**Memo**

To: Professor Pisano

From: Joel Akerman, Taha Ababou, Manuel Segimon, Bora Bulut, Zaiyan Muhammad

Team: PAPO

Date: 11/19/23

Subject: First Prototype Testing Report

1. **Introduction**

This technical report outlines a series of test cases and procedures performed on a dataset of pharmaceutical products. The code is structured to manipulate and analyze this dataset, focusing on specific drugs identified by their generic names and reference item numbers. The main objective is to filter and compare drug items based on their characteristics, using data processing and similarity measures.

**1.1 Purpose**

This document outlines the requirements for PAPO’s procurement software, which will be developed for Beth Israel Medical Center. It will be used by all stakeholders, including developers and testers.

**1.2 Scope**

**1.1.1 In Scope (features that will be tested in this prototype)**

* **Variable Consistency Check**
  + Ensure that the variables generic\_name and reference\_item\_number are correctly assigned and used throughout the code.
* **Data Loading and Integrity Check**
  + Verify that the data is loaded correctly and that the initial data structure aligns with the expected format of the 'Daily Snapshot.csv' file.
* **Data Cleaning Validation**
  + Confirm that the data cleaning steps, such as column removal and reordering, are accurately executed without data loss or corruption.
* **Drug-Specific Data Filtering Test**
  + Check if the filtering process accurately isolates data related to the specified generic drug ('cefepime').
* **Similarity Analysis Accuracy Test**
  + Evaluate the accuracy of the TF-IDF vectorization and cosine similarity calculations in identifying similar drug forms and sizes.
  + Ensure that the reference item is correctly identified and used as a baseline for comparison.
* **Result Sorting and Relevance Test**
  + Assess whether the final sorted list accurately reflects the items most similar to the reference item based on the defined similarity thresholds.

**1.1.2 Out of Scope**

The following features will not be tested for this prototype presentation:

* User Interfaces
* Online Database
* API Calls
* Accuracy Tests with the Procurement Team at Beth Israel Medical Center

**1.3 Roles and Responsibilities**

| **Manuel** | Team Leader/Algorithm Developer |
| --- | --- |
| **Taha** | Developer/QA Tester |
| **Bora** | Developer/QA Tester |
| **Joel** | Product Manager |
| **Zaiyan** | Developer/Market research |

