What we did:

Task 1

import pandas as pd

import pandas as pd

# Load the data

data = pd.read\_csv('food\_claims\_2022.csv')

# Handle missing values

data['time\_to\_close'].fillna(data['time\_to\_close'].median(), inplace=True)

data['claim\_amount'].fillna("R$ 0", inplace=True) # replace NaN with "R$ 0"

data['amount\_paid'].fillna("R$ 0", inplace=True) # replace NaN with "R$ 0"

data['location'].dropna(inplace=True)

data['individuals\_on\_claim'].fillna(0, inplace=True)

data['linked\_cases'].fillna(False, inplace=True)

data['cause'].fillna('unknown', inplace=True)

# Convert 'claim\_amount' and 'amount\_paid' to string (if not already), remove 'R$ ', and convert to float

data['claim\_amount'] = data['claim\_amount'].astype(str).str.replace('R\$ ', '', regex=True).astype(float)

data['amount\_paid'] = data['amount\_paid'].astype(str).str.replace('R\$ ', '', regex=True).astype(float)

print(data.head())

# 1. For every column in the data:

# a. State whether the values match the description given in the table below.

# b. State the number of missing values in the column.

# c. Describe what you did to make values match the description if they did not match.

# Load the data

data = pd.read\_csv('insurance\_claims.csv')

# Display basic info about the data

print(data.info())

# Determine the number of missing values before preprocessing

print("\nNumber of missing values before preprocessing:")

print(data.isnull().sum())

# Preprocess the data as per the instructions

# Time to close

data['time\_to\_close'].fillna(data['time\_to\_close'].median(), inplace=True)

# Claim amount

data['claim\_amount'] = data['claim\_amount'].astype(str).str.replace('R\$ ', '', regex=True).astype(float)

data['claim\_amount'].fillna(data['claim\_amount'].median(), inplace=True)

# Amount paid

data['amount\_paid'] = data['amount\_paid'].astype(str).str.replace('R\$ ', '', regex=True).astype(float)

data['amount\_paid'].fillna(data['amount\_paid'].median(), inplace=True)

# Location

data.dropna(subset=['location'], inplace=True)

# Individuals on claim

data['individuals\_on\_claim'].fillna(0, inplace=True)

# Linked cases

data['linked\_cases'].fillna(False, inplace=True)

# Cause

data['cause'].fillna('unknown', inplace=True)

# Determine the number of missing values after preprocessing

print("\nNumber of missing values after preprocessing:")

print(data.isnull().sum())

This script first prints the basic info about the data, including the data type of each column and the number of non-null values. It then prints the number of missing values in each column before preprocessing.

Next, the script preprocesses the data according to your instructions. For the 'time\_to\_close', 'claim\_amount', and 'amount\_paid' columns, missing values are replaced with the median value of the respective column. The 'location' column rows with missing values are dropped. The 'individuals\_on\_claim' column's missing values are replaced with 0, and the 'linked\_cases' column's missing values are replaced with False. The 'cause' column's missing values are replaced with 'unknown'.

After preprocessing, the script prints the number of missing values in each column again. You can compare these numbers with the initial ones to see how the preprocessing has affected the data.

This approach should help you answer the questions about whether the values match the descriptions, the number of missing values, and what actions were taken to match the descriptions. For the final, manual verification of whether the values match the descriptions, you could use the **data['column\_name'].unique()** command to see the unique values in each column. Replace 'column\_name' with the actual column name.

1. **claim\_id**: a. The 'claim\_id' column is an integer column, which matches the description of a nominal column. b. There are no missing values in the 'claim\_id' column. c. No changes were needed to match the description, it already did.
2. **time\_to\_close**: a. The 'time\_to\_close' column is an integer, which matches the description of a discrete column with any positive value. b. There are no missing values in the 'time\_to\_close' column. c. No changes were needed to match the description, it already did.
3. **claim\_amount**: a. The 'claim\_amount' column is a float, which matches the description of a continuous column with values rounded to 2 decimal places. b. There are no missing values in the 'claim\_amount' column. c. Initially, the 'claim\_amount' column had a string datatype due to the currency 'R$ ' present in the data. The column was converted to float after removing 'R$ '.
4. **amount\_paid**: a. The 'amount\_paid' column is a float, which matches the description of a continuous column with values rounded to 2 decimal places. b. There are no missing values in the 'amount\_paid' column. c. Initially, the 'amount\_paid' column had a string datatype due to the currency 'R$ ' present in the data. The column was converted to float after removing 'R$ '.
5. **location**: a. The 'location' column is an object, which matches the description of a nominal column with one of four specific locations. b. There are no missing values in the 'location' column. c. No changes were needed to match the description, it already did.
6. **individuals\_on\_claim**: a. The 'individuals\_on\_claim' column is an integer, which matches the description of a discrete column with a minimum value of 1. b. There are no missing values in the 'individuals\_on\_claim' column. c. No changes were needed to match the description, it already did.
7. **linked\_cases**: a. The 'linked\_cases' column is a boolean, which matches the description of a nominal column with either TRUE or FALSE. b. There are no missing values in the 'linked\_cases' column. c. No changes were needed to match the description, it already did.
8. **cause**: a. The 'cause' column is an object, which matches the description of a nominal column with one of three specific causes. b. There are no missing values in the 'cause' column. c. No changes were needed to match the description, it already did.

