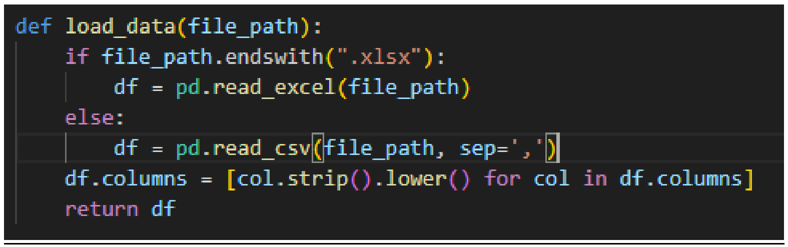
The system created follows the following pipeline:

Input CSV → Data Loading → Entity Resolution Scoring → Data Quality Assessment → Results Export into a SQL database (SSMS this time)

Data Loading



```python
def load_data(file_path):
    if file_path.endswith(".xlsx"):
        df = pd.read_excel(file_path)
    else:
        df = pd.read_csv(file_path, sep=',')
    df.columns = [col.strip().lower() for col in df.columns]
    return df
```

The first snippet of the code handles data based on the file format. In this case we are using a CSV file as a starting point. This part of the code handles the data normalization, more exactly in case of a CSV file it converts all column names to lowercase and strips whitespace.

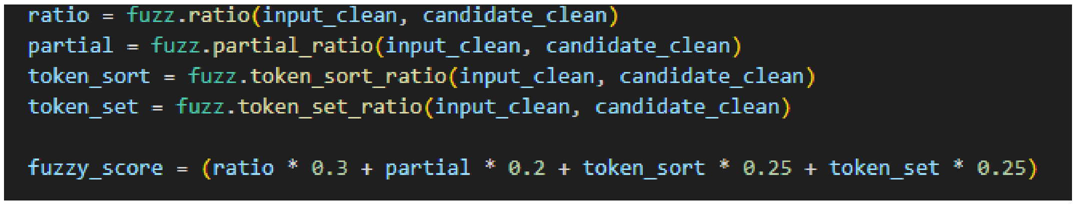
Entity Resolution Scoring

In this section I used the "fuzzywuzzy" library to score the matches.

This part of the code is comprised of 3 subparts that handle: name similarity, if there is a geographic match, compares business context (such as the name of the company, description etc) and in the end it assigns an overall score for the entry.

Name Similarity Scoring

Measures how similar two company names are, accounting for variations in formatting, abbreviations, and word order.



