In [11]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
```

In [12]:
```
df  = pd.read_csv("fraudTest.csv")
df.head(5)
```

Out[12]:

ast	gender	street	...	lat	long	city_pop	job	dob	trans_num	unix_time	merch_lat	merch_long	is_fraud
liott	M	351 Darlene Green	...	33.9659	-80.9355	333497	Mechanical engineer	1968-03-19	2da90c7d74bd46a0caf3777415b3ebd3	1371816865	33.986391	-81.200714	0
ıms	F	3638 Marsh Union	...	40.3207	-110.4360	302	Sales professional, IT	1990-01-17	324cc204407e99f51b0d6ca0055005e7	1371816873	39.450498	-109.960431	0
pez	F	9333 Valentine Point	...	40.6729	-73.5365	34496	Librarian, public	1970-10-21	c81755dbbbea9d5c77f094348a7579be	1371816893	40.495810	-74.196111	0
ıms	M	32941 Krystal Mill Apt. 552	...	28.5697	-80.8191	54767	Set designer	1987-07-25	2159175b9efe66dc301f149d3d5abf8c	1371816915	28.812398	-80.883061	0
sey	M	5783 Evan Roads Apt. 465	...	44.2529	-85.0170	1126	Furniture designer	1955-07-06	57ff021bd3f328f8738bb535c302a31b	1371816917	44.959148	-85.884734	0

In [13]:
```
df.columns
```

Out[13]:
```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

In [47]:
```
dfcol = df[['amt','city_pop','unix_time', 'category']]
dfcol = pd.get_dummies(dfcol, columns=['category'])
dfcol.head()
```

Out[47]:

ıtegory_grocery_net	category_grocery_pos	category_health_fitness	category_home	category_kids_pets	category_misc_net	category_misc_pos	category_pers
0	0	0	0	0	0	0	
0	0	0	0	0	0	0	
0	0	1	0	0	0	0	
0	0	0	0	0	0	1	
0	0	0	0	0	0	0	

In [48]:
```
dfpred = df[['is_fraud']]
dfpred
```

Out[48]:

	is_fraud
0	0
1	0
2	0
3	0
4	0
...	...
555714	0
555715	0
555716	0
555717	0
555718	0

555719 rows × 1 columns

In [49]:
```
X = dfcol
y = dfpred
```

1. Use scikit learn to split data into 70/30

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

2. Make 3 sets of training data (Oversample, Undersample and SMOTE)

In [51]:
```
from imblearn.over_sampling import RandomOverSampler, SMOTE
from imblearn.under_sampling import RandomUnderSampler


# Oversampled training data
oversampler = RandomOverSampler()
X_train_over, y_train_over = oversampler.fit_resample(X_train, y_train)

# Undersampled training data
undersampler = RandomUnderSampler()
X_train_under, y_train_under = undersampler.fit_resample(X_train, y_train)

# SMOTE training data
smote = SMOTE()
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

```

3. Train 3 logistic regression models

In [55]:
```
from sklearn.linear_model import LogisticRegression

# Initialize logistic regression classifier
log_reg = LogisticRegression()

# Fit the model using the oversampled training data
log_reg_over = log_reg.fit(X_train_over, y_train_over)

# Fit the model using the undersampled training data
log_reg_under = log_reg.fit(X_train_under, y_train_under)

# Fit the model using the SMOTE training data
log_reg_smote = log_reg.fit(X_train_smote, y_train_smote)

```

```
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was pas
sed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was pas
sed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\parzu\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was pas
sed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

4. Test the 3 models

In [56]:

```
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score

# Make predictions on the test data using the oversampled model
y_pred_over = log_reg_over.predict(X_test)

# Make predictions on the test data using the undersampled model
y_pred_under = log_reg_under.predict(X_test)

# Make predictions on the test data using the SMOTE model
y_pred_smote = log_reg_smote.predict(X_test)

# Evaluate the oversampled model
print("Oversampled model evaluation:")
print("Accuracy:", accuracy_score(y_test, y_pred_over))
print("F1-Score:", f1_score(y_test, y_pred_over))
print("Recall:", recall_score(y_test, y_pred_over))
print("Precision:", precision_score(y_test, y_pred_over))

# Evaluate the undersampled model
print("\nUndersampled model evaluation:")
print("Accuracy:", accuracy_score(y_test, y_pred_under))
print("F1-Score:", f1_score(y_test, y_pred_under))
print("Recall:", recall_score(y_test, y_pred_under))
print("Precision:", precision_score(y_test, y_pred_under))

# Evaluate the SMOTE model
print("\nSMOTE model evaluation:")
print("Accuracy:", accuracy_score(y_test, y_pred_smote))
print("F1-Score:", f1_score(y_test, y_pred_smote))
print("Recall:", recall_score(y_test, y_pred_smote))
print("Precision:", precision_score(y_test, y_pred_smote))

```

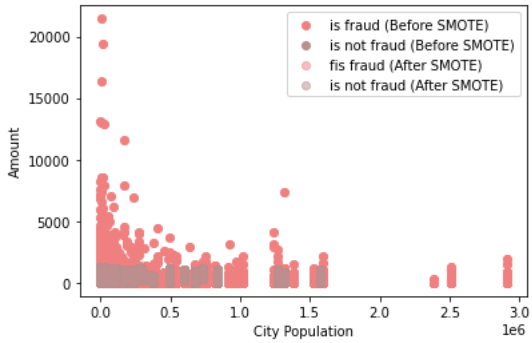
```
Oversampled model evaluation:
Accuracy: 0.16996569015571392
F1-Score: 0.008455144740613357
Recall: 0.8575581395348837
Precision: 0.004248516619620946

Undersampled model evaluation:
Accuracy: 0.16996569015571392
F1-Score: 0.008455144740613357
Recall: 0.8575581395348837
Precision: 0.004248516619620946

SMOTE model evaluation:
Accuracy: 0.16996569015571392
F1-Score: 0.008455144740613357
Recall: 0.8575581395348837
Precision: 0.004248516619620946
```

5. The accuracy is the same for all three models which is unusual, suggesting that the model may have issues.

6. Pick 2 features and plot the 2 classes before and after SMOTE

In [69]:

```
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE

# Get the data
X = df[['city_pop', 'amt']]
y = df['is_fraud']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Plot the original data
plt.scatter(X_train[y_train == 0]['city_pop'], X_train[y_train == 0]['amt'], color='lightcoral', label='is fraud (Before
plt.scatter(X_train[y_train == 1]['city_pop'], X_train[y_train == 1]['amt'], color='rosybrown', label='is not fraud (Bef

# Apply SMOTE to the training data
smote = SMOTE(sampling_strategy='minority')
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Plot the SMOTE data
plt.scatter(X_train_smote[y_train_smote == 0]['city_pop'], X_train_smote[y_train_smote == 0]['amt'], color='lightcoral',
plt.scatter(X_train_smote[y_train_smote == 1]['city_pop'], X_train_smote[y_train_smote == 1]['amt'], color='rosybrown', a

plt.xlabel('City Population')
plt.ylabel('Amount')
plt.legend()
plt.show()

```



In [ ]:

```
1
```