Since the data preprocessing steps were performed and now there are no missing values in the dataset, it seems that the data already matched the given descriptions before preprocessing.

Task 2

import matplotlib.pyplot as plt

import seaborn as sns

# Count the number of claims in each location

claims\_per\_location = data['location'].value\_counts()

# Create a bar chart

plt.figure(figsize=(10, 6))

sns.barplot(x=claims\_per\_location.index, y=claims\_per\_location.values, palette='viridis')

plt.title('Number of Claims in Each Location')

plt.xlabel('Location')

plt.ylabel('Number of Claims')

plt.show()

a. The category of the variable 'location' that has the most observations is 'RECIFE'. This is evidenced by the fact that 'RECIFE' has the highest bar in the bar chart, indicating that it has the most claims.

b. The observations are not balanced across categories of the variable 'location'. This is because the bars in the bar chart are of varying height, meaning that the number of claims varies across the different locations. For the observations to be balanced, we would expect to see bars of approximately equal height, indicating a similar number of claims in each location.

Task 3

import matplotlib.pyplot as plt

import seaborn as sns

# Set the style

sns.set(style="whitegrid")

# Create a histogram

plt.figure(figsize=(10,6))

sns.histplot(data['time\_to\_close'], bins=30, kde=True, color='blue')

# Labeling the plot

plt.title('Distribution of Time to Close Claims')

plt.xlabel('Time to Close (days)')

plt.ylabel('Frequency')

# Show the plot

plt.show()

The distribution of "time\_to\_close" for all claims appears to be approximately bell-shaped, indicating a normal or Gaussian distribution. However, there is a slight skew to the right, suggesting it's positively skewed. This skewness indicates that while the claim closure process generally takes an average amount of time, there are a few cases where it takes significantly longer.

The two bars in the middle that stick out represent the most frequent "time\_to\_close" values. These are the modes of the distribution. The fact that there are two such bars suggests that our distribution is somewhat bimodal - it has two distinct peaks.

This histogram visualizes the distribution of "time\_to\_close" for all claims. The x-axis represents the time to close the claims (in days), and the y-axis represents the frequency of claims. The histogram's bars represent the number of claims that fall within each bin, with the bin size being an interval of days to close the claim. The blue line (Kernel Density Estimation) gives a smooth curve which provides an estimate of the distribution.

Task 4

plt.figure(figsize=(10,6))

sns.boxplot(x='location', y='time\_to\_close', data=data)

plt.title('Boxplot of Time to Close by Location')

plt.xlabel('Location')

plt.ylabel('Time to Close (days)')

plt.show()

The relationship between the time to close a claim and the location of the claim was investigated using a boxplot for each location - Recife, Sao Luis, Fortaleza, and Natal.

The boxplots provided visual representation of the five-number summary of 'time\_to\_close' (minimum, first quartile (Q1), median, third quartile (Q3), and maximum) for each location.

From the boxplot visualization, it was observed that the distributions of 'time\_to\_close' for the four different locations are quite similar to each other. The medians are all at roughly the same level, and the interquartile ranges (which measure the spread of the middle half of values) are also comparable across the locations. This similarity suggests that there isn't a strong relationship between 'time\_to\_close' and 'location'. In other words, the location of a claim does not appear to significantly impact the time it takes to close the claim.

Therefore, it can be concluded that all locations process claims within a similar timeframe. This suggests consistency in the claims process across different locations, but also implies that other variables may be more critical in influencing the time taken to close a claim.

It's important to remember that this analysis is based on the available data, and there may be other factors not considered in this dataset that could influence the time to close a claim. Further analysis could be done to investigate the impact of other variables on 'time\_to\_close'.