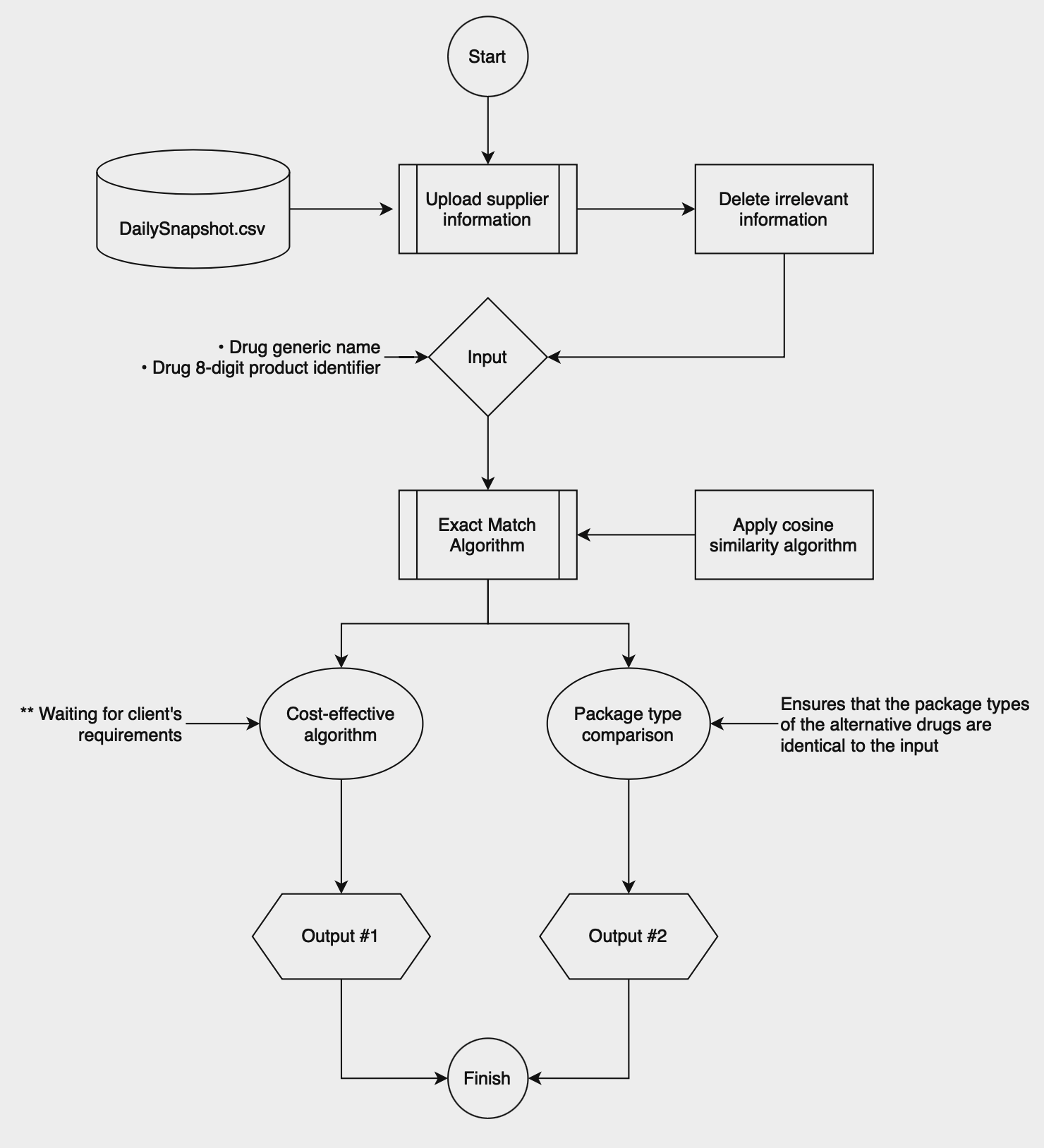
**2.0 Code Overview**

# 

| **Variable Assignment** | The code begins by assigning generic\_name and reference\_item\_number for the drug we want to find alternatives containing the active ingredient of interest. The user sets the variables generic\_name and reference\_item\_number for the aforementioned desired product. |
| --- | --- |
| **Data Loading and Initial Processing** | The dataset is loaded from a CSV file named 'Daily Snapshot.csv' using pandas, a Python data analysis library.  The initial part of the code is dedicated to cleaning and reorganizing the dataset. This includes removing specific columns, reordering columns, and splitting column values for better clarity and analysis. |
| **Data Cleaning and Transformation** | The 'Generic Description' column is split into 'Generic Name' and 'Form', and the original column is dropped.  Similar processing is done for the 'Description' column, splitting it into 'Name' and 'Size'.  The dataset is filtered to remove rows with empty 'Generic Name'. |
| **Filtering Specific Drug Data** | The dataset is further filtered to focus on items with the specified generic name. |
| **Similarity Analysis** | * The code employs TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and cosine similarity measures to find drug form and size similarities. * A reference item is selected based on the provided reference item number, and its form and size are used for comparison. * The dataset is filtered to retain only those highly similar in form and size to the reference item. |
| **Result Sorting** | Finally, the items with high similarity scores are sorted based on their size similarity, providing a clear view of the most similar items to the reference drug. |

# 

**2.1 Visualization**



*Figure A. Prototype Flowchart*

**3.0 Test Methodology**

**3.1 Pre-Testing Setup Procedure**

IDE Preparation:

* 1. Begin by downloading and installing an IDE compatible with Python and Jupyter Notebooks. Visual Studio Code (VS Code) is recommended for its robust features and support for these technologies.

1. Install Python Modules
   1. Ensure that essential Python modules are installed. This includes:
      1. **Scikit-learn:** A machine learning library used for data mining and data analysis.
      2. **Pandas:** A library providing high-performance, easy-to-use data structures and analysis tools.
      3. **Re (Regex):** A module for string searching and manipulation using regular expressions.
      4. **Numpy:** A library for large, multi-dimensional arrays and matrices and a collection of mathematical functions to operate on these arrays.

