**Q1. Using the JSON data file Electronics\_5.json, that is available here :** [**https://nijianmo.github.io/amazon/index.html**](https://nam10.safelinks.protection.outlook.com/?url=https%3A%2F%2Fnijianmo.github.io%2Famazon%2Findex.html&data=05%7C02%7CHimadri.Khanna%40blackrock.com%7C3fefc17428a243d3ce0808dc7fbe5252%7C282a32955c424d939ec16631001cc5f7%7C0%7C0%7C638525702718856733%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0%7C%7C%7C&sdata=Rf8SYUjIkLYD9WEVUbuzSWk9bBFygMdsNt6Fwu8gwE8%3D&reserved=0) **>"Small" subsets for experimentation> Electronics 5-core**

**Context of the data file:**

- reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B

- asin - ID of the product, e.g. 0000013714

- reviewerName - name of the reviewer

- vote - helpful votes of the review

- style - a disctionary of the product metadata, e.g., "Format" is "Hardcover"

- reviewText - text of the review

- overall - rating of the product

- summary - summary of the review

- unixReviewTime - time of the review (unix time)

- reviewTime - time of the review (raw)

- image - images that users post after they have received the product

**Requirements:**

1. Data Ingestion: Design a Python ETL script that ingests data from the above data source.

2. Data Processing:

   - Implement a text processing mechanism including text cleaning to handle inconsistencies, redundancies, misspellings, and irrelevant characters.

   - Transform the processed information into a structured format suitable for storage in a database

   - Send an email on a daily basis with the following information:

     - asin having overall greater than 3 with Format being "Hardcover"

     - asin having the maximum number of verified reviews

     - log file for the ETL process

3. Data Storage: Choose an appropriate flat file format for storing the transformed data

4. Scalability and Performance:

    - Ensure that the solution is scalable to handle growing datasets.

    - Optimize the code and processing pipeline for performance.

    - Consider implementing caching mechanisms or other strategies to speed up repetitive operations.

**Q2. Using the following API, write a python script to extract data and transform it into a tabular fashion, having one row per ID**

* Please clean the data including removing html/css coding from the data i.e only keep the highlighted data-  "page\_description": "\u003Cp\u003EAfterpay&nbsp;\u003Cspan style=\"font-size: 1em;\"\u003Emay get a commission.\u003C/span\u003E\u003C/p\u003E",
* Save the cleaned output as csv files
* Sift through the dataframe to filter only IDs that are parents – save that dataframe separately

[**https://store-directory-api.afterpay.com/api/v1/categories**](https://nam10.safelinks.protection.outlook.com/?url=https%3A%2F%2Fstore-directory-api.afterpay.com%2Fapi%2Fv1%2Fcategories&data=05%7C02%7CHimadri.Khanna%40blackrock.com%7C3fefc17428a243d3ce0808dc7fbe5252%7C282a32955c424d939ec16631001cc5f7%7C0%7C0%7C638525702718867822%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0%7C%7C%7C&sdata=Xlq0ogwjpYyit9sN5Y8Q949dCToIlThn%2F07Bk6B4efc%3D&reserved=0)

**Q3. The attached zip file (Q3.zip) contains 2 csv data files (data1.csv and data2.csv), 1 entity list ,mapping\_data.csv and columns\_to\_keep.csv:**

* Clean and concatenate all data files into one csv file with the following conditions:
  + Using the mapping\_data.csv, map the fieldIDs in fields to FieldNames that would be needed as column names for step 2
  + The final output should have only those columns that exist in Columns\_to\_keep.csv and in the same order, if that column exists in your final output
  + The final output should only contain those entityIDs that are in the entitylist.csv, there can be extra entity IDs in the entitylist
* Create a proper mechanism to log the missing/incorrect/extra entity IDs and columns that were dropped while creating the final output