Capstone Project - Online Payment Fraud Detection

Question

- Blossom Bank also known as BB PLC is a multinational financial services group, that offers retail and investment banking, pension management, asset management and payments services, headquartered in London, UK.

Problem

- Blossom Bank wants to build a Machine Learning model to predict online payment fraud.

Data Dictionary

- The below column reference:
  - step: represents a unit of time where 1 step equals 1 hour
  - type: type of online transaction
  - amount: the amount of the transaction
  - nameOrig: customer starting the transaction •
  - oldbalanceOrg: balance before the transaction •
  - newbalanceOrig: balance after the transaction •
  - nameDest: recipient of the transaction
  - oldbalanceDest: initial balance of recipient before the transaction
  - newbalanceDest: the new balance of the recipient after the transaction
  - isFraud: fraud transaction

1. Problem definition: clearly articulate the problem that is to be solved with your data mining. How will the business benefit from your solution?

- The problem to be solved is the detection of online payment fraud and the business will benefit from my solution through the application of the Machine Learning model that i have trained to predict online payment fraud.

2. Perform exploratory data analysis in python.

- a) Visualize relationships between the label and some key features
- b) Explore correlations
- c) Conduct univariate and multivariate analysis as much as is feasible

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

In [2]:
```
# importing our dataset
df = pd.read_csv(r"C:\Users\THINKPAD\Desktop\Career_Skills\Programmes\Full Stack Data
df.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	21
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	

Checking out the info() of the dataset

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 10 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   step            1048575 non-null  int64
 1   type            1048575 non-null  object
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64
dtypes: float64(5), int64(2), object(3)
memory usage: 80.0+ MB
```

Checking for the statistical description of the dataset

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

Out[4]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	2.696617e+01	1.586670e+05	8.740095e+05	8.938089e+05	9.781600e+05	1.114198e+06
std	1.562325e+01	2.649409e+05	2.971751e+06	3.008271e+06	2.296780e+06	2.416593e+06
min	1.000000e+00	1.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.500000e+01	1.214907e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.000000e+01	7.634333e+04	1.600200e+04	0.000000e+00	1.263772e+05	2.182604e+05
75%	3.900000e+01	2.137619e+05	1.366420e+05	1.746000e+05	9.159235e+05	1.149808e+06
max	9.500000e+01	1.000000e+07	3.890000e+07	3.890000e+07	4.210000e+07	4.220000e+07

Checking for missing values

In [38]:
```
df.isnull().sum()
```

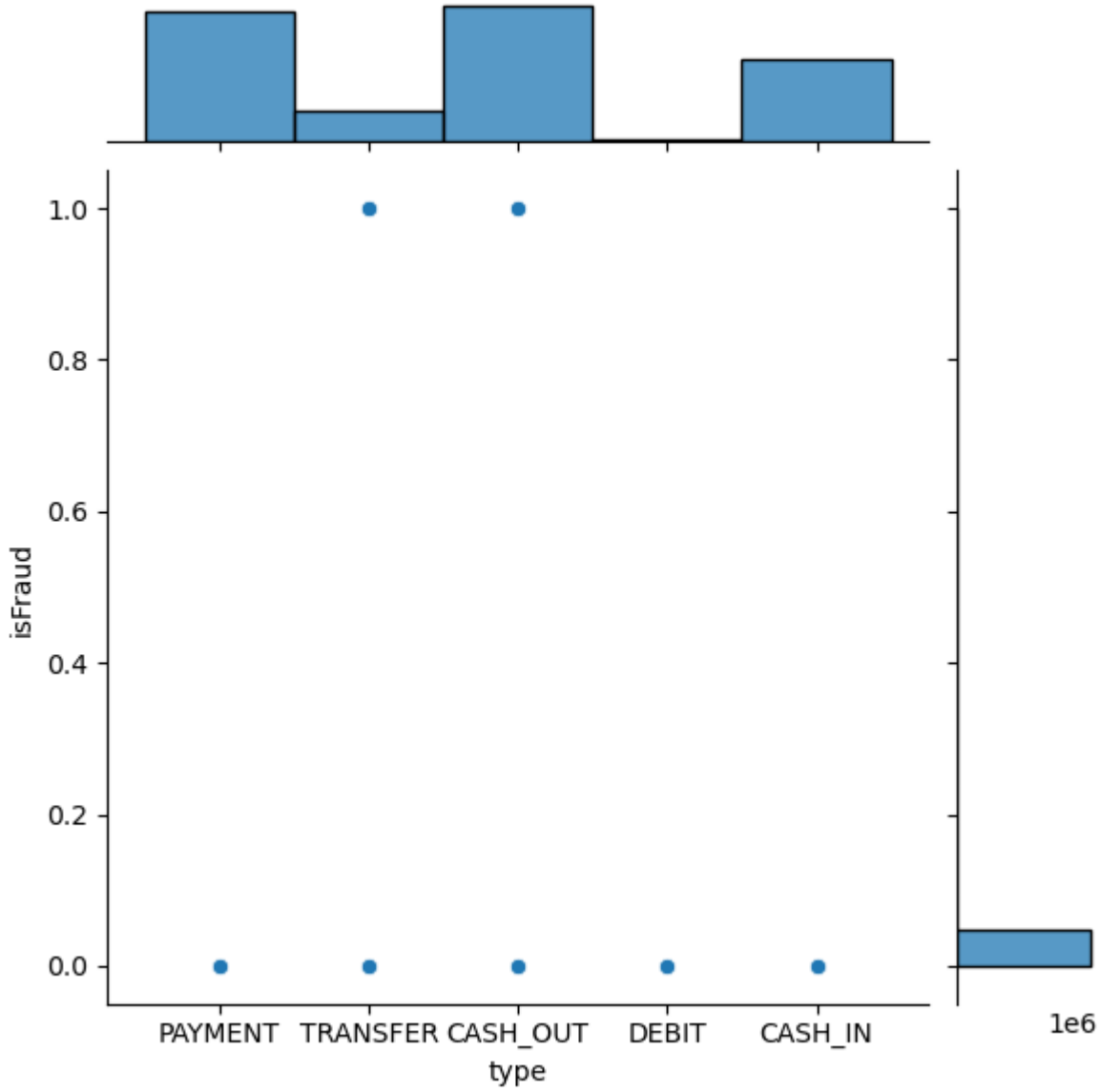
Out[38]:
```
amount            0
oldbalanceOrg     0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

Exploratory Data Analysis

- Using seaborn to create a jointplot to compare the amount of transactions and the number of fraudulent transactions columns. Does the correlation make sense?

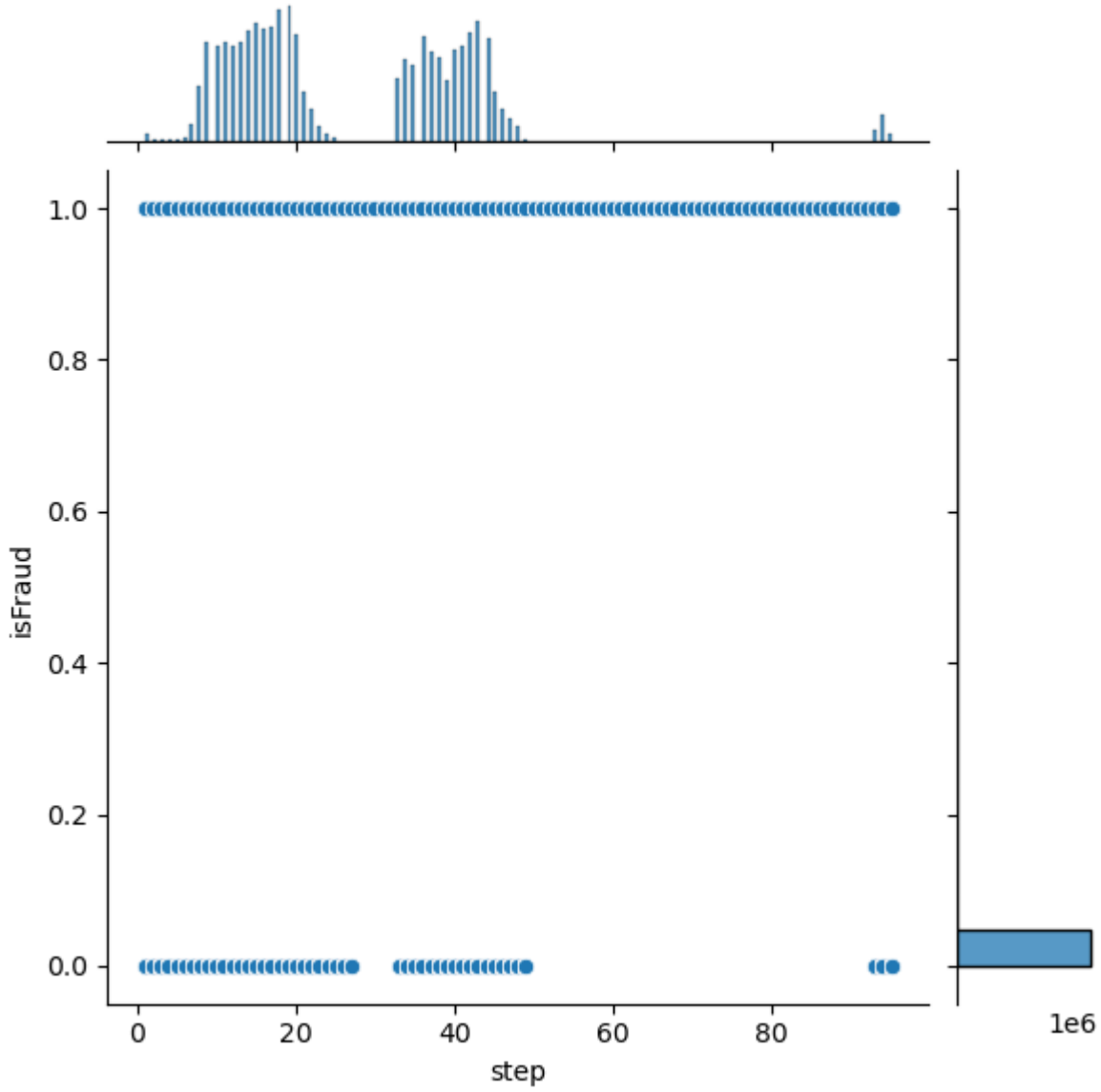
In [6]:
```
sns.jointplot(x='type', y='isFraud', data=df)
```

Out[6]:
```
<seaborn.axisgrid.JointGrid at 0x21ec03e4ac0>
```



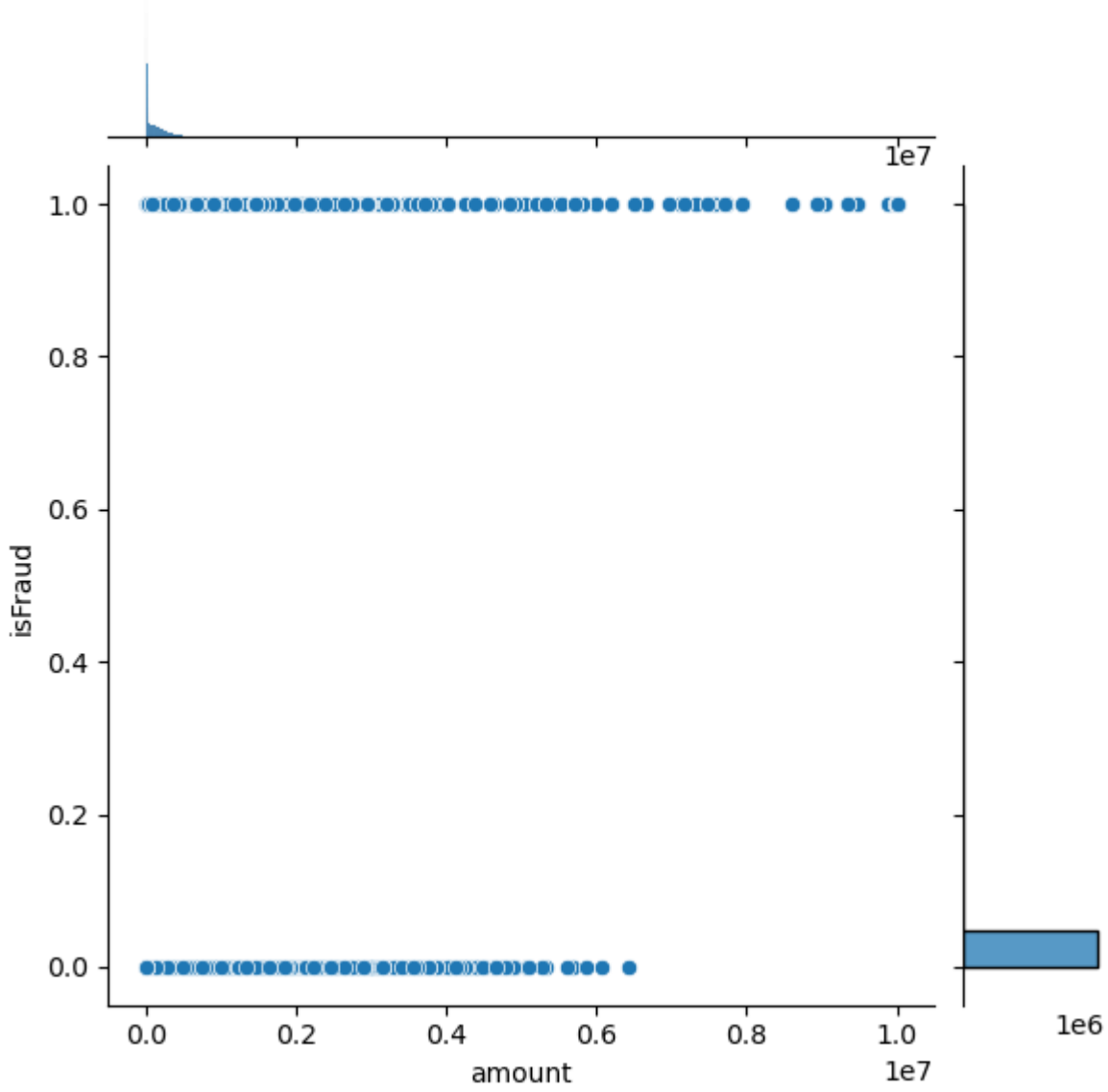
In [11]:
```
sns.jointplot(x='step', y='isFraud', data=df)
```

Out[11]:
```
