Datacamp project looking at length of time restaurant franchise locations take to close claims.

```
WITH converted AS (SELECT id AS claim_id,
                  time_to_close,
    CAST(
        REPLACE(claim_amount, 'R$', '') -- Remove 'R$'
        AS NUMERIC
    ) AS converted_claim_amnt,
                amount_paid,
                location,
                individuals_on_claim,
                linked_cases,
                cause
FROM 'fc.csv'),

linked_na AS (SELECT * -- select records with NA
FROM converted
WHERE linked_cases = 'NA'),

fixed AS (SELECT claim_id,
        time_to_close:: INTEGER AS time_to_close,
        converted_claim_amnt,
    (CASE WHEN amount_paid = 'NA' THEN '28807.83' -- imputiing Median value
          ELSE amount_paid END):: NUMERIC AS      -- median value between

                                    -- 28807.81 - 28807.85 = 28807.83
          amount_paid,
        location,
        individuals_on_claim,
    CASE WHEN claim_id in (SELECT claim_id FROM linked_na) THEN REPLACE(linked_cases,'NA', 'FALSE')
    ELSE linked_cases END AS linked_fix,
    CASE WHEN LOWER(cause) ILIKE '%meat%' THEN 'meat'
             WHEN LOWER(cause) ILIKE '%vegetable%' THEN 'vegetable'
             ELSE cause END AS cause
FROM converted)

-------------------------

SELECT *
FROM fixed;
```

Output/ Columns continued next page

	claim_id	time_to_close	converted_claim_amnt	amount_paid	location
0	1	317	74474.55	51231.37	RECIFE
1	2	195	52137.83	42111.3	FORTALEZA
2	3	183	24447.2	23986.3	SAO LUIS

	claim_id	time_to_close	converted_claim_amnt	amount_paid	location
3	4	186	29006.28	27942.72	FORTALEZA
4	5	138	19520.6	16251.06	RECIFE
5	6	183	47529.14	38011.98	NATAL
6	7	190	39073.26	29826.04	SAO LUIS
7	8	183	29870.56	29727.52	SAO LUIS
8	9	149	26644.46	23362.14	RECIFE
9	10	149	11544.68	9680.82	NATAL

individuals_on_claim	linked_fix	cause
15	FALSE	unknown
12	TRUE	unknown
10	TRUE	meat
11	FALSE	meat
11	FALSE	vegetable
11	FALSE	unknown
12	FALSE	meat
8	TRUE	unknown
9	FALSE	meat
6	FALSE	vegetable

For every column in the data:

a. State whether the values match the description given in the table provided.

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.

claim_id:
a) values matched description provided

b) 0 missing values
c) N/A

time_to_close:
a) values matched description provided
b) 0 missing values
c) N/A

claim_amount:
a) values partially did not match description provided
b) 0 missing values
c) Values for claim_amount were varchar type, as they contained 'R$' at the start of each amount, thus the column was not of type NUMERIC. I used REPLACE() to eliminate 'R$' and cast the field to numeric - to preserve the decimal values, rendering the column as in the description provided. I renamed the column converted_claim_amnt. This was set as a common table expression for later use (converted).

amount_paid:
a) values partially did not match description provided, 36 instances of 'NA'
b) 36 missing values
c) Column contained 36 instances of 'NA'. I changed them to the Median value which I accomplished by ordering data by amount_paid ascending identifying 1964 non-null values resulting in median value being between index (claim_id) 982 and 983 (28807.81 - 28807.85 = -0.04; |-0.04/2| = **0.02**), meaning the median value is 28807.83. I implemented a CASE WHEN clause replacing'NA' with '28807.83' and then cast it as NUMERIC() to preserve decimals values. The column now matches the description provided.

```
CASE WHEN amount_paid = 'NA'  THEN '28807.83'
            ELSE amount_paid END):: NUMERIC AS amount_paid
```

location:
a) values matched description provided
b) 0 missing values
c) N/A

individuals_on_claim:
a) values matched description provided
b) 0 missing values
c) N/A

linked_cases:
a) values partially did not match description provided; 26 instances of 'NA'
b) 26 missing values
c) Column contained 26 'NA' values. I used two common table expressions in addition to the first one fixing claim_amount (see above). First I selected the 26 records with 'NA' values and named the table *linked_na*. Then I filtered the *converted* table with the claim_id in the *linked_na* table within a

CASE WHEN clause replacing'NA' with 'FALSE'. I renamed the column linked_fix. The column now matches the description provided.

```
CASE WHEN claim_id in (SELECT claim_id FROM linked_na)
                    THEN REPLACE(linked_cases,'NA', 'FALSE')
            ELSE linked_cases END AS linked_fix
```

cause:
a) values partially did not match description provided, multiple forms of 'meat'/'vegetable'(s)
b) 0 missing values
c) Column contained multiple forms/cases of 'meat' and 'vegetable'. Implemented a CASE WHEN clause standardizing all 3 categories 'meat', 'vegetable', and 'unkown' as lower case, thus rendering the column as in the description provided.

