Data Cleaning Process Documentation

Objective:

To clean and validate the dataset named messy_Data.csv and ensure it is ready for analysis by following specific data cleaning instructions.

1. Loading the Dataset

Objective: Load the messy dataset into a Jupyter notebook for cleaning.

```
import pandas as pd


df = pd.read_csv(r"C:\Users\user\Downloads\messy_data.csv")

df
```

The dataset comprises 11,000 rows and 8 columns of structured data

2. Inspect the Data

Objective: Examine the dataset to understand its structure and identify errors and inconsistencies.

Steps:

- Check the summary of the dataframe including the number of entries, column data types and also the the structure and integrity of the dataset.
- Check the first few rows of the dataset to understand its structure.
- identify and print the count of missing values (null values) in each column of the data frame.
- check the number of duplicate rows in a Pandas DataFrame, providing insight into the presence of duplicated observations within the dataset.
- Check unique values found in the 'Department' column of a Pandas DataFrame, displaying the distinct department names recorded in the dataset.
- Checks the first 21 entries from the 'Email' column of a Pandas DataFrame , showing a subset of email addresses stored in the dataset.
- Checking a comprehensive statistical summary of DataFrame

```
df.info()


import pandas as pd
```

```
    df = pd.read_csv(r"C:\Users\user\Downloads\messy_data.csv")
    df.head()
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])


duplicate_rows = df.duplicated().sum()
print(f'Number of duplicate rows: {duplicate_rows}')


unique_departments = df['Department'].unique()
print(f'Unique departments: {unique_departments}')


print(df['Email'].head(21))


df.describe()
```

3. Handle Missing Values

Objective: Identify and handle missing values in the dataset.

Assumptions: Missing values are handled based on the nature of the data (numeric or categorical) and the importance of the column for analysis.

Methods: Use of statistical measures (mean, median) for numeric data, defaults for categorical data, and dropping rows for critical missing values ensure the dataset is cleaned and ready for further analysis or modeling.

```
import pandas as pd

file_path = r'C:\Users\user\Downloads\messy_data.csv'

df = pd.read_csv(file_path)

missing_values_count = df.isnull().sum()

print("Missing values in each column:")

print(missing_values_count)

median_salary = df['Salary'].median()
```

```
df['Salary'].fillna(median_salary, inplace=True)

mean_age = df['Age'].mean()

df['Age'].fillna(mean_age, inplace=True)

default_department = 'HR'

df['Department'].fillna(default_department, inplace=True)

default_email = 'unknown@domain.com'

df['Email'].fillna(default_email, inplace=True)

default_join_date = '2000-01-01'

df['Join Date'].fillna(default_join_date, inplace=True)

df.dropna(subset=['Name'], inplace=True)

missing_values_count_after = df.isnull().sum()

print("\nMissing values in each column after handling:")

print(missing_values_count_after)


cleaned_file_path = r'C:\Users\user\Downloads\cleaned_data.csv'

df.to_csv(cleaned_file_path, index=False)


print("\nMissing values handled and cleaned dataset saved.")
```

4. Remove Duplicates

Objective: Identify and remove duplicate rows to ensure data integrity.

Methods Used: Uses the drop_duplicates() method of Pandas DataFrame to remove duplicates based on all columns by default, and shape attribute to count rows.

```
import pandas as pd

file_path = r'C:\Users\user\Downloads\messy_data.csv'
```

```
df = pd.read_csv(file_path)
num_rows_before = df.shape[0]
print(f"Number of rows before removing duplicates: {num_rows_before}")
df = df.drop_duplicates()
num_rows_after = df.shape[0]
print(f"Number of rows after removing duplicates: {num_rows_after}")
cleaned_file_path = r'C:\Users\user\Downloads\cleaned_data.csv'
df.to_csv(cleaned_file_path, index=False)


print("Duplicate rows removed and cleaned dataset saved.")
```

- Display the count of duplicates to check it updated

```
import pandas as pd
file_path = r'C:\Users\user\Downloads\cleaned_data.csv'
df = pd.read_csv(file_path)
num_duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {num_duplicates}")
```

5. Correct Email Formats and Ensure Professional Emails

Objective: Identify and correct invalid email formats. Ensure all emails are professional.

Methods used:

- **Validation Functions**: is_valid_email checks for standard email format, while is_professional_email verifies if the domain is professional.
- **Cleaning**: Normalizes email addresses by stripping whitespace and converting to lowercase.
- **Filtering**: Removes rows with invalid or non-professional emails.
- **Output**: Saves the cleaned data to a new CSV file and displays the cleaned DataFrame for verification in an IPython environment.

- Display all Emails in the Email coloumn

```
import pandas as pd
file_path = r'C:\Users\user\Downloads\cleaned_data.csv'
```

```
df = pd.read_csv(file_path)

emails = df['Email'].tolist()
for email in emails:
    print(email)

import pandas as pd
import re
import numpy as np

def is_valid_email(email):
    if pd.isna(email):
        return False
    email_regex = re.compile(r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,}$")
    return re.match(email_regex, email) is not None

def is_professional_email(email):
    if pd.isna(email) or not isinstance(email, str):
        return False
    professional_domains = ['.com', '.org', '.net', '.edu']
    return any(email.lower().endswith(domain) for domain in professional_domains)
df = pd.read_csv(r"C:\Users\user\Downloads\messy_data.csv")
email_column = 'Email'
df[email_column] = df[email_column].str.strip().str.lower()
df['is_valid_email'] = df[email_column].apply(lambda x: is_valid_email(x) if
pd.notnull(x) else False)
df['is_professional_email'] = df[email_column].apply(lambda x:
is_professional_email(x) if pd.notnull(x) else False)
cleaned_df = df[df['is_valid_email'] & df['is_professional_email']]
output_file_path = r"C:\Users\user\Downloads\cleaned_data.csv"
cleaned_df.to_csv(output_file_path, index=False)
from IPython.display import display
display(cleaned_df)
```

6. Clean Name Fields

Objective: Remove noise from names and ensure consistent formatting.

