Data Engineering Task
By Dmytro Kotov
Mail to: dmytro.kotov97@gmail.com
9 February, 2026

Initially, two datasets were provided: **company_dataset_1.csv** and **company_dataset_2.csv**.

# Matching approach

To complete this part of the task, **dataset_1** (next referred to as **dataset1_cleaned**) was used as the reference dataset. All corresponding companies were matched against **dataset_2** (next referred to as **dataset2_cleaned**), taking into account location related attributes: **country**, **state**, and **postal_zip**. The key objective was to retain all rows from **dataset1_cleaned** and attach all corresponding matches from **dataset2_cleaned**. Where matches existed, the relevant fields were filled with values, where no match was found, **NaN** values were produced. The matching was performed using the pandas.merge() function which basically stands for Left Join

```
company_matches = pd.merge(
   dataset1_cleaned,
   dataset2_cleaned,
   on=['company_name', 'country', 'state', 'postal_zip'],
   how='left',
   indicator=True
).
```

The attribute **indicator=True** is used for explicit judgment whether matches happened or not.

Next, to resolve the next task it was needed to distinguish unique companies for **dataset1_cleaned** and group datasets based on company_name location attributes such as **state**, **postal_zip**, **country** and **city** expand them into columns:

```
locations_dataset1 = dataset1_cleaned.groupby('company_name').apply(
   lambda x: list(zip(x['state'], x['postal_zip'], x['country'], x['city']))
).reset_index(name='locations_dataset1')
```

and

```
locations_dataset2 = dataset2_cleaned.groupby('company_name').apply(
   lambda x: list(zip(x['state'], x['postal_zip'], x['country'], x['city']))
).reset_index(name='locations_dataset2').
```

These columns were concatenated using pandas.merge() function

```
merged = pd.merge(
   locations_dataset1,
```

**locations_dataset2,**
**on='company_name',**
**how='left'**
**).**
As the result of that we receive a data frame with the following columns: **company_name**, **location_dataset1** and **location_dataset2** respectively. Location datasets have the following structure: [('ON', 'L3T0A1', 'Canada', ‘Markham’)].
The **overlapping_locations** column was created by intersection of **location_dataset1** and **location_dataset2:**

**list(set(row[‘locations_dataset1’]).intersection(row[‘locations_dataset2']))).**

For more detailed segment of code see **test_final_kotov.py.**

# Match Rate

To calculate the **match rate**, it was necessary to determine both the total number of rows in the merged dataset and the number of companies in **dataset1_cleaned** that had at least one corresponding match in **dataset2_cleaned**.

The match rate was calculated as:

**Match rate = (number of matched companies / total number of companies) × 100**

In this particular case, the match rate is **41.95%**.

# Unmatch Rate

The **unmatched rate** represents the proportion of companies in **dataset1_cleaned** that have no corresponding match in **dataset2_cleaned**. More generally, it refers to companies that appear in **dataset1_cleaned** but not in **dataset2_cleaned**, as well as companies that appear in **dataset2_cleaned** but not in **dataset1_cleaned**.

To calculate this, set-based operations, specifically **union** and **intersection,** were used. First, all unique companies across both datasets were identified using a union operation. Next, the intersection of the two datasets was computed to determine the set of **matched companies**.

The unmatched companies were then calculated as:

**unmatched_companies = all_companies - matched_companies**

Finally, the unmatched rate was computed as:

**Unmatched rate = (number of unmatched companies / total number of companies) × 100**

In this case, the unmatched rate is **71.05%**.

## One to Many Matches

**One-to-many matches** refer to the proportion of companies in **dataset1_cleaned** that have multiple corresponding matches in **dataset2_cleaned**. This was identified using the **location_dataset2** field in the merged dataset, which contains the list of matched locations for each company from **dataset2_cleaned**.

At this final step, all companies with more than one match in **dataset2_cleaned** were counted. The one-to-many match rate was then calculated as:

**One-to-many rate = (number of companies with multiple matches / total number of companies) × 100**

In this case, the one-to-many match rate is **5.43%**.

## Data Quality Issues

During the initial observation stage, it was noted that both datasets contained highly messy real world data. A significant amount of time was spent cleaning and standardizing this data, which is a crucial step before performing any calculations or matching tasks.

**Quality issues identified:**

- Redundant whitespace before, after, and within data fields.
- Misspelled values, particularly for states and countries.
- Unnecessary special characters (e.g., ., *, -, /, &, #) appearing before or after values.
- Inconsistent use of letter casing (lowercase, uppercase, mixed case).
- Messy and unclear column names.
- Inconsistent company name formats, including differences in casing, punctuation, and legal suffixes (e.g. *"Inc"*, *"Ltd"*, *"INC"*, *"Inc."*, *"Limited"*).

# Normalization / Transformation Applied

The main purpose of normalization and transformation is to ensure consistent data formatting, which is essential for reliable data manipulation and matching. During the dataset investigation, the following data normalization were applied:

- Inconsistent address abbreviations (e.g. *"St", "Street", "Blvd", "Ln", "Ave"*) were standardized into a common full format (e.g. *"Street", "Avenue", "Boulevard"*).

- **Company names** and **address** fields were converted to a consistent case.

- Punctuation was removed.

- Leading and trailing whitespace was trimmed.

- Based on the **postal_zip** field, missing or incorrect **country** and **state** values were restored where possible.

- All state and province values were standardized to a consistent format (e.g. *"On"* —> *"ON", "Ontario"* —> *"ON"*).