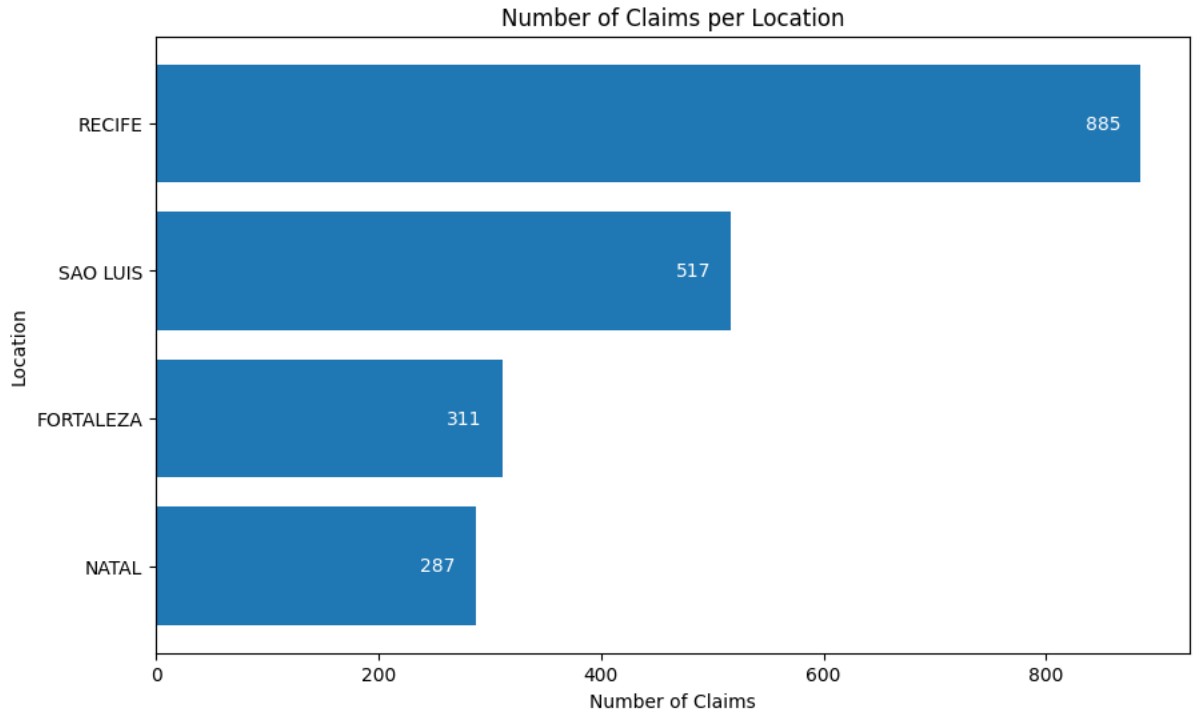
```
CASE WHEN LOWER(cause) ILIKE '%meat%' THEN 'meat'
    WHEN LOWER(cause) ILIKE '%vegetable%' THEN 'vegetable'
    ELSE cause END AS cause
```

```
import matplotlib.pyplot as plt
import pandas as pd

# Create a DataFrame from your data
df = pd.DataFrame(df)

# Group the data by 'location' and count the number of claims for each location
grouped_data = df.groupby('location')['claim_id'].count().reset_index()

# Sort the data frame by the count of claims in ascending order
grouped_data = grouped_data.sort_values(by='claim_id', ascending=True)

# Create a horizontal bar plot
plt.figure(figsize=(10, 6))
bars = plt.barh(grouped_data['location'], grouped_data['claim_id'])
plt.ylabel('Location')
plt.xlabel('Number of Claims')
plt.title('Number of Claims per Location')

# Add labels for the count of claims to the left of the bars
for bar in bars:
    plt.text(bar.get_width() - 50, bar.get_y() + bar.get_height() / 2, str(int(bar.get_width())), ha='left',
va='center',color='white')

plt.show()
```

Number of claims in each location.

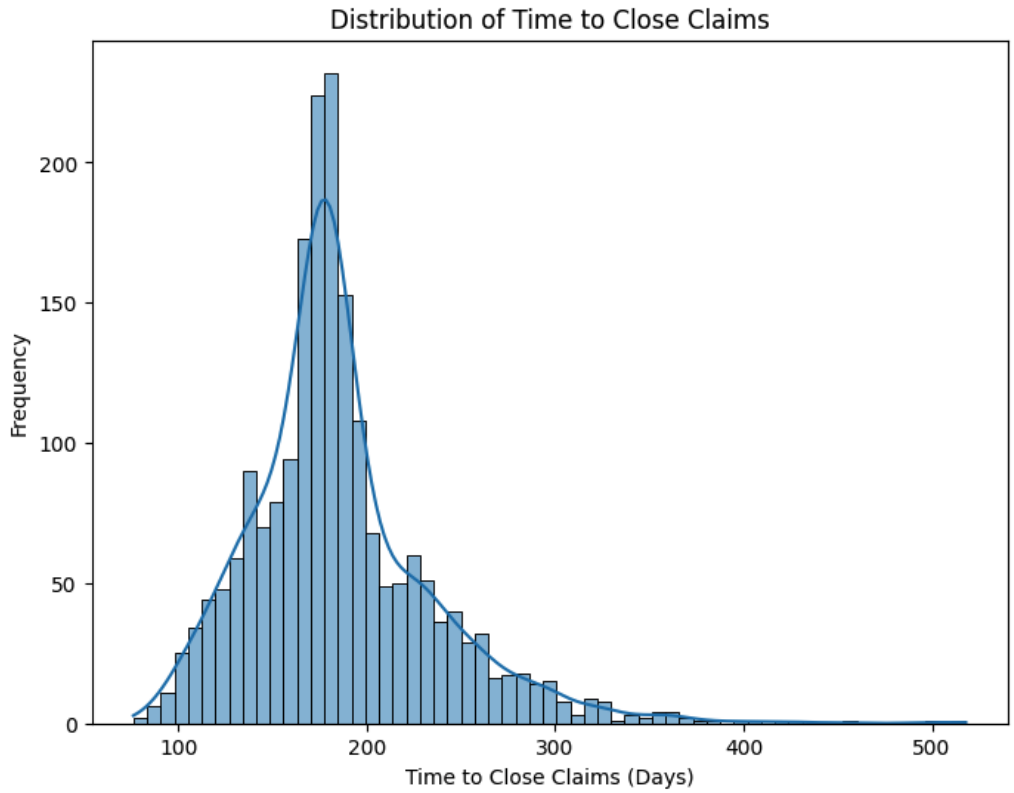
a. State which category of the variable location has the most observations
b. Explain whether the observations are balanced across categories of the variable location

According to the figure, the number of claims is not balanced across locations. The Recife office has the most claims (885 claims), the Sao Luis office has the second most (517 claims), the Fortaleza office has the third most (311 claims), and the Natal office has the fewest (287 claims).




```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Create a DataFrame from your data
df = pd.DataFrame(df)

# Create a histogram
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='time_to_close', bins='auto', kde=True)

#plt.hist(df['time_to_close'], bins=128, edgecolor='k')  # Adjust the number of bins as needed
plt.xlabel('Time to Close Claims (Days)')
plt.ylabel('Frequency')
plt.title('Distribution of Time to Close Claims')
plt.grid(False)
plt.show()
```

Distribution of time to close claims.




```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Create a DataFrame from your data
df = pd.DataFrame(df)

# Calculate the average time to close claims for each location
average_time_by_location = df.groupby('location')['time_to_close'].mean().reset_index()

# Sort the DataFrame by 'time_to_close' in ascending order
average_time_by_location = average_time_by_location.sort_values(by='time_to_close',
ascending=False).reset_index()

average_time_by_location = average_time_by_location[['location', 'time_to_close']]

# Create a horizontal bar plot
plt.figure(figsize=(10, 5))
sns.barplot(x='time_to_close', y='location', data=average_time_by_location, color=(0.0, 0.45, 0.78))
plt.xlabel('Average Time to Close Claims (Days)')
plt.ylabel('Location')
plt.title('Average Number of Days to Close Claims by Location')
```

```
# Add average time labels to the bars
for index, row in average_time_by_location.iterrows():
    plt.text(row['time_to_close'] - 25, index, f'{row["time_to_close"]:.2f} days', va='center', color='white')

plt.show()
```

Relationship between time to close and location.