Popular Democratic Republic of Algeria Ministry of Higher Education and Scientific Research

University of Science and Technology Houari Boumediene

Computer Science Faculté

Système d'Information

Projet Business Intelligence

3ème ING CyberSecurity, 2025/2026

Business Intelligence Project Report

Design, Implementation, and Analytical Outcomes of a Unified Data Warehouse

Student Info :

Korichi Lyna Racha 232331739915 Grp03

I. Summary

This project addresses a critical Business Intelligence challenge frequently encountered in organizations with evolving information systems: the fragmentation of business data across legacy platforms and modern database technologies. In the case studied, historical business data was distributed across Microsoft Access files and SQL Server databases, preventing unified analysis and limiting strategic decision-making.

The primary objective was to design and implement a complete end-to-end Business Intelligence (BI) pipeline, covering Extraction, Transformation, and Loading (ETL), to consolidate heterogeneous data sources into a centralized Data Warehouse. Python was used as the orchestration layer to ensure flexibility and fine-grained control over data quality issues.

The final deliverables include a curated set of Key Performance Indicators (KPIs) and both static dashboards and an interactive 3D multidimensional visualization, enabling stakeholders to analyse sales activity, shipping costs, delivery efficiency, and employee performance across time and regions. This project demonstrates how structured BI methodologies can convert fragmented operational data into actionable analytical insight.

II. Project Context and Objectives

2.1 Business Problem

The organization operates within a data silo architecture resulting from system evolution rather than deliberate design. Specifically:

- Historical data is stored in flat files (Access exports originating from Microsoft Access).
- Current operational data resides in a Microsoft SQL Server database.

This separation introduces several analytical limitations:

- Lack of global visibility over sales and logistics performance.
- Inability to accurately compute aggregate shipping costs across all years.
- Fragmented analysis of employee performance, as staff activity cannot be tracked consistently across systems.

From a BI perspective, this environment prevents longitudinal analysis and undermines the reliability of management reporting.

2.2 Project Realization Structure:

```
projetBI/
├── data/
│   ├── raw/          # Staging: CSV exports from Access & SQL
│   ├── processed/    # Transformation: Normalized Access data
│   └── warehouse/    # Final: Fact and Dimension tables
├── scripts/          # Python ETL Logic
│   ├── extract_data.py       # SQL/Access ODBC extraction
│   ├── transform_access.py   # Text & Date normalization
│   ├── datawarehouse.py      # Star Schema construction
│   ├── kpi_analysis.py       # Business metrics calculation
│   └── visualize_warehouse.py # 3D & Static dashboard generation
├── figures/ # Static results
├── notebook/ # Dynamic results
│   ├── Dashboard_Analysis.ipynb
│   ├──3d_dashboard.html
└
```

2.3 Project Objectives

To address these issues, the project defined the following objectives:

1. **Unification:** Merge SQL Server Access data sources into a single, consistent source of truth.
2. **Data Quality:** Clean and normalize textual and structural data to resolve inconsistencies in naming conventions, casing, accents, and identifiers.
3. **Modelling:** Design and implement a *Star Schema* optimized for analytical workloads and BI querying patterns.
4. **Visualization:** Deliver both static dashboards for reporting and an interactive 3D visualization to support complex pattern discovery.

III. System Architecture and Technical Choices

3.1 Global Architecture

The solution follows a classical **ETL architecture**, separating concerns between data acquisition, transformation, storage, and analysis. This design ensures maintainability, scalability, and minimal impact on operational systems.

Work Flow Overview: following the principle of ETL

1. ***E*xtract:** Pull raw data from SQL Server and Access via *PyODBC.*
2. **Staging:** Store raw snapshots in CSV format (data/raw/).
3. ***T*ransform:** Normalize text, calculate revenue, and resolve duplicates via *Pandas.*
4. ***L*oad:** Populate the Star Schema (Fact and Dimension tables).
5. **Visualize:** Analytical Visualizations and KPIs

