Assignment for Data QA & QC Internship @ Datahut

Problem statement

The dataset messy_Data.csv contains multiple data quality issues that must be addressed to ensure accurate analysis and reporting.The objective of this assignment is to clean the dataset by resolving errors such as missing values, duplicate entries, inconsistent formatting, and noise in certain columns.

Objective

The primary objectives of this data cleaning task are as follows:

1-Identify and Resolve Missing Data:

Handle missing values in critical columns by either filling them with appropriate statistics (e.g., mean, median) or removing rows/columns with excessive missing data.

2-Remove Duplicate Records:

Identify and remove any duplicate rows in the dataset to ensure the uniqueness and accuracy of each record.

3-Correct Inconsistent Email Formats:

Ensure that all email addresses follow a standard format (e.g., username@domain.com) and contain only professional email domains in the final dataset.

4-Standardize Name Fields:

Remove noise from the names and ensure consistent formatting, eliminating any extraneous or irrelevant information.

5-Uniform Date Formats:

Ensure that all dates in the 'Join Date' column follow a consistent format (YYYY-MM-DD) to maintain uniformity.

6-Correct Department Names:

Identify and correct typos or inconsistencies in the 'Department' column and standardize department names for clarity.

7-Handle Salary Outliers:

Ensure that salary values fall within a reasonable range, handling outliers or noise that may exist in the data.

8-Document Data Cleaning Process:

Thoroughly document the steps taken during the data cleaning process, including any assumptions made and the methods used to address specific data quality issues.

9-Prepare Final Clean Dataset:

Save the cleaned dataset as cleaned_dataset.csv and provide a clear summary of the cleaning process, ensuring the dataset is ready for analysis.

Import necessary libraries and dataset

In [33]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [34]:
```
#Load dataset
data=pd.read_csv(r"messy_data.csv")
```

In [35]:
```
data.head()
```

Out[35]:

	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	88552.0	Sales
1	1	379f55b8-87d5-4739-a146-7400b78c24d1	Jeremy Irwin	90.0	Jillian Jenkins	2022-07-07	139227.0	NaN
2	2	18261368-dfa1-47f0-afc6-bddf45926b07	Jennifer Hammondquickly	66.0	jscottgreen.biz	2023-11-21	65550.0	Engineering
3	3	ae7cf7cf-17cf-4c8b-9c44-4f61a9a238e5	Sydney Taylorso	39.0	luke56gonzalez.com	2021-11-05	139932.0	SupportJ
4	4	14ed3e6a-e0f5-4bbe-8d93-8665267f5c90	Julia Lee	71.0	figueroakayla@yahoo.com	NaN	143456.0	Marketing

In [36]:
```
data.tail()
```

Out[36]:

	Unnamed: 0	ID	Name	Age	Email	Join Date	Salary	Department
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

Inspect the data set

In [37]:
```
# shape of the dataset
data.shape
```

Out[37]:
```
(11000, 8)
```

The given data set contain 11000 rows and 8 columns

In [38]:
```
# columns of the data set
print("The columns of the dataset are :\n",data.columns)
```

```
The columns of the dataset are :
 Index(['Unnamed: 0', 'ID', 'Name', 'Age', 'Email', 'Join Date', 'Salary',
       'Department'],
      dtype='object')
```

To gain a comprehensive overview of dataset, including the number of entries, data types of each column, and the presence of any missing values, helping to understand the overall structure and quality of the data.

