DATA CLEANING

Objective:

The goal of this data cleaning task was to process and refine a dataset to remove duplicates, handle missing values, validate and standardize data, and ensure data integrity. This involved various steps to ensure the dataset was cleaned and ready for further analysis.

Dataset:

This dataset is an employee dataset that contains 11000 rows and 8 columns. Columns are Unnamed: 0,ID,Name,Email,Join Date, Salary and Department.

1) Import Necessary Libraries:

```
#Import Necessary Libraries
import pandas as pd
import re
import warnings

# Suppress warnings
warnings.filterwarnings('ignore')
```

2) Load the dataset and convert it into a pandas Data Frame:

```
#Load dataset

df=pd.read_csv("messy_data.csv")
df
```

3) Initial Data Inspection:

	Unnamed: 0	ID	Name	Age	Email	Join Date	Salary	Department
0	0	1e407ff9-6255-489d-a0de-34135d4f74bd	Hunter Thomas	25.0	xlopez@hotmail.com	NaN	88552.000000	Sales
1	1	379f55b8-87d5-4739-a146-7400b78c24d1	Jeremy Irwin	90.0	Jillian Jenkins	2022-07-07	139227.000000	NaN
2	2	18261368-dfa1-47f0-afc6-bddf45926b07	Jennifer Hammondquickly	66.0	jscottgreen.biz	2023-11-21	65550.000000	Engineering
3	3	ae7cf7cf-17cf-4c8b-9c44-4f61a9a238e5	Sydney Taylorso	39.0	luke56gonzalez.com	2021-11-05	139932.000000	SupportJ
4	4	14ed3e6a-e0f5-4bbe-8d93-8665267f5c90	Julia Lee	71.0	figueroakayla@yahoo.com	NaN	143456.000000	Marketing
...	...	...	...	...	...	...	...	...
10995	6523	07c223be-03e6-4f70-a2b5-86df778cc61a	NaN	NaN	NaN	NaN	NaN	NaN
10996	9785	da8a6bbc-5026-4630-848d-f64e80dac56c	Steven Armstrong	38.0	molly89gmail.com	2021-06-24	NaN	Sales
10997	7826	ed19c966-d6d8-4047-b410-b6e595a39340	Stephanie Riossell	NaN	robert96@pollard-frye.com	15/08/2006	122609.594149	HR
10998	7648	783b36b4-d09f-46c9-8a52-7ff96b80863e	Bonnie Benitez	37.0	roypark@warren.net	2020-10-09	147322.005171	Support
10999	7107	fc25a38a-5747-46eb-b6d3-7173f8255809	Caroline Ochoa	53.0	cdavis@hodges.com	2023-08-10	149224.000000	Support

Initial Issues : There are Unnecessary column like Unnamed:0,Missing values in most of the columns, Noise in Name and Salary columns, Incorrect email formats in Email column, Different date formats in Join Date column ,Inconsistency in Department column.

4) Data Cleaning Steps:

1)

```
#Shape of our dataset

df.shape
```

```
(11000, 8)
```

```
#Column names

df.columns
```

```
Index(['Unnamed: 0', 'ID', 'Name', 'Age', 'Email', 'Join Date', 'Salary',
       'Department'],
      dtype='object')
```

```
#Dropping unnecessary columns ,here 'Unnamed: 0' since it doesnot have any meaning here

df=df.drop(columns='Unnamed: 0',axis=1)
```

```
#Rename column name Join Date to Join_Date

df.rename(columns={'Join Date': 'Join_Date','ID':'Id'}, inplace=True)
```

2) Checking for duplicate rows and removing duplicate rows:

```
#Checking for duplicates

df.duplicated().sum()
```

```
291
```

```
# dropping duplicates if any

df=df.drop_duplicates()
```

```
df.duplicated().sum()
```

```
0
```

3) Structure of the dataset:

```
#Structure of the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10709 entries, 0 to 10998
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Id           10709 non-null  object 
 1   Name         8579 non-null   object 
 2   Age          9145 non-null   float64
 3   Email        9617 non-null   object 
 4   Join_Date    8707 non-null   object 
 5   Salary       8661 non-null   float64
 6   Department   8647 non-null   object 
dtypes: float64(2), object(5)
memory usage: 669.3+ KB
```

The info () method provides structure of our dataset including number of rows and number of non-null values in each column ,datatypes of each column etc..

4) Finding the number of Nan values in each column:

```
#Checking for null values
df.isna().sum()
```

```
Id              0
Name         2130
Age          1564
Email        1092
Join_Date    2002
Salary       2048
Department   2062
dtype: int64
```

There are Nan values in every column except Id column. isna() method along with sum() method is used for this purpose.

