**ETL Project - Final Report**

**Objective**

Build a database from the ground up with online data sources utilizing a ETL (Extract, Transform, Load) process, in which joins can be performed with a primary and foreign key reference.

**Purpose**

We all love to read and we would like to find our next book to read. However, the ISBN (International Standard Book Number) standardization has shifted gear in the recent past. It used to be 10 digits long if a book was given an ISBN before 2007, but starting from 2007 ISBN was reformatted to 13 digits long. Older books sometimes do not have a 13-digit ISBN so we need a way to find them using 13-digit ISBN. As a result, the purpose of this project is to build a book database that allows us to easily find a book using only ISBN-13. For the sake of the exercise, we will include a minimal amount of fields in our datasets for easy transfer and speed. Knowing our methodology and framework, one can easily retrieve more fields to look for the information that they need the most.

**Extraction**

Our team utilized two different datasets from two different sources for the project, both datasets have books related information. The unique identifier that was used to join the dataset is the 10-digit ISBN.

* Source 1: [Kaggle Goodreads](https://www.kaggle.com/bahramjannesarr/goodreads-book-datasets-10m?select=book4000k-5000k.csv) - CSV
  + The raw data included approximately 5M rows of data
  + A collection of 900K rows were extracted from the data source
* Source 2: [UCSD Book Graph](https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home) - JSON ([data link](https://drive.google.com/uc?id=1LXpK1UfqtP89H1tYy0pBGHjYk8IhigUK))
  + The original data source included approximately 2M rows of data
  + A collection of 500K rows were extracted from the data source

**Transform**

The datasets were imported into pandas dataframes to perform the cleaning (transform) process. In summary, the following actions were completed:

* Rows with missing information were removed (dropped via dropna(how='any') )
* Rows with duplicated book titles were removed (dropped via drop\_duplicates() )
* The final datasets ready for loading contained 750K (Kaggle) and 200K+ (UCSD) rows after cleaning
* Fields not required for the intended join, meaning not required for aggregations, were removed from the dataframes. The fields we decided to include in this exercise are ISBNs, book title, authors, and book rating.
* Data types from cleaned CSV files were reviewed in order to make sure the table classes in Load phase can be built to match up

**Load**

Data was loaded into a SQLite database using SQLAlchemy, as outlined below. Specifically, to create the database and tables we performed the following actions:

* Created SQLIite database “goodreads” using SQLAlchemy
* Created two classes (which will serve as the anchor points for our Tables) for the cleaned Kaggle and UCSD data sets
* Used pandas to load the two CSV converted dataframes into the SQLite database
* Created scenarios in which we could flexibly customize our SQL queries to retrieve the information that we needed for any potential future analysis:
  1. Joined the two SQL tables on unique key “isbn” (10-digit) to incorporate “isbn13” from UCSD table. The results were also sorted.
  2. Note: The merged dataset has 40K rows, meaning only 40K ISBNs were matches in two datasets
* Used aggregate functions and “having” conditions to determine the Top 10 rated books from authors that have written at least 10 books

**Afterword: *Challenges***

We originally tried to incorporate both web scraping and APIs into the project but faced some challenges. For the sake of mastering the ETL process we decided to drop the ideas and simplify the strategy. Below are a snapshot of the challenges we encountered:

1. **Amazon book price searches using ISBN key:** We initially decided to scrape book prices off of Amazon.com. However, Amazon’s site proved to be too difficult/unpredictable/time-consuming to navigate using automated searches for ISBN using splinter. Further, splinter searches for just 100 ISBNs required 30 minutes of runtime, so 10,000 minimum searches would have taken ~25 hours to complete. As a result, the use of Amazon was ultimately abandoned.
2. **Google API calls to grab book descriptions:** Google Books API’s limitations on calls per minute and calls per day largely restricted our ability to extract large amounts of data. To save time and financial resources, the use of Google Books API was also abandoned.
3. **Logistics of large datasets:** The complete datasets from both sources included more than 1M of rows. The group had to cut down the amount of data and shrink the file sizes down to an acceptable range. Once the data was cleaned and extracted, the group also overcame the challenge of uploading large files to GitHub by installing Git Large File Storage (LFS) extension.
4. **Unzipping google-zip file:** the UCSD dataset utilized google zip, a new format for the group. We found documentation for a python unzip function on the UCSD github page which was then utilized to upload only 500K rows. To help facilitate the flow of the project, only 500K of the 2M rows of UCSD dataset were utilized due to long lead times required for upload, writing csv files and converting to pandas.