**Documentation 2**

**Partner Sustenance Part 2 ( Destination\_partner side)**

**Data Processing Script Documentation**

**Overview**

The Data Processing Script is designed to facilitate the handling and analysis of transaction data stored in CSV files. This script offers a structured approach to processing, ranking, merging, and modifying DataFrames, enabling users to derive insights efficiently. It is particularly useful for organizations that need to analyze transaction performance across various partners and countries.

**Function Descriptions**

**1. process\_data\_and\_save(input\_files, output\_files, group\_column)**

This function is responsible for processing multiple CSV files. It reads the data, sorts it based on specified criteria, and adds rank columns to facilitate further analysis. The processed DataFrames are then saved to new CSV files for easy access.

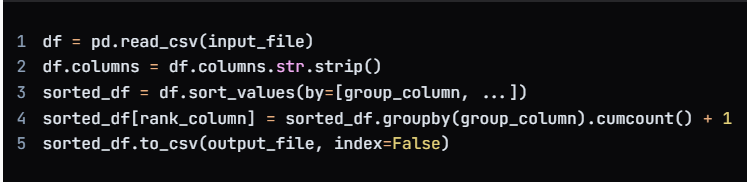
**Parameters:**

* **input\_files**: A list of strings representing the paths to the input CSV files that contain transaction data.
* **output\_files**: A list of strings representing the paths where the processed CSV files will be saved.
* **group\_column**: A string indicating the column name used for grouping the data during sorting.

**Returns:**

* None

**Example Code:**

****This function ensures that the data is organized and ranked according to the specified grouping, making it easier to analyse performance metrics across different segments.

**2. merge\_and\_sort\_csv\_files(file1\_path, file2\_path, file3\_path, file1\_cols, file2\_cols, file3\_cols)**

This function merges three CSV files based on common columns, specifically destination\_country and destination\_partner. It calculates total and average ranks from the ranking columns and sorts the resulting DataFrame. This is particularly useful for generating a comprehensive view of transaction performance across different partners.

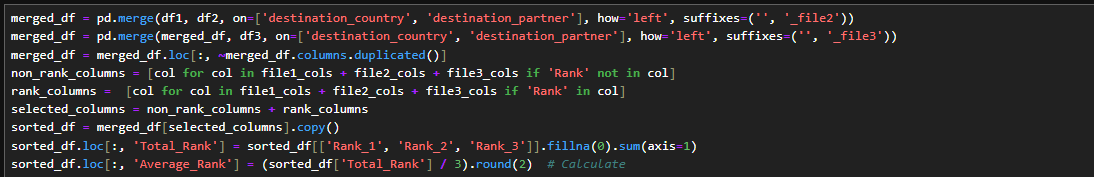
**Parameters:**

* **file1\_path:** The file path to the first CSV file containing relevant transaction data.
* **file2\_path:** The file path to the second CSV file with additional metrics.
* **file3\_path:** The file path to the third CSV file containing further data points.
* **file1\_cols, file2\_cols, file3\_cols:** Lists of strings that specify which columns to include from each file.

**Returns:**

* A DataFrame that contains the merged and sorted results.

**Example Code:**



This function is essential for consolidating data from multiple sources, allowing for a more holistic analysis of transaction performance metrics.

**3. swap\_rows\_based\_on\_columns(df, columns\_to\_compare, columns\_to\_drop=None)**

This function modifies the DataFrame by swapping rows based on specified columns. It allows for a dynamic rearrangement of data, which can be crucial for comparative analysis. Users can also specify columns to drop after the operation to clean up the final DataFrame.

**Parameters:**

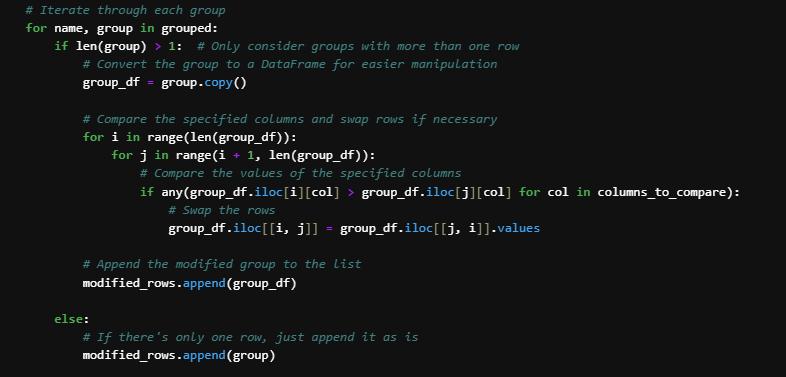
* **df**: The DataFrame that needs to be modified.
* **columns\_to\_compare**: A list of column names that will be used for comparison to determine which rows to swap.
* **columns\_to\_drop**: An optional list of column names that should be dropped from the final DataFrame.

**Returns:**

* A modified DataFrame with swapped rows and any specified columns removed.

**Example Code:**





A black background with white text

Description automatically generated

By utilizing this function, users can ensure that their data is organized in a way that highlights important performance metrics and comparisons.

**Example Usage**

The following code snippet demonstrates how to use the functions defined in the script. This example illustrates a complete workflow from processing the initial data to generating a final output file.

