In [1]:
```
#The first step we're taking in this analysis is importing the dataset  using the pandas library.
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data=pd.read_csv('food_claims_uncleaned.csv')
print(data)
```

```
      claim_id  time_to_close claim_amount  amount_paid   location  \
0            1            317  R$ 74474.55     51231.37     RECIFE
1            2            195  R$ 52137.83     42111.30  FORTALEZA
2            3            183   R$ 24447.2     23986.30   SAO LUIS
3            4            186  R$ 29006.28     27942.72  FORTALEZA
4            5            138   R$ 19520.6     16251.06     RECIFE
...        ...            ...          ...          ...        ...
1995      1996            176   R$ 28982.3     24265.02     RECIFE
1996      1997            166   R$ 5188.44      4772.77  FORTALEZA
1997      1998            179  R$ 11975.85     10087.81     RECIFE
1998      1999            162  R$ 23516.28     23310.24     RECIFE
1999      2000            150    R$ 8051.4      6417.92     RECIFE

      individuals_on_claim linked_cases      cause
0                       15        False    unknown
1                       12         True    unknown
2                       10         True       meat
3                       11        False       meat
4                       11        False  vegetable
...                    ...          ...        ...
1995                    10        False       meat
1996                     2         True       meat
1997                     4         True       meat
1998                     9        False       meat
1999                     4        False  vegetable

[2000 rows x 8 columns]
```

In [3]:
```
# We're moving to data cleaning and validation.
#1.The first approach is checking for null values in the dataset , the result below shows that two columns has null values.
```

In [4]:
```
data.isna().any()
```

Out[4]:
```
claim_id               False
time_to_close          False
claim_amount           False
amount_paid             True
location               False
individuals_on_claim   False
linked_cases            True
cause                  False
dtype: bool
```

In [5]:
```
#2. We need to know the amount of null values these two columns have.
#the result below shows that the two colimns have 36 and 26 null value which makes the data incomplete and dirty.
```

In [6]:
```
data.isna().sum()
```

Out[6]:
```
claim_id                0
time_to_close           0
claim_amount            0
amount_paid            36
location                0
individuals_on_claim    0
linked_cases           26
cause                   0
dtype: int64
```

In [7]:
```
#3. the next step we took is replacing the null values with the mean of the non missing values in the column 'amount_paid'.
```

In [8]:
```
data['amount_paid']=data['amount_paid'].fillna(data['amount_paid'].mean())
data.head()
```

Out[8]:

	claim_id	time_to_close	claim_amount	amount_paid	location	individuals_on_claim	linked_cases	cause
0	1	317	R$ 74474.55	51231.37	RECIFE	15	False	unknown
1	2	195	R$ 52137.83	42111.30	FORTALEZA	12	True	unknown
2	3	183	R$ 24447.2	23986.30	SAO LUIS	10	True	meat
3	4	186	R$ 29006,28	27942,72	FORTALEZA	11	False	meat
4	5	138	R$ 19520.6	16251.06	RECIFE	11	False	vegetable

In [9]:
```
#4. the missing values of the column 'linked_cases' are replaced with "False"
```

In [10]:
```
data['linked_cases']=data['linked_cases'].fillna('False')
data.head()
```

Out[10]:

	claim_id	time_to_close	claim_amount	amount_paid	location	individuals_on_claim	linked_cases	cause
0	1	317	R$ 74474.55	51231.37	RECIFE	15	False	unknown
1	2	195	R$ 52137.83	42111.30	FORTALEZA	12	True	unknown
2	3	183	R$ 24447.2	23986.30	SAO LUIS	10	True	meat
3	4	186	R$ 29006.28	27942.72	FORTALEZA	11	False	meat
4	5	138	R$ 19520.6	16251.06	RECIFE	11	False	vegetable

In [11]:
```
#5.the column 'location' has inconsistencies(the values were in upper case)
#it was rectified by using the capitalize() function.
```

In [12]:
```
data['location']=data['location'].str.capitalize()
data.head()
```

Out[12]:

	claim_id	time_to_close	claim_amount	amount_paid	location	individuals_on_claim	linked_cases	cause
0	1	317	R$ 74474.55	51231.37	Recife	15	False	unknown
1	2	195	R$ 52137.83	42111.30	Fortaleza	12	True	unknown
2	3	183	R$ 24447.2	23986.30	Sao luis	10	True	meat
3	4	186	R$ 29006.28	27942.72	Fortaleza	11	False	meat
4	5	138	R$ 19520.6	16251.06	Recife	11	False	vegetable

In [13]:
```
#5.claim_amount column had 'R$' which made it a text value instead of float so text values were replaced with empty string .
```

In [14]:
```
data['claim_amount']=data['claim_amount'].str.replace('R', '')
data['claim_amount']=data['claim_amount'].str.replace('$', '')
data.head()
```

```
C:\Users\hayod\AppData\Local\Temp\ipykernel_3416\1145647775.py:2: FutureWarning: The default value of regex will change from Tr
ue to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings whe
n regex=True.
  data['claim_amount']=data['claim_amount'].str.replace('$', '')
```

Out[14]:

	claim_id	time_to_close	claim_amount	amount_paid	location	individuals_on_claim	linked_cases	cause
0	1	317	74474.55	51231.37	Recife	15	False	unknown
1	2	195	52137.83	42111.30	Fortaleza	12	True	unknown
2	3	183	24447.2	23986.30	Sao luis	10	True	meat
3	4	186	29006.28	27942.72	Fortaleza	11	False	meat
4	5	138	19520.6	16251.06	Recife	11	False	vegetable

In [15]:
```
#5b.the column was coverted from text type to float type
```

In [16]:
```
data['claim_amount']=data['claim_amount'].astype('float')
data.head()
```

Out[16]:

	claim_id	time_to_close	claim_amount	amount_paid	location	individuals_on_claim	linked_cases	cause
0	1	317	74474.55	51231.37	Recife	15	False	unknown
1	2	195	52137.83	42111.30	Fortaleza	12	True	unknown
2	3	183	24447.20	23986.30	Sao luis	10	True	meat
3	4	186	29006.28	27942.72	Fortaleza	11	False	meat
4	5	138	19520.60	16251.06	Recife	11	False	vegetable

In [ ]:
```
```

In [17]:
```
data['amount_left']=data['claim_amount']-data['amount_paid']
data['percentage_left']=data['amount_left']/data['claim_amount']*100
data.head()
```

Out[17]:

	claim_id	time_to_close	claim_amount	amount_paid	location	individuals_on_claim	linked_cases	cause	amount_left	percentage_left
0	1	317	74474.55	51231.37	Recife	15	False	unknown	23243.18	31.209561
1	2	195	52137.83	42111.30	Fortaleza	12	True	unknown	10026.53	19.230816
2	3	183	24447.20	23986.30	Sao luis	10	True	meat	460.90	1.885287
3	4	186	29006.28	27942.72	Fortaleza	11	False	meat	1063.56	3.666654
4	5	138	19520.60	16251.06	Recife	11	False	vegetable	3269.54	16.749178

In [18]:
```
#6.The column 'cause' had some inconsistencies with trailing spaces and cases irregularity.
#value_counts()
```

In [19]:
```
data['cause'].value_counts()
```

Out[19]:
```
meat          943
unknown       713
vegetable     314
VEGETABLES     16
 Meat          14
Name: cause, dtype: int64
```

In [20]:
```
#6b the irregularities were fixed with the lower(),replace() and strip() function
```

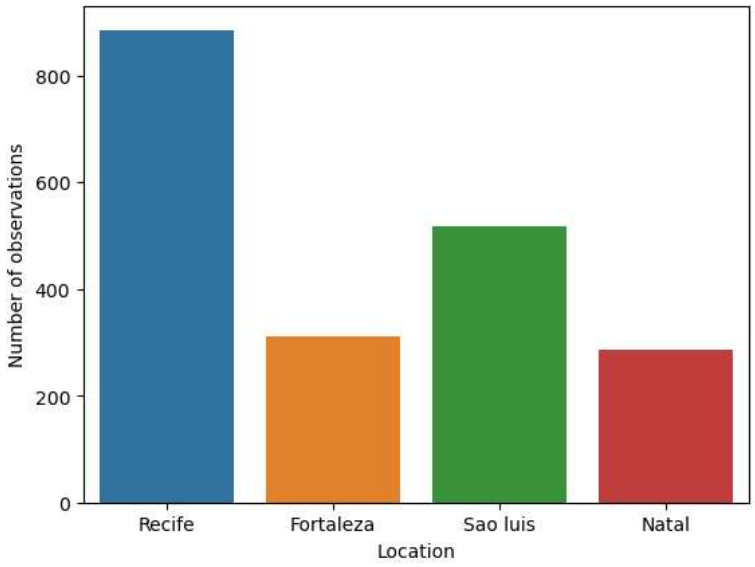
In [21]:
```
data['cause']=data['cause'].str.lower()
data['cause']=data['cause'].str.strip()
data['cause']=data['cause'].str.replace('vegetables','vegetable')
data['cause'].value_counts()
```