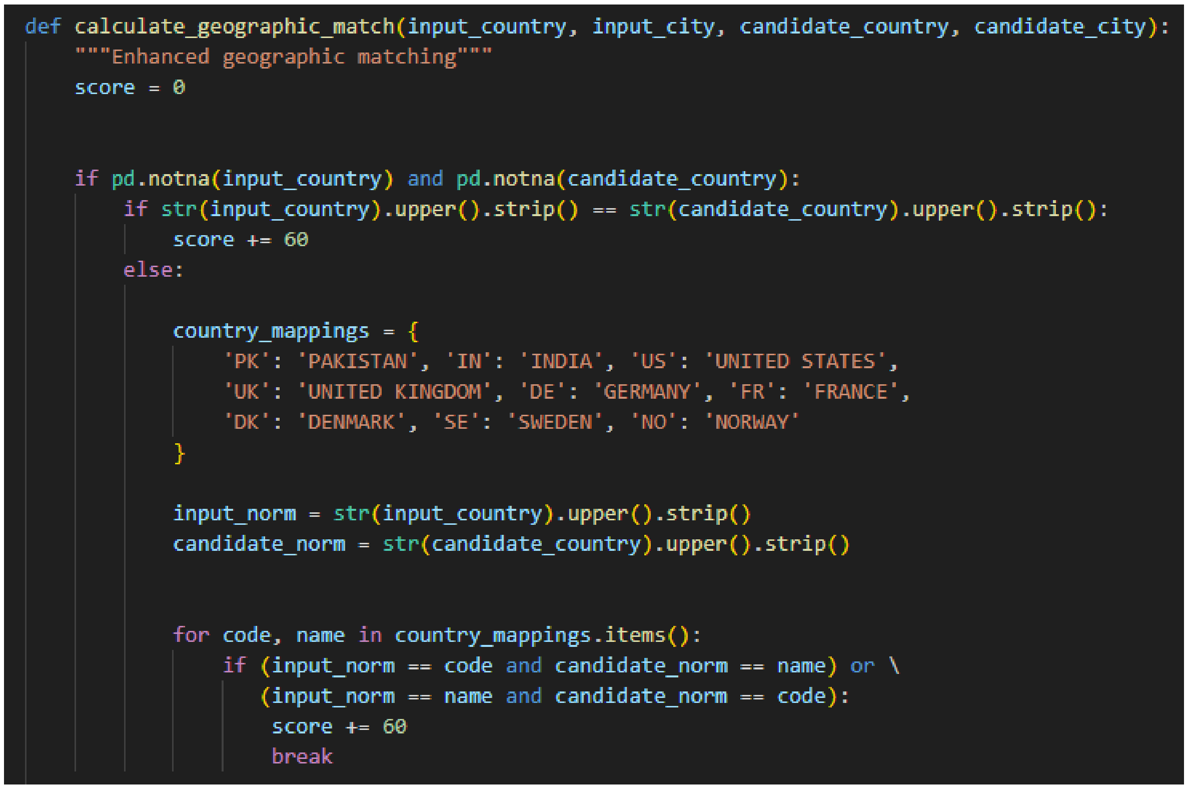
```python
ratio = fuzz.ratio(input_clean, candidate_clean)
partial = fuzz.partial_ratio(input_clean, candidate_clean)
token_sort = fuzz.token_sort_ratio(input_clean, candidate_clean)
token_set = fuzz.token_set_ratio(input_clean, candidate_clean)

fuzzy_score = (ratio * 0.3 + partial * 0.2 + token_sort * 0.25 + token_set * 0.25)
```

Geographical Matching

Here I am determining if two companies are located in the same geographic area.

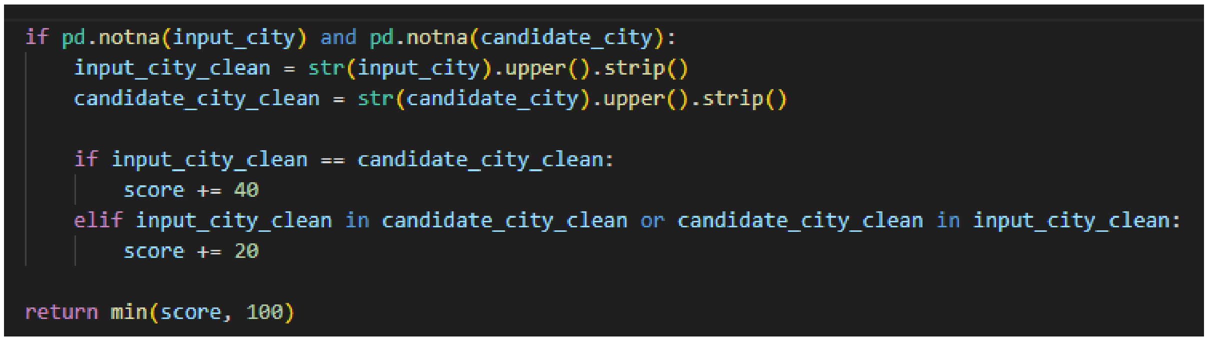
Country matching:

Country counts more heavily due to being a more reliable



```python
def calculate_geographic_match(input_country, input_city, candidate_country, candidate_city):
    """Enhanced geographic matching"""
    score = 0


    if pd.notna(input_country) and pd.notna(candidate_country):
        if str(input_country).upper().strip() == str(candidate_country).upper().strip():
            score += 60
        else:

            country_mappings = {
                'PK': 'PAKISTAN', 'IN': 'INDIA', 'US': 'UNITED STATES',
                'UK': 'UNITED KINGDOM', 'DE': 'GERMANY', 'FR': 'FRANCE',
                'DK': 'DENMARK', 'SE': 'SWEDEN', 'NO': 'NORWAY'
            }

            input_norm = str(input_country).upper().strip()
            candidate_norm = str(candidate_country).upper().strip()


            for code, name in country_mappings.items():
                if (input_norm == code and candidate_norm == name) or \
                   (input_norm == name and candidate_norm == code):
                    score += 60
                    break
```

