Book Inventory: End-to-End E-Commerce Inventory Analytics Project

1. Project Overview / Executive Summary

Book Inventory is a comprehensive, end-to-end data analytics project designed to simulate a real-world inventory management scenario for an online bookstore. The project transforms unstructured web data into strategic business intelligence, enabling stakeholders to make data-driven decisions regarding stock levels, pricing strategies, and revenue potential.

Domain: E-Commerce / Retail

Project Goal: To build a robust data pipeline that extracts product data from the web, cleans and enriches it, and visualizes inventory risks and opportunities.

Tech Stack:

- **Data Collection:** Instant Data Scraper (No-Code Chrome Extension)
- **Data Processing:** Python (Pandas, NumPy)
- **Visualization:** Power BI (DAX, Natural Language Query)

2. Business Problem and Objectives

The Business Scenario

"Books to Scrape," a fictional online retailer, currently manages a catalog of 1,000 distinct SKUs. However, the management team lacks visibility into their inventory valuation and risk exposure. They are struggling to identify which high-value items are critically low on stock and whether their pricing strategy aligns with customer satisfaction (ratings).

Problem Statement

The business suffers from:

1. **Inventory Blindness:** Inability to quantify the total value of stock sitting in the warehouse.
2. **Stockout Risks:** No automated alerts for high-demand items falling below safety stock levels.
3. **Pricing Inefficiency:** Lack of insight into how product pricing correlates with customer ratings.

Objectives & KPIs

- **Objective 1:** Quantify the total potential revenue trapped in current inventory.
- **Objective 2:** Identify and flag top 10% of products at risk of stockout.
- **Objective 3:** Analyze the correlation between Unit Price and Star Rating.

Success KPIs:

- **Potential Revenue ($):** Total value of sellable inventory.
- **Stock Risk Rate (%):** Percentage of unique SKUs below safety stock threshold (5 units).
- **Average Unit Price ($):** Global average price of the catalog.

3. Data Collection

Data Source

The data was acquired via web scraping from the sandbox environment: [Books to Scrape]. This site is designed for scraping practice and mimics a standard paginated e-commerce catalog.

Extraction Process

1. **Tool Selection:** Used **Instant Data Scraper** (Chrome Extension) to simulate a rapid, low-code data acquisition workflow.
2. **Pagination Logic:** Configured the "Next" button selector to iterate through all 50 pages of the catalog.
3. **Extraction:** The scraper identified the HTML table structure and extracted the product pod details.
4. **Export:** Data was exported as a raw CSV file (raw_books_data.csv).

Data Profile

- **Volume:** 1,000 Rows (Unique Books).
- **Format:** CSV (Comma Separated Values).
- **Limitations:** The initial scrape captured the catalog view, which lacked "Stock Level" and "Star Rating" (hidden in CSS classes). These were handled in the processing phase.

4. Data Description

Raw Data Schema (raw_books_data.csv)

Column Name	Data Type	Description
product_pod	String	The book title (often truncated).
price_color	String	Price string containing currency symbol (e.g., £51.77).

thumbnail src	String	Relative URL path to the book cover image.
image_container href	String	Relative URL path to the product detail page.

Cleaned & Enriched Schema (cleaned_books_data.csv)

After processing, the final dataset structure is:

Column Name	Data Type	Description
Title	String	Cleaned, full name of the book.
Price	Float	Numeric price value (GBP removed).
Category	String	**Enriched:** Assigned genre (Fiction, Science, etc.).
Star_Rating	Integer	**Simulated:** Customer rating (1-5 scale).
Stock_Count	Integer	**Simulated:** Current warehouse quantity.
Inventory_Status	String	**Derived:** "In Stock" vs. "Low Stock".
Potential_Revenue	Float	**Derived:** Price * Stock_Count.

5. Data Cleaning and Preprocessing

The ETL process was executed using **Python (Pandas)** in a Google Colab environment.

Step 1: Column Mapping

- **Action:** Renamed product_pod -> Title, price_color -> Price.
- **Reason:** Raw scraper names were technical HTML class names; business-friendly names are required for BI.

Step 2: Currency Normalization

- **Action:** Removed £ symbol and converted Price column from Object (String) to Float.
- **Reason:** Mathematical operations (sum, average) cannot be performed on string data.

Step 3: Data Augmentation (Simulation)

- **Action:** Generated synthetic data for Stock_Count and Star_Rating.
  - *Logic:* Used a weighted random distribution (80% probability of Healthy Stock, 20% probability of Low Stock).
- **Reason:** The raw scrape from the catalog page did not expose inventory levels. To demonstrate the *Inventory Analysis* capability of the dashboard, realistic dummy data was required.

Step 4: URL Standardization

- **Action:** Appended the base domain http://books.toscrape.com/ to relative image paths.
- **Reason:** Power BI requires absolute URLs to render images in the dashboard.

6. Exploratory Data Analysis (EDA)

Key patterns discovered during the Python analysis phase:

1. **Price Distribution:** The prices are normally distributed around £35.00, with a range from £10.00 to £60.00.
2. **Inventory Health:** Due to the weighted simulation, 20% of the inventory is critically low (< 5 units). This represents a significant potential revenue loss if not restocked.
3. **Price-Rating Correlation:** A correlation matrix revealed a coefficient of **0.02** between Price and Rating.
   - *Insight:* There is **no relationship** between how expensive a book is and how highly it is rated.