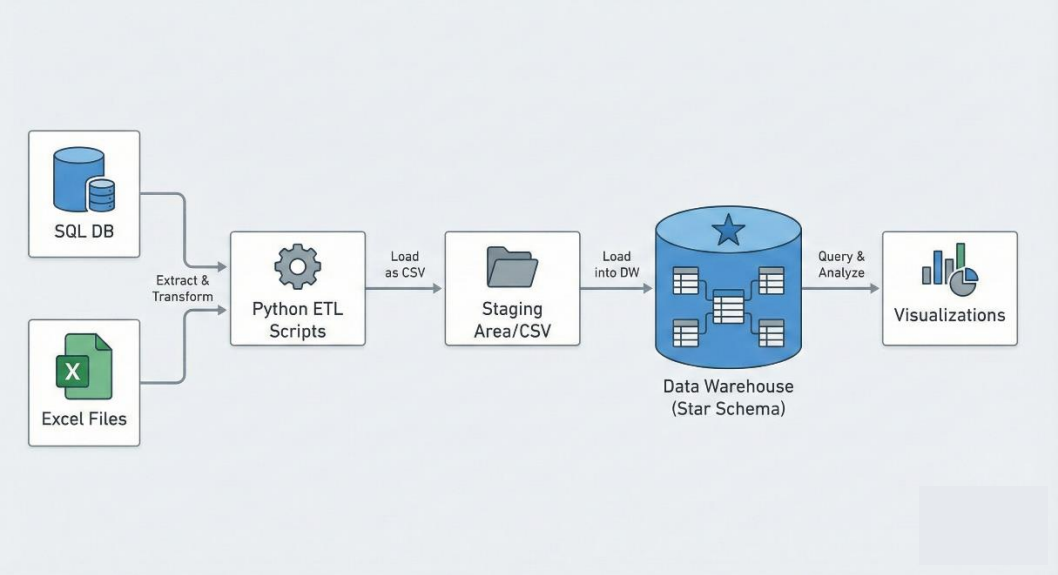

Figure 1 :Architecture Flow Diagram

(Side note: The use of a staging layer enables reproducibility, auditing, and safe reprocessing without repeatedly accessing live systems.)

3.2 Technology Stack and Justification

Python 3.x
Python was selected as the core orchestration engine due to its portability, extensive data engineering ecosystem, and suitability for custom ETL logic. Compared to proprietary ETL tools (e.g., SSIS or Talend), Python provides greater flexibility and transparency.

Pandas :
Used as the primary data manipulation library. Pandas offers high-performance, in-memory DataFrames that are ideal for cleaning, joining, aggregating, and reshaping datasets of moderate size such as Northwind.

PyODBC :
Serves as the database connectivity layer, enabling reliable interaction with ODBC-compliant systems such as SQL Server.

Unidecode :
Applied to normalize textual data by removing accents and enforcing consistent character encoding. This step was essential to ensure accurate deduplication and matching across heterogeneous sources.

Matplotlib & Plotly
Both were used for visualization and graph-making.
Matplotlib was used for static, publication-ready visualizations suitable for reports, while Plotly enabled interactive exploration through 3D visualizations without requiring proprietary BI software.

IV. Data Modelling (Conception)

4.1 Warehouse Strategy: Star Schema:

Source systems were designed using relational normalization (3NF), optimized for transaction processing rather than analytics. For BI purposes, the data was reorganized into a **Dimensional Model (Star Schema)**, the industry standard for analytical systems due to its simplicity and query performance.