<seaborn.axisgrid.JointGrid at 0x21ec2826370>
```



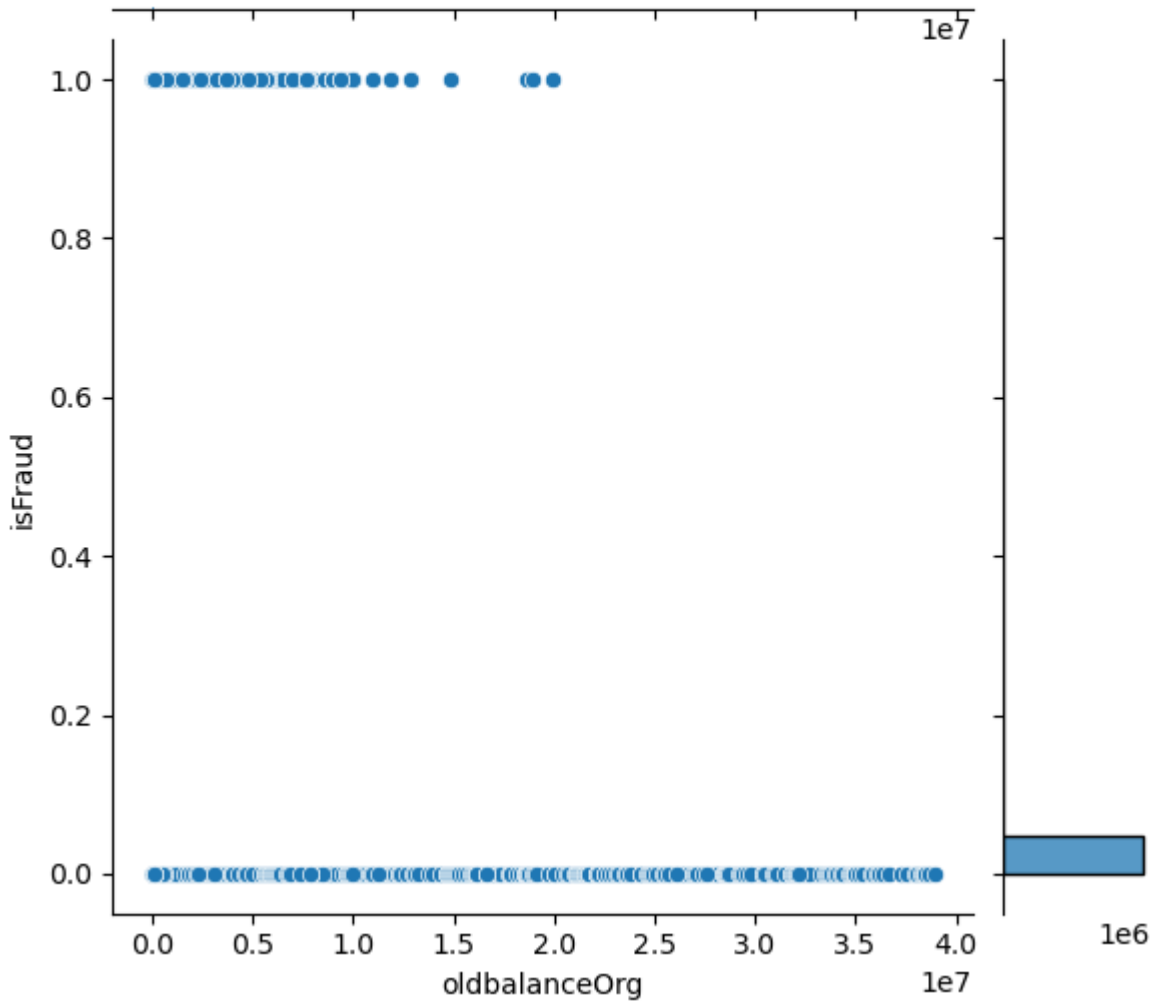
In [12]:
```
sns.jointplot(x='amount', y='isFraud', data=df)
```

Out[12]:
```
<seaborn.axisgrid.JointGrid at 0x21ec1bef370>
```



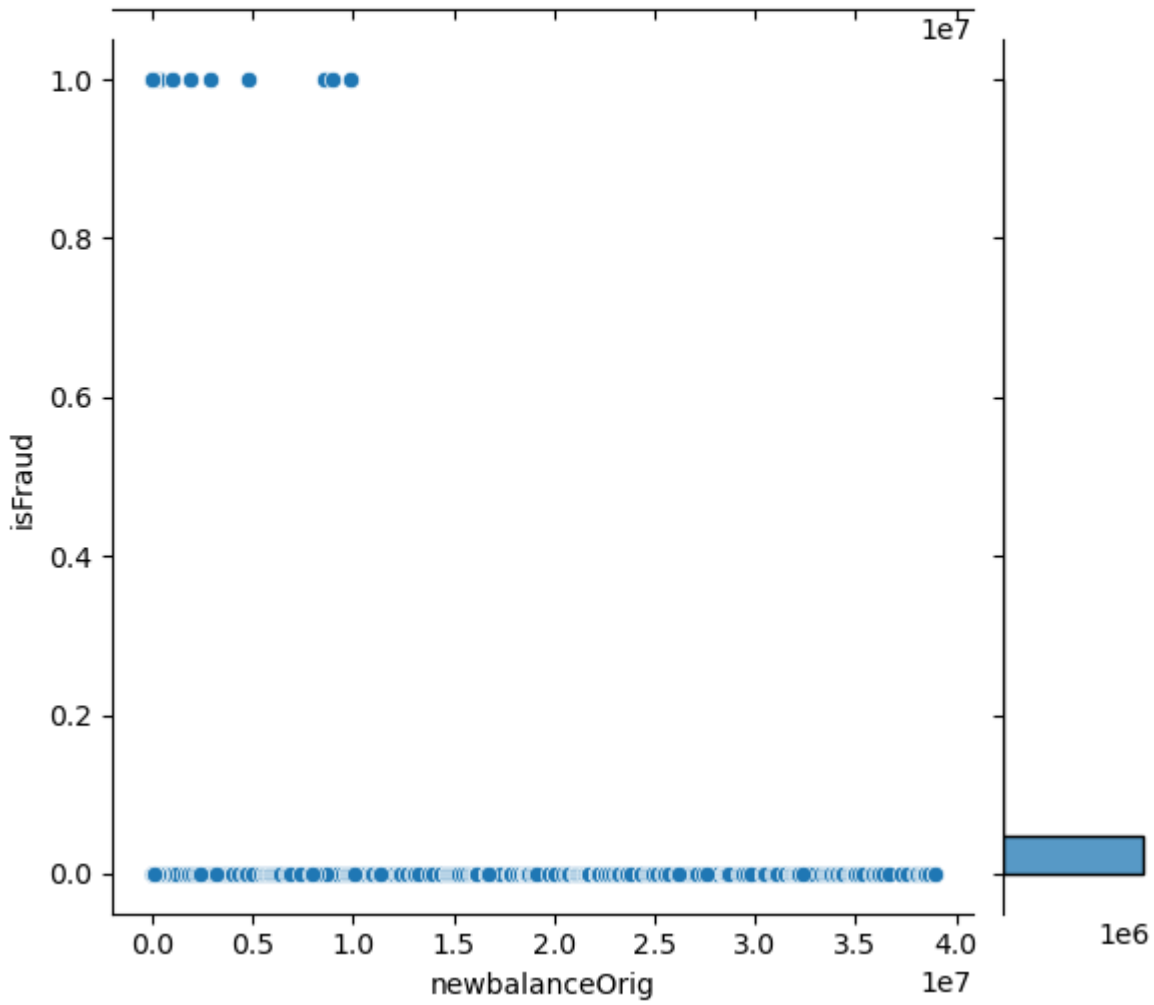
In [13]:
```
sns.jointplot(x='oldbalanceOrg', y='isFraud', data=df)
```

Out[13]:
```
<seaborn.axisgrid.JointGrid at 0x21ec297d670>
```



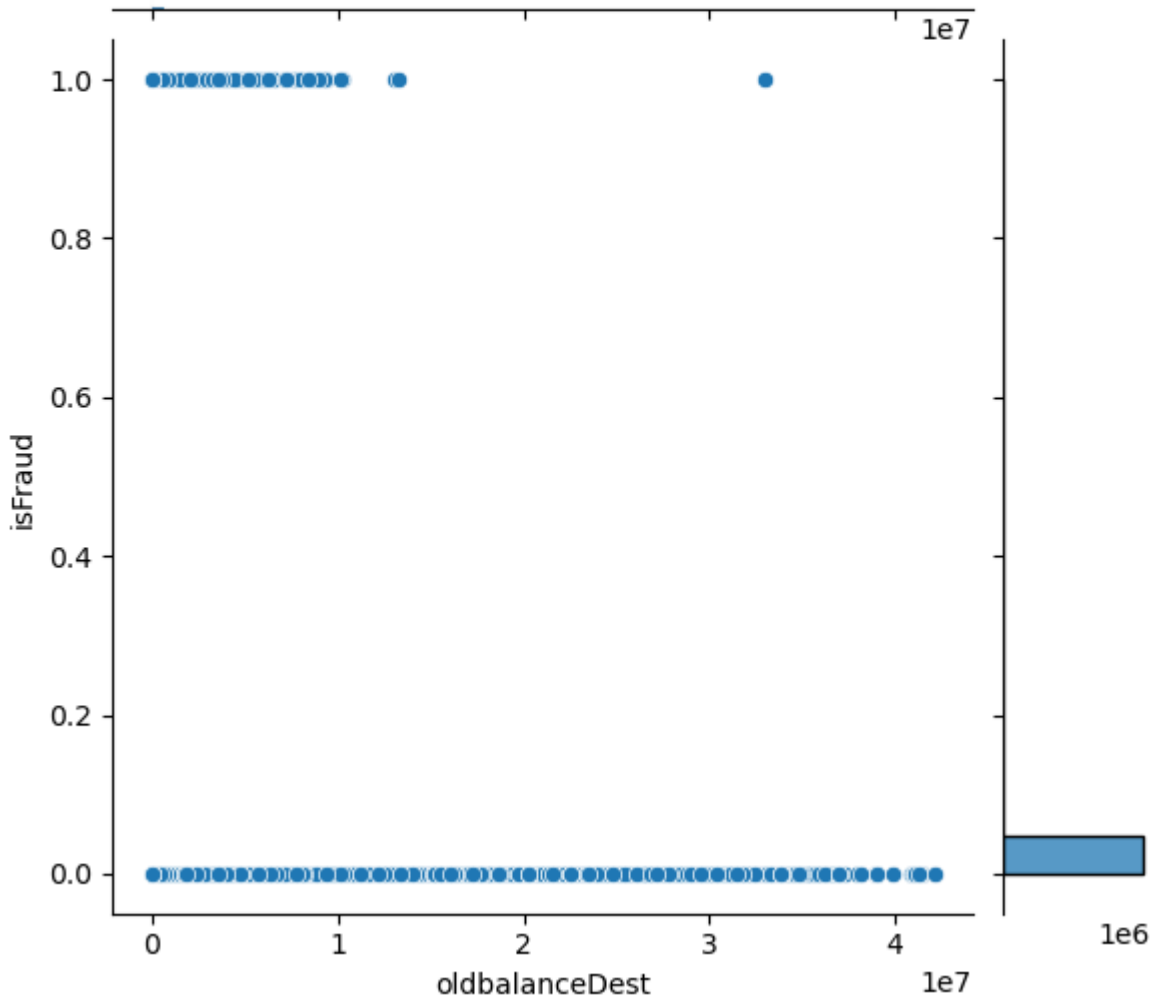
In [14]:
```
sns.jointplot(x='newbalanceOrig', y='isFraud', data=df)
```

Out[14]:
```
<seaborn.axisgrid.JointGrid at 0x21ec297d5b0>
```



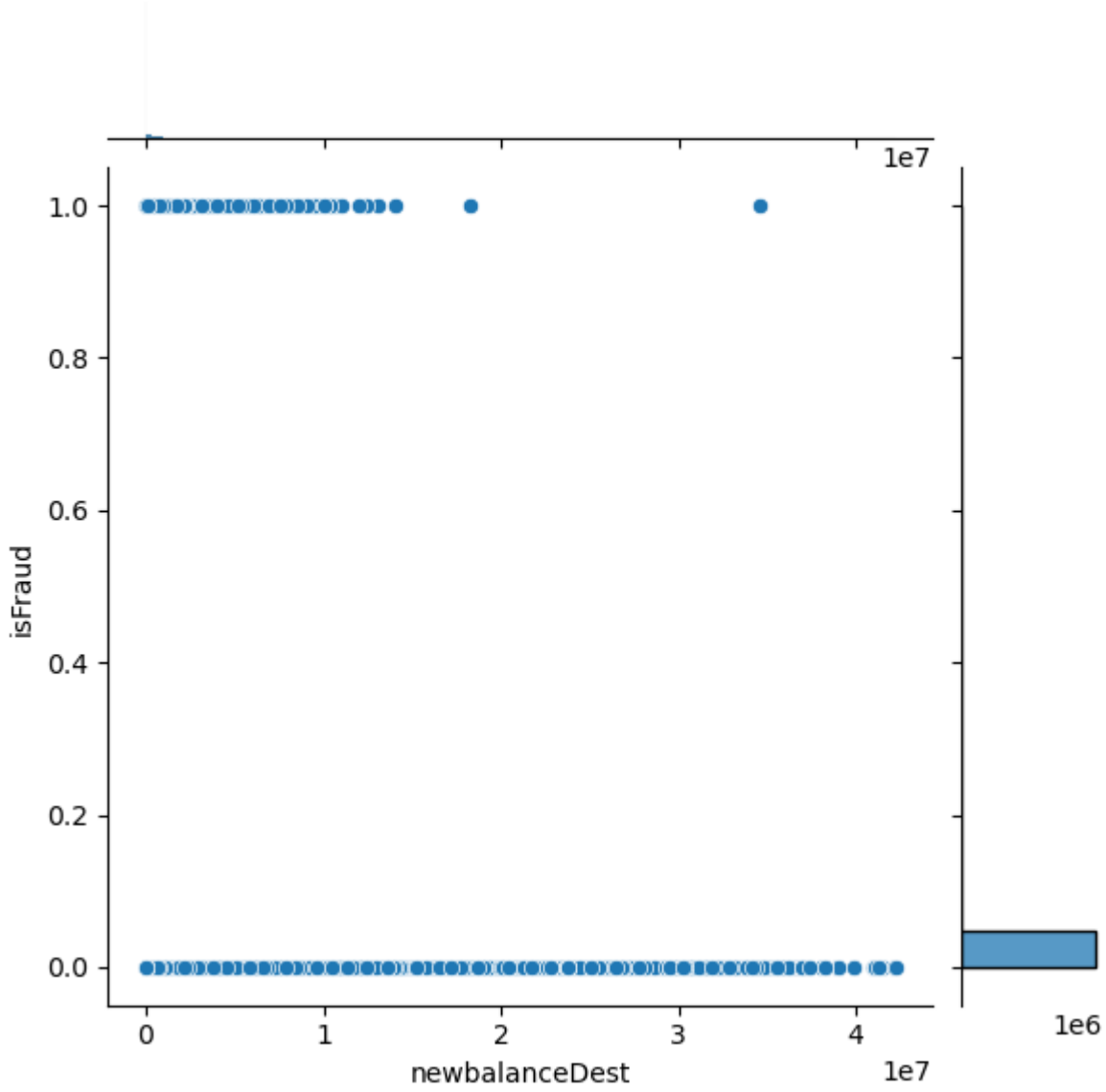
In [15]:
```
sns.jointplot(x='oldbalanceDest', y='isFraud', data=df)
```

Out[15]:
```
<seaborn.axisgrid.JointGrid at 0x21ed19d53a0>
```



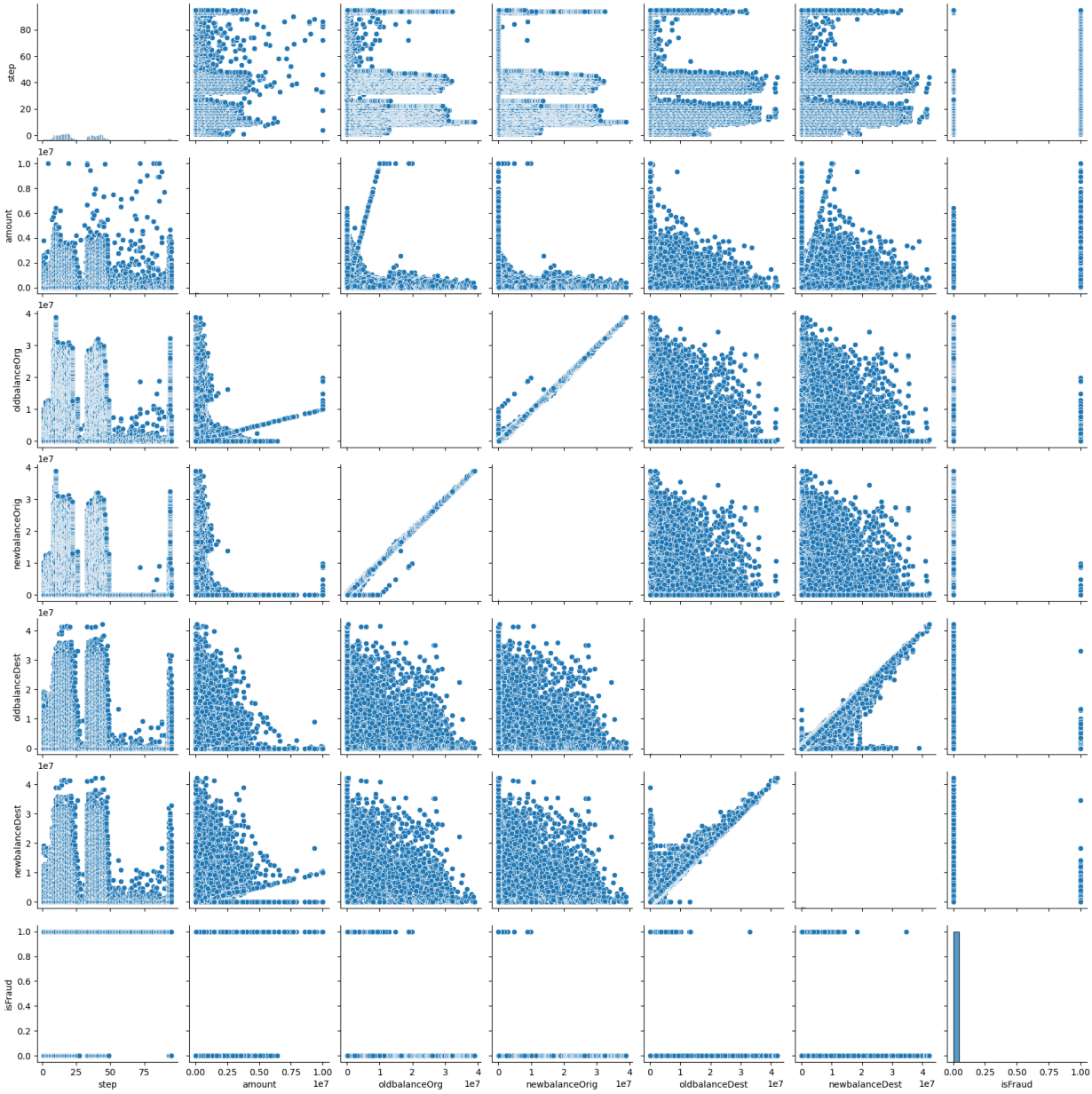
In [16]:
```
sns.jointplot(x='newbalanceDest', y='isFraud', data=df)
```

Out[16]:
```
<seaborn.axisgrid.JointGrid at 0x21ef076d1f0>
```



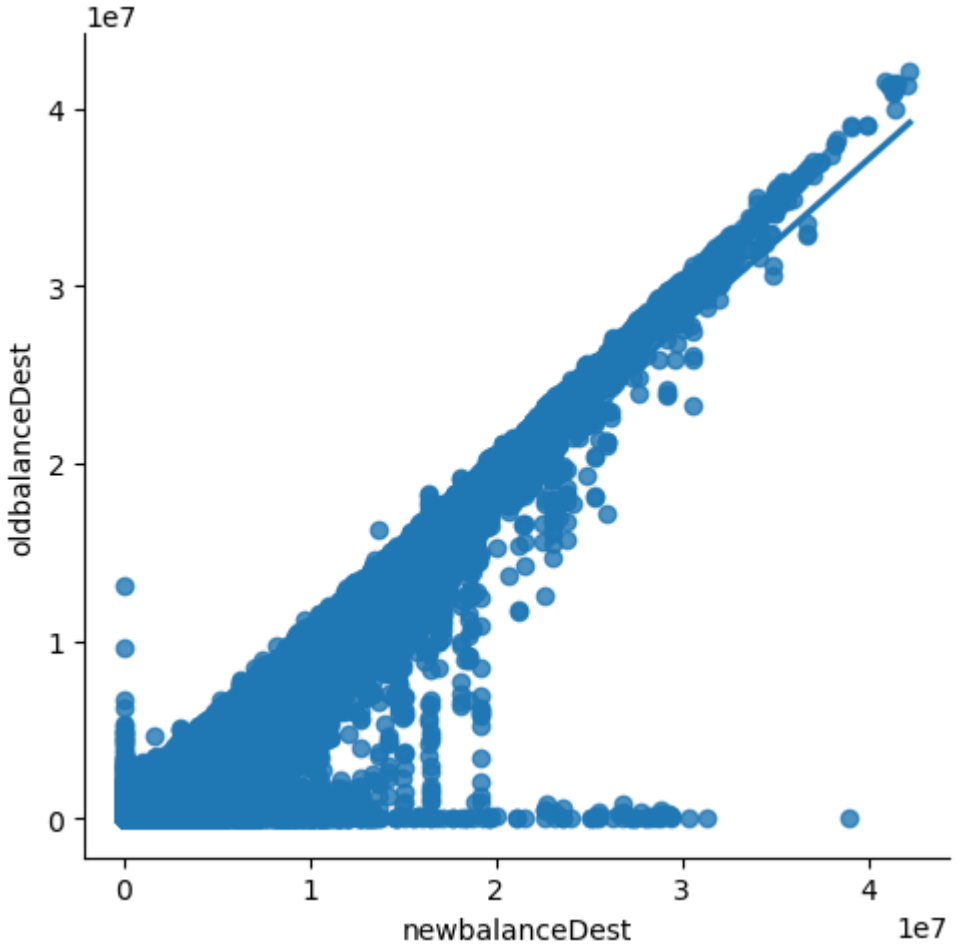
In [17]:
```
sns.pairplot(df)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x21eeed819d0>
```



In [18]:
```
sns.lmplot(x='newbalanceDest',y='oldbalanceDest',data=df)
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x21e97e50c10>
```



3. Perform feature engineering

- a) Encoding categorical variables
- b) Create new features from existing features where necessary, depending on insights from your EDA

In [29]:
```
