Business Objectives :

The project aims to develop a Competitor Benchmarking Tool that enables businesses to compare their product listings with similar competitor offerings in terms of title relevance, price, and user rating. This supports informed pricing strategies, improves market positioning, and enhances competitive intelligence by identifying gaps and opportunities in the product lineup.

Extract – Transform – Load (ETL) :

Data is extracted via web scraping using Selenium and Undetected ChromeDriver, pulling product details such as title, price, availability, and rating. Undeteced ChromeDriver and Random User Agents is used to evade anti-bot detection mechanisms used by modern and newer websites. The data is then transformed and loaded into a structured format using JSON for consistency and future API implementations. The entire ETL is condensed into a single python file (main.py) for integration into automated workflows such as CRON jobs. To ensure efficiency, for newer items that is not recorded by previous pipelines, a hash map (using SHA-256) of previous extracts is stored inside a JSON file. During each run, only new items that are not recorded will be added to the list, this is to minimize redundancy.

Similarity Scoring and Summarizer Tag :

The similarity score is calculated based on title similarity (RapidFuzz), ratings, and price. Title is weighted most heavily because it's the most distinctive identifier. Rating and Price matter but to a lesser extent, because products might be priced or rated differently even if they are otherwise similar. Each component is weighted differently as seen below,

$$Score = 0.6(Title) + 0.2(Ratings) + 0.2(Price)$$

Before identifying products Tag, the class of Tags was defined first, where there are 5 different classes, which are "Best Seller", "Niche", "General", "Academic", and "Indie." These tags were chosen to cover a broad but meaningful spectrum of how books are typically positioned in the market and consumed by readers. Each product has their summary kept inside each product's page that will be extracted using Selenium. Once extracted, the summary is passed to a pre-trained model via an API call for classification with the Hugging Face Inference API is used due to its simplicity and seamless integration.

The model used is bert-large-mnli, a zero-shot text classification model. This model is particularly well-suited for the task because it does not require fine-tuning on specific labels. Trained on the Multi-Genre Natural Language Inference (MNLI) dataset, it can determine whether a piece of text logically entails, contradicts, or is neutral toward a set of candidate labels. This enables the system to classify book summaries into custom categories like Best Seller or Academic even though the model was not explicitly trained on those labels. The tag with the highest confidence score is selected as the final classification for each product (https://huggingface.co/facebook/bart-large-mnli).

Future Upgrades and Implementations :

There are several ways to upgrade this project. For scraping the information, a proxy and captcha solvers can be implemented. All data can be saved into a SQL/NoSQL database for easier data handling. A more accurate way on identifying similarity can also be implemented such as with cosine similarity. For usability, an API can be created using Flask to integrate better to existing products. All the summaries can also be saved beforehand to reduce time consumption and saved into a database for easier use. For the model, it could be saved locally (with the infrastructure available) to avoid usage limits. For additional features that can be made, an analysis of the site could be created to make better predictions and decisions.