City matching:

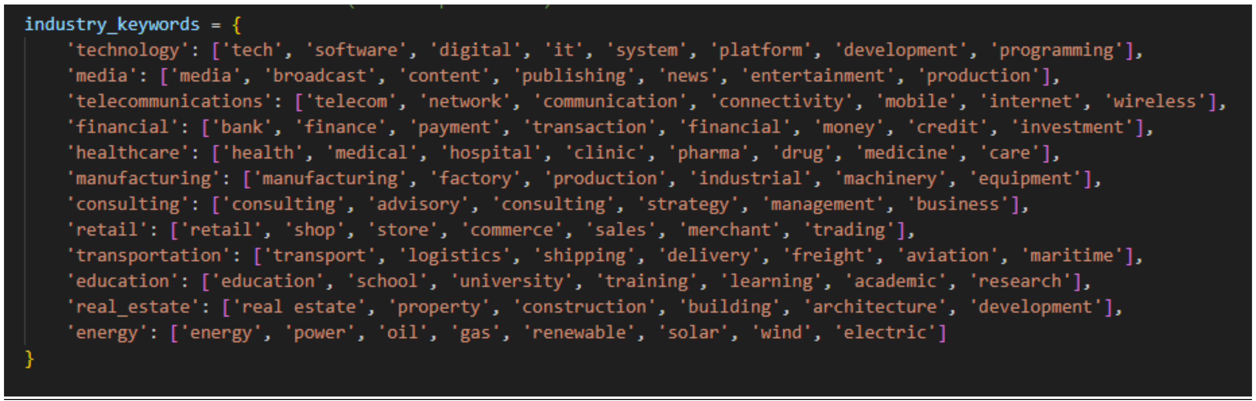


```python
if pd.notna(input_city) and pd.notna(candidate_city):
    input_city_clean = str(input_city).upper().strip()
    candidate_city_clean = str(candidate_city).upper().strip()

    if input_city_clean == candidate_city_clean:
        score += 40
    elif input_city_clean in candidate_city_clean or candidate_city_clean in input_city_clean:
        score += 20

return min(score, 100)
```

Business Context Scoring

This matching works in layers.

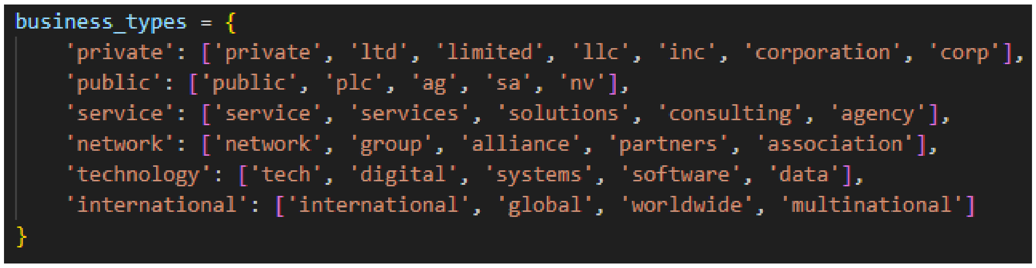
Layer 1: Industry keyword matching - 25 points



```python
industry_keywords = {
    'technology': ['tech', 'software', 'digital', 'it', 'system', 'platform', 'development', 'programming'],
    'media': ['media', 'broadcast', 'content', 'publishing', 'news', 'entertainment', 'production'],
    'telecommunications': ['telecom', 'network', 'communication', 'connectivity', 'mobile', 'internet', 'wireless'],
    'financial': ['bank', 'finance', 'payment', 'transaction', 'financial', 'money', 'credit', 'investment'],
    'healthcare': ['health', 'medical', 'hospital', 'clinic', 'pharma', 'drug', 'medicine', 'care'],
    'manufacturing': ['manufacturing', 'factory', 'production', 'industrial', 'machinery', 'equipment'],
    'consulting': ['consulting', 'advisory', 'consulting', 'strategy', 'management', 'business'],
    'retail': ['retail', 'shop', 'store', 'commerce', 'sales', 'merchant', 'trading'],
    'transportation': ['transport', 'logistics', 'shipping', 'delivery', 'freight', 'aviation', 'maritime'],
    'education': ['education', 'school', 'university', 'training', 'learning', 'academic', 'research'],
    'real_estate': ['real estate', 'property', 'construction', 'building', 'architecture', 'development'],
    'energy': ['energy', 'power', 'oil', 'gas', 'renewable', 'solar', 'wind', 'electric']
}
```