These modules can be installed using pip, the Python package manager, with the command pip install scikit-learn pandas numpy regex.

1. Data and Script Preparation:
   1. Place the client's data file, 'Daily Snapshot.csv', in the newly created directory. This file contains the data set required for the testing.
   2. Copy the Jupyter Notebook containing the Python script into the same directory. This will ensure that all necessary resources are in one accessible location.
2. Environment Verification:
   1. Before commencing the testing, verify that the IDE is functioning correctly and all modules are correctly installed and importable in Python.
   2. Open the Jupyter Notebook in the IDE and run a test cell to confirm that the setup is error-free and ready for the actual testing process.

By following these enhanced setup procedures, the testing environment will be optimally prepared, ensuring a smooth and efficient testing process.

**3.2 Testing Procedure**

1. Pick an active ingredient from the client data and set the ‘generic\_name’ variable to the chosen active ingredient (Test cases are provided at the top of the Jupyter notebook).
2. Pick the 8-digit product code of the drug that contains the aforementioned active ingredient that you want to find alternatives to and set the ‘reference\_item\_number’ to this value.
3. Press run all.
4. The algorithm filters the drugs initially by the active ingredient and then the administration method. It then uses cosine similarity to calculate the similarity based on size, which contains information about the concentration and dosage.
5. The table at the bottom shows the drug we’re finding alternatives to in the first row, followed by the details of the alternatives with form and size similarities above 0.9 in descending order.

**3.3 Measurable Criteria**

| **Data Cleaning and Preprocessing Accuracy** | Evaluate the accuracy of the data cleaning and preprocessing steps in the Python code. |
| --- | --- |
| **Efficiency in Data Transformation** | Measure the time taken to execute data transformation steps, including splitting descriptions and reordering columns. |
| **Similarity Score Precision** | Assess the precision of TF-IDF vectorization and cosine similarity calculations in identifying similar drug forms and sizes. |
| **Filtering and Sorting Capability** | Determine the efficiency and accuracy of the system in filtering and sorting pharmaceutical products based on similarity scores. |
| **Scalability with Large Datasets** | Test the system's performance with increasingly large datasets to assess scalability. |
| **Response Time in Data Analysis** | Measure the response time from data input to the display of results. |
| **Feedback and Iterative Improvement** | Implement a system to collect feedback on the prototype’s performance and use this feedback for iterative improvements. |

**3.4 Test Cases**

The following active ingredient-drug code combinations will be tested:

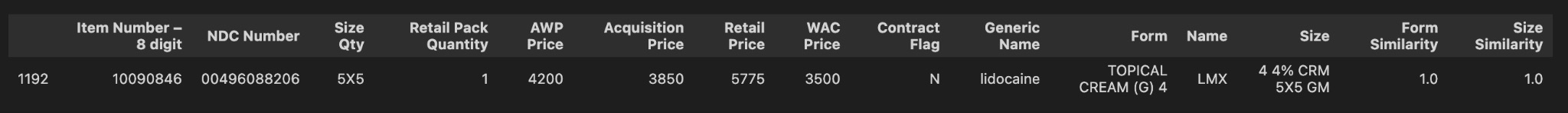
| **Active Ingredient** | **Drug Code** |
| --- | --- |
| lidocaine | 10090846 |
| lidocaine | 10104126 |
| lidocaine | 10136471 |
| cefepime | 10175647 |
| cefepime | 10267015 |
| cefepime | 10027431 |
| ondansetron | 10096566 |
| ondansetron | 10112532 |
| ondansetron | 10126451 |