#data segmentation and droping of irrelevant features
#target = df['isFraud']
#df = df.drop(columns=['step','type'],axis=1)
```

In [25]:
```
df.head()
```

Out[25]:

	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalance
0	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	
1	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	
2	181.00	C1305486145	181.0	0.00	C553264065	0.0	
3	181.00	C840083671	181.0	0.00	C38997010	21182.0	
4	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	

In [26]:
```
target = df['isFraud']
df = df.drop(columns=['nameOrig','nameDest'],axis=1)
```

In [27]:
```
df .head()
```

Out[27]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	9839.64	170136.0	160296.36	0.0	0.0	0
1	1864.28	21249.0	19384.72	0.0	0.0	0
2	181.00	181.0	0.00	0.0	0.0	1
3	181.00	181.0	0.00	21182.0	0.0	1
4	11668.14	41554.0	29885.86	0.0	0.0	0

In [28]:
```
df = pd.get_dummies(df)
df
```

Out[28]:

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	9839.64	170136.00	160296.36	0.00	0.00	0
1	1864.28	21249.00	19384.72	0.00	0.00	0
2	181.00	181.00	0.00	0.00	0.00	1
3	181.00	181.00	0.00	21182.00	0.00	1
4	11668.14	41554.00	29885.86	0.00	0.00	0
...	...	...	...	...	...	...
1048570	132557.35	479803.00	347245.65	484329.37	616886.72	0
1048571	9917.36	90545.00	80627.64	0.00	0.00	0
1048572	14140.05	20545.00	6404.95	0.00	0.00	0
1048573	10020.05	90605.00	80584.95	0.00	0.00	0
1048574	11450.03	80584.95	69134.92	0.00	0.00	0

