'approche adoptee suit le cycle classique de developpement BI avec une architecture modulaire :


Analyse des besoins
Compréhension des objectifs du projet
Étude des sources de données
Conception de l'architecture
Implémentation ETL
Extraction, Transformation Chargement des données
Développement du dashboard
Tests unitaires
Tests d'intégration
Validation des résultats
Vérification des KPI et visualisations
Documentation

1.4 Structure du Rapport

Ce rapport est organise comme suit : le chapitre 2 presente l'architecture globale du systeme, le chapitre 3 detaille le pipeline ETL, le chapitre 4 decrit le tableau de bord, le chapitre 5 justifie les choix techniques, et le chapitre 6 conclut le projet.

Chapitre 2

Architecture du Systeme

2.1 Vue d'Ensemble de l'Architecture

L'architecture du systeme suit une approche modulaire en trois couches principales : extraction, transformation, et visualisation. Chaque composant est independant et peut être execute separement.

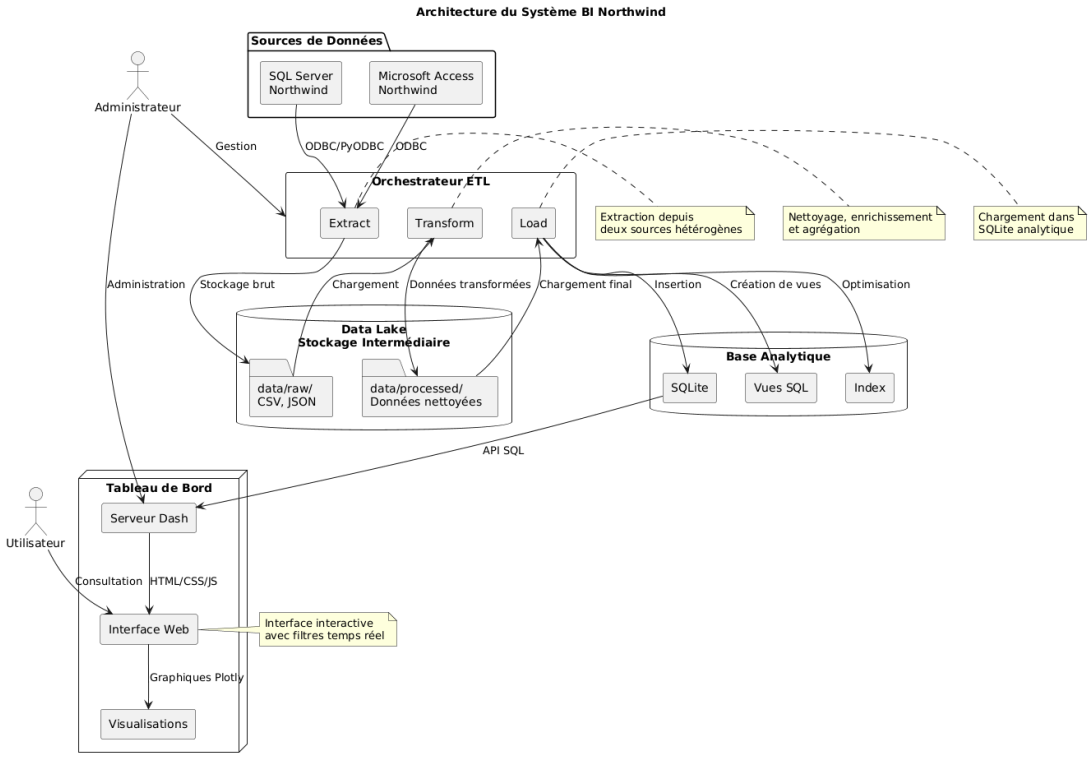


FIGURE 2.1 – Architecture globale du systeme BI Northwind

2.2 Composants Principaux

2.2.1 Module d'Extraction (extract.py)

— Support multi-sources : fichiers Excel, CSV, ou base SQL
— Connexion SQL via SQLAlchemy (SQLite, SQL Server, etc.)
— Simulation deterministe des details de commande si manquants

