I created a Python-based cleaning and fuzzy matching tool that compares donor employer fields with executive company names. I tested it successfully in Google Colab using the matched_name_state dataset. The output includes match scores and flags, and it's ready for the next step — using AI to assess political leaning or validate match quality.

finding **relationships between donors and company executives**, or **to detect possible influence, patterns, or affiliations**.

What You Need To Do

1. **Scroll up to the first code cell** where it says:

python
CopyEdit

```
uploaded = files.upload()
```

2. **Click the play button ▶️ on that cell again**

   🔁 This will reopen the file picker so you can **upload your CSV**.

3. Once you upload the file:
   - It will automatically move to the processing steps
   - The script will clean, match, and output the results as `matched_output.csv`

How to Add the Fix and Run the Notebook in Google Colab

📍 Where to Paste the Fix:

1. **Go to the top of the notebook**
2. Find the **first gray box with code** (where the imports are — e.g., `import pandas as pd`)

3. Click **just above that cell**
4. Click **`+ Code`** (you'll see it at the top-left near the toolbar)

This creates a new empty code cell **above the current first one**.

5. **Paste this line into that new cell**:

python
CopyEdit

```
!pip install fuzzywuzzy python-Levenshtein
```

▶️ How to Run Each Cell:

- **Click the gray play button** at the top-left of each code cell
 OR
- **Click inside the cell** and press **Shift + Enter** on your keyboard

---

✅ After You Run the Install Cell:

- Wait until it says `Successfully installed fuzzywuzzy` (takes a few seconds)
- Then run the next cell (starting with `import pandas as pd`)
- Continue running the rest of the notebook cells one at a time

🟩 Cell 1: Import Libraries

python

CopyEdit

```
import pandas as pd

from fuzzywuzzy import fuzz
```

```
import re

import string

from google.colab import files

import io
```

🧠 **Purpose**: Import required Python libraries:

- `pandas` → for data manipulation.
- `fuzzywuzzy` → for fuzzy string matching.
- `re`, `string` → for regex and punctuation handling.
- `google.colab.files` + `io` → for uploading and reading files in Google Colab.

🟦 Cell 2: Upload CSV File

python

CopyEdit

```
uploaded = files.upload()

file_name = list(uploaded.keys())[0]

df = pd.read_csv(io.BytesIO(uploaded[file_name]))
```

📁 **Purpose**:

- Prompts the user to upload a CSV file.
- Extracts the file name.
- Reads the uploaded CSV into a pandas DataFrame (`df`).

🟨 Cell 3: Define Cleaning Function

python

CopyEdit

```
def clean_name(name):
    if not isinstance(name, str):
        return ""
    name = name.lower()
    name = name.translate(str.maketrans("", "", string.punctuation))
    suffixes = [' inc', ' corp', ' corporation', ' llc', ' ltd', ' limited',
                ' company', ' co', ' incorporated', ' group', ' holdings']
    for suffix in suffixes:
        name = re.sub(suffix + r"$", "", name)
        name = re.sub(suffix + r" ", " ", name)
    return " ".join(name.split())
```

🧹 **Purpose**:

- Cleans employer names to standardize them:
    - Makes text lowercase.
    - Removes punctuation.
    - Strips common business suffixes.

    - Trims extra spaces.

🟪 Cell 4: Apply Cleaning Function

python

CopyEdit

```
df['cleaned_employer'] = df['EMPLOYERorOCCUPATION'].apply(clean_name)
df['cleaned_exec_employer'] = df['exec_employer'].apply(clean_name)
```

🧼 **Purpose**:

- Applies the cleaning function to the employer-related columns.
- Creates two new columns: `cleaned_employer` and `cleaned_exec_employer`.

🟥 Cell 5: Fuzzy Matching

python

CopyEdit

```
df['match_score'] = df.apply(
    lambda row: fuzz.token_sort_ratio(row['cleaned_employer'], row['cleaned_exec_employer'])
    if row['cleaned_employer'] and row['cleaned_exec_employer'] else 0,
    axis=1
)
```

🔍 **Purpose**:

- Compares each pair of cleaned employer strings using fuzzy matching.
- Uses `token_sort_ratio` which ignores word order.
- Scores range from 0 (no similarity) to 100 (perfect match).
- Saves scores in a new column: `match_score`.

🟧 Cell 6: Flag Matching Status

python

CopyEdit

```
df['match_status'] = df['match_score'].apply(lambda x: 'Match' if x >= 90 else 'Needs Review')
```

✅ ❌ **Purpose**:

- Labels each row as:
    - `'Match'` if score ≥ 90.
    - `'Needs Review'` otherwise.
- Helps prioritize manual review.

🟫 Cell 7: Preview Results

python

CopyEdit

```
df[['NAME', 'EMPLOYERorOCCUPATION', 'exec_fullname', 'exec_employer',
    'cleaned_employer', 'cleaned_exec_employer', 'match_score',
'match_status']].head(10)
```

🔎 **Purpose**:

- Displays first 10 rows with relevant columns.
- Helps you check if the cleaning and matching worked properly.

Would you like me to turn this into an actual infographic-style image you can download or include in a presentation?

4o

You said:

what is the criteria for giving the match rating

ChatGPT said:

Great question! The **match rating** in your code is determined using this line:

python

CopyEdit

```
fuzz.token_sort_ratio(row['cleaned_employer'],
row['cleaned_exec_employer'])
```

🔍 What does `fuzz.token_sort_ratio()` do?

It's part of the `fuzzywuzzy` library, and it compares two strings based on how similar their **tokens** (words) are, regardless of order.

📐 How the score is calculated:

1. **Tokenize** each string (split into words).
2. **Sort** the tokens alphabetically.
3. **Join** the tokens back into a string.
4. Use **Levenshtein distance** (edit distance) to calculate similarity between the two strings (after sorting).
5. Return a **score between 0 and 100**, where:
    - **100 = Perfect Match** (after cleaning and sorting words).
    - **0 = Completely Different**.

✅ Example:

String A	String B	Score
`"apple inc"`	`"inc apple"`	100
`"google llc"`	`"google"`	~93
`"amazon.com inc"`	`"amazon"`	~88
`"netflix"`	`"net flix corp"`	~82
`"facebook"`	`"meta"`	~30

Cleaning helps by removing suffixes like "inc", "corp", etc., making matches more accurate.