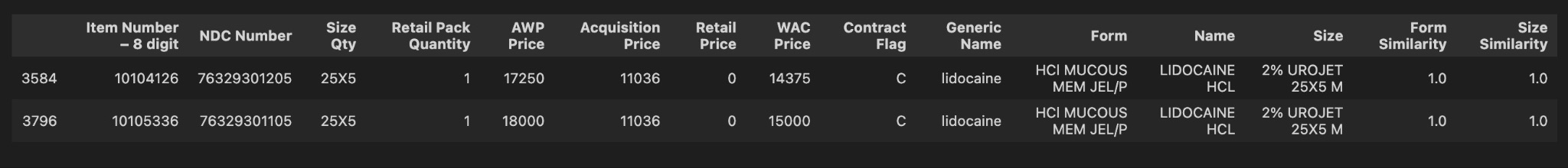
| **Score sheet** | **Achieved?** |
| --- | --- |
| 1. Can filter out non-matching drugs |  |
| 1. Being able to filter by form |  |
| 1. Being able to filter by size |  |
| 1. Being able to display alternative drug options with very similar specifications |  |
| 1. Being able to use the code with different drugs from the dataset |  |

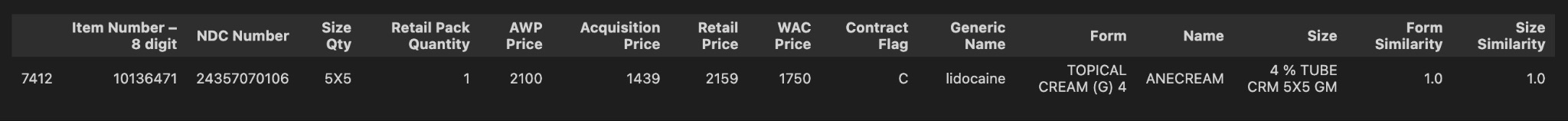
**4.0 Results**

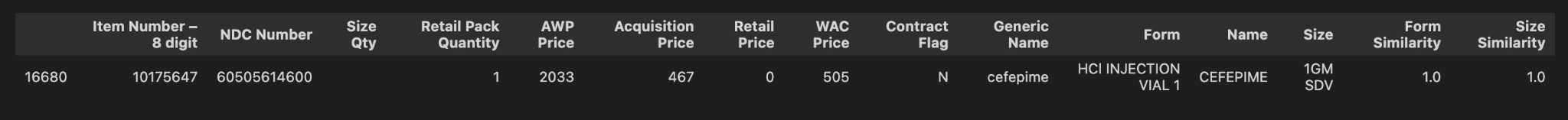
**4.1.1 Test Case Outputs**

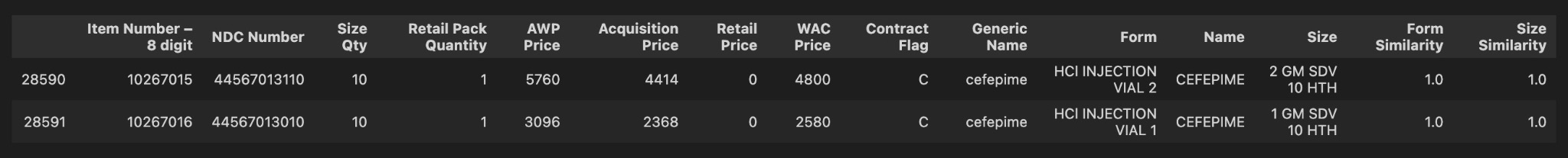
The prototype has been run on the client's dataset, 'Daily Snapshot.csv', with the aforementioned active ingredient-drug code combinations and configured to output matching drugs with form similarity and size similarity above 0.9. The following outputs have been observed:

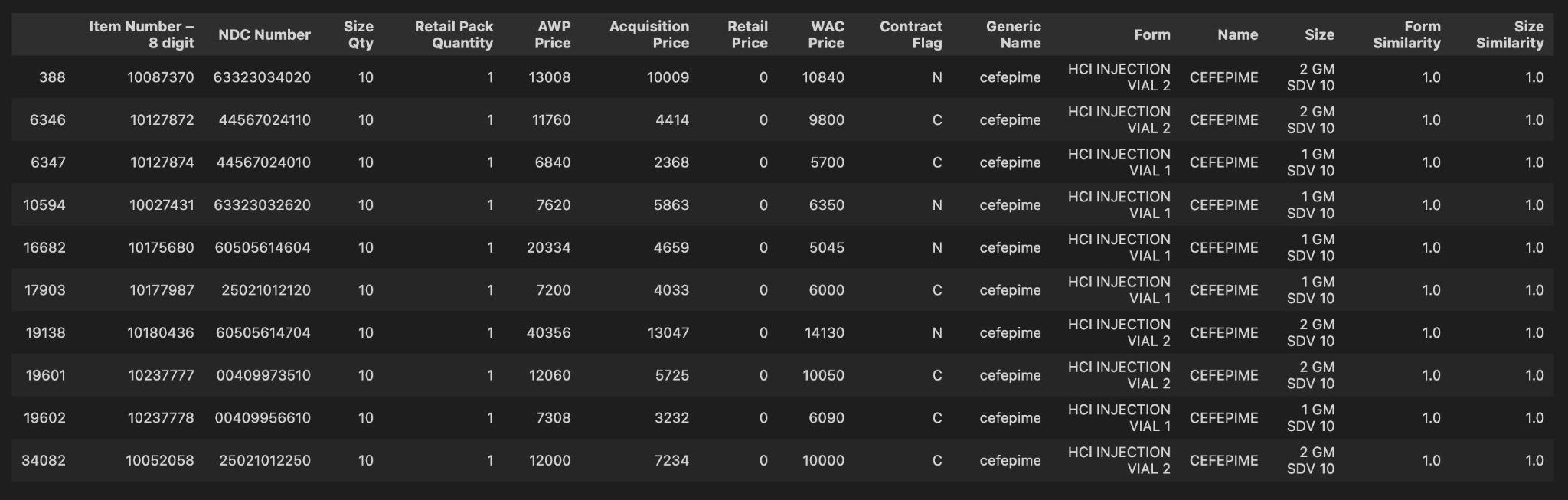
lidocaine - 10090846

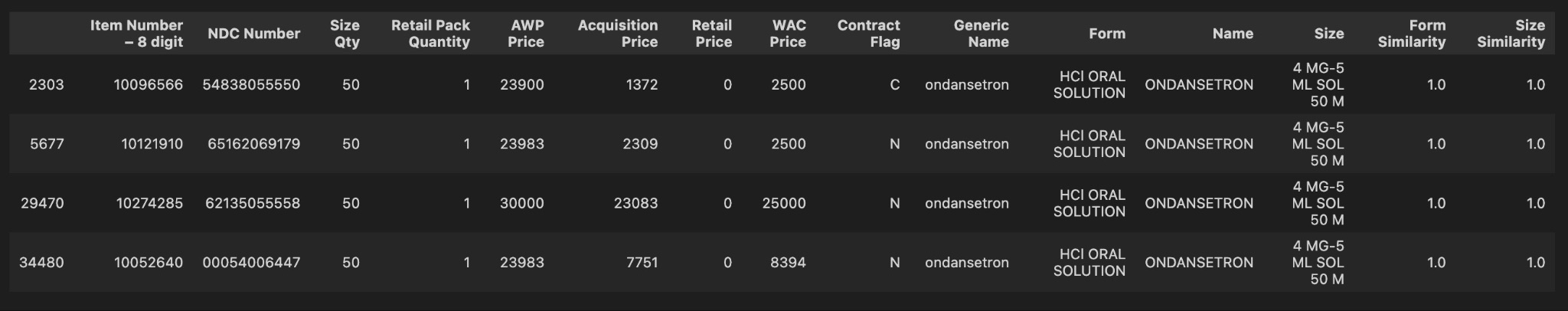
lidocaine - 10104126

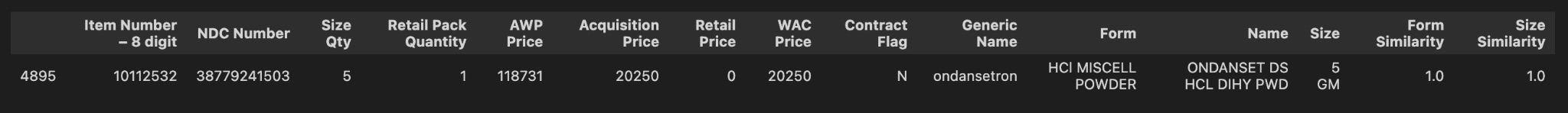
lidocaine - 10136471

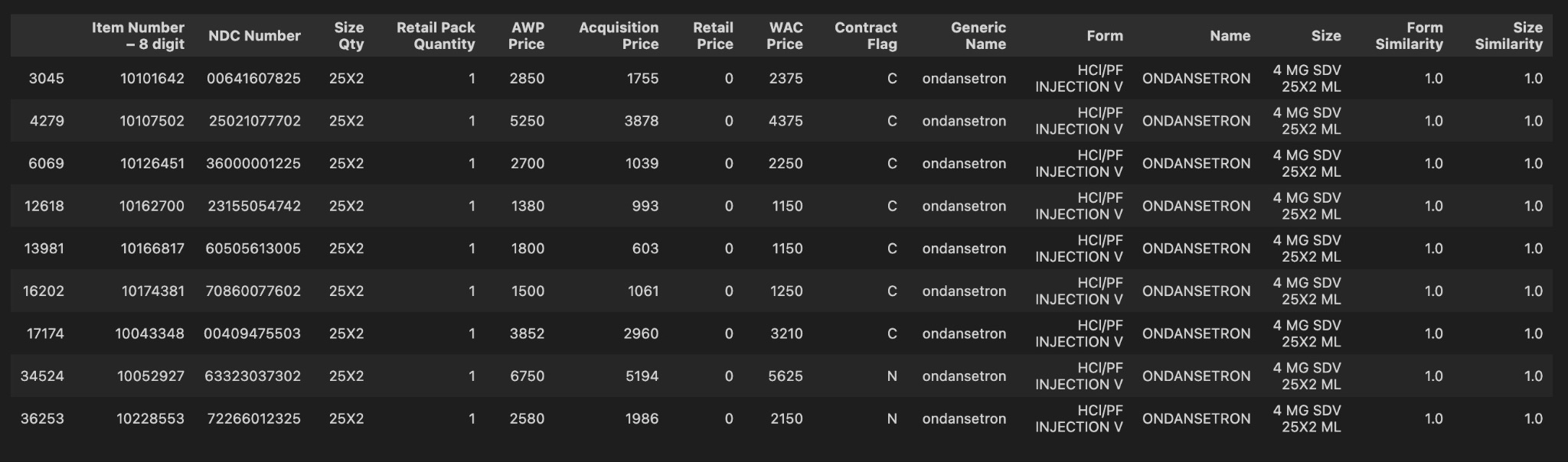
cefepime - 10175647

cefepime - 10267015

cefepime - 10027431

ondansetron - 10096566

ondansetron - 10112532

ondansetron - 10126451

**4.1.2 Implications of the Results**

| **Score sheet** | **Achieved?** |
| --- | --- |
| 1. Can filter out non-matching drugs |  |
| 1. Being able to filter by form |  |
| 1. Being able to filter by size |  |
| 1. Being able to display alternative drug options with very similar specifications |  |
| 1. Being able to use the code with different drugs from the dataset |  |

Outputs show that the prototype is able to filter out non-matching drugs, filter by form and size, display alternative drug options with very similar specifications, and is modular in the sense that it can be used with different drugs from the dataset. It is important to note that first rows correspond to drugs that the algorithm finds alternatives to. Our findings suggest that, due to the stringent size similarity criteria, in many cases, the algorithm is unable to find a valid alternative. This indicates that the majority of the supplier’s drugs are not extremely similar to each other in size and that a reasonable similarity margin should be determined for each drug before applying cosine similarity. Therefore, a machine learning model must be built to intelligently calculate a size similarity threshold for each drug so that the algorithm yields a viable range of alternatives to consider.

**5.0 Conclusion**

This code serves as a comprehensive approach to filtering, analyzing, and comparing pharmaceutical products based on specific characteristics. The testing plan aims to validate each step of the process, ensuring the reliability and accuracy of the results.

**5.1 Future Milestones**

1. Create a platform to include our product
2. Make the supplier data available in real time and automatically
3. Gain access to client software’s API
4. Create a cost-effective algorithm
5. Create the package effective algorithm
6. Run accuracy checks with the procurement team

**6.0 Appendix**

* GitHub repository: <https://github.com/BU-EC463/Drug-Similarity-Calculator/tree/Actual-Data-Snapshot>