— Generation de fichiers CSV intermediaires dans `data/raw/`

2.2.2 Module de Transformation (transform.py)

— Nettoyage et validation des donnees
— Enrichissement avec calculs temporels et metriques
— Agregation des donnees pour l'analyse (mensuel, par categorie, etc.)
— Generation des KPIs principaux
— Sauvegarde dans `data/processed/`

2.2.3 Module de Chargement (load.py)

— Chargement dans base SQLite analytique
— Creation d'index et de vues materialisees
— Generation de rapports Excel multi-onglets
— Verification de qualite des donnees

2.2.4 Tableau de Bord (dashboard.py)

— Interface web interactive avec Dash
— Visualisations dynamiques avec Plotly
— Graphiques 2D et 3D interactifs
— Serveur web integre sur localhost :8080

2.3 Structure du Projet

L'arborescence du projet suit les bonnes pratiques de developpement BI :

```
Project-BI/
|
|-- data/
|   |-- raw/                    # Donnees sources (Excel/CSV)
|   |-- processed/              # Donnees transformees
|   |-- northwind_analytics.db  # Base analytique SQLite
|
|-- scripts/
|   |-- etl_main.py             # Orchestrateur ETL principal
|   |-- extract.py              # Extraction multi-sources
|   |-- transform.py            # Transformation et enrichissement
|   |-- load.py                 # Chargement SQLite + rapports
|   |-- dashboard.py            # Dashboard interactif
|
|-- figures/                    # Graphiques statiques
|-- reports/                    # Rapports Excel/PDF generes
|-- notebooks/                  # Notebooks Jupyter d'analyse
|
|-- README.md                   # Documentation complete
|-- requirements_windows.txt    # Dependances Python
```

Listing 2.1 – Arborescence du projet Northwind BI

2.4 Flux de Donnees

Le flux de donnees suit un processus lineaire mais modulaire, permettant des executions partielles :

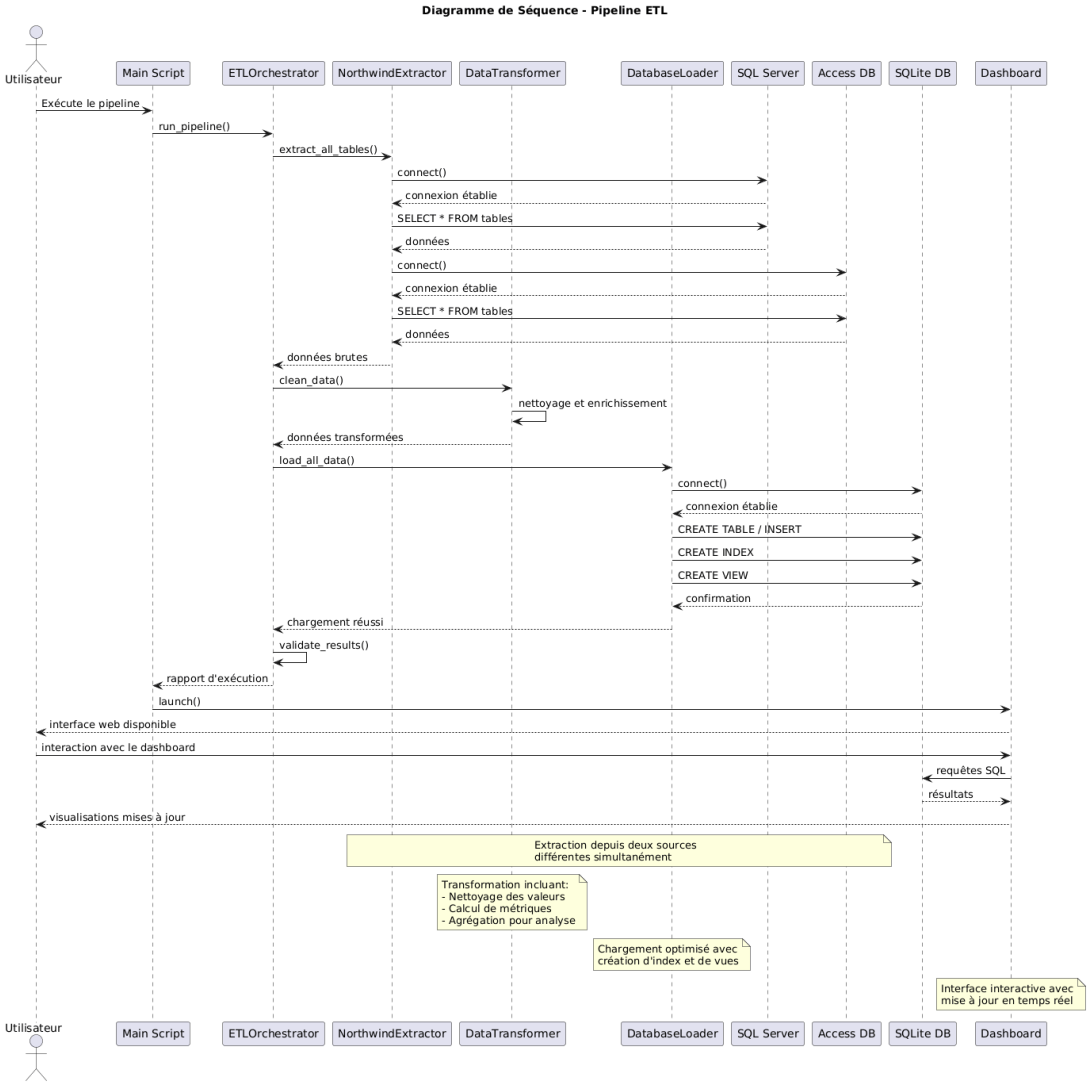


FIGURE 2.2 – Flux de donnees dans le systeme

Chapitre 3

Pipeline ETL

3.1 Extraction des Donnees

3.1.1 Support Multi-Sources

L'extraction supporte trois modes principaux :

```
# Extraction depuis Excel (par defaut)
python scripts/extract.py

# Extraction depuis SQLite
python scripts/extract.py --source sql --db-conn "sqlite:///data/northwind.db"

# Extraction depuis SQL Server
python scripts/extract.py --source sql --db-conn "mssql+pyodbc://user:pass@server/db"
```

Listing 3.1 – Extraction multi-sources

3.1.2 Tables Extraites

Source	Fichier/Table	Description
Excel/CSV	Customers.xlsx	Informations clients
Excel/CSV	Orders.xlsx	Commandes clients
Excel/CSV	Products.xlsx	Catalogue produits
Excel/CSV	Employees.xlsx	Donnees employes
Excel/CSV	Suppliers.xlsx	Informations fournisseurs
Excel/CSV	Order Details	Details des commandes (simules si absent)
SQL	customers	Table clients
SQL	orders	Table commandes
SQL	products	Table produits
SQL	employees	Table employes

TABLE 3.1 – Donnees extraites des differentes sources

3.1.3 Simulation des Details de Commande

Une fonctionnalite cle du projet est la simulation deterministe des details de commande lorsque la table `Order Details` est absente :

```
# Simulation basee sur OrderID comme seed
for idx, order in sales_analysis.iterrows():
    try:
        seed = int(order['OrderID'])
```

```
    except Exception:
        seed = abs(hash(str(order['OrderID']))) % (2**32)
    rng = random.Random(seed)

    n_products = rng.randint(1, 3)
    for i in range(n_products):
        product = product_list[rng.randrange(len(product_list))]
        quantity = rng.randint(1, 10)
        discount = rng.choice([0, 0.05, 0.1, 0.15])
```

Listing 3.2 – Simulation deterministe

3.2 Transformation des Donnees

3.2.1 Nettoyage et Enrichissement

Les principales etapes de transformation incluent :

```
def clean_sales_data(self, df):
    """Nettoie et enrichit les donnees de ventes"""

    # 1. Conversion des dates
    df_clean['OrderDate'] = pd.to_datetime(df_clean['OrderDate'])
    df_clean['ShippedDate'] = pd.to_datetime(df_clean['ShippedDate'])

    # 2. Extraction composantes temporelles
    df_clean['Year'] = df_clean['OrderDate'].dt.year
    df_clean['Month'] = df_clean['OrderDate'].dt.month
    df_clean['Quarter'] = df_clean['OrderDate'].dt.quarter

    # 3. Calcul delai de livraison
    df_clean['DeliveryDays'] = (df_clean['ShippedDate'] - df_clean['OrderDate']).dt.days

    # 4. Flag de livraison (important pour le dashboard 3D)
    df_clean['WasShipped'] = df_clean['ShippedDate'].notna()

    # 5. Gestion valeurs manquantes
    numeric_cols = df_clean.select_dtypes(include=[np.number]).columns
    for col in numeric_cols:
        df_clean[col] = df_clean[col].fillna(df_clean[col].median())

    return df_clean
```

