Fraud

September 30, 2024

1 A. Background

E-commerce users in Indonesia have continued to increase since 2017, with 70.8 million users, and are predicted to reach 189.6 million in 2024. In 2018, the number of users reached 87.5 million, and continued to grow to 129.9 million in 2020. In 2021, it is estimated that there will be 148.9 million users, 166.1 million in 2022, and 180.6 million in 2023.

[1]:
```
import os
os.system('pandoc --version')
```

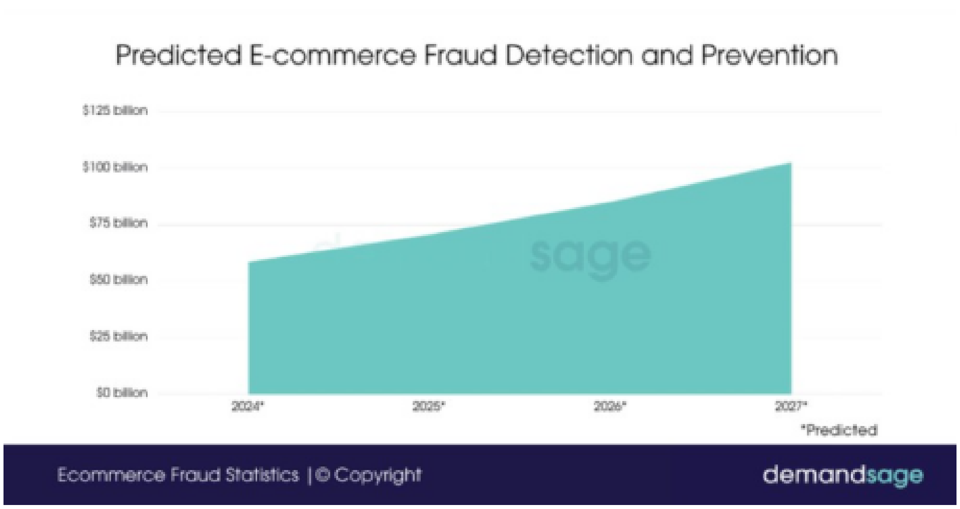
[1]: 0

[2]:
```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
img = mpimg.imread('Img/tempo.jpg')

# Tampilkan gambar
plt.imshow(img)
plt.axis('off')  # Menyembunyikan axis
plt.show()
```

The rapid growth of the e-commerce market and the increase in internet users are the main factors causing fraud in e-commerce. In 2018, the e-commerce market in Indonesia was worth USD 50 billion and is predicted to reach USD 200 billion by 2026. Internet users increased from 560 million in 2018 to 835 million in 2023, with online shoppers growing by 73%.

Based on demandsage.com The e-commerce fraud detection market is estimated to reach $ 102.28 billion by the end of 2027.

[3]:
```
img = mpimg.imread('Img/Pred.jpg')

# Tampilkan gambar
plt.imshow(img)
plt.axis('off')  # Menyembunyikan axis
plt.show()
```