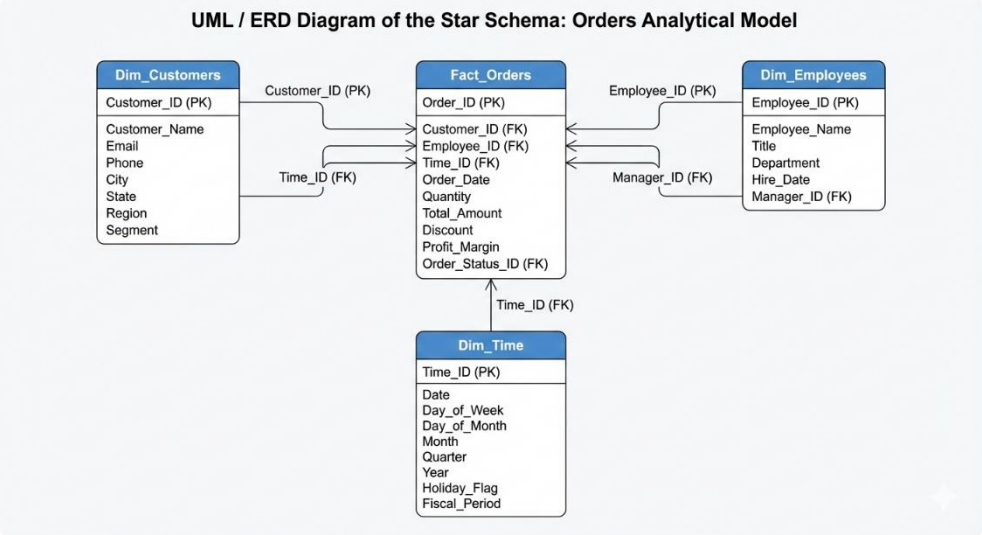

Figure 2 : Star Schema Diagram logic

Core Structure

- Central table: **Fact_Orders**
- Dimension tables: **Dim_Customers**, **Dim_Employees**, **Dim_Time**

4.2 Dimension Tables

Dim_Customers
Contains a unified list of customers across all systems. Surrogate keys were introduced to abstract away inconsistent source identifiers (e.g., numeric access IDs versus alphanumeric SQL IDs).

Dim_Employees
Consolidates employee records using normalized names to merge duplicates and enable consistent performance tracking.

Dim_Time
A generated calendar dimension supporting analysis by year, month, and day, which is essential for trend and time-series analysis.

4.3 Fact Table

Fact_Orders represents the central analytical entity.

- **Metrics:** Freight (shipping cost), delivery delay, order count.
- **Foreign Keys:** Links to customer, employee, and time dimensions.
- **Status:** A derived attribute classifying orders as Delivered, Not Delivered, or Late.

V. ETL Implementation Process

5.1 Extraction

Scripts: extract_data.py

Data was extracted from live SQL Server databases and raw Access files into a dedicated landing zone (data/raw/). This approach ensures that transformation processes do not interfere with production systems.

Key realization code:

```
# Access extraction via pyodbc
conn_str = r"DRIVER={Microsoft Access Driver (*.mdb, *.accdb)};DBQ=" + ACCESS_DB_PATH
# SQL Script execution via regex splitting
commands = re.split(r'\bGO\b', script_content, flags=re.IGNORECASE)
```

5.2 Transformation

Scripts: transform_access.py

Key transformation steps included:

- **Text Normalization:** Conversion to lowercase, removal of accents, trimming of whitespace.
- **Source Tagging:** Addition of a source column to preserve data lineage and auditability.
- **Union Operations:** Vertical concatenation of datasets to form a unified master dataset.

Key Realization code:

```
def normalize_text(s):
    """Normalize: remove accents, lowercase, strip whitespace"""
    s = unidecode.unidecode(str(s)).lower().strip()
    return " ".join(s.split())
```

5.3 Loading (Warehouse Construction)

Script: datawarehouse.py

This phase constructs the analytical warehouse:

1. Generation of dimension tables with surrogate keys.
2. Mapping of original source identifiers to warehouse keys.
3. Calculation of derived attributes (e.g., delivery status).
4. Persistence of *final fact_orders.csv and dim_*.csv tables*.

Key Realization code:

```
# Revenue mapping per order
df["rev"] = (pd.to_numeric(df[price_col]) * pd.to_numeric(df[qty_col]))
rev_map = df.groupby("oid_clean")["rev"].sum().to_dict()
fact["revenue"] = fact["orderid"].map(rev_map)
```

VI. Analysis and Results

6.1 Key Performance Indicators (KPIs)

Script: kpi_analysis.py

Global KPIs provide an immediate overview of operational performance:

Table 1 : KPIs

Metric	Result
Total Orders	878
Total Revenue	1,422,595.59$
Delivered	848
Pending	30
Delivery rate:	96.58%

6.2 Regional Analysis (Static Dashboard)

Order and data analysis was done in different chart shapes to analyse the different factors of revenue:

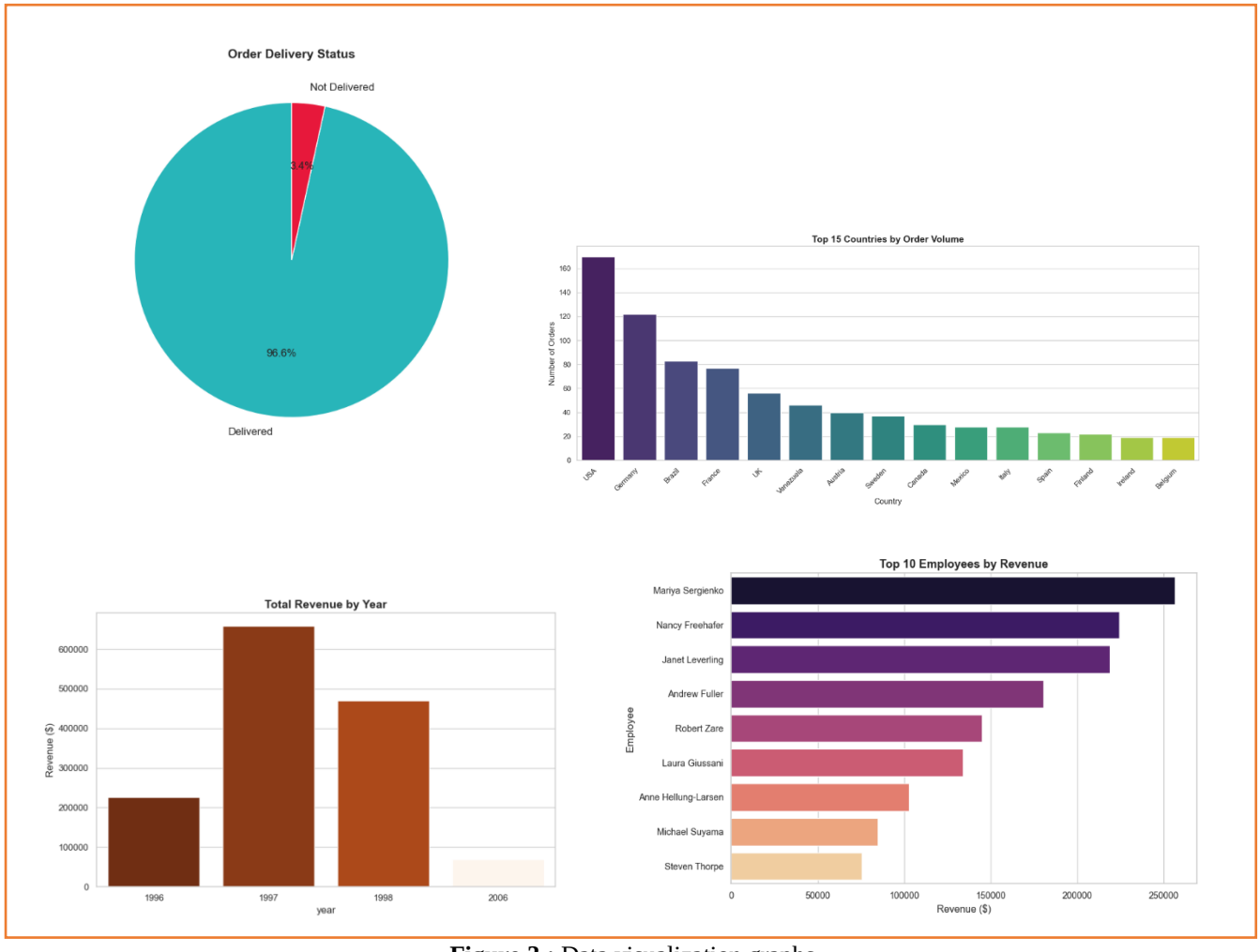

Figure 3 : Data visualization graphs

Interpretation: The visualization highlights orders status, dominant regions, years and employees suggesting where logistics and operational resources should be prioritized.

6.3 Multidimensional 3D Analysis

Script: visualize_warehouse.py

A 3D scatter plot was used to explore complex relationships:

- **X-axis:** Order date
- **Y-axis:** Customers
- **Z-axis:** Employees
- **Bubble size:** revenue
- **Color:** Delivery status

This visualization enables the identification of clusters, anomalies, and correlations that are not easily detectable through traditional 2D charts.