7. Feature Engineering

To support the Power BI dashboard, the following features were engineered in Python:

1. **Potential Revenue (KPI):**
   - *Formula:* df['Price'] * df['Stock_Count']
   - *Purpose:* To quantify the monetary value of the stock.
2. **Inventory Status (Categorical):**
   - *Formula:* np.where(df['Stock_Count'] > 5, 'In Stock', 'Low Stock')
   - *Purpose:* Used for conditional formatting (Red/Green) in the dashboard visuals.

8. Data Modeling

Architecture

A **Flat Schema (Single Table)** approach was chosen for this specific project.

- **Rationale:** The dataset consists of a single dimension (Product) with associated facts (Price, Stock). A complex Star Schema was unnecessary for 1,000 rows and would have overcomplicated the Q&A (Natural Language) engine performance.

Logic

Data transformation was handled upstream in Python (Pandas) rather than Power Query to ensure the raw CSV loaded into Power BI was already "Analysis Ready."

9. Insights and Recommendations

Key Insights

1. **Revenue Concentration:** The top 20% of book titles account for roughly 25% of the total potential revenue, adhering to the Pareto Principle.
2. **Critical Risks:** Approximately 200 items are flagged as "Low Stock." If these are high-velocity items, the business is losing daily sales.
3. **Pricing Opportunity:** Since high prices do not correlate with better ratings, the business can experiment with dynamic pricing on low-rated, high-stock items to clear inventory without damaging brand perception.

Recommendations

1. **Immediate Restock:** Generate a Purchase Order for the 200 items marked "Low Stock" where Potential_Revenue > £500.
2. **Clearance Sale:** Discount books with Star_Rating < 2 and Stock_Count > 40 to free up warehouse space.

10. Dashboard Documentation

Tool: Power BI Desktop

File: Strategic_Inventory_Analysis.pbix

Dashboard Layout

1. **Header:** Project Title and Last Refresh Date.
2. **KPI Cards (Top Row):**
    - *Total Revenue Potential:* Sum of calculated revenue.
    - *Total Books:* Count of SKUs.
    - *Low Stock Count:* (Red text) Count of risky items.
3. **Visuals:**
    - **Scatter Plot:** *Price vs. Star Rating*. Shows the lack of correlation.
    - **Bar Chart:** *Revenue by Category*. Identifies best-performing genres.
    - **Risk Table:** Filtered list showing only "Low Stock" items, sorted by Price (High to Low).

4. **Q&A Feature:**
    - Integrated Power BI Q&A visual allowing users to type: *"Show me top 5 books by price"* to generate instant ad-hoc charts.

11. End-to-End Pipeline and Architecture

Workflow Diagram

[Web Source] -> [Instant Data Scraper] -> [Raw CSV] -> [Python ETL (Colab)] -> [Clean CSV] -> [Power BI Dashboard]

Architecture Description

1. **Ingest:** Chrome Extension extracts HTML table data to local CSV.
2. **Process:** Python script loads raw CSV, applies Regex cleaning, generates synthetic business features, and exports a standardized CSV.
3. **Serve:** Power BI imports the standardized CSV. DAX measures calculate aggregates on the fly.

12. Challenges and Solutions

Challenge	Solution
Dirty Currency Data	The price column contained '£' symbols. Used pandas.str.replace and astype(float) to clean this.
Missing Business Metrics	The web scrape lacked stock levels. Implemented a **Weighted Random Distribution** in NumPy to simulate realistic inventory scenarios for the portfolio.
Relative Image URLs	Images wouldn't load in Power BI. Created a logic to prepend the base domain http://books.toscrape.com/ to all partial URLs.

13. Conclusion

The **Biblio-Metrix** project successfully demonstrated the transformation of a static website into a dynamic business intelligence tool. By identifying over **£50,000+** in potential inventory value and flagging critical stock risks, the dashboard provides the imaginary stakeholders with the visibility needed to optimize supply chain operations.

Future Improvements:

- **Automation:** Replace the Chrome Extension with a BeautifulSoup script scheduled via Airflow.
- **Forecasting:** Use historical sales data (if available) to predict *Days Sales of Inventory (DSI)*.

14. Appendix

A. Data Dictionary Snippet

Title: String (Text)
Price: Decimal (Fixed)
Stock_Count: Whole Number
Inventory_Status: Text (Binary Class)

B. Key DAX Formulas

Low Stock Risk:

```
Low Stock Count = CALCULATE(
    COUNTROWS(cleaned_books_data),
    cleaned_books_data[Inventory_Status] = "Low Stock"
)
```

Total Revenue:

```
Total Revenue Potential = SUM(cleaned_books_data[Potential_Revenue])
```

C. Python Cleaning Snippet

```
# Cleaning Price
df['Price'] = df['Price'].astype(str).str.replace('£', '', regex=False)
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')

# Simulating Stock (Weighted)
stock_good = np.random.randint(5, 51, size=int(len(df)*0.8))
stock_low = np.random.randint(0, 5, size=int(len(df)*0.2))
```