Methods Used:

Steps:

- **Title Case Conversion**: Converts the name to title case.
- **Special Character Removal**: Removes special characters except apostrophes and hyphens.

- **Extra Space Removal**: Removes extra spaces.
- **Noise Words Removal**: Removes common honorifics or titles like 'Mr', 'Mrs', etc., from the beginning of names.
- **Trailing Descriptors Removal**: Removes trailing words or descriptors that often follow names, such as 'View', 'Management', etc.
- **Final Space Cleaning**: Ensures no extra spaces remain after cleaning.
- **NaN Handling**: Converts empty names resulting from cleaning to None.
- **Application**: Applied to the 'Name' column using df['Name'].apply(clean_name).

7. Standardize Date Formats

Purpose: Standardizes the 'Join Date' column to a consistent date format (YYYY-MM-DD).

Method:

- Uses pd.to_datetime() to convert the 'Join Date' column to datetime format.
- Uses dt.strftime('%Y-%m-%d') to format the datetime objects to the desired string format (YYYY-MM-DD).
- **Error Handling**: Uses errors='coerce' to handle errors by converting problematic entries to NaT (Not a Time), which will be handled later.

8. Correct Department Names

Purpose: Standardizes department names to a predefined set of values using a mapping dictionary

Method:

- Uses .str.strip().str.title() to ensure consistent formatting.
- Maps department names using map(department_mapping) to replace variations with standardized names.
- Uses .fillna(df_cleaned['Department']) to retain any values not covered by the mapping.

9. Handle Salary Noise

Purpose: Cleans and filters the 'Salary' column to remove unreasonable values.

Method:

- Uses pd.to_numeric() with errors='coerce' to convert 'Salary' values to numeric format, coercing errors to NaN where necessary.
- Filters the DataFrame to keep only salaries between 15,000 and 200,000, removing outliers or erroneous entries.

```
import pandas as pd

import re

 file_path = r'C:\Users\user\Downloads\cleaned_data.csv'

df = pd.read_csv(file_path)

def clean_name(name):

    if pd.isna(name):

        return None

        cleaned_name = name.strip().title()

    cleaned_name = re.sub(r'[^\w\s\'\-]', '', cleaned_name)

    cleaned_name = re.sub(r'\s+', ' ', cleaned_name).strip()

    noise_words = ['Mr', 'Mrs', 'Ms', 'Dr', 'Miss', 'Mx', 'Sr', 'Jr']

    for word in noise_words:

        if cleaned_name.startswith(word + ' '):

            cleaned_name = cleaned_name[len(word) + 1:]

        elif cleaned_name == word:

            cleaned_name = ''



    trailing_descriptors = [


r'\b(View|Compare|Important|Policy|Management|Party|Mission|Trip|Door|Gun|Perfor
mance|Tv|In|Report|Research|Recognize|Grow|Interest|Every)\b.*'

    ]

    cleaned_name = re.sub('|'.join(trailing_descriptors), '', cleaned_name,
flags=re.IGNORECASE).strip()

    cleaned_name = re.sub(r'\s+', ' ', cleaned_name).strip()

    if cleaned_name == '':

        return None
```

```
    return cleaned_name

df['Name'] = df['Name'].apply(clean_name)



df_cleaned = df.dropna(subset=['Name']).reset_index(drop=True)

df_cleaned['Join Date'] = pd.to_datetime(df_cleaned['Join Date'],
errors='coerce').dt.strftime('%Y-%m-%d')

department_mapping = {

    'Hr': 'HR',

    'Human Resources': 'HR',

    'Eng': 'Engineering',

    'Tech': 'Engineering',

    'Mktg': 'Marketing',

    'Mkt': 'Marketing',

    'Sales': 'Sales',

    'Sls': 'Sales',

    'Sup': 'Support',

    'Supp': 'Support'

}



df_cleaned['Department'] =
df_cleaned['Department'].str.strip().str.title().map(department_mapping).fillna(df_clea
ned['Department'])



df_cleaned['Salary'] = pd.to_numeric(df_cleaned['Salary'], errors='coerce')

df_cleaned = df_cleaned[(df_cleaned['Salary'] >= 15000) & (df_cleaned['Salary'] <=
200000)]
```

```
output_file_path = r'C:\Users\user\Downloads\cleaned_dataset.csv'

df_cleaned.to_csv(output_file_path, index=False)



print("Data cleaning completed. Cleaned dataset saved as 'cleaned_dataset.csv'.")
```

Summary Document

Objective: Provide a summary document detailing the steps taken to clean the data, including assumptions and methodologies used.

Summary:

- Loaded the dataset and inspected its structure and content.
- Handled missing values by filling them with appropriate defaults.
- Removed duplicate rows to ensure each record is unique.
- Corrected email formats and filtered out non-professional emails.
- Cleaned and standardised name fields.
- Standardised date formats in the 'Join Date' column.
- Corrected department names and ensured consistency.
- Handled salary noise by clipping values to a reasonable range.

This document outlines the comprehensive approach taken to clean the dataset, ensuring data quality and consistency for further analysis.

Conclusion

This documentation serves as a detailed record of the data cleaning process, helping to maintain transparency and facilitating reproducibility of the data cleaning steps in the future. Adjust the comments and explanations as needed to reflect the specific details and nuances of your dataset and cleaning requirements.