Part 1: Data Quality

First we check coverage without any input data modification.

In [10]:
```
import pandas as pd

file1 = 'test.csv'
file2 = 'all_addresses.csv'

df1 = pd.read_csv(file1)
df2 = pd.read_csv(file2)

df1_row = df1.shape[0]
print(f"No. of rows in Test.csv: {df1_row}")

df2_row = df2.shape[0]
print(f"No. of rows in all_addresses.csv: {df2_row}")

#For comparing zip codes we need them in same data type.
df1['zip'] = df1['zip'].astype(str).str.rstrip('.0')


non_matching_rows = pd.merge(df1, df2, on=['address', 'city', 'state', 'zip'],
non_matching_rows = non_matching_rows[non_matching_rows['_merge'] == 'left_only


print(f"Non-Matching Rows: {non_matching_rows.shape[0]}")
print(f"Total no.of rows: {total_number_of_rows}")


# Calculate coverage percentage

number_of_matching_rows = total_number_of_rows - non_matching_rows.shape[0]
coverage_percentage = (number_of_matching_rows / total_number_of_rows) * 100

print(f"Coverage Percentage: {coverage_percentage:.2f}%")

# Save non-matching rows to a new CSV file
non_matching_rows.to_csv('non_matching_rows.csv', index=False)
print("Non-matching rows saved to 'non_matching_rows_initially.csv'")
```

```
No. of rows in Test.csv: 99249
No. of rows in all_addresses.csv: 130000
Non-Matching Rows: 50428
Total no.of rows: 99249
Coverage Percentage: 49.19%
Non-matching rows saved to 'non_matching_rows_initially.csv'
```

Now we modify the input data to add missing information, fix format and increase the coverage percentage.

In [2]:
```
#This function take two strings and return the number of characters that are di
def count_different_words(str1, str2):
    if pd.isnull(str1) or pd.isnull(str2):
        return float('inf')
```

```python
    words1 = str1.split()
    words2 = str2.split()
    min_length = min(len(words1), len(words2))
    different_count = 0

    for i in range(min_length):
        if words1[i] != words2[i]:
            different_count += 1

    different_count += abs(len(words1) - len(words2))
    return different_count

#This function corrent the address if 3 or less than 3 words are different betw
def replace_missing_words(original, replacement):
    orig_words = original.split()
    repl_words = replacement.split()

    min_length = min(len(orig_words), len(repl_words))
    new_words = []

    for i in range(min_length):
        if count_different_words(orig_words[i], repl_words[i]) <= 3:
            new_words.append(repl_words[i])
        else:
            new_words.append(orig_words[i])

    new_words.extend(orig_words[min_length:])
    new_address = ' '.join(new_words)
    return new_address
```

In [12]:

```python
import pandas as pd

file1 = 'test.csv'
file2 = 'all_addresses.csv'

df1 = pd.read_csv(file1)
df2 = pd.read_csv(file2)

net_rows = df1.shape[0]
total_number_of_rows = df1.shape[0]


#For comparing zip codes we need them in same data type.
df1['zip'] = df1['zip'].astype(str).str.rstrip('.0')

#We do some pre-processing to fix few minor things such as removing Unit # and
df1['address'] = df1['address'].str.replace(r'\bUnit \d+\b', '', regex=True).st
df1['address'] = df1['address'].str.replace(',', '', regex=True).str.strip()
df1['address'] = df1['address'].str.replace(r'(\d+)([A-Za-z])', r'\1 \2', regex

#We compare state and zip to fix the address if 3 or less than 3 words are diff
def update_address(row):
    match_rows = df2[
        (df2['state'] == row['state']) &
        (df2['zip'] == row['zip'])
    ]
    def replace_func(match_row):
        if count_different_words(row['address'], match_row['address']) <= 3:
            return replace_missing_words(row['address'], match_row['address'])
        else:
            return row['address']

    new_addresses = match_rows.apply(replace_func, axis=1).dropna()

    if not new_addresses.empty:
        return new_addresses.iloc[0]
    else:
        return row['address']

print('Starting address update')
df1['address'] = df1.apply(update_address, axis=1)

print('Starting City update for missing rows')
for index, row in df1.iterrows():
    if pd.isnull(row['city']):
        match_rows = df2[
            (df2['state'] == row['state']) &
            (df2['zip'] == row['zip'])
        ]

        for _, match_row in match_rows.iterrows():
            if row['address'] == match_row['address']:
                df1.at[index, 'city'] = match_row['city']
                break

def update_info(row):
    match_rows = df2[
        (df2['address'] == row['address'])
    ]

    if not match_rows.empty:
        match_row = match_rows.iloc[0]

        if row['state'] != match_row['state']:
            row['state'] = match_row['state']

        if row['city'] != match_row['city']:
            row['city'] = match_row['city']

        if row['zip'] != match_row['zip']:
            row['zip'] = match_row['zip']

    return row


print('Updating incorrent info column value')
df1 = df1.apply(update_info, axis=1)



# Perform an outer merge to capture non-matching rows
non_matching_rows = pd.merge(df1, df2, on=['address', 'city', 'state', 'zip'],
non_matching_rows = non_matching_rows[non_matching_rows['_merge'] == 'left_only


print(f"Non-Matching Rows: {non_matching_rows.shape[0]}")
print(f"Total no.of rows: {total_number_of_rows}")
```

```python
# Calculate coverage percentage

number_of_matching_rows = total_number_of_rows - non_matching_rows.shape[0]
coverage_percentage = (number_of_matching_rows / total_number_of_rows) * 100

print(f"Coverage Percentage: {coverage_percentage:.2f}%")

# Save non-matching rows to a new CSV file
non_matching_rows.to_csv('non_matching_rows.csv', index=False)
print("Non-matching rows saved to 'non_matching_rows_final.csv'")
```

```
Starting address update
Starting City update for missing rows
Updating incorrent info column value
Non-Matching Rows: 21566
Total no.of rows: 99249
Coverage Percentage: 78.27%
Non-matching rows saved to 'non_matching_rows_final.csv'
```

Coverage percentage was increased from 49.1% to 78.27%. In slides we discuss some more ways to increase the coverage in future.