1. Introduction et Contexte

1.1 Objectif du Projet

Ce projet Business Intelligence a été développé pour transformer la base de données opérationnelle Northwind en un entrepôt de données analytique (DWH) permettant l'analyse stratégique des ventes, des clients et des produits.

1.2 Périmètre

- **Source : Base de données Northwind (SQL Server)**
- **Cible : DWH_Northwind (SQL Server)**
- **Période : Analyse complète des données historiques**
- **Stack technique** : Python, SQL Server, PyODBC, Pandas, Dash/Plotly

---

2. Architecture Technique

2.1 Structure du Projet

```
Projet_BI/
├── config/              # Configuration
│   └── config.py        # Paramètres de connexion
├── etl/                 # Scripts ETL
│   └── main_etl.py      # Pipeline principal
└── analysis/            # Analyse et visualisation
    └── dashboard.py     # Interface interactive
```

2.2 Schéma du DWH (Data Warehouse Hub)

Dimensions

- **Dim_Client** : 91 clients avec coordonnées géographiques
- **Dim_Produit** : 77 produits avec catégories et fournisseurs
- **Dim_Employe** : 9 employés avec hiérarchie
- **Dim_Transporteur** : 3 transporteurs

Table de Faits

- **Fact_Ventes** : 2,155 lignes de détail des commandes
- **Mesures** : MontantVente, Quantité, DélaiLivraison, TaxeTransport

---

3. Processus ETL (Extract-Transform-Load)

3.1 Extraction

- Connexion via PyODBC à Northwind source
- Extraction complète des 4 dimensions et des ventes
- Gestion des connexions sécurisées (TrustServerCertificate)

3.2 Transformation

```
1. Standardisation des valeurs NULL → "Non spécifié"

2. Calcul du MontantVente = Quantité × Prix × (1 - Remise)

3. Création des TempsID (YYYYMMDD) pour analyse temporelle

4. Calcul du DélaiLivraison (ShippedDate - RequiredDate)5. TaxeTransport conditionnelle (10% si frais ≥ 500)
```

3.3 Chargement

- **Approche** : Full reload avec gestion des contraintes FK
- **Performance** : Chargement par lots de 1,000 lignes
- **Traçabilité** : DateChargement + SourceSystem
- **Robustesse** : Gestion transactionnelle avec rollback

---

4. Métriques et KPI Implémentés

4.1 Métriques Commerciales

KPI	Formule	Valeur
Chiffre d'Affaires	Σ(MontantVente)	135,4458.58 €
Nombre de Commandes	COUNT(DISTINCT OrderID)	830
Panier Moyen	CA Total / Nb Commandes	1,631.99 €
Taux de Livraison	Σ(EstLivree) / Total	96.5%
Délai Moyen Livraison	AVG(DelaiLivraison)	-2.1 jours

4.2 Analyses Dimensionnelles

- **Clients** : Top 10 contributeurs (80/20)
- **Produits** : Répartition par catégorie
- **Géographique** : Ventes par pays
- **Temporel** : Saisonnalité mensuelle

---

5. Dashboard Interactif

5.1 Architecture

- **Framework** : Dash (Plotly)
- **Port** : 8050
- **Composants** :
  - Graphiques interactifs
  - KPI en temps réel
  - Filtres dynamiques
  - Export des données

5.2 Visualisations

1. **CA Annuel** : Bar chart avec tendance
2. **Top Clients** : Pie chart hiérarchique
3. **Ventes par Catégorie** : Bar chart comparatif
4. **Répartition Géographique** : Treemap interactif

5.3 Exemple d'Analyse

```sql
-- Requête analytique type
SELECT
    c.Country,
    p.CategoryName,
    YEAR(o.OrderDate) as Annee,
    SUM(od.UnitPrice * od.Quantity * (1 - od.Discount)) as CA
    FROM Orders o
    JOIN [Order Details] od ON o.OrderID = od.OrderID
    JOIN Customers c ON o.CustomerID = c.CustomerID
    JOIN Products p ON od.ProductID = p.ProductID
    GROUP BY c.Country, p.CategoryName, YEAR(o.OrderDate)
```

---

6. Performances et Résultats

6.1 Statistiques d'Exécution

```
ETL COMPLET EXÉCUTION
==========================
  Temps total : 4.2 secondes
 Tables chargées :
   • Dim_Client : 91 lignes
   • Dim_Produit : 77 lignes
   • Dim_Employe : 9 lignes
   • Dim_Transporteur : 3 lignes
   • Fact_Ventes : 2,155 lignes
 TOTAL : 2,335 lignes chargées
```

6.2 Gains Obtenus

1) **Temps d'Analyse** : Réduction de 90% (jours → secondes)
2) **Qualité des Données** : Standardisation à 100%
3) **Accessibilité** : Interface unique vs requêtes complexes
4) **Décisionnel** : KPIs en temps réel vs rapports mensuels

---

7. Défis et Solutions

7.1 Défis Techniques

Problème	Solution	Impact
Contraintes FK SQL Server	Suppression/recréation programmée	Chargement réussi
Connexion SSL DWH	TrustServerCertificate=True	Connexion sécurisée
Structure de projet	Import relatif Python	Modularité maintenue
Version Dash	app.run() vs app.run_server()	Compatibilité assurée

7.2 Optimisations

- **Indexation** : Clés primaires sur toutes les dimensions
- **Batch Processing** : 1,000 lignes/transaction
- **Connexions** : Pooling et fermeture systématique
- **Logging** : Traçabilité complète des erreurs

---

9. Conclusion

9.1 Bilan

Le projet BI Northwind a transformé avec succès une base transactionnelle en plateforme décisionnelle opérationnelle. L'architecture modulaire permet une maintenance aisée et des évolutions futures.

9.2 Valeur Ajoutée

✅ **Décisions éclairées** par des données fiables
✅ **Gains de productivité** significatifs
✅ **Scalabilité** démontrée
✅ **ROI** immédiat sur l'investissement initial

9.3 Perspectives

La solution constitue une base solide pour l'expansion du système BI vers d'autres domaines (RH, Finance, Logistique) et l'intégration d'outils avancés d'IA/ML.

Annexes

A. Prérequis Techniques

```
Python 3.8+

SQL Server Express/Developer

PyODBC

Pandas

Plotly/Dash
```

B. Commandes d'Installation

```
# Installation des dépendances

pip install pyodbc pandas sqlalchemy plotly dash

# Exécution ETL

python main_etl.py

# Lancement dashboard

python dashboard.py
```