5) Removing Rows that have Nan values in all columns :

```
#Removing rows that contains Nan values in 6 columns
columns_to_check = ['Name', 'Age', 'Email', 'Join_Date', 'Salary',
       'Department'] # Specify the column names here

# Remove rows where all specified columns have null values
df= df.dropna(subset=columns_to_check, how='all')
```

dropna() method is used for removing rows.

6) Statistical Summary of numerical data:

This statistical summary provides us with the count of non null values, minimum value, maximum value,mean,standard deviation,25%,50%,75% of numerical columns.

```
#Summary Statistics
df.describe()
```

	Age	Salary
count	9145.000000	8661.000000
mean	54.167523	89839.028667
std	21.079546	34879.814556
min	18.000000	24655.136613
25%	36.000000	59686.000000
50%	54.000000	89168.000000
75%	72.000000	119424.000000
max	90.000000	176156.206747

There are some inconsistency that we will deal later.

7) Describing categorical columns:

```
df.describe(include="object")
```

	Id	Name	Email	Join_Date	Department
count	9617	8579	9617	8707	8647
unique	8908	7929	9160	3338	264
top	468d3baa-8db4-477d-8090-e7399b2820a7	Elizabeth Williams	mmurphy@gmail.com	2020-07-20	Marketing
freq	2	6	3	11	1408

In order to describe , used describe() method, from above we can see that there are duplication in columns we will deal with it later.

8) Cleaning Email Formats:

- **Validates email addresses:** Uses a regular expression to check the validity of each email
- address. Invalid email addresses are replaced with Nan.
- **Removes leading and trailing whitespace:** Used the str.strip()method to remove any extra spaces at the beginning or end of each email address.
- **Converts email addresses to lowercase**: Used the str.lower() method to ensure all email addresses are in lowercase.
- **Handling Missing values**:Dropping rows with missing values in email column.
- **Duplicates in email addresses**: There are duplicates in emails,but retained that emails because there are relevant information in other columns.

```
import numpy as np
# Function to validate email addresses
def validate_email(email):
    if pd.isnull(email):
        return False
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None

# Apply email validation and replace unprofessional emails with NaN
df['Email'] = df['Email'].apply(lambda x: x if validate_email(x) else np.nan)

# Remove leading and trailing whitespace
df['Email'] = df['Email'].str.strip()

# Convert to lowercase
df['Email'] = df['Email'].str.lower()


# Drop rows with NaN values in 'Email'
df = df.dropna(subset=['Email'])
```

Output:

Email
nancyortega@gmail.com
aliciawright@yahoo.com
mlewis@hotmail.com
gmedina@wilson-scott.com
sbryant@scott.info
...
usmith@gmail.com
cassandra78@hotmail.com
joseph11@williams.com
hpope@gmail.com

9) Cleaning Name fields:

- **Removing Noises added to the names:** There is noise in the Name column, meaning some extra words are appended to the end of the last names. To remove this noise and standardize the names, I defined a function clean_name(). This function extracts only the first name and removes all titles from the names to standardize them.
- **Handling Missing Values:** Imputation is used. Imputing Nan values with Unknown value, rather than dropping rows in order to prevent loss of information.

```
# List of titles to remove
titles = ['Mr.', 'Mrs.', 'Ms.', 'Dr.', 'Miss', 'Prof.']

# Function to remove titles and extract first name
def clean_name(name):
    if pd.notnull(name):
        for title in titles:
            name = name.replace(title, '').strip()
        return name.split()[0] if name else np.nan
    return np.nan

# Apply the function to the 'Name' column to get the 'FirstName'
df['FirstName'] = df['Name'].apply(clean_name)

# Get the position of the 'Name' column
name_col_position = df.columns.get_loc('Name')

# Insert the 'FirstName' column at the same position as the 'Name' column
df.insert(name_col_position, 'FirstName', df.pop('FirstName'))

# Optionally, drop the original 'Name' column if not needed
df.drop(columns=['Name'], inplace=True)

# Print the final DataFrame
df
```

Output:

FirstName
Hunter
Julia
Lisa
Unknown
Jay
...
Rhonda
Scott
Tina
Alexandra
Unknown

10) Standardising Date Formats:

- **Datatype Conversion**: Date values were in string format, so I converted them to datetime format using the pd.to_datetime() method.
- There were different date formats in this column, so I defined a convert_to_yyyy_mm_dd() function to convert all date formats to YYYY-MM-DD. For this, I used the strftime('%Y-%m-%d') method.
- **Handling Missing Values:** Dropped rows that have Nan values in Join_ Date column.

```
import numpy as np
# Function to convert date strings to YYYY-MM-DD format
def convert_to_yyyy_mm_dd(date_str):
    try:
        return pd.to_datetime(date_str, dayfirst=True).strftime('%Y-%m-%d')
    except:
        return np.nan  # Return NaN for any invalid dates

# Apply conversion to 'Join Date' column
df['Join_Date'] = df['Join_Date'].apply(convert_to_yyyy_mm_dd)

df
```

