

# Project: The "TechStore" Data Platform

**Course:** Business Intelligence (BI)

**Level:** 4th Year Artificial Intelligence Engineering

**Team Composition:** Maximum 4 students per group

---

## 1. Project Overview

### The Business Scenario

You have been hired as the **Lead Data Engineer** for "*TechStore*," a rapidly growing electronics retail chain operating across multiple cities in Algeria.

The company is currently facing a "**Data Silo**" crisis. Critical business information is scattered across different isolated systems, making it impossible to get a clear picture of performance:

- **Sales & Inventory Data:** Locked in a central **ERP system**<sup>1</sup> (MySQL) that is difficult for managers to query directly.
- **Marketing Data:** Managed separately in **Excel spreadsheets** by the advertising team, meaning no one knows if the ad spend is actually driving sales in the ERP.
- **HR & Targets:** Sales targets and manager bonuses are tracked in **local Excel file** by the Human Resources (HR) department, disconnected from the actual sales results.
- **Logistics & Shipping:** Delivery costs are provided by third-party couriers in **separate rate sheets**, hiding the true profit margin of sold items.
- **Competitor Intelligence:** Completely unknown; staff currently checks competitor websites manually to fix prices.
- **Historical Archives:** Sales records from previous years (before 2023) exist only as **scanned paper invoices**, making year-over-year comparison impossible.

Management is struggling to answer complex questions like: "*Did the high marketing spend actually help the store manager hit their target?*" or "*Are we losing money on shipping heavy items to the South?*"

### Your Mission

Your goal is to build a **Unified Business Intelligence Platform** that transforms raw, scattered data into visual strategic insights. You are not just building a database; you are building a **Decision Support System**. Your responsibilities are twofold:

1. **The Backend (Data Engineering):** Engineer a robust ETL pipeline to extract data from fragmented sources (MySQL, Excel, Web, and Paper), clean it, and structure it into a central **Data Warehouse**.
2. **The Frontend (Analytics & Visualization):** Develop an Interactive Dashboard that brings this data to life.

---

<sup>1</sup>**Enterprise Resource Planning (ERP):** A software suite that integrates core business processes, such as finance, HR, manufacturing, sales, and supply chain, into a single system to facilitate real-time data flow across the organization.

## 2. Educational Objectives

This project is designed to bridge the gap between theoretical BI concepts and practical engineering skills. By the end of this project, you will demonstrate mastery of:

- **Advanced ETL (Extract, Transform, Load):** Writing Python scripts to programmatically connect to SQL databases, parse Excel files, scrape web data, and digitize images.
- **Data Integration:** Merging heterogeneous data sources (Structured vs. Unstructured) to create a single source of truth.
- **Dimensional Modeling:** Designing a rigorous **Star Schema** optimized for analytical queries (OLAP).
- **Business Analysis & Visualization:** Translating raw data into actionable strategic insights (ROI<sup>2</sup>, Profit Margins, Performance vs Targets) by executing complex SQL queries and rendering them into interactive dashboards.

## 3. The Architecture (The Pipeline)

You will build a pipeline consisting of four distinct stages:

1. **The Source Layer:** Connecting to a remote MySQL production server (simulating the ERP), reading departmental flat files, and extracting data from the web and images.
2. **The Staging Area:** Using **Pandas** to clean, normalize, and merge these datasets.
3. **The Warehouse Layer:** Storing the finalized data in a local database (like **SQLite**).
4. **The Presentation Layer:** Building a Python-based Dashboard to visualize the results.

## 4. Data Extraction

### Source 1: The ERP System (MySQL)

It is the main software used to run the business. Think of it as the company's "central brain" that tracks every single sale, product, and customer in real-time. Technically, this is an **OLTP (Online Transaction Processing)** system powered by a **MySQL** database, optimized for fast daily transactions.

- **Access:** you can connect via Python (for example using `mysql-connector`<sup>3</sup>).
- **Connection Information:**
  - Host: `boughida.com`
  - Database: `techstore_erp`
  - User: `student_user_4ing`
  - Password: `bi_guelma_2025`
- The database can be viewed graphically using phpMyAdmin at <https://boughida.com/phpmyadmin>.

---

<sup>2</sup>**Return on Investment (ROI):** A key performance indicator (KPI) used to evaluate the efficiency of an investment. In marketing, it measures the revenue generated relative to the cost of the advertising campaign, calculated as:  $ROI = \frac{\text{Revenue} - \text{Cost}}{\text{Cost}} \times 100$ .

<sup>3</sup><https://pypi.org/project/mysql-connector-python/>

**Tables to Extract:**

- `table_sales`: The central transaction log recording every item sold, quantity, and revenue.
- `table_products`: The master catalog of all items, including their manufacturing `Unit_Cost`.
- `table_reviews`: Internal customer feedback containing text reviews and star ratings.
- `table_subcategories` / `table_categories`: Tables defining the product hierarchy (e.g., Laptops → Computers).
- `table_stores`: A list of all physical store locations and their assigned IDs.
- `table_customers`: Profiles of registered clients including their home cities.
- `table_cities`: Shared geographic data linking stores and customers to specific regions.

**Source 2: Departmental Files (Excel files)**

These datasets are managed manually by non-technical departments (Marketing, HR, Logistics) who prefer spreadsheets.

- `marketing_expenses.xlsx`: Tracks the monthly advertising budget spent for each product category (in USD).
- `monthly_targets.xlsx`: Lists the revenue goals set by HR for each store manager to evaluate performance.
- `shipping_rates.xlsx`: Provides the delivery cost based on the destination region.

**Source 3: Competitor Pricing (Web Scraping)**

You must develop a Python script (using BeautifulSoup<sup>4</sup> for example) to scrape real-time product prices from a specified competitor's website (TechWorld Algeria). This external data will be compared against internal ERP prices allowing management to identify and fix overpriced products.

- **Website Link:** <https://boughida.com/competitor/>
- **Output:** You must extract `Competitor_Product_Name` and `Competitor_Price` for each product in the website.

**Source 4: Legacy Archives (OCR)**

You must build an image processing pipeline to digitize a set of scanned paper invoices (format: .jpg) stored in the `legacy_invoices` directory. These files represent the "Legacy Archives"—sales records from the previous year (2022) before the company adopted the modern ERP system. Using OCR libraries like `pytesseract`, you need to read these images and extract:

- **Output:** The date, total transaction amount (`Total_Revenue`), quantity, customer ID, and product name.

**IMPORTANT NOTE:** This step is **OPTIONAL**. Alternatively, instead of building an automated OCR pipeline, you may manually read the invoices and directly populate your `DataFrame` with the data.

**BONUS:** Students who successfully implement the automated OCR pipeline will receive **extra credit** on their final evaluation.

---

<sup>4</sup><https://pypi.org/project/beautifulsoup4/>

## 5. Transformation Requirements

Extracting data is only the first step in ETL process. You must write Python code using Pandas to clean, merge, and enrich the data before loading it into the data warehouse.

1. **"Net Profit" Calculation:** The ERP only gives you Revenue and Product Cost. You must calculate the *true* profit by integrating the external Excel files.
  - Formula:  $\text{Net\_Profit} = \text{Total\_Revenue} - (\text{Product\_Cost} \times \text{Quantity}) - \text{Shipping\_Cost} - \text{Marketing\_Cost}$
2. **Sentiment Analysis:** Use the VADER<sup>5</sup> sentiment analyzer to analyze the text of customer reviews. The output must be a Sentiment Score ranging from -1.0 to +1.0 for each product.
3. **Currency Harmonization:** The Marketing Department tracks expenses in **USD**, but the ERP records sales in **DZD**. You must apply a currency conversion rate before calculating net profit.
4. **Data Cleaning & Quality Control:** Your task is to audit the provided files and apply transformations:
  - **Duplicates:** Identify and remove duplicate records.
  - **Standardization:** Unify inconsistent data formats (Date formats, ID columns).
  - **Text Cleaning:** Fix case sensitivity (e.g., "Audio" vs "audio") and trim whitespace.
  - **Business Logic:** Detect invalid numerical values (negative costs) and handle Missing Values (Nulls).

## 6. Data Warehouse Design (The Loading Phase)

Once cleaned, load the data into a local **SQLite**<sup>6</sup> Database named `techstore_dw.db`. Note that **SQLite** is selected for this project to ensure the Data Warehouse (DW) is **portable and self-contained**, thereby simplifying the final submission.