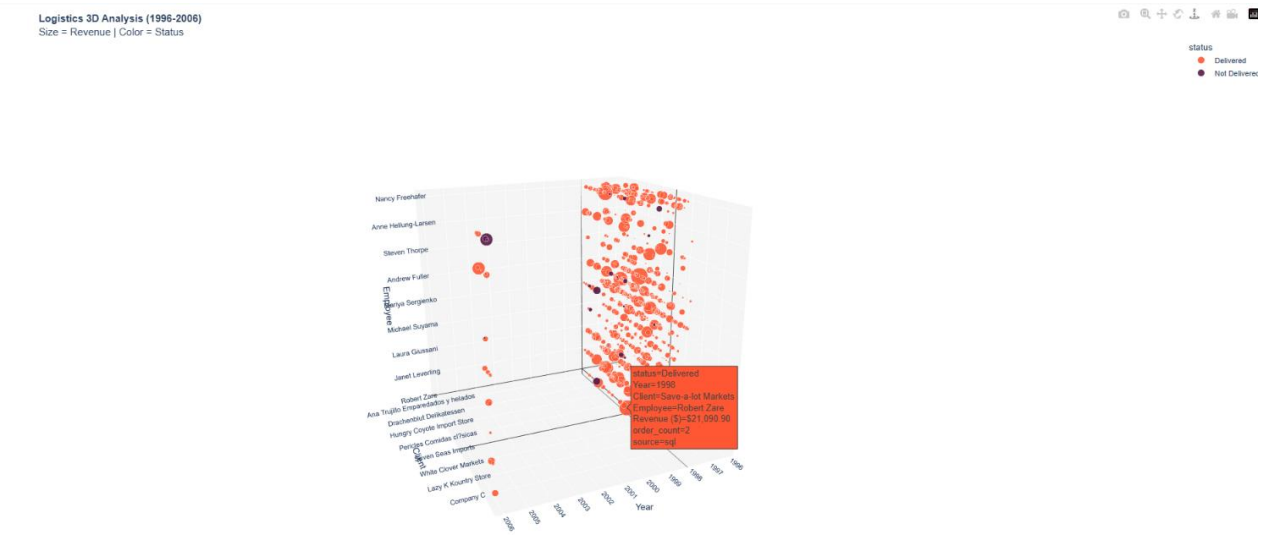

Figure 4 : Order Status Graph

VII. Conclusion and Future Improvements

7.1 Conclusion

This project successfully unified fragmented legacy and modern data sources into a coherent Business Intelligence platform. Through dimensional modelling and a robust ETL pipeline, raw operational data was transformed into a reliable analytical asset. The resulting KPIs and visualizations provide tangible business value by supporting informed decision-making.

7.2 Future Improvements

- **Automation:** Integrate Apache Airflow to schedule ETL execution.
- **Financial Expansion:** Incorporate order line details to compute revenue metrics.
- **Cloud Migration:** Move the warehouse to a cloud-based data lake for scalability and accessibility

Sources:

- Access to CSV Sources:

MS Access export table as .CSV file

https://learn.microsoft.com/en-us/answers/questions/4924885/ms-access-export-table-as-csv-file

How to export a Big Access DataBase on a CSV file

https://learn.microsoft.com/en-us/answers/questions/5612631/how-to-export-a-big-access-database-on-a-csv-file

Using the Wizard to Export to Text Files with Microsoft Access

https://nolongerset.com/export-specs-via-the-wizard/

SQL to CSV Sources

Export-Csv (Microsoft.PowerShell.Utility)

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/export-csv?view=powershell-7.5

- ETL Documentation Sources:

Guide for ETL Data Modeling Process

https://rivery.io/data-learning-center/etl-process-guide/

ETL tool documentation

https://etltool.readthedocs.io/en/latest/

ETL Pipeline Documentation

https://www.meegle.com/en_us/topics/etl-pipeline/etl-pipeline-documentation

ETL Pipeline Documentation: Here are my Tips and Tricks!

https://dethwench.com/etl-pipeline-documentation-best-practices/

ETL: A Complete Guide To Extract, Transform, And Load

https://www.rudderstack.com/learn/etl/etl-guide/