In [8]:
```
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('new_data.csv')
data.head()
```

Out[2]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanc
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	2
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6362620 entries, 0 to 6362619
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 534.0+ MB
```

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

Out[4]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
count	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06	6.362620e+06
mean	2.433972e+02	1.798619e+05	8.338831e+05	8.551137e+05	1.100702e+06	1.224996e+06
std	1.423320e+02	6.038582e+05	2.888243e+06	2.924049e+06	3.399180e+06	3.674129e+06
min	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.560000e+02	1.338957e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00

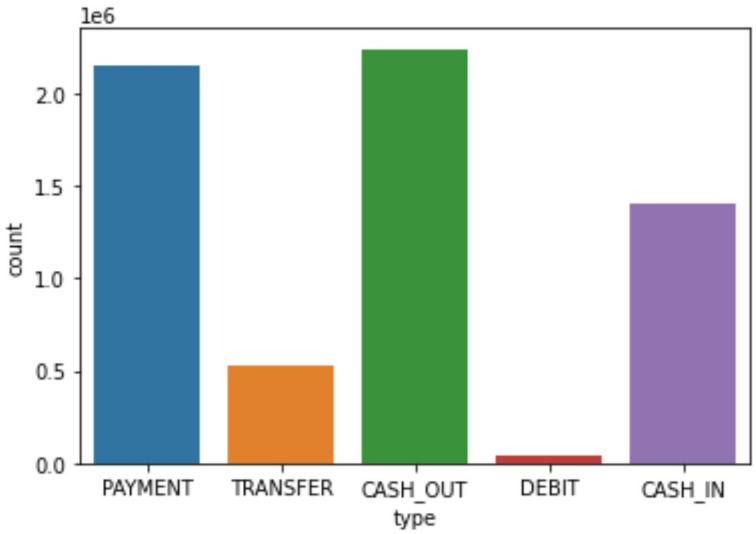
	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
50%	2.390000e+02	7.487194e+04	1.420800e+04	0.000000e+00	1.327057e+05	2.146614e+05
75%	3.350000e+02	2.087215e+05	1.073152e+05	1.442584e+05	9.430367e+05	1.111909e+06
max	7.430000e+02	9.244552e+07	5.958504e+07	4.958504e+07	3.560159e+08	3.561793e+08

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
```

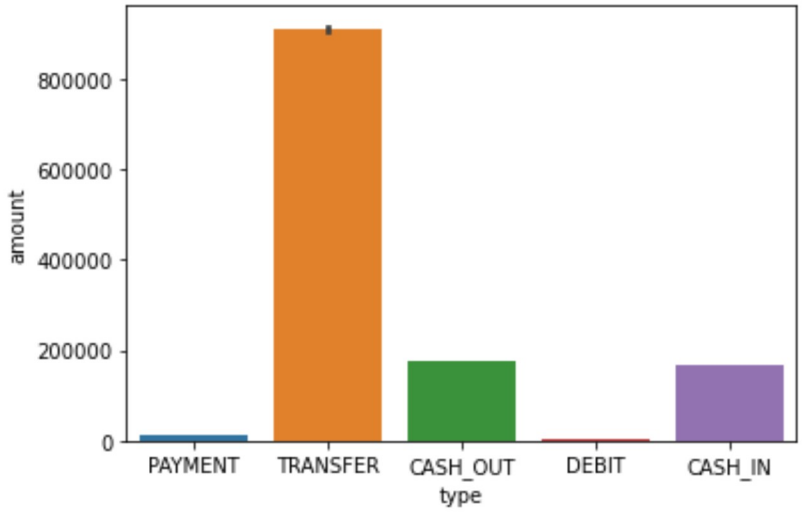
In [9]:
```
sns.countplot(x = 'type', data=data)
```