Data in DW must be denormalized, so you are required to design a **Star Schema** optimized for analytics:

- **The Fact Table (Fact\_Sales):** A central table containing all sales transactions from both the ERP and legacy archives. Includes numerical measures: **Quantity**, **Total\_Revenue**, and calculated **Net\_Profit**.
- **The Dimension Tables:**
  - **Dim\_Product:** Flattened hierarchy, including Sentiment Score and Competitor Price.
  - **Dim\_Store:** Links stores to regions and includes Sales Targets.
  - **Dim\_Customer:** Tracking who is buying and their location.
  - **Dim\_Date:** Temporal dimension (Day, Month, Year) for time-based analysis (While this dimension is technically optional, it is strongly recommended because it enables consistent and efficient drill-down analysis across multiple time granularities).

---

<sup>5</sup><https://www.geeksforgeeks.org/python/python-sentiment-analysis-using-vader/>

<sup>6</sup><https://docs.python.org/3/library/sqlite3.html>

## 7. The Dashboard & Analytics (Presentation Layer)

**Goal:** Build an interactive Dashboard (using `Streamlit`<sup>7</sup> package) connected to your local SQLite Data Warehouse.

### Requirement A: Global KPIs

The top row must display:

1. **Total Real Revenue:** Sum of all sales.
2. **Net Profit:** Final profit after deducting all costs.
3. **Target Achievement (%)**: Gauge chart showing Actual Sales vs. HR Targets.
4. **Average Sentiment Score:** Derived from internal reviews.

### Requirement B: Advanced SQL Analysis

Using SQL Window Functions and aggregations (see chapter 4), visualize:

- **YTD Growth:** Line chart showing revenue accumulation over time.
- **Top Products:** Leaderboard of top 3 best-selling products per category.
- **Marketing ROI:** Bar chart comparing "Marketing Spend" vs. "Revenue Generated".
- **Price Competitiveness:** Compare the company's product prices with competitor prices collected through web scraping in order to identify products that are overpriced (at risk of losing customers) and products that could generate higher profit margins.
- **Custom KPIs:** Propose, calculate, and visualize **3 additional KPIs** of your choice.

### Requirement C: Interactive Filters

Enable OLAP slicing & dicing by allowing analysis based on *date range*, *store or region*, and *product category*.

## 8. Deliverables

**Deadline:** Monday, January 27, 2026, at 23:59.

**Submission:** Submit a single ZIP file named `Group_Number_BI_Project.zip` via Email to: [boughida.adil@gmail.com](mailto:boughida.adil@gmail.com)

The ZIP file must contain the following four items:

1. `etl_pipeline.ipynb`: Notebook running the Extraction, Transformation (NLP & OCR), and Loading.
2. `techstore_dw.db`: The populated SQLite Data Warehouse.
3. `dashboard_app.py`: The source code for the dashboard application. This script must include the SQL queries used to fetch KPIs and aggregations directly from the SQLite Data Warehouse.

---

<sup>7</sup><https://streamlit.io/>

4. `project_report.pdf`: A concise report (max 6 pages) explaining:

- A clear description of each student's individual contribution.
- **System Architecture**: A diagram or explanation of your data flow (Source -> ETL -> DW -> Dashboard).
- **Star Schema Design**: Screenshot/Diagram of Fact and Dimension tables.
- **KPI Logic**: The SQL queries used for calculation (including your 3 custom KPIs).
- **Business Interpretation**: Select 2 or 3 key insights from your dashboard and provide a recommendation. Example: "*We noticed that marketing for Printers costs more than the profit it generates. We recommend stopping these ads immediately to save money.*"

**Important:** Your code must run on any computer without modification. Use relative paths (e.g., `flat_files/shipping_rates.xlsx`), and do not use absolute paths (e.g., `C:/Users/...`)

## Final Note: Teamwork Strategy

To successfully complete this project by the deadline, **effective teamwork is essential**. You are strongly encouraged to divide the workload among group members based on their technical strengths.

**Example of Work Division (4 Members):**

- **Member 1 (Data Extraction Engineer)**: Responsible for extracting data from the primary sources, including connecting to the **MySQL ERP** and building the **Web Scraper** for competitor pricing.
- **Member 2 (ETL & Unstructured Data Specialist)**: Focuses on processing the flat files (CSV/Excel), performing Pandas transformations, and handling the **OCR/Bonus task** to digitize the legacy invoices.
- **Member 3 (Database Architect)**: Designs the Star Schema, creates the portable SQLite database, and writes the complex SQL queries for the KPIs.
- **Member 4 (Frontend Developer)**: Builds the Dashboard, implements the charts, and ensures the UI connects correctly to the backend database using the queries provided by Member 3.