Out[21]:
```
meat         957
unknown      713
vegetable    330
Name: cause, dtype: int64
```

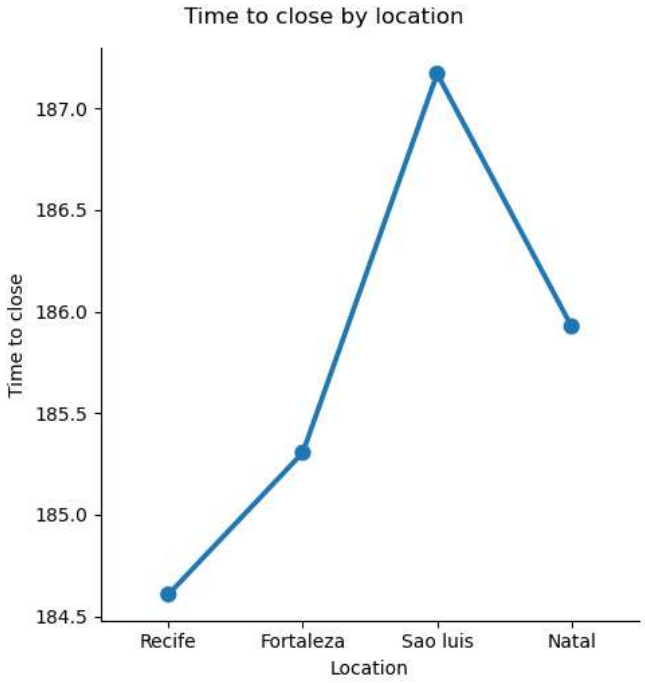
In [22]:
```
#The irregularities have been fixed so we're going to save the clean data as a new csv file.
data.to_csv('food.csv',index=False)
```

In [23]:
```
#The first visualization shows that Recife has the highest number of observations.
```

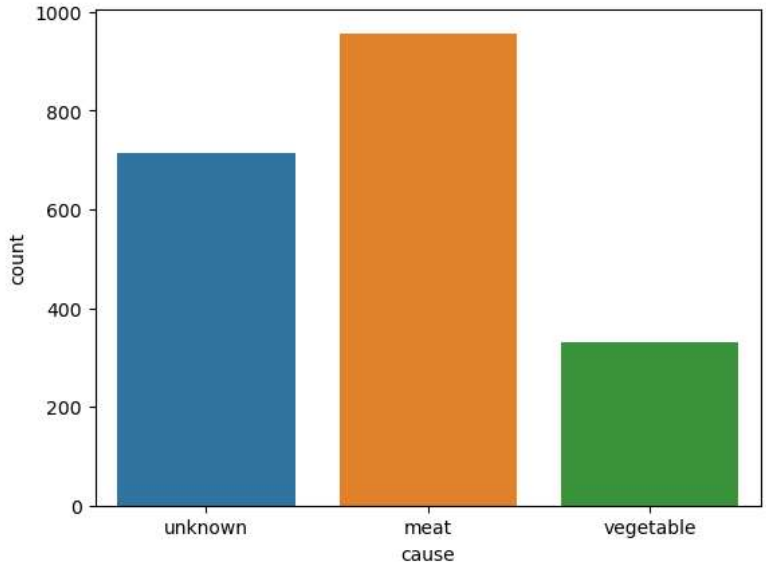
In [24]:
```
sns.countplot(x='location',data=data)
plt.xlabel('Location')
plt.ylabel('Number of observations')
plt.show()
```



In [25]:
```
g=sns.catplot(x='location',y='time_to_close',data=data,kind='point',errorbar=None)
g.fig.suptitle('Time to close by location',y=1.03)
g.set(xlabel='Location',ylabel="Time to close")
plt.show()
#the diagram shows that sao lius has the highest average time interval of claims.
#the time intervals varies from an average of 184.5 to 187.0 among all locations
```



In [26]:
```
g=sns.countplot(x='cause',data=data)
plt.show()
#this plot shows that meat is the main cause of all claims and then unknown causes then vegetables.
#unknown causes here describes the remaining kinds of food prepared across the location
```



In [27]:
```
#distribution of time_to_close for all claims.the illustration shows that 200 days has the highest time rande for the food claims
data['time_to_close'].hist(bins= 5)
plt.xlabel('Time to close')
plt.ylabel('Distribution')
plt.suptitle('Distribution of time to close')
plt.show()
```