```
import pandas as pd
import numpy as np
import seaborn as sns
import threading
```

```
ds = pd.read_csv('fraudTest.csv')
```

```
ds.head()
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel

5 rows × 23 columns

```
ds.isna().sum()
```

```
Unnamed: 0               0
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
unix_time                0
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
```

```
ds.shape
```

```
(555719, 23)
```

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 555719 entries, 0 to 555718
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   Unnamed: 0             555719 non-null  int64  
 1   trans_date_trans_time  555719 non-null  object 
 2   cc_num                 555719 non-null  int64  
 3   merchant               555719 non-null  object 
 4   category               555719 non-null  object 
 5   amt                    555719 non-null  float64
 6   first                  555719 non-null  object 
 7   last                   555719 non-null  object 
```

```
 8   gender                 555719 non-null  object
 9   street                 555719 non-null  object
 10  city                   555719 non-null  object
 11  state                  555719 non-null  object
 12  zip                    555719 non-null  int64
 13  lat                    555719 non-null  float64
 14  long                   555719 non-null  float64
 15  city_pop               555719 non-null  int64
 16  job                    555719 non-null  object
 17  dob                    555719 non-null  object
 18  trans_num              555719 non-null  object
 19  unix_time              555719 non-null  int64
 20  merch_lat              555719 non-null  float64
 21  merch_long             555719 non-null  float64
 22  is_fraud               555719 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 97.5+ MB
```

```
ds.describe(include='all')
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category
count	555719.000000	555719	5.557190e+05	555719	555719
unique	NaN	544760	NaN	693	14
top	NaN	2020-12-19 16:02:22	NaN	fraud_Kilback LLC	gas_transport
freq	NaN	4	NaN	1859	56370
mean	277859.000000	NaN	4.178387e+17	NaN	NaN
std	160422.401459	NaN	1.309837e+18	NaN	NaN
min	0.000000	NaN	6.041621e+10	NaN	NaN
25%	138929.500000	NaN	1.800429e+14	NaN	NaN
50%	277859.000000	NaN	3.521417e+15	NaN	NaN
75%	416788.500000	NaN	4.635331e+15	NaN	NaN
max	555718.000000	NaN	4.992346e+18	NaN	NaN

11 rows × 23 columns

```
ds.columns
```

```
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')
```

```
ds = ds.drop(columns=['Unnamed: 0','merchant','category','city','state','cc_num','first', 'last','trans_num','unix_time','street','merch
```

```
ds.head()
```

	trans_date_trans_time	amt	gender	lat	long	city_pop	dob	is_fraud
0	2020-06-21 12:14:25	2.86	M	33.9659	-80.9355	333497	1968-03-19	0
1	2020-06-21 12:14:33	29.84	F	40.3207	-110.4360	302	1990-01-17	0
2	2020-06-21 12:14:53	41.28	F	40.6729	-73.5365	34496	1970-10-21	0

```
ds['gender'].unique()
```

```
array(['M', 'F'], dtype=object)
```

```
# Binarizing Gender column
def gender_binarizer(x):
    if x=='F':
        return 1
    if x=='M':
        return 0

ds['gender'] = ds['gender'].transform(gender_binarizer)
```

```
ds = ds.loc[:99999,ds.dtypes!= object]
```

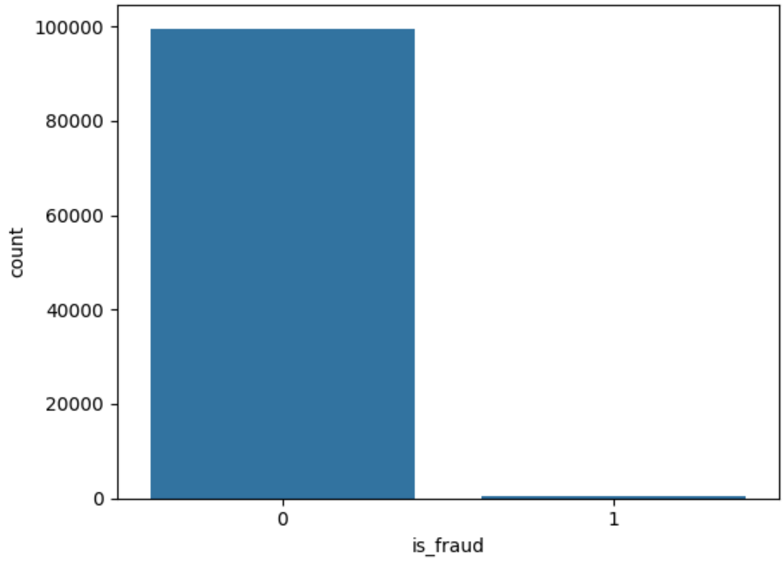
```
ds.head()
```

	amt	gender	lat	long	city_pop	is_fraud
0	2.86	0	33.9659	-80.9355	333497	0
1	29.84	1	40.3207	-110.4360	302	0
2	41.28	1	40.6729	-73.5365	34496	0
3	60.05	0	28.5697	-80.8191	54767	0
4	3.19	0	44.2529	-85.0170	1126	0

Next steps: Generate code with ds View recommended plots

```
sns.countplot(x=ds['is_fraud'])
```

```
<Axes: xlabel='is_fraud', ylabel='count'>
```




```
X_in = ds.drop('is_fraud',axis=1)
y_in =ds['is_fraud']
```

```
y_in.value_counts()
```

```
is_fraud
0    99598
1      402
Name: count, dtype: int64
```

```
from imblearn.over_sampling import SMOTEN
X,y = SMOTEN().fit_resample(X_in,y_in)
```

```
y.value_counts()
```

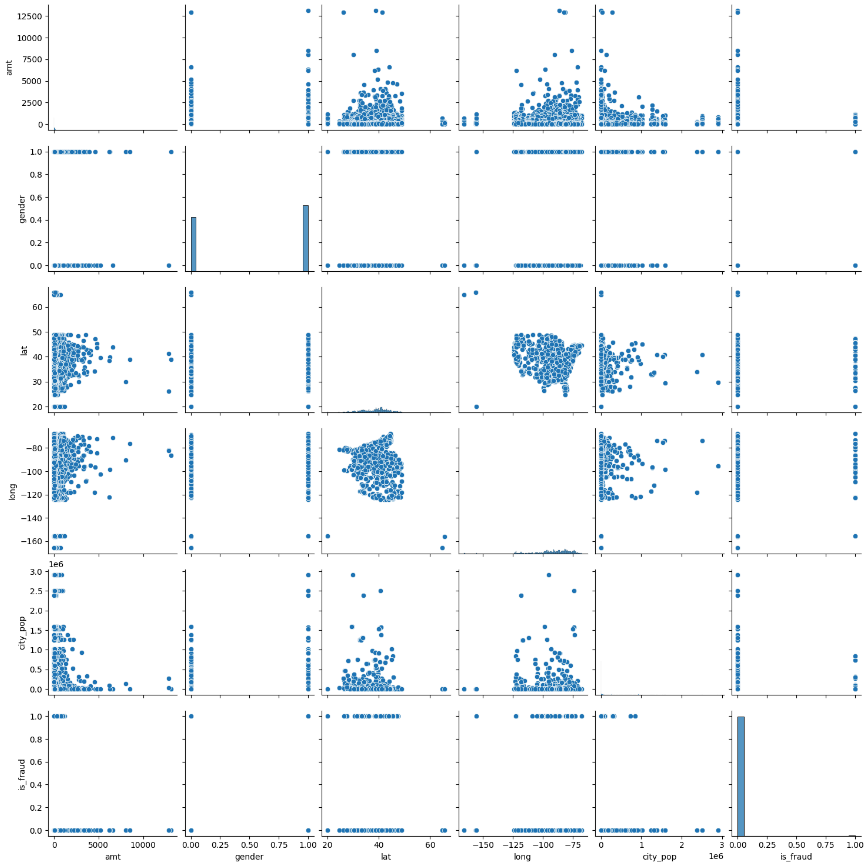
```
is_fraud
0    99598
1    99598
Name: count, dtype: int64
```