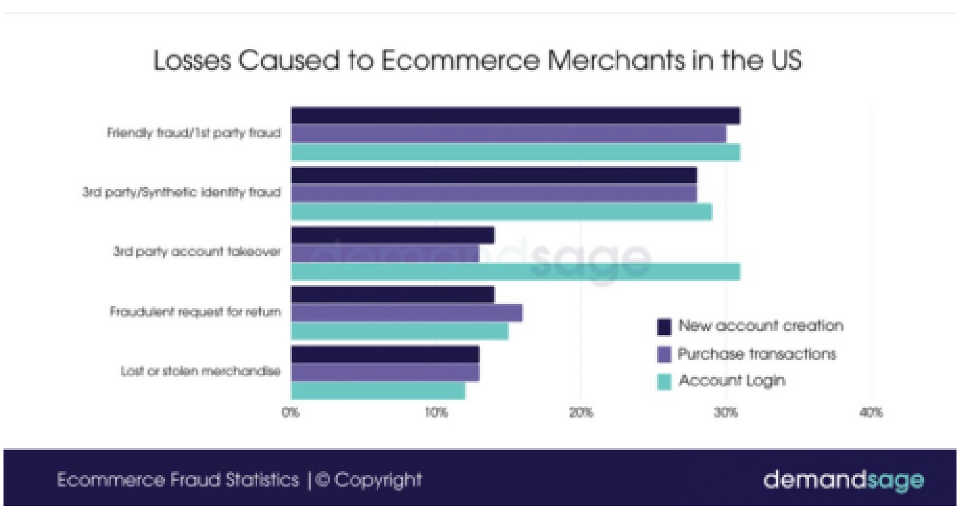
Based on year-over-year history, there has been an increase of more than 50% in attacks compared to the previous year. However, only a slight increase has been observed in successful fraud attempts.

[4]:
```
img = mpimg.imread('Img/No.jpg')

# Tampilkan gambar
plt.imshow(img)
plt.axis('off')  # Menyembunyikan axis
plt.show()
```

63% of fraud costs incurred by e-commerce merchants in the United States were attributed to domestic transactions. The remaining 37% were attributed to international fraud in the United States in 2022. This represents a significant decrease in domestic fraud compared to 71% recorded in 2021, while international fraud increased.

The largest proportion of fraud losses experienced by online merchants in the United States were due to friendly fraud.

[5]:
```
img = mpimg.imread('Img/Loss.jpg')

# Tampilkan gambar
plt.imshow(img)
plt.axis('off')  # Menyembunyikan axis
plt.show()
```



This is a practice where consumers make purchases and then ask for a refund. The second largest part is synthetic fraud, where fraudsters create fake identities for the purpose of defrauding.

Fraud di e-commerce include :

Phishing/Pharming/Whaling: Stealing confidential information such as credit card details and passwords, related to Cyber Security Fraud such as account takeovers.

Card Testing: Testing stolen credit cards for validity, part of Cyber Security Fraud.

Identity Theft: Theft of identity for fraud, including Cyber Security Fraud.

First-party Misuse: Providing false information for illegal gain, related to Buyer Fraud (false claims, denied payments) and Merchant Fraud (counterfeit goods, non-compliance).

1.1 A.1. Project Idea

In general, this project will attempt to implement several models until the best one is obtained.

- Logistic Regression: Very popular for fraud detection because its output is binary (fraud or non-fraud). This model works well for linearly separable datasets.
- Decision Trees: Provides an easily understandable interpretation in detecting patterns that lead to fraud. Can be used for binary classification (fraud or not) with splits based on transaction features.
- Random Forest: An ensemble model that combines several decision trees to improve accuracy. Useful for handling complex data and reducing the risk of overfitting.
- Support Vector Machines (SVM): Can optimally separate data to detect fraud, especially if the data is not linear. Can be used with kernels to accommodate various patterns present in the dataset.
- Gradient Boosting Machines (GBM): Such as XGBoost or LightGBM, very effective for detecting fraud patterns by addressing class imbalance in the dataset.
- K-Nearest Neighbors (KNN): Can be used to classify transactions as fraud based on proximity to similar transactions. This model is simple but can be efficient if the dataset is not too large.
- Neural Networks (Multilayer Perceptron): Can be used to detect complex non-linear patterns in fraud transactions. It is very powerful but requires more data and computation.

The models based on their purposes:

1. Classification of Fraud Campaign Classifying campaigns related to fraudulent activities based on behavioral patterns and transaction characteristics.

Recommended Models:

Logistic Regression: Suitable for binary classification (fraud vs. non-fraud) and also provides the probability of fraud detection. Random Forest: Utilizes multiple decision trees to enhance accuracy and recognize important features that influence fraud campaigns. Gradient Boosting Machines (GBM): Useful for handling complex data with non-linear relationships. Feature Processing:

Categorical Encoding: Encoding campaign categories (such as types of promotions) using methods like one-hot encoding or target encoding. Feature Engineering: Create additional features such as frequency of promotion usage or campaigns per customer.

