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('fraud_dataset_example.csv')
data.head()
```

Out[2]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101613 entries, 0 to 101612
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   step            101613 non-null  int64
 1   type            101613 non-null  object
 2   amount          101613 non-null  float64
 3   nameOrig        101613 non-null  object
 4   oldbalanceOrg   101613 non-null  float64
 5   newbalanceOrig  101613 non-null  float64
 6   nameDest        101613 non-null  object
 7   oldbalanceDest  101613 non-null  float64
 8   newbalanceDest  101613 non-null  float64
 9   isFraud         101613 non-null  int64
 10  isFlaggedFraud  101613 non-null  int64
dtypes: float64(5), int64(3), object(3)
memory usage: 8.5+ MB
```

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

Out[4]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
count	101613.000000	1.016130e+05	1.016130e+05	1.016130e+05	1.016130e+05	1.016130e+05	101613.000000	101613.0
mean	8.523457	1.740901e+05	9.071753e+05	9.234992e+05	8.810428e+05	1.183998e+06	0.001142	0.0
std	1.820681	3.450199e+05	2.829575e+06	2.867319e+06	2.399949e+06	2.797761e+06	0.033768	0.0
min	1.000000	3.200000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.0
25%	8.000000	1.001659e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.0
50%	9.000000	5.338541e+04	2.019047e+04	0.000000e+00	2.105800e+04	5.178343e+04	0.000000	0.0
75%	10.000000	2.124984e+05	1.947150e+05	2.192178e+05	5.919217e+05	1.063122e+06	0.000000	0.0
max	10.000000	1.000000e+07	3.893942e+07	3.894623e+07	3.400874e+07	3.894623e+07	1.000000	0.0

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]:
```
<Axes: xlabel='type', ylabel='count'>
```

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

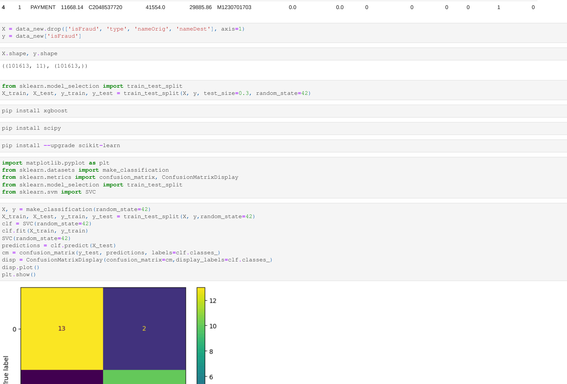
Out[8]:
```
<Axes: xlabel='type', ylabel='amount'>
```

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

Out[9]:
```
0    101497
1       116
Name: isFraud, dtype: int64
```

In [9]:
```
plt.figure(figsize=(15, 6))
sns.histplot(data['step'], bins=50)
```

Out[9]:
```
<Axes: xlabel='step', ylabel='Count'>
```



In [10]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(data.corr(),
cmap='BrBG',
fmt='.2f',
linewidths=2,annot=True)
```

```
C:\Users\NEETI PANDEY\AppData\Local\Temp\ipykernel_22704\2102605044.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(data.corr(),
```

Out[10]:
```
<Axes: >
```


In [11]:
```
type_new = pd.get_dummies(data['type'], drop_first=True)
data_new = pd.concat([data, type_new], axis=1)
data_new.head()
```

Out[11]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	CASH_OUT	DEBIT	PAYMENT	TRANSFER
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0	0	0	1	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0	0	0	1	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0	0	0	0	1
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0	1	0	0	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0	0	0	1	0

In [12]:
```
X = data_new.drop(['isFraud', 'type', 'nameOrig', 'nameDest'], axis=1)
y = data_new['isFraud']
```

In [13]:
```
X.shape, y.shape
```

Out[13]:
```
((101613, 11), (101613,))
```

In [14]:
```
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)
```

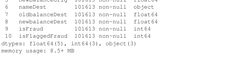
In [ ]:
```
pip install xgboost
```

In [ ]:
```
pip install scipy
```

In [ ]:
```
pip install --upgrade scikit-learn
```

In [15]:
```
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

In [16]:
```
X, y = make_classification(random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=42)
clf = SVC(random_state=42)
clf.fit(X_train, y_train)
SVC(random_state=42)
predictions = clf.predict(X_test)
cm = confusion_matrix(y_test, predictions, labels=clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=clf.classes_)
disp.plot()
plt.show()
```



In [18]:
```
data["type"] = data["type"].map({"CASH_OUT": 1, "PAYMENT": 2,
                                 "CASH_IN": 3, "TRANSFER": 4,
                                 "DEBIT": 5})
data["isFraud"] = data["isFraud"].map({0: "No Fraud", 1: "Fraud"})
print(data.head())
```

```
   step  type    amount   nameOrig  oldbalanceOrg  newbalanceOrig  \
0     1     2   9839.64  C1231006815         170136.0       160296.36
1     1     2   1864.28  C1666544295          21249.0        19384.72
2     1     4    181.00  C1305486145            181.0            0.00
3     1     1    181.00   C840083671            181.0            0.00
4     1     2  11668.14  C2048537720          41554.0        29885.86

     nameDest  oldbalanceDest  newbalanceDest   isFraud  isFlaggedFraud
0  M1979787155             0.0             0.0  No Fraud               0
1  M2044282225             0.0             0.0  No Fraud               0
2   C553264065             0.0             0.0     Fraud               0
3    C38997010         21182.0             0.0     Fraud               0
4  M1230701703             0.0             0.0  No Fraud               0
```

In [19]:
```
#splitting the data
from sklearn.model_selection import train_test_split
x = np.array(data[["type", "amount", "oldbalanceOrg", "newbalanceOrig"]])
y = np.array(data[["isFraud"]])
```

In [20]:
```
# training a machine learning model
from sklearn.tree import DecisionTreeClassifier
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.10, random_state=42)
model = DecisionTreeClassifier()
model.fit(xtrain, ytrain)
print(model.score(xtest, ytest))
```

```
0.9986213184412517
```

In [21]:
```
#prediction
#features = [type, amount, oldbalanceOrg, newbalanceOrig]
features = np.array([[4, 9000.60, 9000.60, 0.0]])
print(model.predict(features))
```

```
['Fraud']
```

In [ ]: