In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('D:\Final Year Project\Online_Fraud.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldb
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15999 entries, 0 to 15998
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   step            15999 non-null  int64
 1   type            15999 non-null  object
 2   amount          15999 non-null  float64
 3   nameOrig        15999 non-null  object
 4   oldbalanceOrg   15999 non-null  float64
 5   newbalanceOrig  15999 non-null  float64
 6   nameDest        15999 non-null  object
 7   oldbalanceDest  15999 non-null  float64
 8   newbalanceDest  15999 non-null  float64
 9   isFraud         15999 non-null  int64
 10  isFlaggedFraud  15999 non-null  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 1.3+ MB
```

In [5]:
```
data.describe()
```

Out[5]:

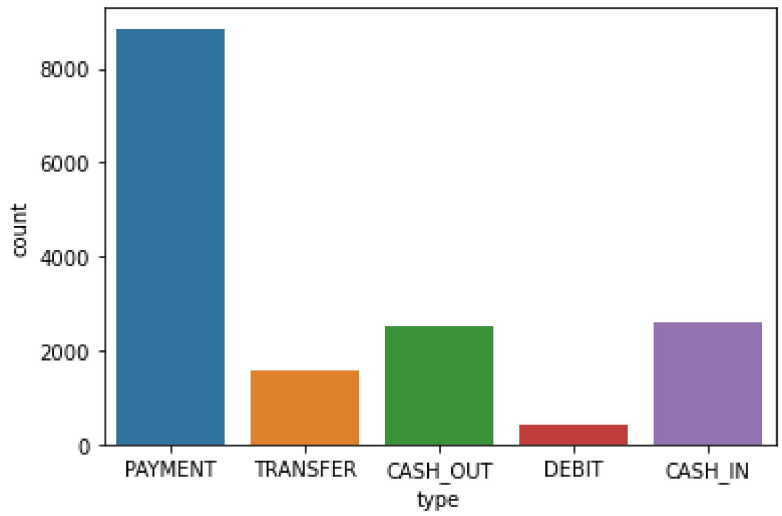
	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceI
count	15999.000000	1.599900e+04	1.599900e+04	1.599900e+04	1.599900e+04	1.599900e
mean	5.361585	1.144297e+05	7.172959e+05	7.322475e+05	8.054745e+05	1.131963e
std	2.501721	2.817263e+05	1.917880e+06	1.959206e+06	2.485147e+06	3.128831e
min	1.000000	2.390000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e
25%	3.000000	4.738340e+03	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e
50%	7.000000	1.328598e+04	1.991800e+04	6.808590e+03	0.000000e+00	0.000000e
75%	7.000000	1.253290e+05	1.178770e+05	1.119765e+05	2.398809e+05	2.816413e
max	8.000000	1.000000e+07	1.290000e+07	1.300000e+07	2.130000e+07	2.530000e

In [6]:
```
obj = (data.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:", len(object_cols))

int_ = (data.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:", len(num_cols))

fl = (data.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:", len(fl_cols))
```