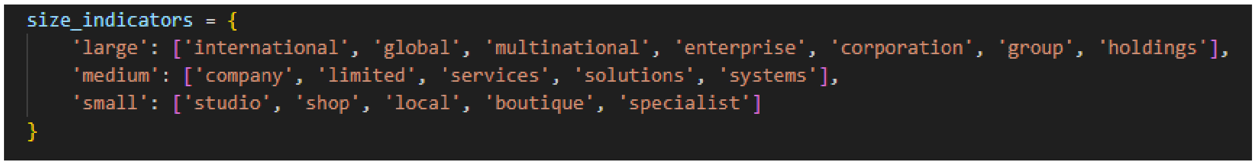
Here we have a dictionary that is looking for keywords to award more score to a match.

Layer 2: Business type matching - 15 points



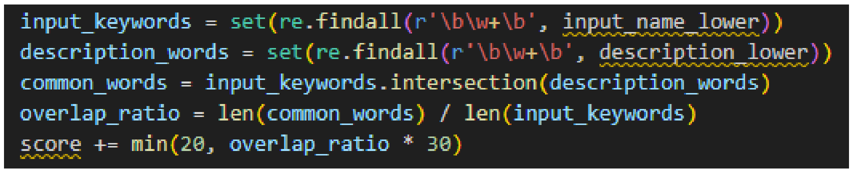
```python
business_types = {
    'private': ['private', 'ltd', 'limited', 'llc', 'inc', 'corporation', 'corp'],
    'public': ['public', 'plc', 'ag', 'sa', 'nv'],
    'service': ['service', 'services', 'solutions', 'consulting', 'agency'],
    'network': ['network', 'group', 'alliance', 'partners', 'association'],
    'technology': ['tech', 'digital', 'systems', 'software', 'data'],
    'international': ['international', 'global', 'worldwide', 'multinational']
}
```

Layer 3: Company Size Indicators - 10 points



```python
size_indicators = {
    'large': ['international', 'global', 'multinational', 'enterprise', 'corporation', 'group', 'holdings'],
    'medium': ['company', 'limited', 'services', 'solutions', 'systems'],
    'small': ['studio', 'shop', 'local', 'boutique', 'specialist']
}
```

Layer 4: Fuzzy Description Matching - 20 points



```python
input_keywords = set(re.findall(r'\b\w+\b', input_name_lower))
description_words = set(re.findall(r'\b\w+\b', description_lower))
common_words = input_keywords.intersection(description_words)
overlap_ratio = len(common_words) / len(input_keywords)
score += min(20, overlap_ratio * 30)
```

Layer 5 & 6: Tags and Industry Classification - 15 + 20 points

Uses structured business tags if available, also leverages formal industry classifications

Real Example:

- Input: "24-SEVEN MEDIA NETWORK (PRIVATE) LIMITED"
- Candidate Description: "MNET provides secure electronic inter-bank connectivity platform for online financial transactions"
- Industry Match: Media vs Financial (0 points)
- Business Type: "PRIVATE" and "LIMITED" vs financial services (15 points)
- Description Overlap: "NETWORK" appears in both (10 points)
- **Total Business Score: 25/100**

After all these factors are calculated the next step is to calculate the overall score according to the following formula:

```
overall_score = (name_score * 0.4 + geo_score * 0.4 + business_score * 0.2)
```