Output:

Join_Date
2016-10-15
2022-12-02
2022-08-12
2023-08-09
2023-06-24
...
2022-10-06
2021-05-02
2023-09-30
2022-02-13
2020-11-28

11) Correct Department Names:

- **Handling Noise:** There was noise in the Department column, so I used regular expressions to remove the noise and standardize the entries.
- **Handling Missing Values:** Dropped rows with Nan values in Department column.

```
#Standardising Department Names to ensure Consistency.

# Define a regex pattern to match variations of HR,Support,Sales,Marketing,Engineering (case insensitive)
pattern1 = r'HR[a-zA-Z]*'
pattern2 = r'Support[a-zA-Z]*'
pattern3 = r'Sales[a-zA-Z]*'
pattern4 = r'Marketing[a-zA-Z]*'
pattern5 = r'Engineering[a-zA-Z]*'

# Apply regex replacement to standardize HR,Support,Sales,Marketing,Engineering  variations
df['Department'] = df['Department'].str.replace(pattern1, 'HR', regex=True)
df['Department'] = df['Department'].str.replace(pattern2, 'Support', regex=True)
df['Department'] = df['Department'].str.replace(pattern3, 'Sales', regex=True)
df['Department'] = df['Department'].str.replace(pattern4, 'Marketing', regex=True)
df['Department'] = df['Department'].str.replace(pattern5, 'Engineering', regex=True)



# Display unique values in the standardized department column
df['Department'].unique()
```

```
array(['Support', 'Sales', 'HR', 'Engineering', 'Marketing', nan],
      dtype=object)
```

Output:

Department
Support
Sales
HR
Engineering
Sales
...
Engineering
HR
Marketing
Marketing
Marketing

12) Handling Salary Noise:

- **Handling Noise:** Removed noise in Salary column using round() function,that means rounding the Salary to 2 decimal digits**.**
- **Handling Missing Values:** Imputing missing values with median **.**The median is less sensitive to outliers compared to the mean. Salary data often contains outliers (extremely high or low values) that can skew the mean.
- **Checked for Outliers:** Used IQR for outlier detection,but could not find any potential outliers.

```
df['Salary'].isna().sum()
```

```
548
```

```
#Imputing Nan values with median
df['Salary']=df['Salary'].fillna(df['Salary'].median())
```

```
df['Salary'].isna().sum()
```

```
0
```

```
#dealing noise in Salary column
# Round the 'Salary' column to two decimal places
df['Salary'] = df['Salary'].round(2)
```

Salary
123018.00
83354.00
37701.73
112970.00
41511.14
...
33029.00
30614.00
150844.40
67036.00
88530.00

```
#checking for outliers
Q1 = df['Salary'].quantile(0.25)
Q3 = df['Salary'].quantile(0.75)

# Calculate IQR (Interquartile Range)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Find outliers based on the bounds
outliers = df[(df['Salary'] < lower_bound) | (df['Salary'] > upper_bound)]
# Print outliers in a list format
print("Outliers in Salary column:")
print(outliers['Salary'].tolist())
```

```
Outliers in Salary column:
[]
```

13) Cleaning Age Column:

- **Datatype Conversion:**Converting Age from float to int.
- **Handling Missing Values:** Imputing Nan values with mean of the Ages in Age column.

```
#Imputing Age column with mean value of Age column
mean_age=df['Age'].mean()
df['Age']=df['Age'].fillna(mean_age)
```

Converting Age Column from float to int

```
df['Age']=df['Age'].astype(int)
df['Age']
```

```
5        81
8        71
9        44
12       23
13       57
```

14) Removing Duplicate Id's:

- Sort dataframe by Id and Join_Date in descending order, to retain the rows with latest Join_Date.
- Drop duplicates keeping the latest date for each ID.

15)Converting DataFrame to csv file:

```
#Converting dataframe to a csv file
df_latest.to_csv('cleaned_dataset.csv',index=False)
```

Outcome:

The messy dataset has been systematically cleaned and pre-processed, ensuring its suitability for further analysis. Inconsistencies were identified and rectified, missing values were handled appropriately, duplicates were removed, errors were corrected, and data formats were standardized. As a result, the dataset has been transformed into a clean, reliable, and structured format, ready for accurate and meaningful analysis.