In [1]:
```
#Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#read csv file
data = pd.read_csv('onlinefraud.csv')
data.head()
```

Out[2]:

| | step | type | amount | nameOrig | oldbalanceOrg | newbalanceOrig | nameDest | oldbal |
|---|---|---|---|---|---|---|---|---|
| **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 [3]:
```
data.shape
```

Out[3]:
```
(525391, 11)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 525391 entries, 0 to 525390
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   step            525391 non-null  int64  
 1   type            525391 non-null  object 
 2   amount          525391 non-null  float64
 3   nameOrig        525391 non-null  object 
 4   oldbalanceOrg   525391 non-null  float64
 5   newbalanceOrig  525391 non-null  float64
 6   nameDest        525391 non-null  object 
 7   oldbalanceDest  525391 non-null  float64
 8   newbalanceDest  525391 non-null  float64
 9   isFraud         525391 non-null  int64  
 10  isFlaggedFraud  525391 non-null  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 44.1+ MB
```

In [5]:
```
data.isna().sum()
```

Out[5]:
```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

In [6]:
```
data.describe()
```

Out[6]:

| | step | amount | oldbalanceOrg | newbalanceOrig | oldbalanceDest | newbalanceD |
|---|---|---|---|---|---|---|
| **count** | 525391.000000 | 5.253910e+05 | 5.253910e+05 | 5.253910e+05 | 5.253910e+05 | 5.253910e |
| **mean** | 14.229581 | 1.645037e+05 | 9.038810e+05 | 9.235996e+05 | 9.817080e+05 | 1.156670e |
| **std** | 3.851198 | 2.696783e+05 | 3.002894e+06 | 3.039871e+06 | 2.331355e+06 | 2.499715e |
| **min** | 1.000000 | 1.000000e-01 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e |
| **25%** | 11.000000 | 1.315852e+04 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e |
| **50%** | 15.000000 | 8.034980e+04 | 1.840700e+04 | 0.000000e+00 | 1.211080e+05 | 2.213392e |
| **75%** | 18.000000 | 2.198746e+05 | 1.663035e+05 | 2.059076e+05 | 8.981122e+05 | 1.195243e |
| **max** | 20.000000 | 1.000000e+07 | 3.890000e+07 | 3.890000e+07 | 4.150000e+07 | 4.150000e |

In [7]:
```
#Understanding the transaction type
print(data.type.value_counts())
```

```
CASH_OUT    191642
PAYMENT     172514
CASH_IN     114831
TRANSFER     42603
DEBIT         3801
Name: type, dtype: int64
```

In [8]:
```
#Count the occurrences of fraud and no fraud and print them
occ = data['isFraud'].value_counts()
occ
```

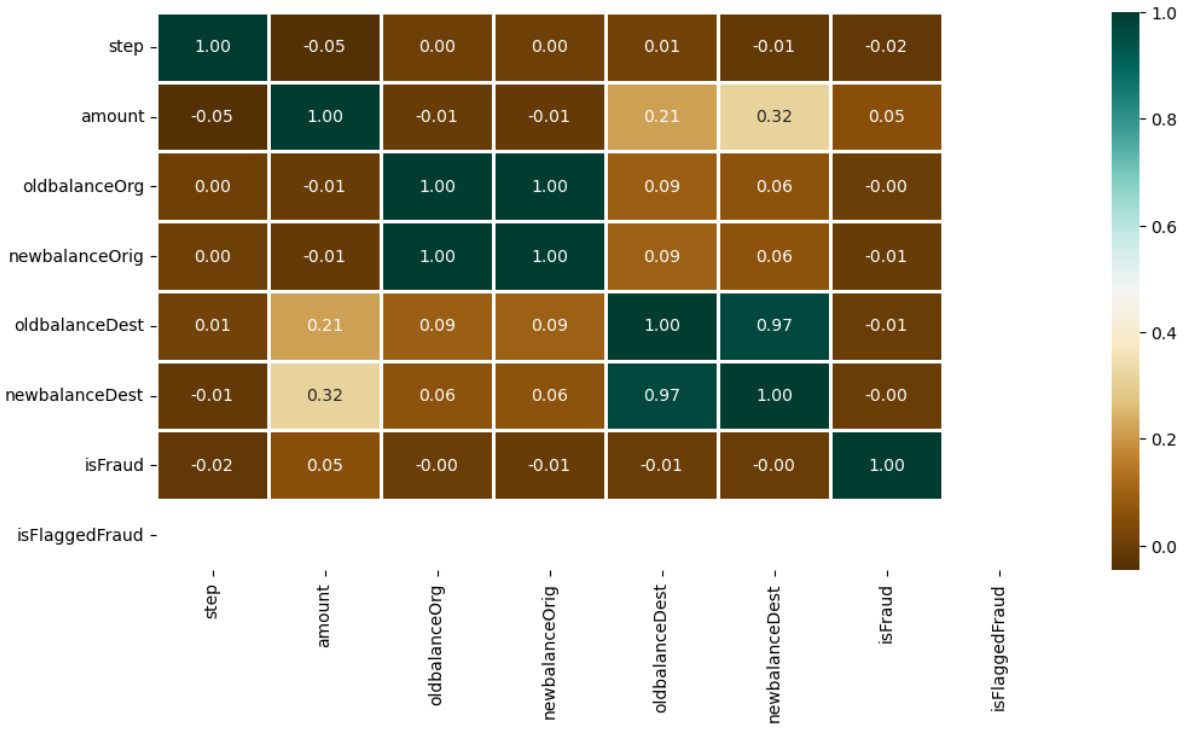
Out[8]:
```
0    525158
1       233
Name: isFraud, dtype: int64
```

In [9]:
```
#Print the ratio of fraud cases
ratio_cases = occ/len(data.index)
print(f'Ratio of fraudulent cases: {ratio_cases[1]}\nRatio of non-fraudulent cases:
```

```
Ratio of fraudulent cases: 0.00044347923736797926
Ratio of non-fraudulent cases: 0.999556520762632
```

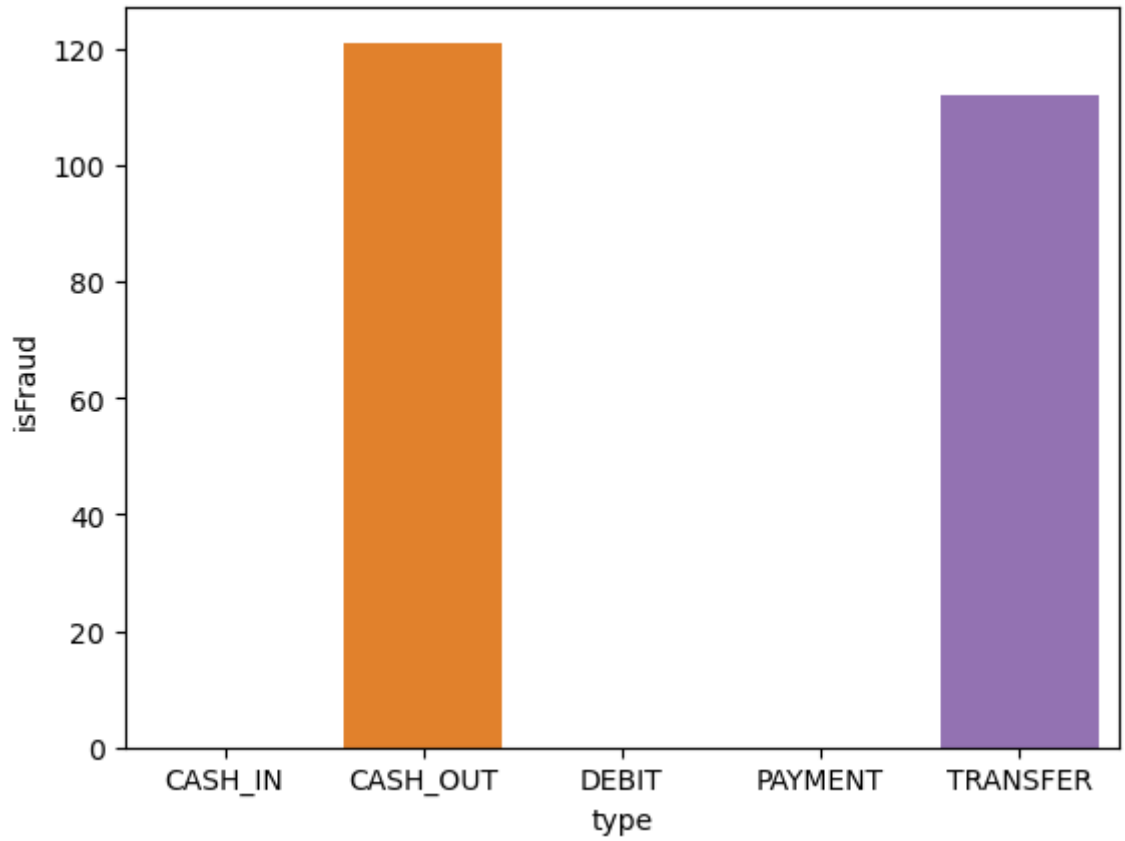
In [10]:
```
plt.figure(figsize=(12, 6))
sns.heatmap(data.corr(),
            cmap='BrBG',
            fmt='.2f',
            linewidths=2,
            annot=True)
```

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



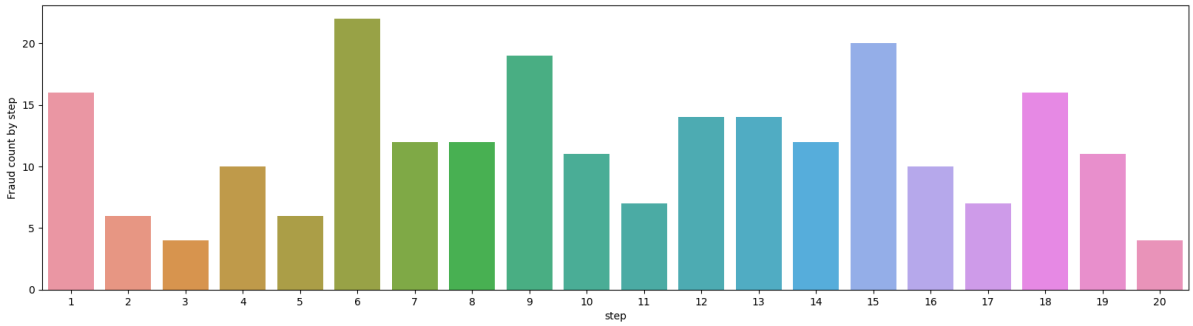

In [11]:
```
t=pd.DataFrame(data.groupby(by=data['type'])['isFraud'].sum())
sns.barplot(data=t,x=t.index, y='isFraud')
```

Out[11]:
```
<AxesSubplot:xlabel='type', ylabel='isFraud'>
```



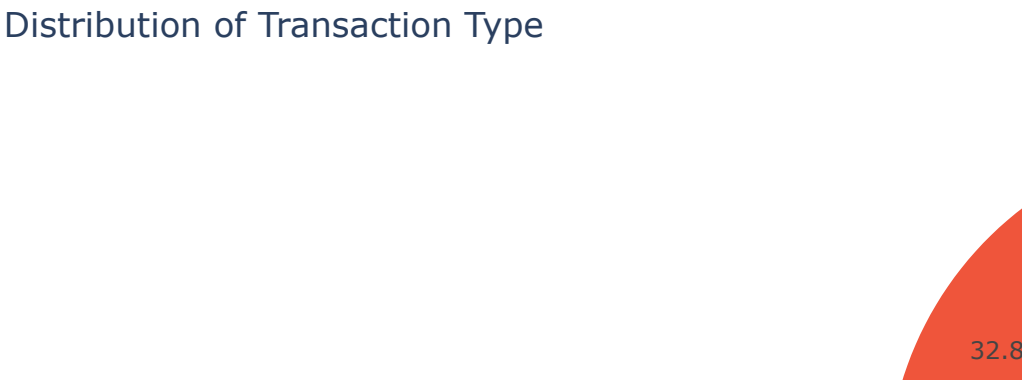

In [12]:
```
t=pd.DataFrame(data.groupby(by=data['step'])['isFraud'].sum())
t2=t.sort_values(by='isFraud', ascending=False).head(20)
t2 = t2.rename_axis('step').reset_index()
plt.figure(figsize=(20,5))
sns.barplot(data=t2,x='step', y='isFraud',)
plt.ylabel('Fraud count by step')
```

Out[12]: Text(0, 0.5, 'Fraud count by step')




In [13]:
```
type = data["type"].value_counts()
transactions = type.index
quantity = type.values

import plotly.express as px
figure = px.pie(data,
             values=quantity,
             names=transactions,hole = 0.5,
             title="Distribution of Transaction Type")
figure.show()
```




In [14]:
```
# Checking correlation
correlation = data.corr()
print(correlation["isFraud"].sort_values(ascending=False))
```

```
isFraud           1.000000
amount            0.052494
oldbalanceOrg    -0.000483
newbalanceDest   -0.001764
oldbalanceDest   -0.005179
newbalanceOrig   -0.006235
step             -0.020222
isFlaggedFraud         NaN
Name: isFraud, dtype: float64
```

In [15]:
```
#transform the categorical into numeric
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 [16]:
```
# 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 [17]:
```
# training a machine learning model
from sklearn.tree import DecisionTreeClassifier
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.40, random_state=
model = DecisionTreeClassifier()
model.fit(xtrain, ytrain)
print(model.score(xtest, ytest))
```

```
0.9993338313736873
```

In [18]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(xtrain,ytrain)
lr.score(xtrain,ytrain)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataCon
versionWarning:

A column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples, ), for example using ravel().
```

Out[18]:
```
0.999866765640762
```

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(xtrain,ytrain)
model.score(xtrain,ytrain)
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_14260\2079082207.py:3: DataConversionWa
rning:

A column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
```

Out[19]:
```
1.0
```

In [20]:
```
# prediction
#features = [type, amount, oldbalanceOrg, newbalanceOrig]
features = np.array([[4, 5000.60, 3000.60, 0.0]])
print(model.predict(features))
```

```
['No Fraud']
```

In [21]:
```
#prediction
features=np.array([[1,181.00,181.00,0.0]])
print(model.predict(features))
```

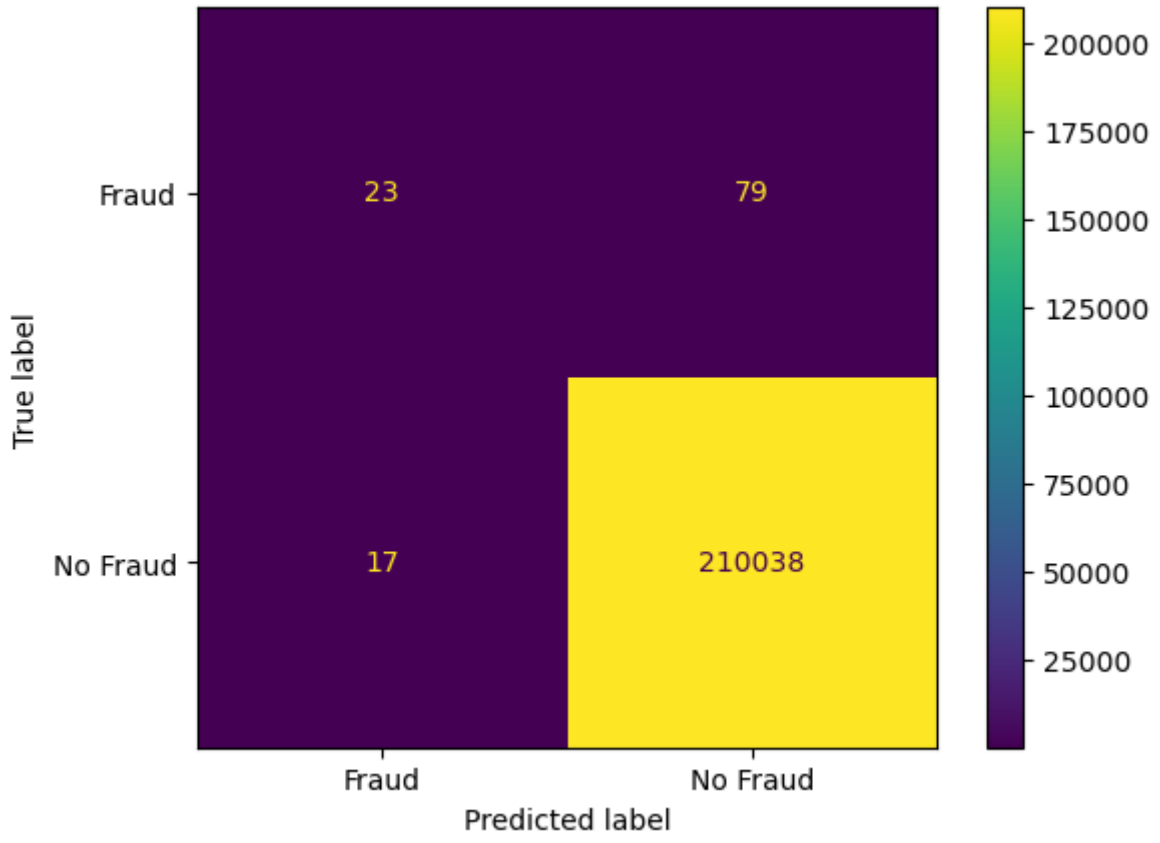
```
['Fraud']
```

In [22]:
```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(model, xtest, ytest)
plt.show()
```

```
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureW
arning:

Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is
deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: Confus
ionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
```




In [23]:
```
#import classification_report
from sklearn.metrics import classification_report
```

In [24]:
```
classifier = DecisionTreeClassifier()
classifier.fit(xtrain, ytrain)
ypred = classifier.predict(xtest)
report = classification_report(ytest, ypred)
print(report)
```

```
              precision    recall  f1-score   support

       Fraud       0.27      0.25      0.26       102
    No Fraud       1.00      1.00      1.00    210055

    accuracy                           1.00    210157
   macro avg       0.63      0.63      0.63    210157
weighted avg       1.00      1.00      1.00    210157
```

In [25]:
```
classifier = RandomForestClassifier()
classifier.fit(xtrain, ytrain)
ypred = classifier.predict(xtest)
report = classification_report(ytest, ypred)
print(report)
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_14260\803100067.py:2: DataConversionWar
ning:

A column-vector y was passed when a 1d array was expected. Please change the shape
of y to (n_samples,), for example using ravel().
```

```
              precision    recall  f1-score   support

       Fraud       0.59      0.24      0.34       102
    No Fraud       1.00      1.00      1.00    210055

    accuracy                           1.00    210157
   macro avg       0.79      0.62      0.67    210157
weighted avg       1.00      1.00      1.00    210157
```