PYTHON_PROJECT_TRANSACTIONS_KELLY_NGUYEN

December 31, 2024

1. INTRODUCTION

With the rapid advancement of technology, online and digital payments have become a common part of our daily transactions. However, this convenience has also led to an increase in financial fraud, making it a critical issue to address.

This project aims to analyze the patterns of both fraudulent and legitimate transactions and compare them to detect suspicious activities. The analysis is based on data from millions of real transactions conducted in October 2024.

1. Transaction Categories
2. Regional and Currency Diversity
3. Consumer behaviour

The dataset used for this project includes key features essential for fraud detection, such as merchant type and category, device types, geographic locations, currencies, card types. It is available at Transactions

First, let's import libraries to help with reading, cleaning and visualising the data

[1]:
```
#Importing "pandas" library for reading the dataset
import pandas as pd
#Importing libraries for visualisation of the dataset
import matplotlib.pyplot as plt
import seaborn as sns #visualisation
import numpy as np
```

2. DATA EXPLORATION

Now, we import the dataset and inspect its initial overview.

[2]:
```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/Dataset -Python/synthetic_fraud_data.
↪csv')
```

[3]:
```
df.shape
```

[3]:
```
(7483766, 24)
```

[4]:
```
df.columns
```

[4]:
```
Index(['transaction_id', 'customer_id', 'card_number', 'timestamp',
       'merchant_category', 'merchant_type', 'merchant', 'amount', 'currency',
       'country', 'city', 'city_size', 'card_type', 'card_present', 'device',
       'channel', 'device_fingerprint', 'ip_address', 'distance_from_home',
       'high_risk_merchant', 'transaction_hour', 'weekend_transaction',
       'velocity_last_hour', 'is_fraud'],
      dtype='object')
```

This data has 7,483,766 rows and 24 columns. Now, let's display the first few rows of the dataset to examine its structure and have a look at the dataset information

[5]:
```
df.head()
```

[5]:
```
  transaction_id customer_id       card_number  \
0    TX_a0ad2a2a  CUST_72886  6646734767813109
1    TX_3599c101  CUST_70474   376800864692727
2    TX_a9461c6d  CUST_10715  5251909460951913
3    TX_7be21fc4  CUST_16193   376079286931183
4    TX_150f490b  CUST_87572  6172948052178810

                         timestamp merchant_category merchant_type  \
0 2024-09-30 00:00:01.034820+00:00        Restaurant     fast_food
1 2024-09-30 00:00:01.764464+00:00     Entertainment        gaming
2 2024-09-30 00:00:02.273762+00:00           Grocery      physical
3 2024-09-30 00:00:02.297466+00:00               Gas         major
4 2024-09-30 00:00:02.544063+00:00        Healthcare       medical

         merchant     amount currency    country  …  device channel  \
0       Taco Bell     294.87      GBP         UK  … iOS App  mobile
1           Steam    3368.97      BRL     Brazil  …    Edge     web
2     Whole Foods  102582.38      JPY      Japan  … Firefox     web
3           Exxon     630.60      AUD  Australia  … iOS App  mobile
4  Medical Center  724949.27      NGN    Nigeria  …  Chrome     web

                 device_fingerprint       ip_address distance_from_home  \
0  e8e6160445c935fd0001501e4cbac8bc   197.153.60.199                  0
1  a73043a57091e775af37f252b3a32af9  208.123.221.203                  1
2  218864e94ceaa41577d216b149722261   10.194.159.204                  0
3  70423fa3a1e74d01203cf93b51b9631d   17.230.177.225                  0
4  9880776c7b6038f2af86bd4e18a1b1a4  136.241.219.151                  1

  high_risk_merchant transaction_hour weekend_transaction  \
0              False                0               False
1               True                0               False
2              False                0               False
3              False                0               False
4              False                0               False
```

```
                                velocity_last_hour  is_fraud
0  {'num_transactions': 1197, 'total_amount': 334…     False
1  {'num_transactions': 509, 'total_amount': 2011…      True
2  {'num_transactions': 332, 'total_amount': 3916…     False
3  {'num_transactions': 764, 'total_amount': 2201…     False
4  {'num_transactions': 218, 'total_amount': 4827…      True

[5 rows x 24 columns]
```

[6]:
```
df.duplicated().sum()
```

[6]:
```
0
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7483766 entries, 0 to 7483765
Data columns (total 24 columns):
 #   Column               Dtype
---  ------               -----
 0   transaction_id       object
 1   customer_id          object
 2   card_number          int64
 3   timestamp            object
 4   merchant_category    object
 5   merchant_type        object
 6   merchant             object
 7   amount               float64
 8   currency             object
 9   country              object
 10  city                 object
 11  city_size            object
 12  card_type            object
 13  card_present         bool
 14  device               object
 15  channel              object
 16  device_fingerprint   object
 17  ip_address           object
 18  distance_from_home   int64
 19  high_risk_merchant   bool
 20  transaction_hour     int64
 21  weekend_transaction  bool
 22  velocity_last_hour   object
 23  is_fraud             bool
dtypes: bool(4), float64(1), int64(3), object(16)
memory usage: 1.1+ GB
```

[8]:
```
df.isnull().sum()
```

```
[8]: transaction_id         0
     customer_id            0
     card_number            0
     timestamp              0
     merchant_category      0
     merchant_type          0
     merchant               0
     amount                 0
     currency               0
     country                0
     city                   0
     city_size              0
     card_type              0
     card_present           0
     device                 0
     channel                0
     device_fingerprint     0
     ip_address             0
     distance_from_home     0
     high_risk_merchant     0
     transaction_hour       0
     weekend_transaction    0
     velocity_last_hour     0
     is_fraud               0
     dtype: int64
```

The dataset appears to be generally well-structured and straightforward and there is no missing data found and the rows are not duplicated.

To continue, we will examine the columns that describe transaction characteristics. To begin with, we create 2 variables called categorical and numerical to make it easier for inspecting the columns given their different characteristics.

```
[9]: # List of categorical variables
categorical = [i for i in df.columns if df[i].dtypes == 'O' or df[i].dtypes ==␣
 ↪'bool']
# List of numerical variables
numerical = [i for i in df.columns if i not in categorical]
print('categorical:', categorical, '\n', 'numerical: ', numerical)
```

```
categorical: ['transaction_id', 'customer_id', 'timestamp', 'merchant_category',
'merchant_type', 'merchant', 'currency', 'country', 'city', 'city_size',
'card_type', 'card_present', 'device', 'channel', 'device_fingerprint',
'ip_address', 'high_risk_merchant', 'weekend_transaction', 'velocity_last_hour',
'is_fraud']
 numerical:  ['card_number', 'amount', 'distance_from_home', 'transaction_hour']
```

Since the columns with boolean data type go with True/False, so treating them as categorical data can be more efficient for certain analyses and machine learning models. Let's inspect the distribution of categorical variables.

[10]:
```
df[categorical].describe()
```

[10]:
```
       transaction_id customer_id                          timestamp  \
count         7483766     7483766                            7483766
unique        7477306        4869                            7483754
top        TX_706baadf  CUST_91730  2024-10-23 07:29:30.447871+00:00
freq                3        4015                                  2

       merchant_category merchant_type merchant currency  country  \
count            7483766       7483766  7483766  7483766  7483766
unique                 8            17      105       11       12
top           Healthcare        online    Chegg      EUR  Nigeria
freq              936770       1401650   156105  1065751   849840

                 city city_size    card_type card_present   device  channel  \
count         7483766   7483766      7483766      7483766  7483766  7483766
unique             11         2            5            2        9        3
top      Unknown City    medium  Basic Debit        False     Edge      web
freq          6983706   7284598      1548363      6832719  1189560  4563141

                     device_fingerprint      ip_address high_risk_merchant  \
count                           7483766         7483766            7483766
unique                           785462         7477187                  2
top     30d9029c7fd056ffb7c77cdc22a00d16  193.254.92.164              False
freq                               2373               3            5611803

       weekend_transaction                                velocity_last_hour  \
count              7483766                                           7483766
unique                   2                                           7483740
top                  False  {'num_transactions': 0, 'total_amount': 371.88…
freq               5554103                                                 3

       is_fraud
count   7483766
unique        2
top       False
freq    5989047
```

The summary table displays the most common values for each variable along with their respective frequencies in the dataset.