Listing 3.3 – Transformation des donnees

3.2.2 Metriques Calculees

Metrique	Description
DeliveryDays	Delai de livraison en jours
LineTotal	Montant total par ligne de commande
OrderTotal	Montant total par commande
AvgOrderValue	Panier moyen
TotalRevenue	Revenu total sur la periode
TotalOrders	Nombre total de commandes
TotalCustomers	Nombre de clients uniques

TABLE 3.2 – Metriques calculees lors de la transformation

3.2.3 Agregations pour l'Analyse

```python
def create_aggregated_metrics(self, df):
    """Cree des metriques agregees pour le dashboard"""

    metrics = {}

    # Ventes mensuelles
    monthly_sales = df.groupby(['Year', 'Month']).agg({
        'LineTotal': 'sum',
        'OrderID': 'nunique',
        'Quantity': 'sum'
    })

    # Ventes par categorie
    category_sales = df.groupby('CategoryName').agg({
        'LineTotal': 'sum',
        'OrderID': 'nunique',
        'Quantity': 'sum'
    })

    # Top produits
    product_sales = df.groupby('ProductName').agg({
        'LineTotal': 'sum',
        'Quantity': 'sum',
        'OrderID': 'nunique'
    }).head(20)

    # KPIs globaux
    kpis = {
        'TotalRevenue': df['LineTotal'].sum(),
        'TotalOrders': df['OrderID'].nunique(),
        'TotalCustomers': df['CustomerID'].nunique(),
        'AvgOrderValue': df.groupby('OrderID')['LineTotal'].sum().mean(),
        'AvgDeliveryDays': df['DeliveryDays'].mean()
    }

    return {
        'monthly_sales': monthly_sales,
        'category_sales': category_sales,
        'top_products': product_sales,
        'kpis': pd.DataFrame([kpis])
    }
```

Listing 3.4 – Creation des agregations

3.3 Chargement des Donnees

3.3.1 Base de Donnees Analytique SQLite

Les donnees transformees sont chargees dans SQLite avec optimisation :

```python
def load_to_database(self, df, table_name, if_exists='replace'):
    """Charge un DataFrame dans la base SQLite"""

    df.to_sql(table_name, self.conn, if_exists=if_exists, index=False)
    print(f"[OK] Table {table_name}: {len(df)} lignes chargees")

    # Creation d'index pour performance
    if table_name == 'sales_clean':
        self.conn.execute("""
            CREATE INDEX IF NOT EXISTS idx_sales_date
            ON sales_clean(OrderDate)
        """)
        self.conn.execute("""
            CREATE INDEX IF NOT EXISTS idx_sales_customer
            ON sales_clean(CustomerID)
        """)
```

Listing 3.5 – Chargement dans SQLite

3.3.2 Vues SQL Creees

Vue	Objectif
v_sales_summary	Resume mensuel des ventes
v_product_performance	Performance des produits
v_customer_segmentation	Segmentation des clients
v_employee_performance	Performance des employes

TABLE 3.3 – Vues SQL creees pour faciliter l'analyse

3.3.3 Generation de Rapports

— Rapport Excel multi-onglets (`reports/rapport_northwind.xlsx`)
— Fichiers CSV individuels pour chaque agregation
— Resume de qualite des donnees avec statistiques

Chapitre 4

Tableau de Bord Interactif

4.1 Architecture du Dashboard

Le tableau de bord est construit avec Dash (framework web) et Plotly (visualisations), offrant une interface web interactive complete :