In [39]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11000 entries, 0 to 10999
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  11000 non-null  int64
 1   ID          11000 non-null  object
 2   Name        8667 non-null   object
 3   Age         9253 non-null   float64
 4   Email       9731 non-null   object
 5   Join Date   8808 non-null   object
 6   Salary      8761 non-null   float64
 7   Department  8745 non-null   object
dtypes: float64(2), int64(1), object(5)
memory usage: 687.6+ KB
```

To generate descriptive statistics for dataset, summarizing key metrics such as mean, median, standard deviation, and percentiles for each numerical column.

In [40]:
```
#descriptive statistics
data.describe()
```

Out[40]:

	Unnamed: 0	Age	Salary
count	11000.000000	9253.000000	8761.000000
mean	5012.947818	54.162650	89886.585012
std	2884.739158	21.072919	34896.320117
min	0.000000	18.000000	24655.136613
25%	2509.750000	36.000000	59723.844874
50%	5024.500000	54.000000	89241.000000
75%	7510.250000	72.000000	119491.000000
max	9999.000000	90.000000	176156.206747

To generate descriptive statistics for all the categorical columns

In [41]:
```
data.describe(include='object')
```

Out[41]:

	ID	Name	Email	Join Date	Department
count	11000	8667	9731	8808	8745
unique	10000	7929	9160	3338	264
top	47f408c8-c6e1-4c59-a9bd-c080526fa46f	Elizabeth Williams	fwilliams@yahoo.com	2022-03-31	Support
freq	2	6	3	12	1425

There is missing data in multiple columns, particularly in Name, Email, Join Date, and Department, which may need cleaning or imputation.

The presence of duplicate values in ID suggests potential issues with data uniqueness, which should be addressed to maintain data integrity.

The Department column is heavily skewed towards a specific department (Support), which may indicate an imbalance in department representation.

Handling duplicates

In [42]:
```
# check for the duplicate values
duplicate_count = data.duplicated().sum()
print('The number of duplicate values present in the data set are',duplicate_count)
```

```
The number of duplicate values present in the data set are 291
```

In [43]:
```
# Remove duplicate rows and keep the first occurrence
data = data.drop_duplicates()

# View the cleaned Data
print(data.duplicated().sum())
```

```
0
```

Handling missing values

In [44]:
```
# check for the missing values
data.isnull().sum()
```

Out[44]:
```
Unnamed: 0       0
ID               0
Name          2130
Age           1564
Email         1092
Join Date     2002
Salary        2048
Department    2062
dtype: int64
```

name , age, email,join date,salary departrment columns contain null values

In [45]:
```
# check rows with more than 5 null values
rows_with_many_nulls = data[data.isnull().sum(axis=1) > 5]

print("Rows with more than 5 null values:")
rows_with_many_nulls.shape
```

```
Rows with more than 5 null values:
```

Out[45]:
```
(1092, 8)
```

There are 1269 rows(entries) in which 5 of the columns are null.hence they are irrelevent

In [46]:
```
# remove  rows with more than 5 null values
data = data.dropna(thresh=len(data.columns) - 5)
```

In [47]:
```
# checking null values after deleting irrelevent rows
data.isnull().sum()
```

Out[47]:
```
Unnamed: 0       0
ID               0
Name          1038
Age            472
Email            0
Join Date      910
Salary         956
Department     970
dtype: int64
```

In [48]:
```
# fill the missing values in the name column with unknown
data['Name'] = data['Name'].fillna('Unknown')
# Check if any missing values remain in the 'Name' column
print(data['Name'].isnull().sum())
```

```
0
```

The misssing values in the name columns are now filled with unknown values

In [49]:
```
# fill the missing values in the age column with median
data['Age'] = data['Age'].fillna(data['Age'].median())
# Check if any missing values remain in the 'Age' column
print(data['Age'].isnull().sum())
```

```
0
```

The missing values in the age columns are now filled with the median value,which is less sensitive to outliers

In [50]:
```
# fill the missing values in the department column with mode
data['Department'] = data['Department'].fillna(data['Department'].mode()[0])
# Check if any missing values remain in the 'department' column
print(data['Department'].isnull().sum())
```

```
0
```

The missing values in the department columns are now filled with the mode

In [51]:
```
# Fill missing values in the 'Salary' column using the mean salary of the specific 'Department'
data['Salary'] = data.groupby('Department')['Salary'].transform(lambda x: x.fillna(x.mean()))

# Check if any missing values remain in the 'Salary' column
print(data['Salary'].isnull().sum())
```

```
0
```

In [52]:
```
# checking null values after filling the rows
data.isnull().sum()
```

Out[52]:
```
Unnamed: 0      0
ID              0
Name            0
Age             0
Email           0
Join Date     910
Salary          0
Department      0
dtype: int64
```

The data set does not contain any missing values other than join date column.The null values in the join date column will handle after standardising the date formats

Correct Email format

In [53]:
```
import re
def is_valid_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None

