Data Science Assignment

Data Cleaning and Data Wrangling

Name: Kiruthika P

Reg no.: RA2211026010468

1. You are working with a dataset of employee salaries. Some salary values are missing, and the experience column contains values in different formats (e.g., "5 years", "Six Years", "7 YRS").

   You need to:

   a. Fill the missing salary values with the median salary. b. Standardize the experience column so that all values are numerical.

   Input:

   data = {'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'], 'Salary': [50000, np.nan, 55000, np.nan, 60000], 'Experience': ['5 years', 'Six Years', '7 YRS', '3 years', 'Ten Years']}

```
import pandas as pd
import numpy as np

# Initial dataset
data = {
    'Employee': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Salary': [50000, np.nan, 55000, np.nan, 60000],
    'Experience': ['5 years', 'Six Years', '7 YRS', '3 years', 'Ten Years']
}
df = pd.DataFrame(data)

# a. Fill missing salary values with median
df['Salary'].fillna(df['Salary'].median(), inplace=True)

# b. Standardize Experience column to numerical values
experience_map = {
    'five': 5, 'six': 6, 'seven': 7, 'three': 3, 'ten': 10
}
def convert_experience(exp):
    exp = exp.lower()
    for word, num in experience_map.items():
        if word in exp:
            return num
    # If already a number present
    return int(''.join(filter(str.isdigit, exp)))

df['Experience'] = df['Experience'].apply(convert_experience)

print(df)
```

```
  Employee   Salary  Experience
0    Alice  50000.0           5
1      Bob  55000.0           6
2  Charlie  55000.0           7
3    David  55000.0           3
4      Eve  60000.0          10
<ipython-input-1-b7e614e9142c>:13: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

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

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

2. You have a dataset containing customer reviews where the "Feedback" column contains text comments like "Great Service!", "Very Poor Support!", etc. You also have a "Spending" column.

   Your tasks are: a. Create a new column "Sentiment" by classifying reviews that contain the word "Poor" as "Negative" and others as "Positive". b. Bin the Spending column into three categories: "Low", "Medium", "High".

   Input: data = {'Customer': ['John', 'Sarah', 'Mike', 'Lisa', 'Tom'], 'Feedback': ['Great Service!', 'Very Poor Support!', 'Excellent Product!', 'Poor Experience', 'Loved it!'], 'Spending': [150, 700, 1200, 500, 300]}

```
# Initial dataset
```

```
data = {
    'Customer': ['John', 'Sarah', 'Mike', 'Lisa', 'Tom'],
    'Feedback': ['Great Service!', 'Very Poor Support!', 'Excellent Product!', 'Poor Experience', 'Loved it!'],
    'Spending': [150, 700, 1200, 500, 300]
}
df = pd.DataFrame(data)

# a. Sentiment classification
df['Sentiment'] = df['Feedback'].apply(lambda x: 'Negative' if 'poor' in x.lower() else 'Positive')

# b. Bin Spending into categories
bins = [0, 400, 800, float('inf')]
labels = ['Low', 'Medium', 'High']
df['Spending_Category'] = pd.cut(df['Spending'], bins=bins, labels=labels)

print(df)
```

```
  Customer            Feedback  Spending Sentiment Spending_Category
0     John      Great Service!       150  Positive               Low
1    Sarah  Very Poor Support!       700  Negative            Medium
2     Mike  Excellent Product!      1200  Positive              High
3     Lisa     Poor Experience       500  Negative            Medium
4      Tom           Loved it!       300  Positive               Low
```

---

3. Given the two datasets:

   a. A sales dataset containing sales representatives and their respective sales.

   b. A target dataset with the target sales assigned to each representative.

   Write python code to:

   i. Merge both datasets based on the representative name.

   ii. Detect and remove outliers in the sales column using the Interquartile Range (IQR) method.

   Input: sales_data = {'Rep': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'], 'Sales': [50000, 75000, 1200000, 65000, 90000]} target_data = {'Rep': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'], 'Target': [55000, 80000, 95000, 70000, 85000]}

```
# Input datasets
sales_data = {
    'Rep': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Sales': [50000, 75000, 1200000, 65000, 90000]
}
target_data = {
    'Rep': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Target': [55000, 80000, 95000, 70000, 85000]
}

# Create DataFrames
sales_df = pd.DataFrame(sales_data)
target_df = pd.DataFrame(target_data)

# i. Merge based on Rep name
merged_df = pd.merge(sales_df, target_df, on='Rep')

# ii. Remove outliers using IQR
Q1 = merged_df['Sales'].quantile(0.25)
Q3 = merged_df['Sales'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

filtered_df = merged_df[(merged_df['Sales'] >= lower_bound) & (merged_df['Sales'] <= upper_bound)]

print(filtered_df)
```

```
     Rep  Sales  Target
0  Alice  50000   55000
1    Bob  75000   80000
3  David  65000   70000
4    Eve  90000   85000
```