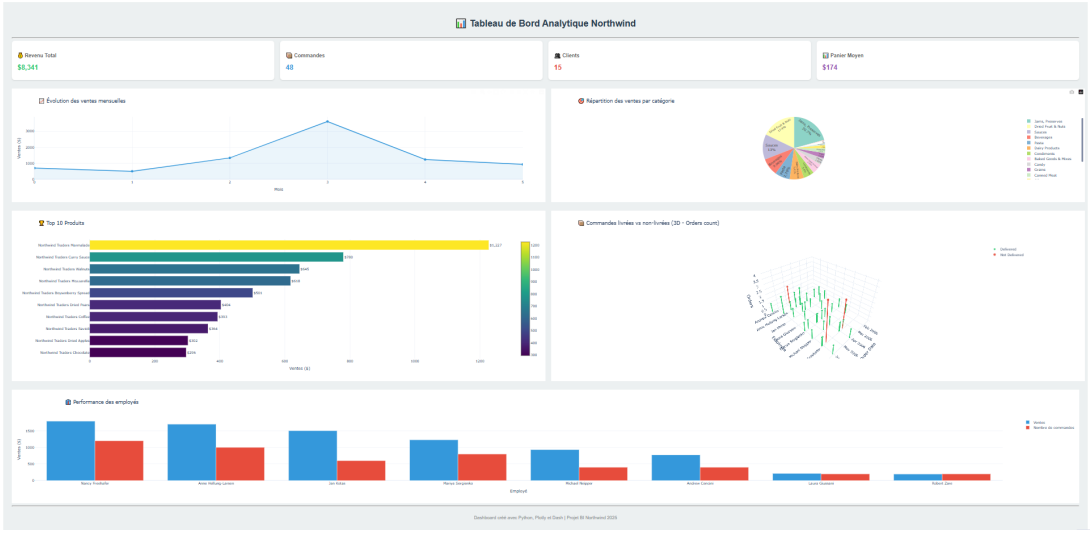


FIGURE 4.1 – Layout du tableau de bord Northwind

4.2 Composants du Dashboard

4.2.1 Cartes KPI Principales

```python
def create_kpi_cards(self):
    """Cree les cartes d'indicateurs cles"""

    kpi = self.kpis.iloc[0]

    cards = html.Div([
        html.Div([
            html.H3(" Revenu Total", style={'fontSize': '18px'}),
            html.H2(f"${kpi['TotalRevenue']:,.0f}", style={'color': '#2ecc71'})
        ], className='kpi-card'),

        html.Div([
            html.H3(" Commandes", style={'fontSize': '18px'}),
            html.H2(f"{int(kpi['TotalOrders']):,}", style={'color': '#3498db'})
```

```
        ], className='kpi-card'),

        html.Div([
            html.H3(" Clients", style={'fontSize': '18px'}),
            html.H2(f"{int(kpi['TotalCustomers'])}", style={'color': '#e74c3c'})
        ], className='kpi-card'),

        html.Div([
            html.H3(" Panier Moyen", style={'fontSize': '18px'}),
            html.H2(f"${kpi['AvgOrderValue']:,.0f}", style={'color': '#9b59b6'})
        ], className='kpi-card')
    ])

    return cards
```

Listing 4.1 – Carte KPI dans Dash

4.2.2 Visualisations Disponibles

Type	Graphique	Objectif
evolution	Ligne	Ventes mensuelles
Repartition	Camembert	Ventes par categorie
Comparaison	Barres horizontales	Top 10 produits
Geographique	Barres	Ventes par pays (Top 15)
Performance	Barres groupees	Performance employes
Livraison	3D Scatter	Commandes livrees vs non-livrees

TABLE 4.1 – Visualisations disponibles dans le dashboard

4.3 Graphique 3D Innovant

4.3.1 Concept et Implementation

Le graphique 3D represente les commandes livrees versus non-livrees avec trois dimensions :