# Find the number of invalid emails
invalid_emails_count = data['Email'].apply(lambda x: not is_valid_email(x)).sum()
print(f"Number of invalid emails: {invalid_emails_count}")
```

```
Number of invalid emails: 2410
```

In [54]:
```
# Remove rows with invalid emails
data = data[data['Email'].apply(is_valid_email)]
```

The number of invalid emails present in the dataset are 2410. removed the rows with invalid email.

Clean name field

In [55]:
```
data['Name'].iloc[30:40]
```

Out[55]:
```
52        Joshua Harmonview
53            Angela Waller
57           Jason Stephens
59            Jessica Evans
61            Kimberly Page
63    Mrs. Sheri Duffyparty
68        Dennis Martindoor
69          Robert Peterson
71     Allison Mcknighttrip
72            Miguel Wilcox
Name: Name, dtype: object
```

In [56]:
```
df=data.copy()
```

In [57]:
```
# Function to clean and format names
def clean_name(name):
    # Remove titles like Mr., Mrs., Dr., etc. (add more titles as necessary)
    titles = ['Mr', 'Mrs', 'Ms', 'Miss', 'Dr', 'Prof', 'Sir']
    name = re.sub(r'\b(?:' + '|'.join(titles) + r')\b\.?\s*', '', name, flags=re.IGNORECASE)

    # Remove any non-alphabetical characters (except spaces)
    name = re.sub(r'[^a-zA-Z\s]', '', name)

    # Remove multiple spaces and strip leading/trailing spaces
    name = re.sub(r'\s+', ' ', name).strip()

    # Capitalize the first letter of each word (title case)
    name = name.title()

    return name

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

In [58]:
```
df['Name'].iloc[30:40]
```

Out[58]:
```
52       Joshua Harmonview
53           Angela Waller
57          Jason Stephens
59           Jessica Evans
61           Kimberly Page
63         Sheri Duffyparty
68       Dennis Martindoor
69         Robert Peterson
71    Allison Mcknighttrip
72           Miguel Wilcox
Name: Name, dtype: object
```

Standardise date format

In [59]:
```
# check unique date format
df['Join Date'].unique()
```

Out[59]:
```
array([nan, '15/10/2016', '2022-02-12', ..., '30/07/2003', '01/05/1982',
       '15/08/2006'], dtype=object)
```

In [60]:
```
# Convert the 'Join Date' column to a consistent datetime format (YYYY-MM-DD)
df['Join Date'] = pd.to_datetime(df['Join Date'], errors='coerce', format='%Y-%m-%d')

# Convert datetime objects back to string format as YYYY-MM-DD
df['Join Date'] = df['Join Date'].dt.strftime('%Y-%m-%d')

# View the cleaned DataFrame
df['Join Date'].unique()
```

Out[60]:
```
array([nan, '2022-02-12', '2022-12-08', ..., '2020-08-28', '2022-01-29',
       '2022-11-09'], dtype=object)
```

The join dates are now formated to standard format.The null values present in the column are handled by forward and backward filling method

In [61]:
```
# Forward fill to fill most of the missing 'Join Date' values
df['Join Date'] = df['Join Date'].ffill()

# Backward fill to handle any remaining null values
df['Join Date'] = df['Join Date'].bfill()
# Check if any missing values remain in the 'join date' column
print(df['Join Date'].isnull().sum())
```

```
0
```