```
Categorical variables: 3
Integer variables: 0
Float variables: 5
```

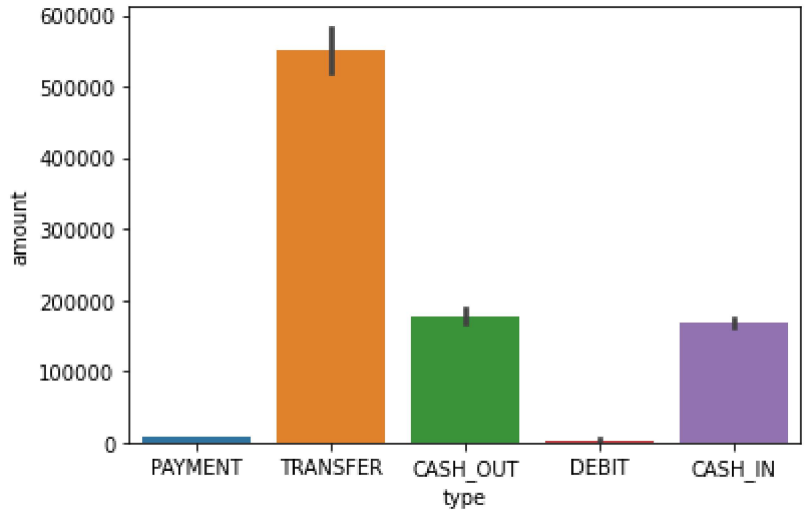
In [7]:
```
sns.countplot(x='type', data=data)
```

Out[7]:
```
<AxesSubplot:xlabel='type', ylabel='count'>
```



In [8]:
```
sns.barplot(x='type', y='amount', data=data)
```

Out[8]:
```
<AxesSubplot:xlabel='type', ylabel='amount'>
```



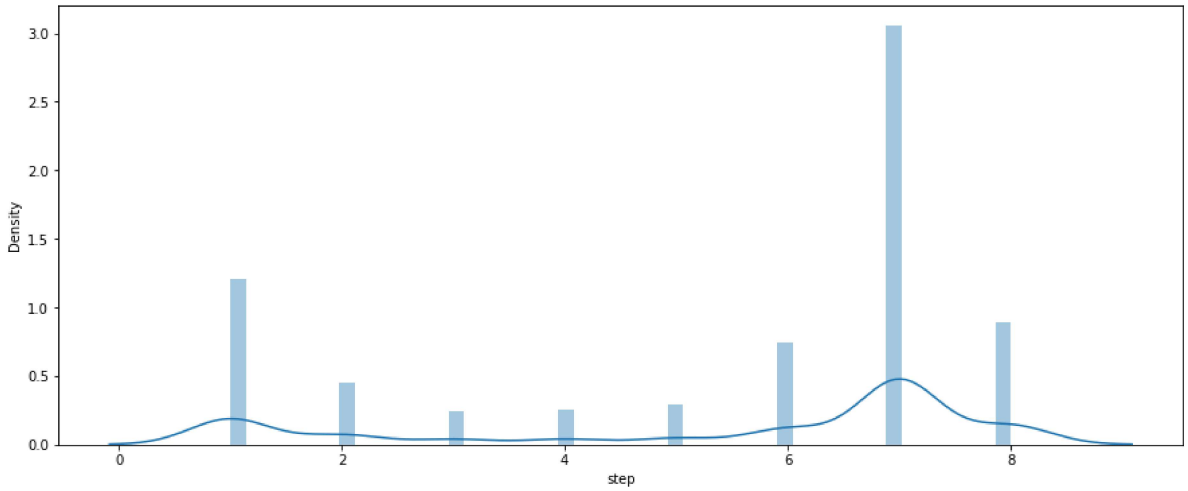
In [9]:
```
data['isFraud'].value_counts()
```

Out[9]:
```
0    15920
1       79
Name: isFraud, dtype: int64
```

In [10]:
```
plt.figure(figsize=(15, 6))
sns.distplot(data['step'], bins=50)
```

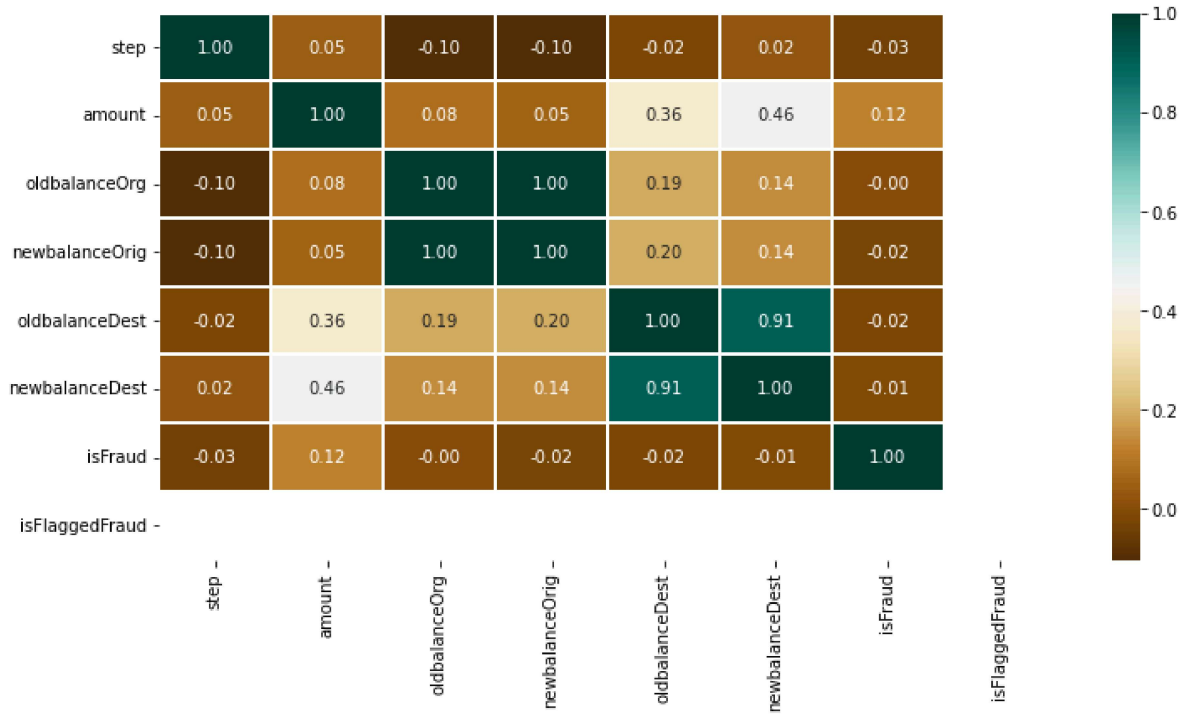
```
C:\Users\pavan\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:
```
<AxesSubplot:xlabel='step', ylabel='Density'>
```



In [11]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(data.corr(),
            cmap='BrBG',
            fmt='.2f',
            linewidths=2,
            annot=True)
```

Out[11]: <AxesSubplot:>



In [12]:
```
type_new = pd.get_dummies(data['type'], drop_first=True)
data_new = pd.concat([data, type_new], axis=1)
data_new.head()
```

Out[12]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldb
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	

In [13]:
```
X = data_new.drop(['isFraud', 'type', 'nameOrig', 'nameDest'], axis=1)
y = data_new['isFraud']
```

In [14]:
```
X.shape, y.shape
```

Out[14]: ((15999, 11), (15999,))

In [15]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
```

In [16]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\pavan\anaconda3\lib\site-p
ackages (2.0.3)
Requirement already satisfied: scipy in c:\users\pavan\anaconda3\lib\site-pac
kages (from xgboost) (1.7.1)
Requirement already satisfied: numpy in c:\users\pavan\anaconda3\lib\site-pac
kages (from xgboost) (1.20.3)
Note: you may need to restart the kernel to use updated packages.
```

In [17]:
```
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score as ras
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

In [18]:
```
models = [LogisticRegression(), XGBClassifier(),
        SVC(kernel='rbf', probability=True),
        RandomForestClassifier(n_estimators=7,
                               criterion='entropy',
                               random_state=7)]

for i in range(len(models)):
    models[i].fit(X_train, y_train)
    print(f'{models[i]} : ')

    train_preds = models[i].predict_proba(X_train)[:, 1]
    print('Training Accuracy : ', ras(y_train, train_preds))

    y_preds = models[i].predict_proba(X_test)[:, 1]
    print('Validation Accuracy : ', ras(y_test, y_preds))
    print()
```

```
LogisticRegression() :
Training Accuracy :  0.9235271994534576
Validation Accuracy :  0.9183679803834842

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
Training Accuracy :  1.0
Validation Accuracy :  0.9992162705340999

SVC(probability=True) :
Training Accuracy :  0.8931257806515723
Validation Accuracy :  0.8807567257179659

RandomForestClassifier(criterion='entropy', n_estimators=7, random_state=7) :
Training Accuracy :  0.9999879236220852
Validation Accuracy :  0.9435094008737419
```

In [20]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(models[1], X_test, y_test)
plt.show()
```