[11]:
```
df[numerical].describe()
```

[11]:
```
        card_number        amount  distance_from_home  transaction_hour
count  7.483766e+06  7.483766e+06        7.483766e+06      7.483766e+06
mean   4.222100e+15  4.792468e+04        3.220519e-01      1.215467e+01
```

```
std    2.341170e+15  1.775562e+05       4.672628e-01       6.536767e+00
min    3.700086e+14  1.000000e-02       0.000000e+00       0.000000e+00
25%    4.004400e+15  3.635300e+02       0.000000e+00       7.000000e+00
50%    5.010745e+15  1.177450e+03       0.000000e+00       1.200000e+01
75%    5.999914e+15  2.242953e+04       1.000000e+00       1.800000e+01
max    6.999728e+15  6.253153e+06       1.000000e+00       2.300000e+01
```

The table shows that variables are of different ranges.

3. ANALYSIS

In order to analyse deeper about the characteristics of both fraudulent and legitimate transactions. We will separate the orginal dataset to two dataframes which are separately for fraudulent transactions and legitimate transations using the column "is_fraud" to show whether a transaction is suspicious.

[12]:
```python
fraud_transactions = df[df['is_fraud']==True]
```

[13]:
```python
legit_transactions = df[df['is_fraud']==False]
```

Let's check out their information

[14]:
```python
fraud_transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1494719 entries, 1 to 7483755
Data columns (total 24 columns):
 #   Column               Non-Null Count    Dtype
---  ------               --------------    -----
 0   transaction_id       1494719 non-null  object
 1   customer_id          1494719 non-null  object
 2   card_number          1494719 non-null  int64
 3   timestamp            1494719 non-null  object
 4   merchant_category    1494719 non-null  object
 5   merchant_type        1494719 non-null  object
 6   merchant             1494719 non-null  object
 7   amount               1494719 non-null  float64
 8   currency             1494719 non-null  object
 9   country              1494719 non-null  object
 10  city                 1494719 non-null  object
 11  city_size            1494719 non-null  object
 12  card_type            1494719 non-null  object
 13  card_present         1494719 non-null  bool
 14  device               1494719 non-null  object
 15  channel              1494719 non-null  object
 16  device_fingerprint   1494719 non-null  object
 17  ip_address           1494719 non-null  object
 18  distance_from_home   1494719 non-null  int64
 19  high_risk_merchant   1494719 non-null  bool
 20  transaction_hour     1494719 non-null  int64
```

```
 21  weekend_transaction  1494719 non-null  bool
 22  velocity_last_hour   1494719 non-null  object
 23  is_fraud             1494719 non-null  bool
dtypes: bool(4), float64(1), int64(3), object(16)
memory usage: 245.2+ MB
```

[15]:
```python
fraud_percentage = (len(fraud_transactions)/len(df)*100)
print(fraud_percentage)
```

```
19.972818498066346
```

The dataset contains 1,494,719 fraudulent transactions, making up 19.97% of the total

[16]:
```python
legit_transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5989047 entries, 0 to 7483765
Data columns (total 24 columns):
 #   Column               Dtype
---  ------               -----
 0   transaction_id       object
 1   customer_id          object
 2   card_number          int64
 3   timestamp            object
 4   merchant_category    object
 5   merchant_type        object
 6   merchant             object
 7   amount               float64
 8   currency             object
 9   country              object
 10  city                 object
 11  city_size            object
 12  card_type            object
 13  card_present         bool
 14  device               object
 15  channel              object
 16  device_fingerprint   object
 17  ip_address           object
 18  distance_from_home   int64
 19  high_risk_merchant   bool
 20  transaction_hour     int64
 21  weekend_transaction  bool
 22  velocity_last_hour   object
 23  is_fraud             bool
dtypes: bool(4), float64(1), int64(3), object(16)
memory usage: 982.4+ MB
```

[17]:
```python
legit_percentage = (len(legit_transactions)/len(df)*100)
print(legit_percentage)
```

```
80.02718150193365
```

There are 5,989,047 legitimate transactions in the dataset, accounting for 80.03% of the total.

3.1. Transaction Categories

3.1.1 Column "merchant_category"

First, let's check out "merchant_category" column to see the categories of merchant of all transactions.

[18]:
```python
df.merchant_category.value_counts()
```

[18]:
```
merchant_category
Healthcare       936770
Restaurant       936178
Entertainment    936173
Retail           935883
Travel           935790
Gas              935401
Grocery          934029
Education        933542
Name: count, dtype: int64
```

The transaction payments are distributed across eight categories: healthcare, restaurants, entertainment, retail, travel, gas, grocery, and education. Interestingly, these categories are evenly distributed

Now, we will examine the merchant category associated with fraudulent transactions

[19]:
```python
fraud_transactions.merchant_category.value_counts()
```

[19]:
```
merchant_category
Travel           187477
Grocery          186987
Restaurant       186951
Entertainment    186890
Gas              186829
Healthcare       186769
Retail           186613
Education        186203
Name: count, dtype: int64
```

The fraud transactional counts across all merchant categories are similar, ranging between approximately 186,000 and 187,000 transactions. The Travel category records the highest number of transactions. This is likely due to the significant growth in online bookings and digital payments within the travel industry.

[20]:
```python
fraud_category_percentage = (fraud_transactions['merchant_category'].
 ↪value_counts() / df['merchant_category'].value_counts()) * 100
print(fraud_category_percentage)
```

```
merchant_category
Education        19.945862
Entertainment    19.963191
Gas              19.973145
Grocery          20.019400
Healthcare       19.937551
Restaurant       19.969600
Retail           19.939779
Travel           20.034089
Name: count, dtype: float64
```

Let's analyse legitimate transaction activity

[21]:
```
legit_transactions.merchant_category.value_counts()
```

[21]:
```
merchant_category
Healthcare       750001
Entertainment    749283
Retail           749270
Restaurant       749227
Gas              748572
Travel           748313
Education        747339
Grocery          747042
Name: count, dtype: int64
```

[22]:
```
legit_category_percentage = (legit_transactions['merchant_category'].
  ↪value_counts() / df['merchant_category'].value_counts()) * 100
print(legit_category_percentage)
```

```
merchant_category
Education        80.054138
Entertainment    80.036809
Gas              80.026855
Grocery          79.980600
Healthcare       80.062449
Restaurant       80.030400
Retail           80.060221
Travel           79.965911
Name: count, dtype: float64
```

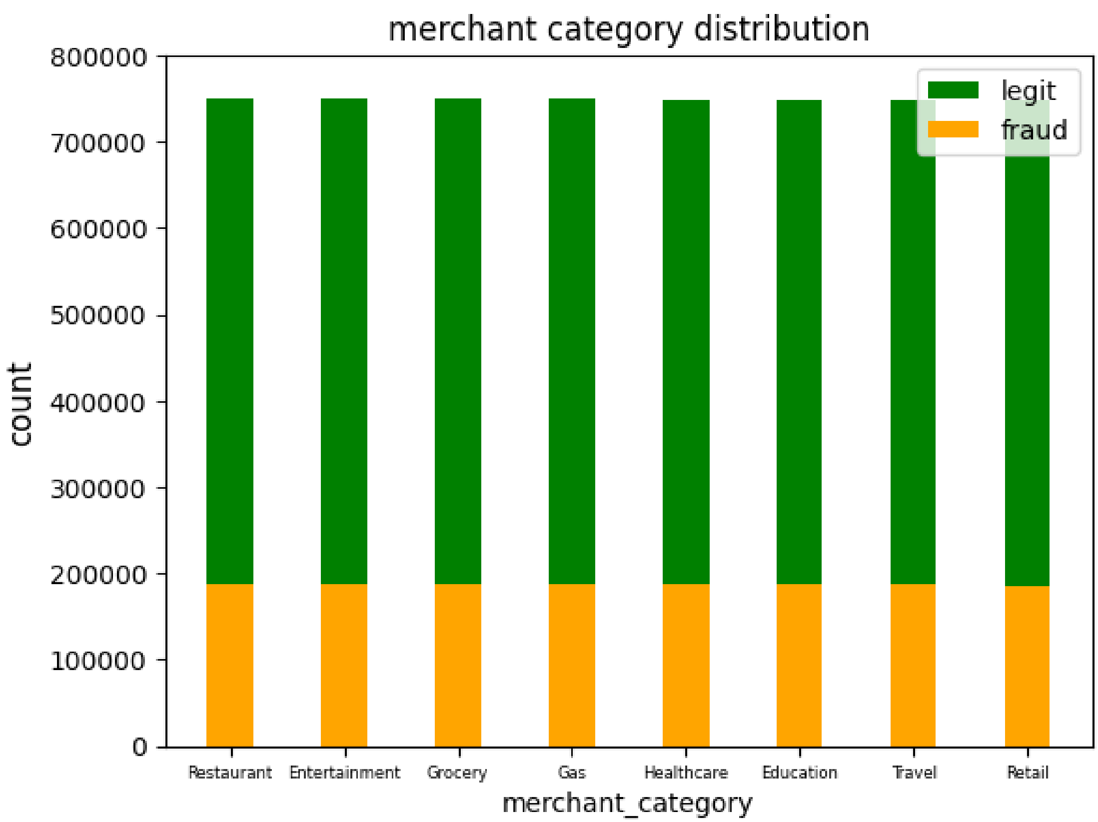
Now, we will display the distribution of merchant categories in a bar chart for comparison

[23]:
```
df.merchant_category.unique()

index = np.arange(8)
width = 0.4
plt.bar(index, legit_transactions.merchant_category.value_counts(),width,␣
  ↪color="green", label="legit")
```

```
plt.bar(index, fraud_transactions.merchant_category.value_counts(), width,␣
  ↪color="orange", label="fraud")
plt.xlabel("merchant_category")
plt.ylabel("count",fontsize=12)
plt.ylim(0, 800000)
plt.title("merchant category distribution")
plt.xticks(index,df.merchant_category.unique(),fontsize=6)

plt.legend(loc='best')
plt.show()
```



From the chart, we can clearly see how legitimate and fraudulent transactions contribute to the total transactions across the eight merchant categories. Fraudulent transactions account for approximately 20% of the total, while legitimate transactions make up around 80%.

3.1.2 Merchant_type

Let's have a deeper look into "merchant_type" to see more specific types of merchant that the dataset contains

[24]:
```
df['merchant_type'].value_counts()
```

[24]:
```
merchant_type
online        1401650
physical       935039
medical        468393
pharmacy       468377
local          467902
major          467499
supplies       466765
fast_food      312805
events         312598
streaming      312091
premium        311695
casual         311678
gaming         311484
hotels         234311
booking        234026
transport      233977
airlines       233476
Name: count, dtype: int64
```

We observe 17 specific types within the merchant category. The highest transaction count belongs to the "online" type, followed by "physical," then "medical," "pharmacy," and "local."

Online transactions and digital payments are the most prevalent, with physical transactions also being significant.

Combining merchant type and category to have a more detailed view of transaction distributions.

[25]:
```python
df_type_category= df.groupby(['merchant_type', 'merchant_category']).size().
 ↪reset_index(name='count')
print(df_type_category)
```

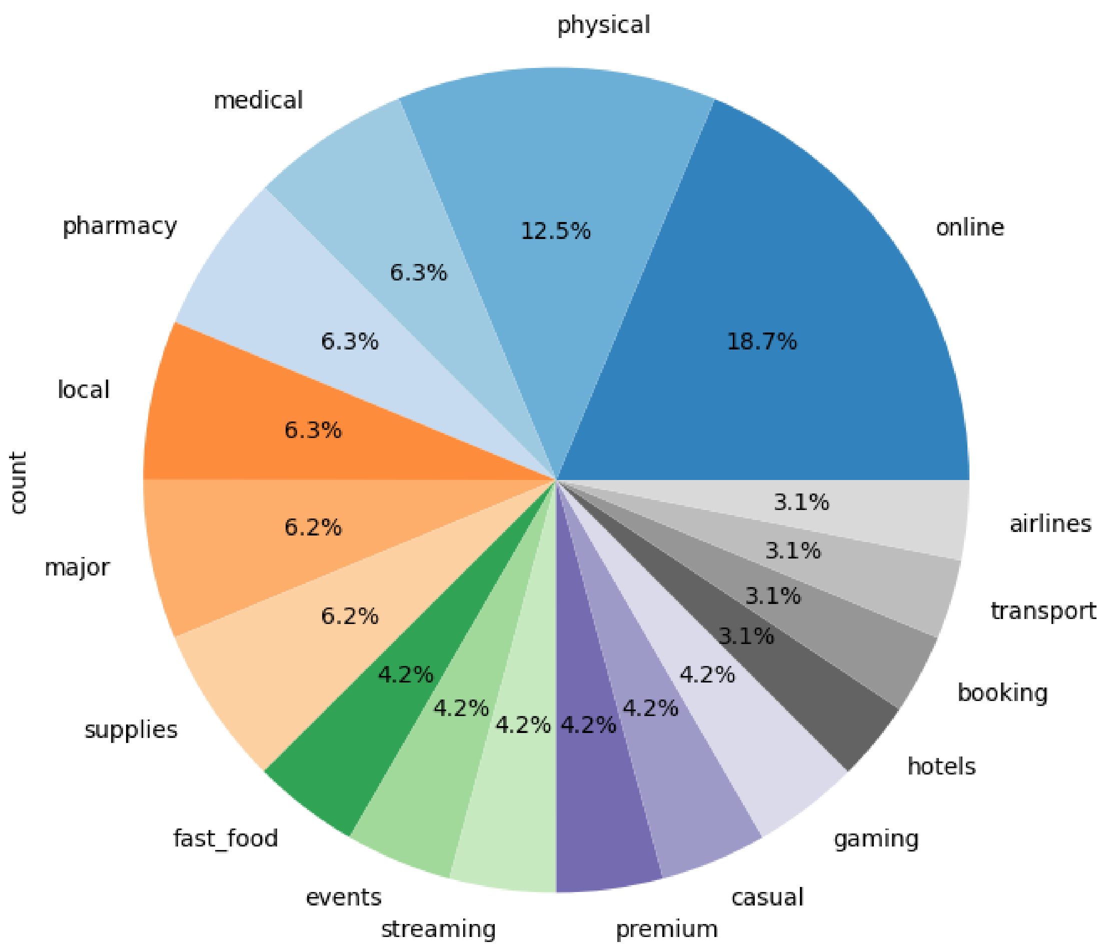
```
   merchant_type merchant_category   count
0       airlines            Travel  233476
1        booking            Travel  234026
2         casual        Restaurant  311678
3         events     Entertainment  312598
4      fast_food        Restaurant  312805
5         gaming     Entertainment  311484
6         hotels            Travel  234311
7          local               Gas  467902
8          major               Gas  467499
9        medical        Healthcare  468393
10        online         Education  466777
11        online           Grocery  467231
12        online            Retail  467642
13      pharmacy        Healthcare  468377
14      physical           Grocery  466798
15      physical            Retail  468241
```

```
16        premium        Restaurant  311695
17      streaming     Entertainment  312091
18       supplies         Education  466765
19      transport            Travel  233977
```

We will now show them in percentage to see the differences.

[26]:
```
df['merchant_type'].value_counts().plot.pie(autopct='%1.1f%%', figsize=(8, 8),
    colormap ='tab20c')
```

[26]:
```
<Axes: ylabel='count'>
```



From the pie chart, we can clearly see that Online holds the highest percentage of transactions at 18.7%, followed by Physical at 12.5%. Both Medical and Pharmacy hold an equal percentage of 6.3% each. The smallest percentage is 3.1% and belongs to four different merchant type including airlines, transport, booking and hotels.

Let's dig deeper into online transactions

[27]:
```
print(df[df['merchant_type']=='online']['merchant_category'].value_counts())
```

```
merchant_category
Retail       467642
Grocery      467231
Education    466777
Name: count, dtype: int64
```

The online merchant type appears across multiple categories including Education, Grocery, and Retail, indicating a significant volume of online transactions in these sectors.

We will now analyze the distribution of fraudulent and legitimate transactions across different merchant types.

[28]:
```
fraud_transactions['merchant_type'].value_counts()
```

[28]:
```
merchant_type
online       279363
physical     187200
pharmacy      93569
major         93416
local         93413
supplies      93240
medical       93200
fast_food     62786
events        62525
streaming     62206
gaming        62159
premium       62083
casual        62082
transport     47069
booking       46846
airlines      46820
hotels        46742
Name: count, dtype: int64
```

[29]:
```
legit_transactions['merchant_type'].value_counts()
```

[29]:
```
merchant_type
online       1122287
physical      747839
medical       375193
pharmacy      374808
local         374489
major         374083
supplies      373525
events        250073
```

```
fast_food     250019
streaming     249885
premium       249612
casual        249596
gaming        249325
hotels        187569
booking       187180
transport     186908
airlines      186656
Name: count, dtype: int64
```

Let's group the merchant types and categories of both fraudulent and legitimate transactions to analyze their combinations.

[30]:
```python
#fraudulent transactions
fraudulent_combination = fraud_transactions.groupby(['merchant_type',␣
↪'merchant_category']).size()

fraudulent_combination_df = fraudulent_combination.reset_index(name='count')
print(fraudulent_combination_df)
```

```
   merchant_type merchant_category  count
0       airlines            Travel  46820
1        booking            Travel  46846
2         casual        Restaurant  62082
3         events     Entertainment  62525
4      fast_food        Restaurant  62786
5         gaming     Entertainment  62159
6         hotels            Travel  46742
7          local               Gas  93413
8          major               Gas  93416
9        medical        Healthcare  93200
10        online         Education  92963
11        online           Grocery  93396
12        online            Retail  93004
13      pharmacy        Healthcare  93569
14      physical           Grocery  93591
15      physical            Retail  93609
16       premium        Restaurant  62083
17     streaming     Entertainment  62206
18      supplies         Education  93240
19     transport            Travel  47069
```

[31]:
```python
#legitimate transactions
legitimate_combination = legit_transactions.groupby(['merchant_type',␣
↪'merchant_category']).size()

legitimate_combination_df = legitimate_combination.reset_index(name='count')
```

```python
print(legitimate_combination_df)
```

```
   merchant_type merchant_category   count
0       airlines            Travel  186656
1        booking            Travel  187180
2         casual        Restaurant  249596
3         events     Entertainment  250073
4      fast_food        Restaurant  250019
5         gaming     Entertainment  249325
6         hotels            Travel  187569
7          local               Gas  374489
8          major               Gas  374083
9        medical        Healthcare  375193
10        online         Education  373814
11        online           Grocery  373835
12        online            Retail  374638
13      pharmacy        Healthcare  374808
14      physical           Grocery  373207
15      physical            Retail  374632
16       premium        Restaurant  249612
17     streaming     Entertainment  249885
18      supplies         Education  373525
19     transport            Travel  186908
```

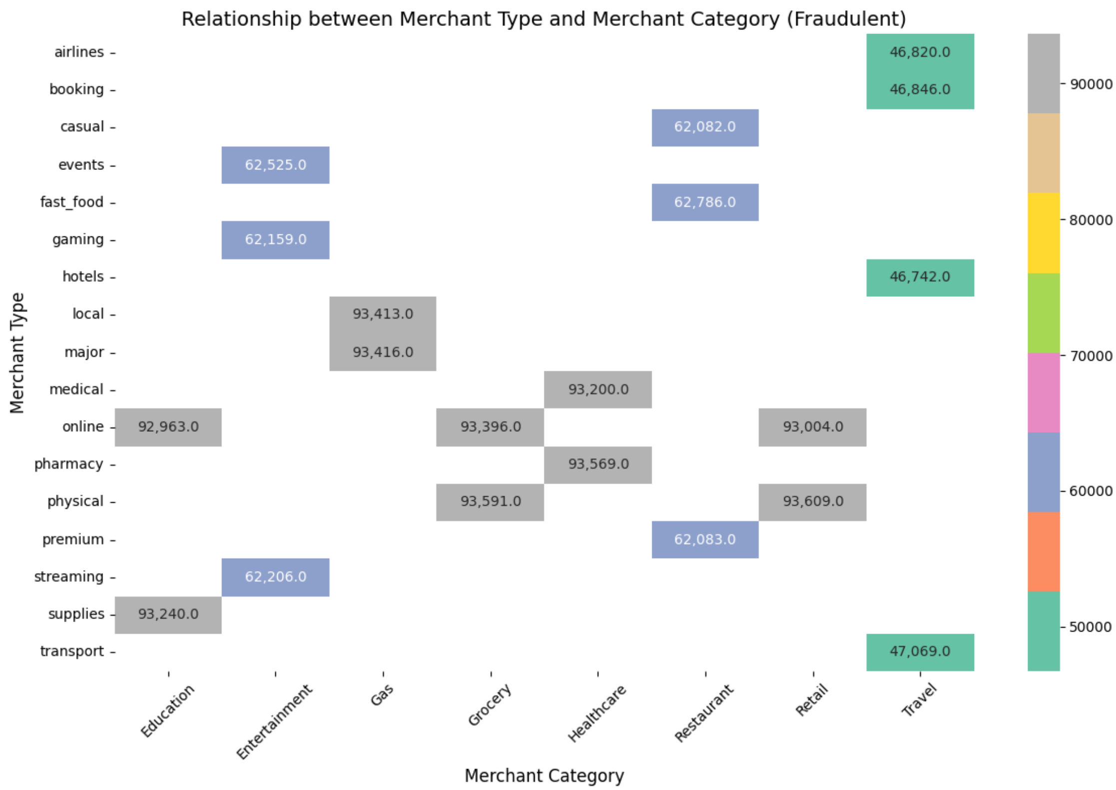
We are now creating a heatmap to see the distribution of merchant type and category to total suspicious and verified transactions

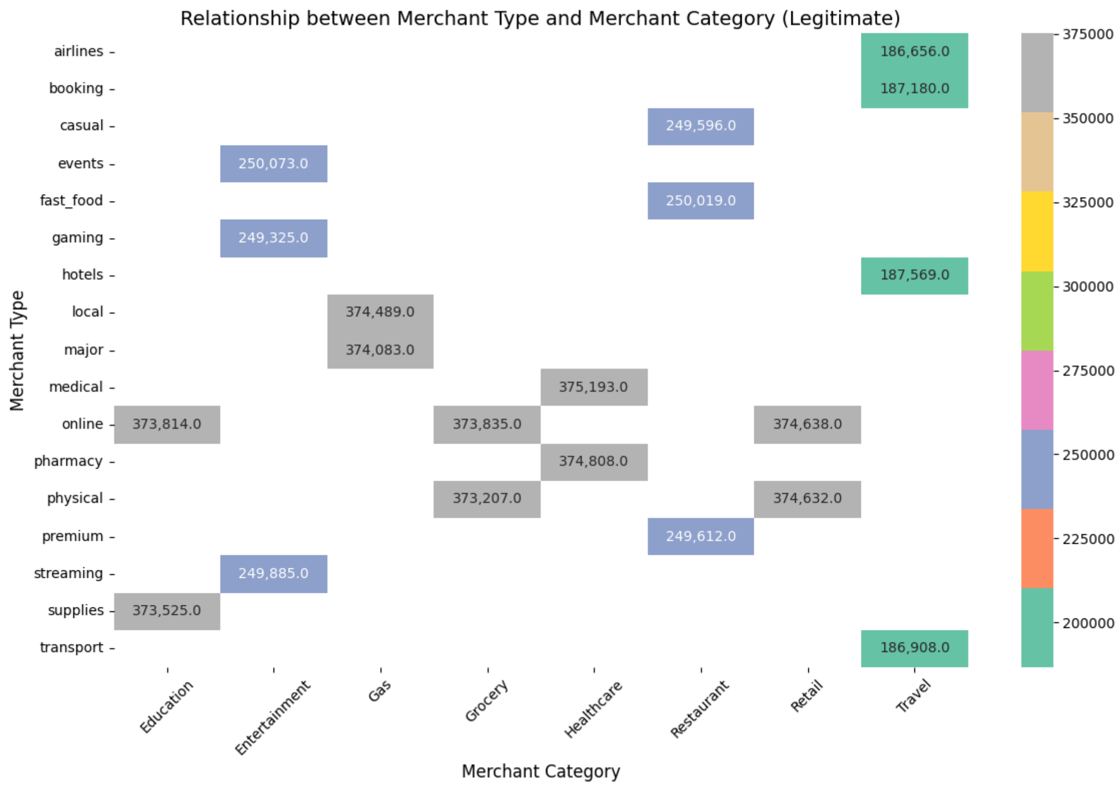
[32]:
```python
# Create the pivot table
fraud_combination = fraudulent_combination_df.pivot(index='merchant_type',␣
 ↪columns='merchant_category', values='count')
legit_combination = legitimate_combination_df.pivot(index='merchant_type',␣
 ↪columns='merchant_category', values='count')

# The first heatmap for fraudulent combination
plt.figure(figsize=(12, 8))
sns.heatmap(fraud_combination, annot=True, fmt=",", cmap="Set2", cbar=True)
plt.title("Relationship between Merchant Type and Merchant Category␣
 ↪(Fraudulent)", fontsize=14)
plt.xlabel("Merchant Category", fontsize=12)
plt.ylabel("Merchant Type", fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()

# The second heatmap for legitimate combination
plt.figure(figsize=(12, 8))
sns.heatmap(legit_combination, annot=True, fmt=",", cmap="Set2", cbar=True)
```

```
plt.title("Relationship between Merchant Type and Merchant Category␣
↪(Legitimate)", fontsize=14)
plt.xlabel("Merchant Category", fontsize=12)
plt.ylabel("Merchant Type", fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()
```





The first heatmap illustrates the relationship between merchant type and merchant category for fraudulent payments. The second heatmap displays the same relationship, but specifically for legitimate transactions. Their contributions are quite similar within merchant types, with online transactions accounting for the highest count.

3.2 Regional and Currency Diversity

We will now analyse column currency and country to see which currency were the most indicated and which country has the most transactions

[33]:
```
#currency of dataset
currency_df= df['currency'].value_counts().reset_index(name='country_count')
print(currency_df)
#country by currency
country_df = df.groupby(['currency', 'country']).size().
  ↪reset_index(name='count')
print(country_df)
#Merging currency and country
merged_df = pd.merge(currency_df,country_df, on='currency', how='inner')
print(merged_df)
```

```
  currency  country_count
0      EUR        1065751
1      NGN         849840
```

2	BRL	804800
3	RUB	793730
4	MXN	785704
5	SGD	588668
6	GBP	538493
7	CAD	532632
8	JPY	527393
9	USD	500060
10	AUD	496695

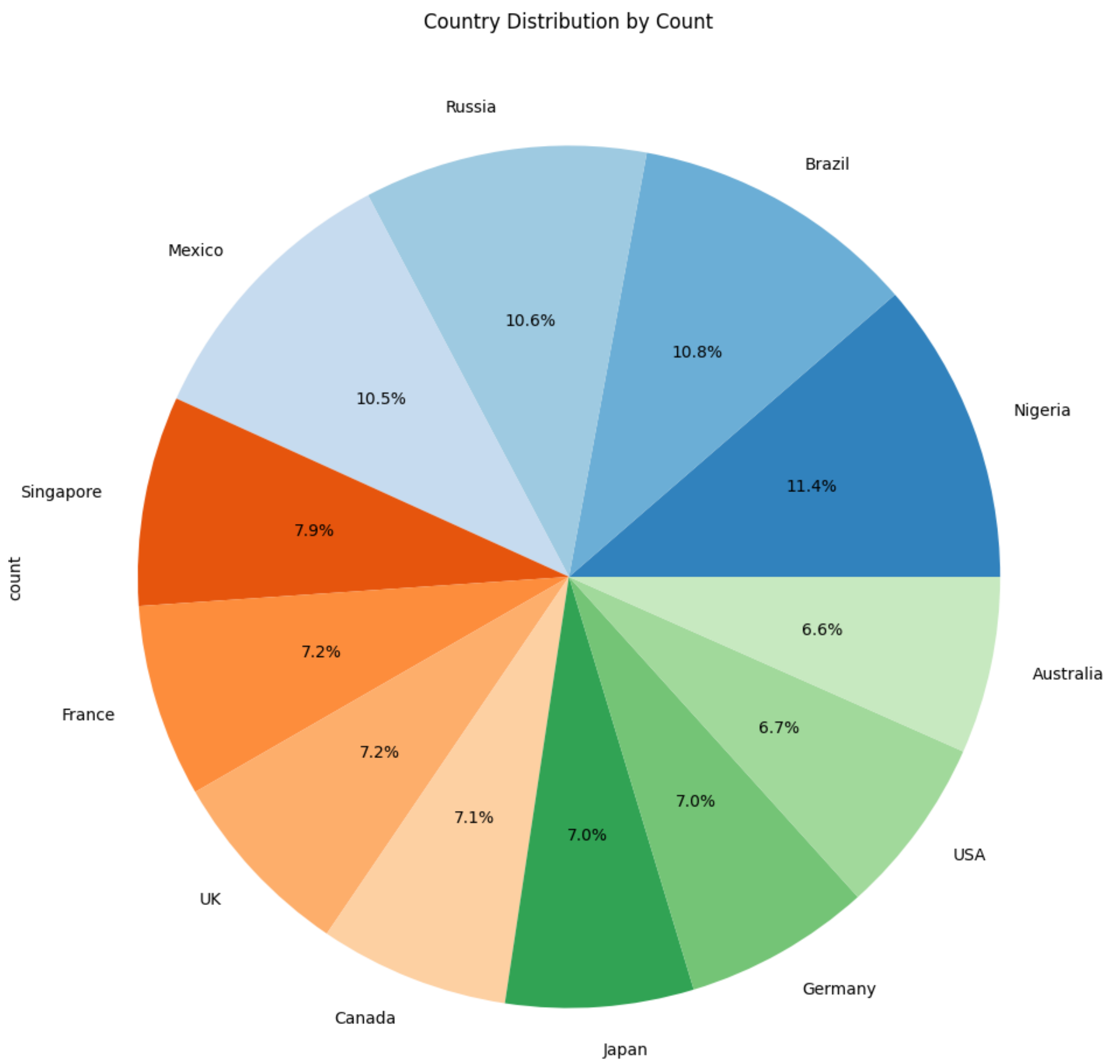
	currency	country	count
0	AUD	Australia	496695
1	BRL	Brazil	804800
2	CAD	Canada	532632
3	EUR	France	541287
4	EUR	Germany	524464
5	GBP	UK	538493
6	JPY	Japan	527393
7	MXN	Mexico	785704
8	NGN	Nigeria	849840
9	RUB	Russia	793730
10	SGD	Singapore	588668
11	USD	USA	500060

	currency	country_count	country	count
0	EUR	1065751	France	541287
1	EUR	1065751	Germany	524464
2	NGN	849840	Nigeria	849840
3	BRL	804800	Brazil	804800
4	RUB	793730	Russia	793730
5	MXN	785704	Mexico	785704
6	SGD	588668	Singapore	588668
7	GBP	538493	UK	538493
8	CAD	532632	Canada	532632
9	JPY	527393	Japan	527393
10	USD	500060	USA	500060
11	AUD	496695	Australia	496695

There are 11 unique currency types in the dataset. The Euro (EUR) has the highest count of 1,065,751 but it is split between France (541,287) and Germany (524,464). Therefore, Nigeria (NGN) - 849,840, Brazil (BRL) -804,800, Russia (RUB) - 793,730, and Mexico (MXN) -785,704 stand out as the countries with the highest number of transactions

[34]:
```
df['country'].value_counts().plot.pie(autopct='%1.1f%%', figsize=(12, 12),␣
↪colors=plt.get_cmap('tab20c').colors, title='Country Distribution by Count')
```

[34]: <Axes: title={'center': 'Country Distribution by Count'}, ylabel='count'>



Let's have a deeper look at fraudulent data to analyse suspicious patterns

[35]:
```
#currency of fraudulent transaction
fraud_currency_df= fraud_transactions['currency'].value_counts().
 ↪reset_index(name='fraudulent_count')
#country of fraudulent transaction
fraud_country_df = fraud_transactions.groupby(['currency', 'country']).size().
 ↪reset_index(name='country_count')
#merging fraudulent currency and country
merged_fraud = pd.merge(fraud_currency_df,fraud_country_df, on='currency',␣
 ↪how='inner')
print(merged_fraud)
```

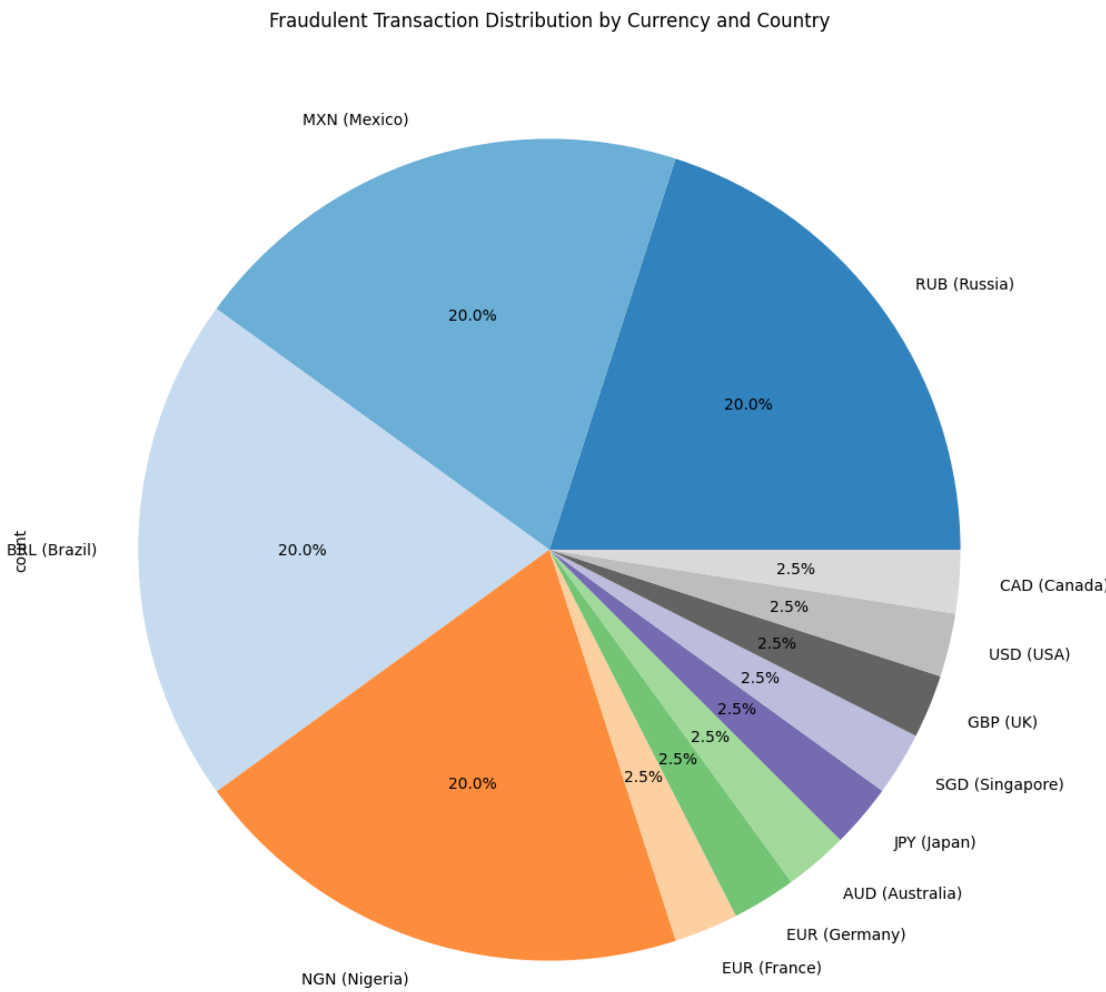
```
   currency  fraudulent_count    country  country_count
0       RUB            299425     Russia         299425
1       MXN            298841     Mexico         298841
2       BRL            298629     Brazil         298629
3       NGN            298600    Nigeria         298600
4       EUR             74631     France          37426
5       EUR             74631    Germany          37205
6       AUD             37652  Australia          37652
7       JPY             37592      Japan          37592
8       SGD             37414  Singapore          37414
9       GBP             37345         UK          37345
10      USD             37312        USA          37312
11      CAD             37278     Canada          37278
```

Drawing a pie chart to show the distribution of fraudulent tranasctions by curency and country

[36]:
```
#merge currency and country
merged_fraud['currency_country'] = merged_fraud['currency'] + " (" +␣
↪merged_fraud['country'] + ")"
merged_fraud['proportion'] = merged_fraud['country_count'] /␣
↪merged_fraud['country_count'].sum()
print(merged_fraud[['currency', 'country','proportion']])
#
fraud_transactions.country.value_counts().plot.pie(autopct='%1.1f%%',␣
↪figsize=(12, 12), colormap='tab20c', title='Fraudulent Transaction␣
↪Distribution by Currency and␣
↪Country',labels=merged_fraud['currency_country'])
```

```
   currency    country  proportion
0       RUB     Russia    0.200322
1       MXN     Mexico    0.199931
2       BRL     Brazil    0.199789
3       NGN    Nigeria    0.199770
4       EUR     France    0.025039
5       EUR    Germany    0.024891
6       AUD  Australia    0.025190
7       JPY      Japan    0.025150
8       SGD  Singapore    0.025031
9       GBP         UK    0.024985
10      USD        USA    0.024963
11      CAD     Canada    0.024940
```

[36]:
```
<Axes: title={'center': 'Fraudulent Transaction Distribution by Currency and
Country'}, ylabel='count'>
```



We can observe the distribution of fraudulent transactions across various currencies. RUB (299,425) has the highest count, followed by MXN, BRL, and NGN. These four currencies account for approximately 80% of the total fraudulent transactions. This suggests that these currencies are major contributors to the overall fraudulent activity. AUD, JPY, SGD, GBP, USD, and CAD have lower counts, with values that are much smaller compared to RUB, MXN, and BRL.

We will now continue to analyse legitimate transactions

[37]:
```
# currencies of legitimate tranasctions
legit_currency_df=legit_transactions.currency.value_counts().
 ↪reset_index(name='legitimate_count')
print(legit_currency_df)

##country by currency of legitimate transactions
```

```
legit_country_df = legit_transactions.groupby(['currency', 'country']).size().
  ↪reset_index(name='country_count')
#merging fraudulent currency and country
merged_legit = pd.merge(legit_currency_df,legit_country_df, on='currency',␣
  ↪how='inner')
print(merged_legit)
```

```
   currency  legitimate_count
0       EUR            991120
1       SGD            551254
2       NGN            551240
3       BRL            506171
4       GBP            501148
5       CAD            495354
6       RUB            494305
7       JPY            489801
8       MXN            486863
9       USD            462748
10      AUD            459043
   currency  legitimate_count    country  country_count
0       EUR            991120     France         503861
1       EUR            991120    Germany         487259
2       SGD            551254  Singapore         551254
3       NGN            551240    Nigeria         551240
4       BRL            506171     Brazil         506171
5       GBP            501148         UK         501148
6       CAD            495354     Canada         495354
7       RUB            494305     Russia         494305
8       JPY            489801      Japan         489801
9       MXN            486863     Mexico         486863
10      USD            462748        USA         462748
11      AUD            459043  Australia         459043
```

The Euro (EUR) ranks highest with a total legitimate count of 991,120, split between France (503,861) and Germany (487,259), followed by SGD (551,254), NGN (551,240), and BRL (506,171).

Next, let's merge legitimate_count and fraudulent_count by currency and country to see their proportion.

[38]:
```
currency_merged = pd.merge(legit_currency_df, fraud_currency_df, on='currency',␣
  ↪how='inner')
print(currency_merged)
```

```
  currency  legitimate_count  fraudulent_count
0      EUR            991120             74631
1      SGD            551254             37414
2      NGN            551240            298600
3      BRL            506171            298629
4      GBP            501148             37345
```

5	CAD	495354	37278
6	RUB	494305	299425
7	JPY	489801	37592
8	MXN	486863	298841
9	USD	462748	37312
10	AUD	459043	37652

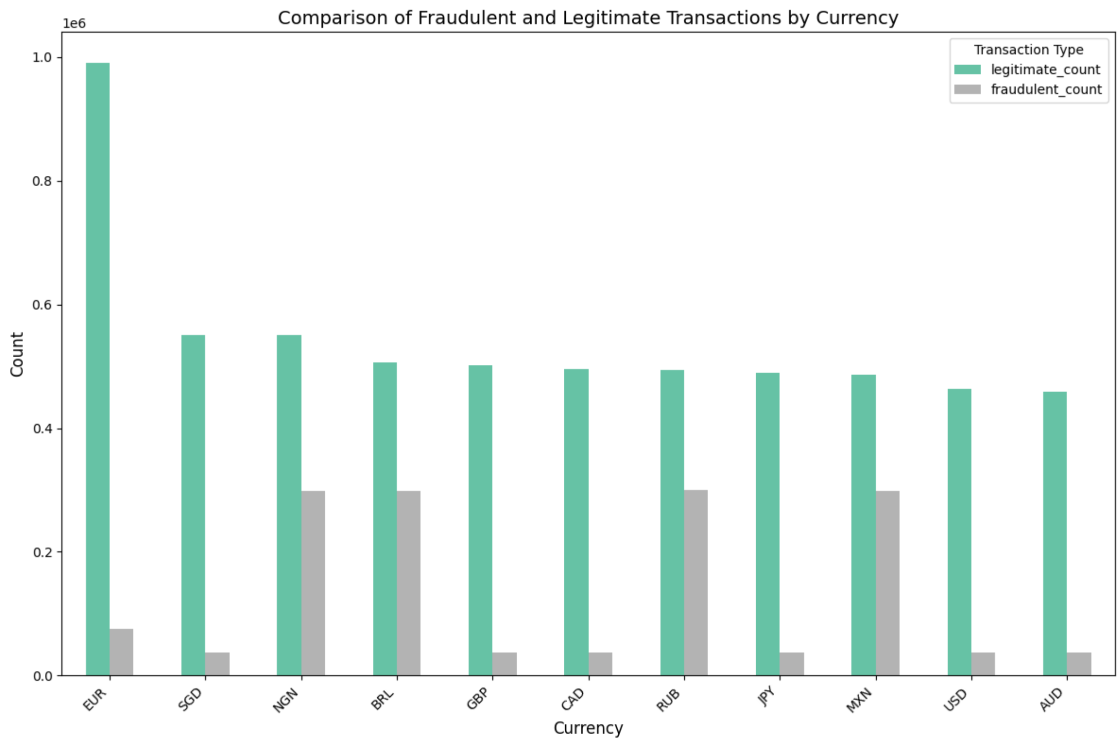
Drawing a stacked bar chart for comparison.

[39]:
```
# Plotting the data
fig, ax = plt.subplots(figsize=(12, 8))

# Plotting both fraudulent and legitimate counts as a bar chart
currency_merged.set_index('currency')[['legitimate_count', 'fraudulent_count']].
↪plot(kind='bar', stacked=False, ax=ax, colormap='Set2')

# Adding chart labels and title
plt.title('Comparison of Fraudulent and Legitimate Transactions by Currency',␣
↪fontsize=14)
plt.xlabel('Currency', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.legend(title='Transaction Type', fontsize=10)
plt.tight_layout()

# Display the chart
plt.show()
```



Further to the previous analysis, fraudulent transactions account for about 20% of total payments. We can easily notice that four currencies—RUB (Russia), MXN (Mexico), BRL (Brazilia), and NGN (Nigeria)—are from regions with unstable economies, making them more likely to be targeted for fraudulent transactions due to the potential for money laundering or other illegal activities. These currencies need to be flagged for further analysis, particularly focusing on their involvement in specific merchant categories or transaction types. Meanwhile, other currencies and corresponding countries such as EUR (Germany and France), SGD (Singapore), GBP (UK), CAD (Canada), JPY (Japan), USD (USA), and AUD(Australia) have a substantial amount of legitimate transactions and come from stable, developed countries which align with stronger regulatory frameworks and advanced financial systems.

[40]:
```
#fraud transactions by city
fraud_location= fraud_transactions.groupby(['country', 'city']).size().
↪reset_index(name='count')
print(fraud_location)
```

```
     country          city   count
0  Australia  Unknown City   37652
1     Brazil  Unknown City  298629
2     Canada  Unknown City   37278
3     France  Unknown City   37426
4    Germany  Unknown City   37205
5      Japan  Unknown City   37592
6     Mexico  Unknown City  298841
```

```
7     Nigeria  Unknown City  298600
8      Russia  Unknown City  299425
9   Singapore  Unknown City   37414
10         UK  Unknown City   37345
11        USA       Chicago    3701
12        USA        Dallas    3648
13        USA       Houston    3687
14        USA   Los Angeles    3771
15        USA      New York    3696
16        USA  Philadelphia    3739
17        USA       Phoenix    3786
18        USA   San Antonio    3736
19        USA     San Diego    3771
20        USA      San Jose    3777
```

The city where payments occured is not recorded outside of the USA, therefore we cannot dig deeper into the location.

[41]:
```
fraud_amount = fraud_transactions.groupby(['merchant_category',␣
↪'merchant_type', 'amount']).size().reset_index(name='count')
print(fraud_amount)
```

```
        merchant_category merchant_type     amount  count
0               Education        online       0.01      4
1               Education        online       0.02      8
2               Education        online       0.03     11
3               Education        online       0.04     12
4               Education        online       0.05     16
…                       …             …        …    …
1292415            Travel     transport  207595.52      1
1292416            Travel     transport  207942.87      1
1292417            Travel     transport  208333.64      1
1292418            Travel     transport  208390.21      1
1292419            Travel     transport  208484.99      1

[1292420 rows x 4 columns]
```

3.3 Consumer behaviour

We will continue with chanel and device that customer used to process transactions

[42]:
```
#channel used in all dataset.
df.channel.value_counts()
```

[42]:
```
channel
web       4563141
mobile    2269578
pos        651047
Name: count, dtype: int64
```

There are three channels which are web/browser channel has the highest transaction count, 4,563,141, followed by mobile and POS Point Of Sale)

[43]:
```
#devices used in all dataset.
df.device.value_counts()
```

[43]:
```
device
Edge               1189560
iOS App            1143461
Chrome             1132384
Android App        1126117
Firefox            1120952
Safari             1120245
Chip Reader         217324
Magnetic Stripe     217204
NFC Payment         216519
Name: count, dtype: int64
```

Web browsers and mobile apps play significant roles in today's digital payment platforms, surpassing traditional methods. Among them, Edge leads with 1,189,560 transactions, while iOS App and Android App also demonstrate strong usage with 1,143,461 and 1,126,117 transactions, respectively. Other web browsers, such as Chrome, Firefox, and Safari, also surpass a million transactions each.

Meanwhile, the data highlights the continued presence of traditional payment methods. These include Chip Reader (217,324 transactions), Magnetic Stripe (217,204 transactions), and NFC Payment (216,519 transactions), which collectively account for a significantly smaller share. Traditional payment methods, though less used, could still pose unique fraud risks that need monitoring.

Let's see the change in the device and channel used with fraudulent transactional activity.

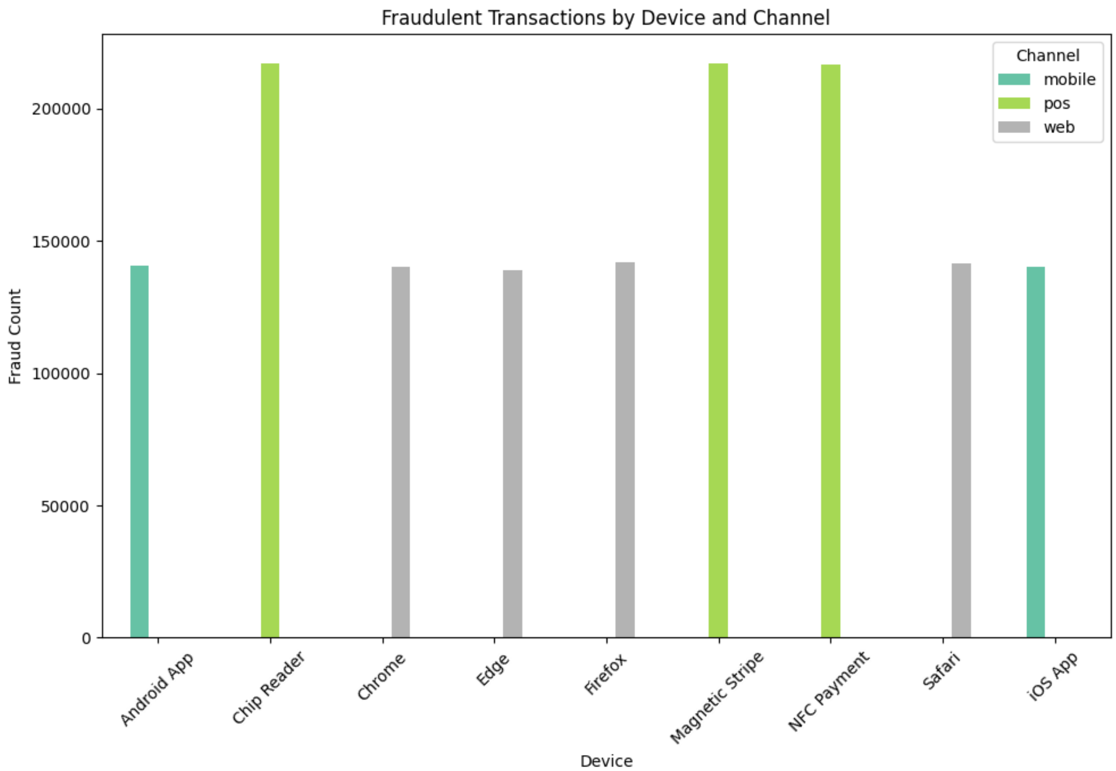
[44]:
```
fraud_channel = fraud_transactions.groupby(['device', 'channel']).size().
    ↪reset_index(name='count')
print(fraud_channel)
# Create a pivot table to make the data easier to plot
fraud_channel_pivot = fraud_channel.pivot(index='device', columns='channel',␣
    ↪values='count')

# Plot a grouped bar chart
fraud_channel_pivot.plot(kind='bar', figsize=(10, 7), colormap='Set2')

plt.title('Fraudulent Transactions by Device and Channel')
plt.xlabel('Device')
plt.ylabel('Fraud Count')
plt.xticks(rotation=45)
plt.legend(title='Channel')
plt.tight_layout()
plt.show()
```

```
            device channel   count
0      Android App  mobile  140844
```

```
1       Chip Reader     pos  217324
2            Chrome     web  140087
3              Edge     web  138885
4           Firefox     web  142171
5   Magnetic Stripe     pos  217204
6       NFC Payment     pos  216519
7            Safari     web  141379
8           iOS App  mobile  140306
```



As we can see from the data, all payment methods are associated with fraudulent activity. POS payment methods, such as Chip Reader, Magnetic Stripe, and NFC Payment, account for significant fraud counts, highlighting the need to strengthen security measures in these systems. Additionally, other browsers and mobile apps also contribute to high fraud counts. This suggests that enhanced security measures should be prioritized for these platforms as well.

While newer technologies like mobile/browser payments or digital wallets often have multiple layers of security (such as two-factor authentication, biometrics, and tokenization), traditional payment methods can be more vulnerable to certain types of fraud. Hence, fraud activity is often higher on these platforms, which highlights the need for stronger security protocols and the adoption of more secure, modern payment methods.

Next, let's see if card was physically present during the transactions.

[45]:
```
card_present = fraud_transactions.card_present.value_counts()
print(card_present)
```

```
card_present
False    843672
True     651047
Name: count, dtype: int64
```

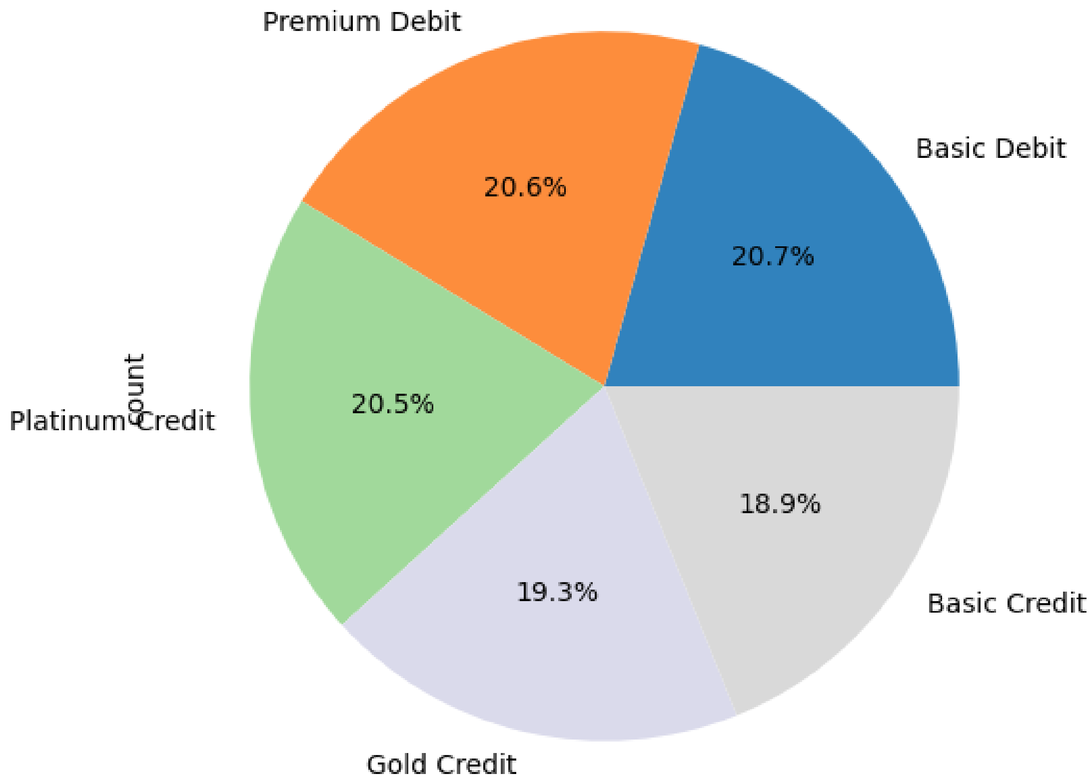
There are 651,047 transactions using a card, and 843,672 transactions are online.

The type of card used in fraudulent transactions.

[46]:
```
#Type of card used in the transaction
card=fraud_transactions.card_type.value_counts()
print(card)
fraud_transactions['card_type'].value_counts().plot.pie(autopct='%1.1f%%',␣
↪figsize=(6, 6), colormap ='tab20c')
```

```
card_type
Basic Debit        309239
Premium Debit      307502
Platinum Credit    306190
Gold Credit        289060
Basic Credit       282728
Name: count, dtype: int64
```

[46]:
```
<Axes: ylabel='count'>
```



The card types include Debit (Premium and Basic) and Credit (Platinum, Basic, Gold). Fraud activity seems to be more prevalent with credit cards than with debit cards. One reason could be that credit card balances are often larger than debit card balances. Additionally, debit card transactions are processed in real time, while credit card transactions are updated after some time, which may also contribute to the higher count of fraud with credit cards compared to debit cards.

4. Conclusion

Fraudulent transactional activity mostly originate from unstable countries: Russia, Mexico, Nigeria and Brazil with corresponding currencies RUB, MXN, NGN, BRL

Online transactions exhibit the highest fraud counts.

Beyond Point of Sale (POS) payment methods, fraud risks are mainly detected through Web and Mobile platforms.

Credit cards are more commonly used for fraudulent activities than debit cards.

Online and digital payments is often targeted by fraudsters.

Implementing robust fraud detection measures and enabling the identification of anomalies indicative of fraudulent activities are essential.

Understanding these transaction patterns is crucial for developing targeted strategies to detect and prevent fraud, ensuring the integrity and security of financial transactions across various sectors.

Thank you for your attention