Out[9]: <AxesSubplot:xlabel='type', ylabel='count'>



In [10]:
```
sns.barplot(x = 'type',y='amount',data=data)
```

Out[10]: <AxesSubplot:xlabel='type', ylabel='amount'>

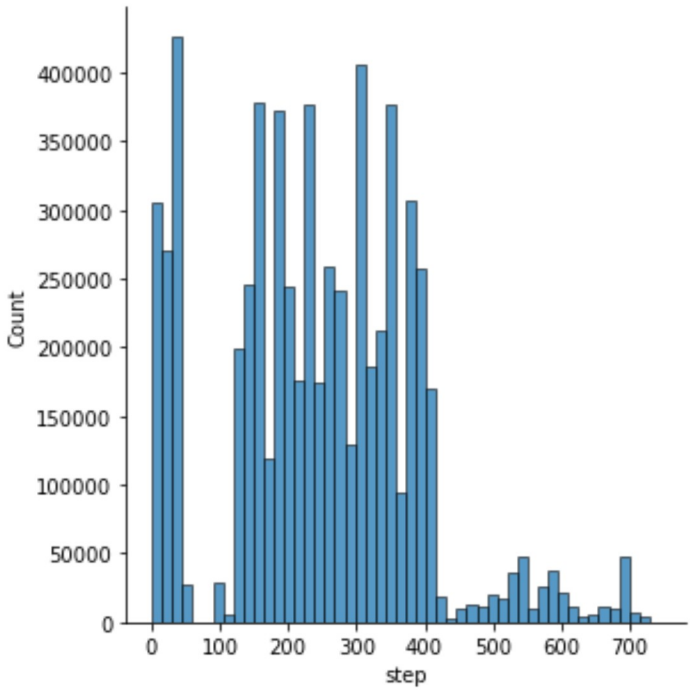


In [11]:
```
data['isFraud'].value_counts()
```

Out[11]:
```
0    6354407
1       8213
Name: isFraud, dtype: int64
```

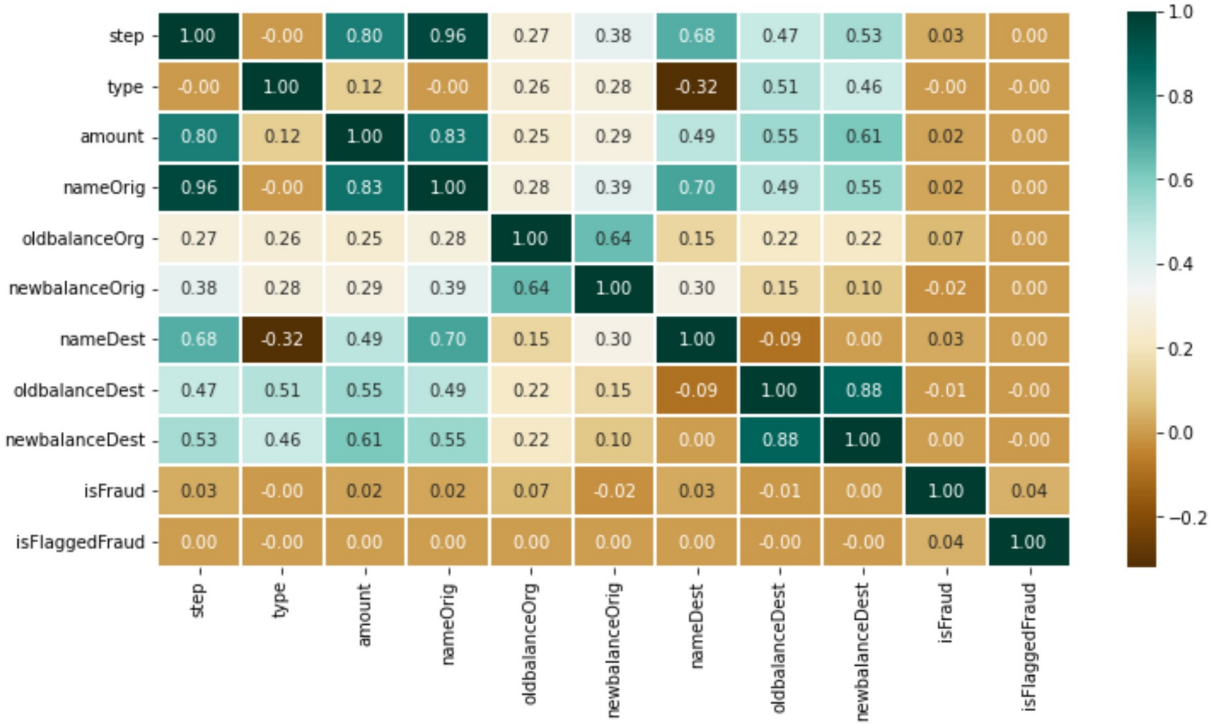
In [14]:
```
plt.figure(figsize=(15,6))
sns.displot(data['step'], bins = 50)
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x23ffbd47af0>

<Figure size 1080x432 with 0 Axes>
```



In [16]:
```
plt.figure(figsize=(12,6))
sns.heatmap(data.apply(lambda x: pd.factorize(x)[0]).corr(),
            cmap = 'BrBG',fmt='.2f',linewidths = 2,annot = True)
```

Out[16]: <AxesSubplot:>



In [18]:
```
type_new = pd.get_dummies(data['type'],drop_first=True)
data_new = pd.concat([data,type_new],axis=1)
data_new.head()
```

Out[18]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanc
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	2
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	

In [19]:
```
x = data_new.drop(['isFraud','type','nameOrig','nameDest'],axis=1)
y = data_new['isFraud']
```

In [20]:
```
x.shape,y.shape
```

Out[20]: ((6362620, 11), (6362620,))

In [22]:
```
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 [23]:
```
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 [25]:
```
models = [LogisticRegression(),XGBClassifier(),
        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()
```

```
C:\Users\ak500\AppData\Roaming\Python\Python39\site-packages\sklearn\linear_model\_lo
gistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression() :
Training Accuracy : 0.8852989436757212
Validation Accuracy :  0.8846597918484596

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 : 0.9999774189140321
Validation Accuracy :  0.999212631773824

RandomForestClassifier(criterion='entropy', n_estimators=7, random_state=7) :
Training Accuracy : 0.9999992846155892
Validation Accuracy :  0.9635718404867615
```

In [26]:

```
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = ConfusionMatrixDisplay.from_estimator(models[1],x_test,y_test)
cm.plot(cmap = 'Blues')
plt.show()
```