```
ds.corr()
```

	amt	gender	lat	long	city_pop	is_fraud
amt	1.000000	-0.002058	0.004497	0.000323	0.002691	0.181555
gender	-0.002058	1.000000	-0.044486	-0.053116	0.027901	-0.000072
lat	0.004497	-0.044486	1.000000	-0.017368	-0.154416	0.009932
long	0.000323	-0.053116	-0.017368	1.000000	-0.051689	-0.003700
city_pop	0.002691	0.027901	-0.154416	-0.051689	1.000000	-0.003776
is_fraud	0.181555	-0.000072	0.009932	-0.003700	-0.003776	1.000000

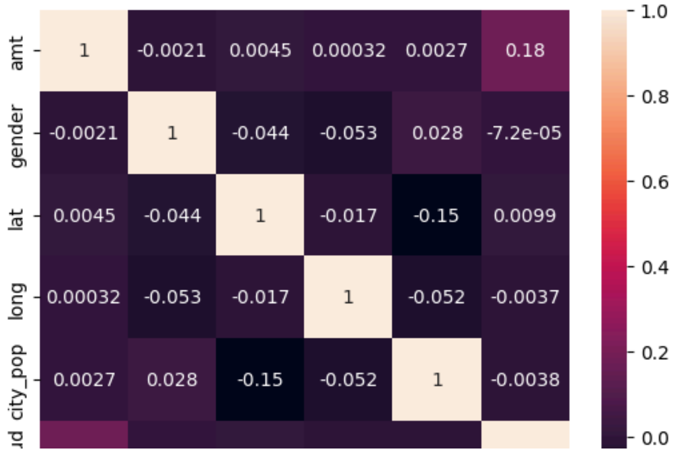
```
sns.pairplot(ds)
```

```
<seaborn.axisgrid.PairGrid at 0x7c21c0945270>
```




```
sns.heatmap(ds.corr(),annot=True)
```

```
<Axes: >
```




```
#splitting
from sklearn.model_selection import train_test_split

X_test, X_train, y_test, y_train = train_test_split(X, y, test_size=0.2, random_state=0)


#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


X_train
```

```
array([[-0.05998004, -1.08837945,  0.81060419,  0.16883475,  0.08114222],
       [ 0.52685656, -1.08837945,  0.45671234,  0.60325075,  0.80572182],
       [-0.82311466, -1.08837945, -0.45185704,  0.4407713 , -0.28821054],
       ...,
       [-0.51908479,  0.91879721, -0.61805674, -0.09117796, -0.3035924 ],
       [-0.32571843, -1.08837945,  0.41304893,  0.4451996 , -0.30612877],
       [ 0.89674553,  0.91879721,  0.11827964,  1.03009958, -0.30540468]])
```

```
#LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
lo = LogisticRegression()
lo.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
yp1 = lo.predict(X_test)


from sklearn.metrics import precision_score, recall_score, accuracy_score, f1_score
from sklearn.metrics import mean_squared_error, mean_absolute_error


print("LOGISTIC REGRESSION")
print("Accuracy= ",accuracy_score(y_test,yp1))
print("Precision= ",precision_score(y_test,yp1))
print("Recall= ",recall_score(y_test,yp1))
print("F1_Score= ",f1_score(y_test,yp1))
print("MSE= ",mean_squared_error(y_test,yp1))
print("MAE= ",mean_absolute_error(y_test,yp1))
```

```
LOGISTIC REGRESSION
Accuracy=  0.7954203167750195
Precision=  0.8528456751149995
Recall=  0.7132189326796153
F1_Score=  0.7768079032224937
MSE=  0.20457968322498055
```