Correct Department names

In [62]:
```
# check unique values
df['Department'].unique()
```

Out[62]:
```
array(['Sales', 'Marketing', 'SupportE', 'HR', 'Engineering', 'SalesA',
       'SalesX', 'Engineeringi', 'MarketingR', 'Salesw', 'SalesK',
       'MarketingU', 'Support', 'Supporty', 'Marketinge', 'Marketingv',
       'HRv', 'SalesM', 'EngineeringL', 'MarketingQ', 'HRs', 'Supportj',
       'Salesd', 'Marketingu', 'SalesU', 'MarketingH', 'Salesj', 'Saleso',
       'SalesZ', 'Engineeringe', 'Marketingr', 'Engineeringm', 'HRt',
       'SalesW', 'Engineeringp', 'HRF', 'SalesC', 'SupportC', 'SalesY',
       'EngineeringK', 'Marketingz', 'SalesN', 'HRo', 'SupportB', 'HRi',
       'Salest', 'Supportg', 'Supporta', 'HRn', 'SupportQ', 'Supportm',
       'Supporto', 'EngineeringH', 'Marketingw', 'EngineeringM',
       'Supportz', 'HRr', 'Salesy', 'MarketingD', 'HRE', 'Supportu',
       'Marketingm', 'HRw', 'Supportk', 'MarketingN', 'SupportS',
       'EngineeringD', 'Marketingp', 'Salesl', 'Salesr', 'SupportT',
       'Engineeringn', 'MarketingO', 'SalesF', 'Salesz', 'Marketingt',
       'Engineeringv', 'HRL', 'HRI', 'HRW', 'EngineeringR', 'HRD',
       'Marketingk', 'Salese', 'Saless', 'Salesn', 'Engineeringl',
       'SalesE', 'HRy', 'SalesD', 'HRm', 'HRz', 'MarketingG',
       'Marketingn', 'SalesI', 'Salesu', 'Engineeringg', 'MarketingE',
       'SupportA', 'Engineeringq', 'EngineeringQ', 'EngineeringA',
       'Supporte', 'Engineeringu', 'EngineeringO', 'Supportv', 'HRK',
       'Engineeringb', 'SalesV', 'HRT', 'SupportY', 'MarketingX',
       'Supportp', 'HRU', 'HRp', 'HRg', 'SalesS', 'Marketingl',
       'Engineeringt', 'SalesJ', 'EngineeringX', 'Engineeringc',
       'Marketingh', 'MarketingF', 'EngineeringC', 'Engineeringz', 'HRP',
       'Salesv', 'Engineeringk', 'SupportH', 'Salesk', 'Supporth',
       'Supportq', 'Salesi', 'HRJ', 'Supportd', 'HRZ', 'SupportJ',
       'MarketingZ', 'Salesc', 'EngineeringJ', 'Engineeringx',
       'EngineeringT', 'Salesq', 'Engineeringh', 'Supportl', 'SupportX',
       'EngineeringG', 'SalesH', 'MarketingC', 'SupportV', 'HRq',
       'Marketingb', 'Supporti', 'SupportL', 'HRM', 'Supportt', 'HRX',
       'EngineeringE', 'EngineeringF', 'SupportP', 'HRf', 'MarketingB',
       'Engineeringf', 'HRB', 'Marketinga', 'HRj', 'SupportI', 'HRA',
       'Marketingj', 'MarketingV', 'EngineeringZ', 'HRl', 'Marketingx',
       'Engineeringj', 'HRN', 'MarketingJ', 'Marketingi', 'Supportw',
       'HRh', 'Engineerings', 'Supportb', 'Engineeringa', 'SalesB',
       'SalesO', 'Marketingf', 'Marketingg', 'SupportK', 'Engineeringd',
       'HRk', 'Marketingd', 'SalesR', 'MarketingS', 'MarketingL',
       'MarketingT', 'HRR', 'Engineeringw', 'Salesg', 'SupportO',
       'EngineeringV', 'HRb', 'SupportN', 'SupportG', 'Marketingc',
       'Supportn', 'EngineeringU', 'Supportx', 'HRY', 'Salesp', 'HRH',
       'MarketingK', 'SalesL', 'Marketingq', 'SupportD', 'SalesT',
       'Supports', 'HRQ', 'Supportr', 'Engineeringr', 'EngineeringW',
       'Salesm', 'HRC', 'SupportF', 'HRe', 'SupportW', 'EngineeringI',
       'Marketingy', 'MarketingM', 'MarketingI', 'SupportU', 'Salesa',
       'Salesf', 'MarketingY', 'Supportf', 'HRd', 'HRu', 'Salesh',
       'EngineeringB', 'SupportM', 'EngineeringN', 'HRV', 'HRa', 'HRG',
       'SalesP', 'Supportc', 'HRx', 'EngineeringS', 'EngineeringP',
       'Engineeringy', 'HRS', 'SupportR', 'Salesx', 'Marketings',
       'MarketingW', 'HRO', 'HRc', 'SupportZ', 'SalesG', 'SalesQ',
       'Salesb', 'MarketingP', 'Engineeringo'], dtype=object)
```

In [63]:
```
# Define a function to extract the base department name
def clean_department(dept):
    # List of valid departments
    valid_departments = ['HR', 'Sales', 'Marketing', 'Engineering', 'Support']

    # Iterate over valid departments and check if the base name matches
    for valid_dept in valid_departments:
        if dept.startswith(valid_dept):
            return valid_dept  # Return the clean department name
    return dept  # If not found, return as-is (can handle later)

# Apply the cleaning function to the 'Department' column
df['Department'] = df['Department'].apply(clean_department)

# View the cleaned Department column
print(df['Department'].unique())
```

```
['Sales' 'Marketing' 'Support' 'HR' 'Engineering']
```

Handle Salary noice

In [65]:
```
#Visualize the salary distribution
plt.figure(figsize=(5, 5))
sns.boxplot(df['Salary'].dropna())  # Drop NaN for visualization
plt.title('Salary Distribution with Boxplot')
plt.show()
```

Salary column does not conain outliers.

In [67]:
```
#check dataset after cleaning
df.head()
```

Out[67]:

	Unnamed: 0	ID	Name	Age	Email	Join Date	Salary	Department
0	0	1e407ff9-6255-489d-a0de-34135d4f74bd	Hunter Thomas	25.0	xlopez@hotmail.com	2022-02-12	88552.000000	Sales
4	4	14ed3e6a-e0f5-4bbe-8d93-8665267f5c90	Julia Lee	71.0	figueroakayla@yahoo.com	2022-02-12	143456.000000	Marketing
5	5	f92ea13f-24d3-4792-91e4-5b0e07d90c2c	Lisa Bailey	81.0	milleryvonne@fowler-york.com	2022-02-12	123018.000000	Support
8	8	12cc2f16-d641-402e-b22b-3332bafc4f1d	Unknown	71.0	andrew74@hotmail.com	2022-02-12	83354.000000	Sales
9	9	d06f507d-62e5-4bd6-a252-4e60e109d324	Jay Fisher	44.0	denisemaynard@jordan.com	2022-12-08	37701.725993	HR

In [69]:
```
# Lets sort the DataFrame by 'Unnamed: 0' column
df_sorted = df.sort_values(by='Unnamed: 0').reset_index(drop=True)
df_sorted.head()
```

Out[69]:

	Unnamed: 0	ID	Name	Age	Email	Join Date	Salary	Department
0	0	1e407ff9-6255-489d-a0de-34135d4f74bd	Hunter Thomas	25.0	xlopez@hotmail.com	2022-02-12	88552.000000	Sales
1	4	14ed3e6a-e0f5-4bbe-8d93-8665267f5c90	Julia Lee	71.0	figueroakayla@yahoo.com	2022-02-12	143456.000000	Marketing
2	5	f92ea13f-24d3-4792-91e4-5b0e07d90c2c	Lisa Bailey	81.0	milleryvonne@fowler-york.com	2022-02-12	123018.000000	Support
3	8	12cc2f16-d641-402e-b22b-3332bafc4f1d	Unknown	71.0	andrew74@hotmail.com	2022-02-12	83354.000000	Sales
4	9	d06f507d-62e5-4bd6-a252-4e60e109d324	Jay Fisher	44.0	denisemaynard@jordan.com	2022-12-08	37701.725993	HR

In [70]:
```
# Save the DataFrame to a CSV file
df_sorted.to_csv('cleaned_dataset.csv', index=False)
```

The given dataset is cleaned and saved as a new csv file as cleaned_dataset