```
!pip install fuzzywuzzy
```

```
Collecting fuzzywuzzy
  Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl.metadata (4.9 kB)
Downloading fuzzywuzzy-0.18.0-py2.py3-none-any.whl (18 kB)
Installing collected packages: fuzzywuzzy
Successfully installed fuzzywuzzy-0.18.0
```

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

```
Collecting python-Levenshtein
  Downloading python_Levenshtein-0.26.1-py3-none-any.whl.metadata (3.7 kB)
Collecting Levenshtein==0.26.1 (from python-Levenshtein)
  Downloading levenshtein-0.26.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.2 kB)
Collecting rapidfuzz<4.0.0,>=3.9.0 (from Levenshtein==0.26.1->python-Levenshtein)
  Downloading rapidfuzz-3.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Downloading python_Levenshtein-0.26.1-py3-none-any.whl (9.4 kB)
Downloading levenshtein-0.26.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (162 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 162.6/162.6 kB 5.1 MB/s eta 0:00:00
Downloading rapidfuzz-3.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 40.0 MB/s eta 0:00:00
Installing collected packages: rapidfuzz, Levenshtein, python-Levenshtein
Successfully installed Levenshtein-0.26.1 python-Levenshtein-0.26.1 rapidfuzz-3.10.1
```

```
# Import necessary libraries
import numpy as np  # For numerical operations
import pandas as pd  # For data manipulation
from fuzzywuzzy import fuzz  # For fuzzy string matching
from fuzzywuzzy import process  # For advanced fuzzy operations
```

```
# Load the dataset
df = pd.read_csv('tested.csv')  # Replace 'test.csv' with your dataset filename
```

```
# Display the first few rows of the dataset
print("Dataset Preview:")
print(df.head())

# Display dataset information
print("\nDataset Information:")
print(df.info())

# Display summary statistics
print("\nDataset Summary Statistics:")
print(df.describe(include='all'))
```

```
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
```

```
75%     1204.750000    1.000000    3.000000                NaN   NaN
max     1309.000000    1.000000    3.000000                NaN   NaN

                Age       SibSp       Parch    Ticket        Fare  \
count   332.000000  418.000000  418.000000       418  417.000000
unique         NaN         NaN         NaN       363         NaN
top            NaN         NaN         NaN  PC 17608         NaN
freq           NaN         NaN         NaN         5         NaN
mean     30.272590    0.447368    0.392344       NaN   35.627188
std      14.181209    0.896760    0.981429       NaN   55.907576
min       0.170000    0.000000    0.000000       NaN    0.000000
25%      21.000000    0.000000    0.000000       NaN    7.895800
50%      27.000000    0.000000    0.000000       NaN   14.454200
75%      39.000000    1.000000    0.000000       NaN   31.500000
max      76.000000    8.000000    9.000000       NaN  512.329200

                  Cabin Embarked
count                91      418
unique               76        3
top     B57 B59 B63 B66        S
freq                  3      270
mean                NaN      NaN
std                 NaN      NaN
min                 NaN      NaN
25%                 NaN      NaN
50%                 NaN      NaN
75%                 NaN      NaN
max                 NaN      NaN
```

```
# Check for missing values
print("\nMissing Values in the Dataset:")
print(df.isnull().sum())

# Display columns with missing values
columns_with_na = df.columns[df.isnull().any()]
print("\nColumns with Missing Values:", columns_with_na)
```

```
Missing Values in the Dataset:
PassengerId      0
Survived         0
Pclass           0
```

```
Name            0
Sex             0
Age            86
SibSp           0
Parch           0
Ticket          0
Fare            1
Cabin         327
Embarked        0
dtype: int64

Columns with Missing Values: Index(['Age', 'Fare', 'Cabin'], dtype='object')
```

```
# Handle missing values
df['Age'].fillna(df['Age'].mean(), inplace=True)  # Fill missing 'Age' with mean
df['Fare'].fillna(df['Fare'].median(), inplace=True)  # Fill missing 'Fare' with median
df.drop('Cabin', axis=1, inplace=True)  # Drop 'Cabin' column due to excessive missing data

# Verify no missing values remain
print("\nMissing Values After Processing:")
print(df.isnull().sum())
```

```
Missing Values After Processing:
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
<ipython-input-8-91e6e71abc8c>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me
```

```
  df['Age'].fillna(df['Age'].mean(), inplace=True)  # Fill missing 'Age' with mean
<ipython-input-8-91e6e71abc8c>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignme
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Fare'].fillna(df['Fare'].median(), inplace=True)  # Fill missing 'Fare' with median
```

```
# Display unique values in specific columns
print("\nUnique Values in Embarked Column:")
print(df['Embarked'].unique())

# Count distribution of a specific column
print("\nValue Counts for 'Survived':")
print(df['Survived'].value_counts())
```

```
Unique Values in Embarked Column:
['Q' 'S' 'C']

Value Counts for 'Survived':
Survived
0    266
1    152
Name: count, dtype: int64
```

```
# Function to find potential duplicates using fuzzy matching
def find_duplicates(dataframe, column, threshold=80):
    duplicates = []
    for i, name in enumerate(dataframe[column]):
        for j, other_name in enumerate(dataframe[column]):
            if i != j:  # Avoid self-comparison
                similarity = fuzz.token_sort_ratio(name, other_name)
                if similarity >= threshold:
                    duplicates.append((name, other_name, similarity))
    return duplicates
```

```
# Apply the function to the 'Name' column
potential_duplicates = find_duplicates(df, 'Name', threshold=80)

# Display potential duplicates
print("\nPotential Duplicates Found:")
for duplicate in potential_duplicates[:10]:  # Show only the first 10 for brevity
    print(duplicate)
```

```
Potential Duplicates Found:
('Brady, Mr. John Bertram', 'Crafton, Mr. John Bertram', 82)
('Davison, Mr. Thomas Henry', 'Conlon, Mr. Thomas Henry', 80)
('Kiernan, Mr. John', 'Kennedy, Mr. John', 80)
('Kiernan, Mr. John', 'Lingane, Mr. John', 80)
('Kennedy, Mr. John', 'Kiernan, Mr. John', 80)
('Crafton, Mr. John Bertram', 'Brady, Mr. John Bertram', 82)
('Lingane, Mr. John', 'Kiernan, Mr. John', 80)
('Dennis, Mr. William', 'Dibden, Mr. William', 82)
('Dibden, Mr. William', 'Dennis, Mr. William', 82)
('Dibden, Mr. William', 'Gilbert, Mr. William', 80)
```

```
# Deduplicate dataset by retaining unique names
unique_names = set()
deduplicated_rows = []

for _, row in df.iterrows():
    name = row['Name']
    if name not in unique_names:
        deduplicated_rows.append(row)
        unique_names.add(name)

# Create a new DataFrame with deduplicated rows
df_cleaned = pd.DataFrame(deduplicated_rows)

# Check the shape of the cleaned dataset
print("\nShape of Cleaned Dataset:")
print(df_cleaned.shape)
```

```
Shape of Cleaned Dataset:
(418, 11)
```

```
# Save the cleaned dataset
df_cleaned.to_csv('cleaned_data.csv', index=False)

print("\nCleaned dataset has been saved as 'cleaned_data.csv'.")
```

```
Cleaned dataset has been saved as 'cleaned_data.csv'.
```

Start coding or generate with AI.