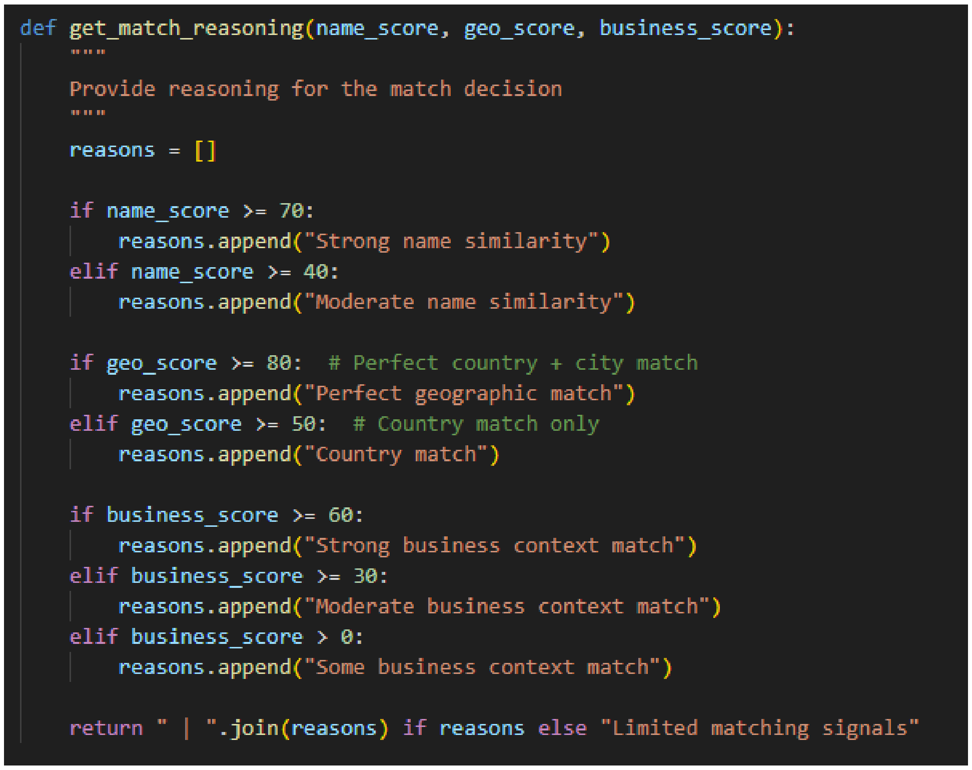
After that, a recommendation is assigned based on the overall score:

```
if overall_score >= 80:
    recommendation = 'STRONG_ACCEPT'
elif overall_score >= 70:
    recommendation = 'ACCEPT'
elif overall_score >= 50:
    recommendation = 'REVIEW'
else:
    recommendation = 'REJECT'
```

Business Logic:

- **STRONG_ACCEPT**: Automate processing, high confidence
- **ACCEPT**: Good for most business uses, acceptable risk
- **REVIEW**: Requires human analyst review
- **REJECT**: Not a match, don't process

There is also a match reasoning that helps identity potential issues with the match.



```python
def get_match_reasoning(name_score, geo_score, business_score):
    """
    Provide reasoning for the match decision
    """
    reasons = []

    if name_score >= 70:
        reasons.append("Strong name similarity")
    elif name_score >= 40:
        reasons.append("Moderate name similarity")

    if geo_score >= 80:  # Perfect country + city match
        reasons.append("Perfect geographic match")
    elif geo_score >= 50:  # Country match only
        reasons.append("Country match")

    if business_score >= 60:
        reasons.append("Strong business context match")
    elif business_score >= 30:
        reasons.append("Moderate business context match")
    elif business_score > 0:
        reasons.append("Some business context match")

    return " | ".join(reasons) if reasons else "Limited matching signals"
```

This code adds and appends data in the match_reason column to show some information about how the score was given.

Example Output: **"Strong name similarity | Country match | Moderate business context match"**

All these results are exported into three CSV files:

- veridion_analysis_entity_resolution.csv - Complete matching results
- veridion_analysis_best_matches.csv - Top recommendations
- veridion_analysis_quality_issues.csv - Data quality problems (human review and validation is recommended here)

In the end all these files are pushed in an SQL database (using SSMS for this case) where the data can be used to generate reports.