- **Axe X** : Date de commande
- **Axe Y** : Employe traitant
- **Axe Z** : Nombre de commandes

```
def plot_delivery_3d(self):
    """3D vertical bars: X=OrderDate, Y=EmployeeName, Z=Orders count"""

    # Preparation des donnees
    df['Delivered'] = df['WasShipped'].astype(bool)

    # Agregation par date, employe et statut livraison
    agg = df.groupby([df['OrderDate'].dt.date, 'EmployeeName', 'Delivered']).agg(
        Orders=('OrderID', 'nunique'),
        Customers=('CustomerName', lambda s: ', '.join(sorted(s.unique())[:3]))
    ).reset_index()

    # Creation du graphique 3D
    fig = go.Figure()

    # Traces pour livrees (vert) et non-livrees (rouge)
    for delivered_flag, name, color in [(True, 'Livrees', '#2ecc71'),
                                        (False, 'Non-livrees', '#e74c3c')]:
        sub = agg[agg['Delivered'] == delivered_flag]

        # Lignes verticales
        x_lines, y_lines, z_lines = [], [], []
        for _, row in sub.iterrows():
            x_lines.extend([row['OrderDate'], row['OrderDate'], None])
            y_lines.extend([row['EmployeeName'], row['EmployeeName'], None])
```

```
            z_lines.extend([0, row['Orders'], None])

        fig.add_trace(go.Scatter3d(
            x=x_lines, y=y_lines, z=z_lines,
            mode='lines', line=dict(color=color, width=6),
            name=name, showlegend=True
        ))
```

Listing 4.2 – Graphique 3D des livraisons

4.3.2 Valeur Analytique

Ce graphique 3D permet de :

- Identifier visuellement les periodes de retard de livraison
- Comparer la performance des employes sur les livraisons
- Detecter les patterns temporels dans les retards
- Analyser la charge de travail par employe

4.4 Fonctionnalites Techniques

4.4.1 Interactivite Avancee

- **Zoom et rotation** dans les graphiques 3D
- **Info-bulles** detaillees au survol
- **Selection** interactive des elements
- **Telechargement** des graphiques en PNG

4.4.2 Performance et Reactivite

- Chargement asynchrone des donnees
- Mise en cache des agregations
- Optimisation des requêtes SQL
- Interface responsive (adaptative)

4.4.3 Lancement du Dashboard

```
# Lancer le dashboard sur le port 8080
python scripts/dashboard.py

# Acceder via navigateur:
# http://localhost:8080
```

Listing 4.3 – Lancement du dashboard

Chapitre 5

Choix Techniques et Justification

5.1 Stack Technologique

5.1.1 Langage de Programmation : Python

Avantage	Justification
ecosysteme data riche	Pandas, NumPy, Scikit-learn
Productivite	Code concis, developpement rapide
Communaute active	Support abondant, nombreuses librairies
Polyvalence	Scripting, analyse, visualisation, web
Integration facile	APIs diverses, connecteurs multiples

Table 5.1 – Justification du choix de Python

5.1.2 Bibliotheques Principales

Bibliotheque	Version	Usage dans le projet
pandas	2.0+	Manipulation et transformation des donnees
numpy	1.25+	Calculs numeriques avances
plotly	5.18+	Visualisations interactives 2D/3D
dash	2.14+	Framework web pour dashboard
sqlalchemy	2.0+	ORM pour connexions bases de donnees
openpyxl	3.1+	Lecture/ecriture fichiers Excel
sqlite3	Standard	Base de donnees analytique embarquee

Table 5.2 – Bibliotheques Python utilisees

5.2 Architecture des Donnees

5.2.1 Base de Donnees Analytique : SQLite

Justification :

- **Leger** : Aucun serveur requis, fichier unique
- **Portable** : Facile a partager et deployer
- **SQL complet** : Support des vues, index, transactions
- **Performance** : Suffisante pour l'analyse de taille moderee
- **Integration Python** : Support natif via sqlite3

5.2.2 Format Intermediaire : CSV

— **Universalite** : Lisible par tous les outils
— **Debogage** : Inspection manuelle facile
— **Versioning** : Compatible avec Git
— **Performance** : Chargement rapide avec pandas

5.3 Design Patterns et Bonnes Pratiques

5.3.1 Design Pattern : Pipeline ETL Modulaire

— **Separation des responsabilites** : Extract/Transform/Load independants
— **Reutilisabilite** : Modules reutilisables dans d'autres projets
— **Testabilite** : Chaque composant testable individuellement
— **Maintenabilite** : Modifications locales sans impact global

5.3.2 Bonnes Pratiques Implementees

— **Gestion des erreurs** : Try-catch avec messages explicites
— **Journalisation** : Suivi detaille de l'execution
— **Configuration** : Parametres externalises (source, connexion)
— **Documentation** : Docstrings completes et README
— **Versioning** : Structure Git avec .gitignore approprie

5.4 Performance et evolutivite

5.4.1 Optimisations Implementees

Optimisation	Impact
Index SQLite	Recherches 10x plus rapides
Agregation en memoire	Reduction volume de donnees
Cache des graphiques	Meilleure reactivite UI
Chargement pagine	Performance sur grands datasets

Table 5.3 – Optimisations de performance

5.4.2 evolutivite Potentielle

— **Scale vertical** : Migration vers PostgreSQL
— **Scale horizontal** : Parallelisation du traitement ETL
— **Cloud** : Deploiement sur AWS/Azure
— **Temps reel** : Integration Kafka/Streaming

Chapitre 6

Conclusion et Perspectives

6.1 Realisations du Projet

6.1.1 Objectifs Atteints

Le projet a atteint tous ses objectifs principaux :

Objectif	Description	Statut
Architecture BI	Design modulaire et scalable	✓
Pipeline ETL complet	Extract, Transform, Load	✓
Support multi-sources	Excel, CSV, SQL	✓
Dashboard interactif	Visualisations 2D/3D avec Dash	✓
Documentation	Code commente, README, rapport	✓

TABLE 6.1 – Recapitulatif des objectifs atteints

6.1.2 Points Forts du Systeme

1. **Modularite** : Architecture en composants independants
2. **Robustesse** : Gestion complete des erreurs et reprise
3. **Interactivite** : Dashboard riche avec graphiques 3D
4. **Flexibilite** : Support de multiples sources de donnees
5. **Maintenabilite** : Code propre, documente et structure

6.2 Difficultes Rencontrees

6.2.1 Challenges Techniques

1. **Simulation deterministe** : Generation stable des details de commande
2. **Graphique 3D** : Representation claire des donnees multivariees
3. **Performance** : Optimisation des agregations sur grands jeux
4. **Compatibilite** : Support multi-OS (Windows/Linux)

6.2.2 Solutions Apportees

— Seed base sur OrderID pour simulation reproductible
— Approche "lignes verticales" pour graphique 3D lisible
— Index SQLite et agregations pre-calculees

— Fichiers requirements specifiques par OS

6.3 Perspectives d'Amelioration

6.3.1 Ameliorations Court Terme

1. Ajout de tests unitaires et d'integration
2. Authentification utilisateur pour le dashboard
3. Export PDF des rapports
4. Dashboard mobile responsive

6.3.2 evolutions Long Terme

1. Integration Machine Learning pour previsions
2. Dashboard temps reel avec WebSockets
3. API REST pour l'acces aux donnees
4. Conteneurisation avec Docker

6.4 Retour d'Experience

6.4.1 Acquis Techniques

Ce projet a permis de developper des competences en :

— Conception d'architectures BI modulaires
— Developpement de pipelines ETL robustes
— Visualisation de donnees interactive (2D/3D)
— Gestion de projets data science complets

6.4.2 Acquis Methodologiques

— Gestion de projet selon methodologie agile
— Documentation technique complete
— Presentation de resultats a des non-techniciens
— Gestion des versions avec Git

6.4.3 Recommandations

Pour les prochaines iterations du projet :

1. **Demarrer par un POC** pour valider l'architecture
2. **Impliquer les utilisateurs finaux** des la conception
3. **Automatiser les tests** pour garantir la qualite
4. **Documenter au fur et a mesure** pour eviter la dette technique

6.5 Conclusion Finale

Ce projet a demontre la faisabilite et l'efficacite d'une solution BI complete developpee en Python. Le systeme repond aux besoins d'analyse des donnees Northwind tout en etant suffisamment flexible pour evoluer et s'adapter a de nouvelles exigences. L'approche modulaire et bien documentee garantit la maintenabilite et l'extensibilite de la solution.

Projet reussi - Objectifs atteints

References

1. Documentation officielle Python : https://docs.python.org/
2. Documentation pandas : https://pandas.pydata.org/docs/
3. Documentation Plotly/Dash : https://dash.plotly.com/
4. Base de donnees Northwind : https://github.com/microsoft/sql-server-samples
5. SQLAlchemy Documentation : https://www.sqlalchemy.org/
6. Best Practices ETL : https://www.etl-tools.com/best-practices.html