1048575 rows × 6 columns

4. Model selection, training, and validation

- a) Train and test at least 2 supervised learning model

In [42]:
```
# import sklearn library to split data
from sklearn.model_selection import train_test_split
```

In [43]:
```
# Use model_selection.train_test_split from sklearn to split the data into training ar
x_train, x_test, y_train, y_test = train_test_split(df, target, test_size=0.2, random_
```

In [44]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(838860, 6)
(209715, 6)
```

5. Model evaluation

- a) Analyse the results of your trained model
- b) What metrics are most important for the problem? For instance, should the business be more concerned with better results on false negatives or true positives?

In [45]:
```
# Import LinearRegression from sklearn.linear_model to train our model on our training
from sklearn.linear_model import LinearRegression
```

In [46]:
```
model = LinearRegression()
model.fit(x_train, y_train)
```

Out[46]:
```
LinearRegression()
```

In [48]:
```
prediction = model.predict(x_test)
#prediction
```

In [49]:
```
# evaluation of model/ measure of performance
from sklearn import metrics
```

In [50]:
```
MSE = metrics.mean_squared_error(y_test, prediction)
MSE
```

Out[50]:
```
8.31646790667558e-28
```

In [51]:
```
score = metrics.r2_score(y_test, prediction)
score
```

Out[51]:
```
1.0
```

In [ ]:

In [ ]: