**Lab 2: Data Cleaning Process: Handling missing values and performing deduplication.**

1. **Create a dataset as below:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CustomerID** | **Name** | **Age** | **JoinDate** | **MonthlyCharges** | **Churn** |
| **C001** | **John** | **25** | **12/1/2024** | **29.85** | **No** |
| **C002** | **Alice** | **34** | **11/15/2023** | **56.95** | **Yes** |
| **C003** | **BOB** | **17** | **6/1/2022** | **4000** | **No** |
| **C004** | **BOBY** | **29** | **6/1/2022** | **75.5** | **No** |
| **C004** | **Eve** | **29** | **12/5/2024** | **75.5** | **No** |
| **C005** | **eve** | **120** | **invalid\_date** | **45.99** | **Yes** |
| **C006** | **Steve** | **-5** |  | **60** | **No** |
| **C007** | **Ramu** |  | **1/1/2024** | **49.99** |  |
| **C008** | **mary** | **220** | **3/5/2023** | **-30** | **Yes** |
| **C008** | **Bob** | **30** | **3/5/2023** | **55** | **No** |

import pandas as pd

# Creating the dataset

data = {

    "CustomerID": ["C001", "C002", "C003", "C004", "C004", "C005", "C006", "C007", "C008", "C008"],

    "Name": ["John", "Alice", "BOB", "BOBY", "Eve", "eve", "Steve", "Ramu", "mary", "Bob"],

    "Age": [25, 34, 17, 29, 29, 120, -5, None, 220, 30],

    "JoinDate": ["12/1/2024", "11/15/2023", "6/1/2022", "6/1/2022", "12/5/2024", "invalid\_date", None, "1/1/2024", "3/5/2023", "3/5/2023"],

    "MonthlyCharges": [29.85, 56.95, 4000, 75.5, 75.5, 45.99, 60, 49.99, -30, 55],

    "Churn": ["No", "Yes", "No", "No", "No", "Yes", "No", None, "Yes", "No"]

}

# Converting the dictionary into a pandas DataFrame

df = pd.DataFrame(data)

# Display the dataset

print(df)

1. **Find out the missing values in the dataset using isnull(), isna(), notna(), notnull() and using any() or sum() functions individually with each of them.**

# Checking for missing values using isnull()

print("Missing values using isnull():")

print(df.isnull())

# Checking for missing values using isna()

print("\nMissing values using isna():")

print(df.isna())

# Checking for non-missing values using notna()

print("\nNon-missing values using notna():")

print(df.notna())

# Checking for non-missing values using notnull()

print("\nNon-missing values using notnull():")

print(df.notnull())

# Checking if any column has missing values using any()

print("\nColumns with missing values using any():")

print(df.isnull().any())

# Checking if any column has missing values using sum()

print("\nCount of missing values in each column using sum():")

print(df.isnull().sum())

1. **Visualize the missing values using msno.matrix() function.**

import missingno as msno

import pandas as pd

# Assuming 'df' is your DataFrame

# Visualizing missing values using msno.matrix()

msno.matrix(df)

1. **Fill the missing values with mean, median or mode wherever necessary. And display the new dataset.**

import pandas as pd

# Assuming 'df' is your original DataFrame

# Fill missing values for 'Age' and 'MonthlyCharges' using mean

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

df['MonthlyCharges'] = df['MonthlyCharges'].fillna(df['MonthlyCharges'].mean())

# Fill missing values for 'Churn' using mode (since it's categorical)

df['Churn'] = df['Churn'].fillna(df['Churn'].mode()[0])

# Fill missing values for 'JoinDate' using mode (since it's categorical)

df['JoinDate'] = df['JoinDate'].fillna(df['JoinDate'].mode()[0])

# Fill missing values for 'Name' using mode (since it's categorical)

df['Name'] = df['Name'].fillna(df['Name'].mode()[0])

# Display the new dataset with missing values filled

print(df)

df['Age'] = df['Age'].replace(-5, df['Age'].mean())  # Replace -5 with the mean of Age

df['MonthlyCharges'] = df['MonthlyCharges'].replace(-30, df['MonthlyCharges'].mean())

df['JoinDate'] = df['JoinDate'].replace('invalid\_date', df['JoinDate'].mode()[0])  # Replace 'invalid\_date' with the mode of JoinDate

# Display the new dataset with missing values filled

print(df)

1. **Display the unique names.**

import pandas as pd

# Assuming 'df' is your DataFrame

# Displaying the unique names in the 'Name' column

unique\_names = df['Name'].unique()

# Printing the unique names

print(unique\_names)

1. **Identify the duplicated row with reference to the attribute “CustomerID” and remove the duplicated row.**

import pandas as pd

# Assuming 'df' is your DataFrame

# Identify the duplicated rows based on 'CustomerID'

duplicated\_rows = df[df.duplicated(subset='CustomerID', keep=False)]

# Display duplicated rows

print("Duplicated rows based on 'CustomerID':")

print(duplicated\_rows)

# Remove duplicated rows based on 'CustomerID'

df\_cleaned = df.drop\_duplicates(subset='CustomerID', keep='first')

# Display the cleaned dataset

print("\nDataset after removing duplicated rows:")

print(df\_cleaned)

\*\*\*