2. Fraud Technique Analysis Analyzing fraud techniques used by perpetrators to exploit systems, such as card testing or identity theft.

Recommended Models:

Support Vector Machines (SVM): To separate hard-to-separate data linearly and detect complex fraud techniques. Neural Networks (Multilayer Perceptron): Suitable for detecting complicated non-linear patterns in fraud techniques. XGBoost: Combines several models to improve predictions for specific fraud techniques. Feature Processing:

Time Series Analysis: Analyzing transaction times and possible fraud techniques that may occur within specific periods. Anomaly Detection: Identifying fraud techniques based on anomalous patterns in the data.

3. Monitoring Fraud Trend Monitoring fraud trends over time to identify increases in suspicious activity and anticipate potential risks.

Recommended Models:

Time Series Models: Such as ARIMA or Prophet to periodically monitor changes in fraud trends. LSTM (Long Short-Term Memory): Suitable for detecting fraud patterns that depend on time. Random Forest: Can be used to monitor trend variations based on historical features. Feature Processing:

Temporal Feature Engineering: Using time-based features, such as hours, days, or months, to observe trend patterns. Clustering: Segmenting fraud trends based on specific patterns such as geographical areas or types of fraud.

4. Fraud Ring GNN (Graph Neural Network) Detecting fraud networks or fraud rings involving multiple interconnected perpetrators using graph analysis.

Recommended Models:

Graph Neural Networks (GNN): To analyze relationships between fraud perpetrators in complex networks. DeepWalk or Node2Vec: Algorithms to represent nodes (perpetrators) in a fraud network to predict their involvement in fraud networks. XGBoost: Can be used as an additional model to analyze features from graph representation. Feature Processing:

Graph Feature Engineering: Creating graph-based features, such as degree centrality and clustering coefficient to identify key perpetrators in a fraud ring. Network Analysis: Performing analysis on network structures to find fraud patterns related to actors within the fraud ring.

1.2 A.2. The problem to be solved

First-party Misuse: Fraud where individuals or organizations intentionally provide false information for illegal gain. This relates to Buyer Fraud and Merchant Fraud, involving false claims, payment denials, fake accounts, promotion abuse, and the sale of counterfeit goods.

1. First-party Fraud has a significant impact on e-commerce businesses and financial institutions, including chargebacks, revenue declines, and coupon abuse.
2. Approximately 60% of chargebacks are caused by customers themselves.
3. Retail businesses experience a 2.4% decline in annual revenue due to this fraud.
4. 73% of e-commerce companies experience coupon abuse.
5. 70% of financial institutions report losses exceeding $500,000, with 62% of those losses coming from First-party Fraud.

Developing a Machine Learning model to predict and prevent First-party Fraud in e-commerce transactions by identifying suspicious activities that may indicate fraudulent behavior.

- Feature Classification of Fraud Campaign Classifying campaigns related to fraudulent activities based on behavioral patterns and transaction characteristics.
- Feature Fraud Technique Analysis Analyzing fraud techniques used by perpetrators to exploit systems, such as card testing or identity theft.
- Feature Monitoring Fraud Trend Monitoring fraud trends over time to identify increases in suspicious activity and anticipate potential risks.
- Feature Fraud Ring GNN (Graph Neural Network) Detecting fraud networks or fraud rings involving multiple interconnected perpetrators using graph analysis.

1.3 A.3. Project Purposes

Project Purposes: Aimed at e-commerce companies and financial institutions looking to detect and prevent First-party Fraud and other frauds negatively impacting business.

Benefits and Business Impact:

Loss Reduction: Reducing the risk of chargebacks and fraud claims while protecting company revenue. Increased Consumer Trust: Protecting customers from fraud, creating a safe transaction experience. Operational Efficiency: Automating fraud detection, saving on manual monitoring costs. Effective Use of Promotions: Preventing coupon and promotion abuse, optimizing marketing strategies. Better Customer Experience: Enhancing transaction smoothness for legitimate customers.

1.4 A.4. Project Impact

- Reduction of Fraud Losses: Mitigating the financial impact of First-party Fraud through early detection and prevention.
- Increased Accuracy: Enhancing fraud detection accuracy using machine learning (ML) models, thus reducing false positives and false negatives.
- Operational Efficiency: Streamlining the fraud detection process, reducing manual intervention, and speeding up response time.
- Reduction of Chargebacks: Improving profit margins by reducing chargebacks from 60% of cases caused by customer fraud.
- Revenue Enhancement: Avoiding a 2.4% decline in annual revenue for sellers.
- Reduction of Coupon Abuse: Decreasing coupon abuse by 73%.
- Cost Savings: Reducing fraud management costs currently reaching 10% of annual revenue.
- Customer Trust: Enhancing customer reputation and loyalty.
- Financial Security: Mitigating losses for financial institutions and improving financial health.

1.5 A.5. Target User for Project

Main User

1. Fraud Prevention and Security Team
2. Risk Management Team
3. E-commerce Platform Managers
4. Financial Institutions

1.6 A.6. Project Output

The output of this project is an interactive dashboard that allows users to detect fraud more easily with 4 main Features: Classification of Fraud Campaign Feature, Fraud Technique Analysis Feature, Fraud Trend Monitoring Feature, and Fraud Ring GNN (Graph Neural Network) Feature.

1.7 A.7. Data Collection

1. Data Collection

This internal data is taken from one of the leading e-commerce platforms in Indonesia, which has been masked to protect sensitive information. This dataset contains information about transactions made by customers, used to analyze shopping behavior patterns and detect potential fraud. This data aligns with the project needs as it uses primary data from the field, which can provide more accurate results.

2. Data Description

[ ]:
```
var = pd.read_csv(r'C:\Dimas\Docs\Me\Coding\Algoritma␣
↪Bootcamp\Material\Capstone\DCD\Shopee\Clean\Test\Var.csv',␣
↪encoding='latin-1')
var
```

1.8 A.7. Data Preparation

1. Target and Predictor

Target Variable: Buyer Status
Buyer Status is a category used to classify customers based on their behavior or account status. There are four possible classes:

- Frozen: Customer accounts are temporarily frozen, usually due to suspicious activity or policy violations.
- Banned: Customer accounts are permanently banned from the platform due to serious violations.
- Delete: Customer accounts are deleted, either by the customer themselves or by the system.
- Normal: Customer accounts are in an active state and can conduct transactions.

Predictor Features
The following features are used to predict the Buyer Status category:

1. Address:
    - The residential address of the customer, which may indicate geographical location or shopping behavior patterns.

2. Recipient Phone:
    - Only the first 6 digits of the recipient's phone number. This can help identify customers based on area codes.
3. Zip Code:
    - The zip code of the customer's location, which provides further information about the geographical area where they reside.
4. City:
    - The name of the city where the customer lives. This can provide insights into customer demographics and preferences.
5. pv_voucher_activity_name:
    - The name of the voucher activity used by the customer. This can indicate how often customers use vouchers or promotions.

Reasons for Choosing Predictor Features:
These features were selected because they can provide useful insights into customer behavior and characteristics. Address, zip code, and city provide important geographical context in fraud risk analysis, while information about voucher activity can indicate potential abuse.

Development Options
- Fuzzy Wuzzy:
- This algorithm can be used for standardization or grouping of text data. For example, if there are variations in the spelling of city names or addresses, this algorithm will help align the data so that analysis can be performed more effectively.

2 B. Data Exploratory

2.1 B.1. Data Preparation

[6]:
```
import pandas as pd
import openpyxl
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import seaborn as sns
from faker import Faker
import random
import string
```

[8]:
```
data = pd.read_csv(r'C:\Dimas\Docs\Me\Coding\Algoritma␣
↪Bootcamp\Material\Capstone\DCD\Shopee\Clean\Test\Third.csv',␣
↪encoding='latin-1')
```

```
C:\Users\sendd\AppData\Local\Temp\ipykernel_22064\956667483.py:1: DtypeWarning:
Columns (39) have mixed types. Specify dtype option on import or set
low_memory=False.
  data = pd.read_csv(r'C:\Dimas\Docs\Me\Coding\Algoritma
Bootcamp\Material\Capstone\DCD\Shopee\Clean\Test\Third.csv', encoding='latin-1')
```

[8]:
```
               order_id      checkout_id        order_sn shipping_traceno  \
0      103297621228623  103297621284800  2204108N5SQ32F     JP5929909549
1      103451378688478  103451378625388  220412D4C48CEX              NaN
2      103074208315718  103074208361763  220408253JJ2A6     JP8423988280
3      103451371680118  103451371670424  220412D4BWJGBP              NaN
4      103451368345982  103451368318277  220412D4BTCRBX              NaN
…                  …                …               …              …
65137  103191844356054  103191844318773  2204095JND13XP              NaN
65138  103302777287765  103302777238703  2204108SYEWH2N              NaN
65139  103193040397746  103193040359774  2204095KS1NDDJ              NaN
65140  103354992227224  103354992291538  220411AAKJY0VR     JP2168283178
65141  103297359333463  103297359395117  2204108MWYXP2Q              NaN

       grand_total      cogs    shop_id order_fe_status  \
0           112000    104000   33255986      TO_RECEIVE
1             9900     39900  270505142          TO_PAY
2                0  50000000   72176016      TO_RECEIVE
3             3250     33250   96014510          TO_PAY
4             1125      4500    5149374          TO_PAY
…                …         …        …              …
65137        12144     15300   48382819         TO_SHIP
65138        18163     30700  200051117         TO_SHIP
65139        32128     49990  155829176         TO_SHIP
65140         7800     42900   48382819         TO_SHIP
65141         6430     41450    5241910         TO_SHIP

      order_logistics_status specific_purchased_time  … seller_latitude  \
0                PICKUP DONE     2022-04-10 20:47:02  …             NaN
1                    INVALID     2022-04-12 15:29:38  …             NaN
2                PICKUP DONE     2022-04-08 06:43:29  …             NaN
3                      READY     2022-04-12 15:29:31  …             NaN
4                      READY     2022-04-12 15:29:29  …             NaN
…                        …                     …  …             …
65137                  READY     2022-04-09 15:24:05  …             NaN
65138                INVALID     2022-04-10 22:12:59  …             NaN
65139                  READY     2022-04-09 15:44:00  …             NaN
65140        REQUEST CREATED     2022-04-11 12:43:13  …             NaN
65141                  READY     2022-04-10 20:42:40  …             NaN

      buyer_longitude  buyer_latitude sender_user_id hashed_ba  \
0                 NaN             NaN            NaN       NaN
1                 NaN             NaN            NaN       NaN
2                 NaN             NaN            NaN       NaN
3                 NaN             NaN            NaN       NaN
4                 NaN             NaN            NaN       NaN
…                   …               …            …        …
65137             NaN             NaN            NaN       NaN
```

```
65138              NaN              NaN             NaN       NaN
65139              NaN              NaN             NaN       NaN
65140              NaN              NaN             NaN       NaN
65141              NaN              NaN             NaN       NaN

       registration_phone_number                      SZ  \
0                   6.282333e+12  '1491123671126906758
1                   6.288202e+13                     NaN
2                   6.289540e+13  '1430741364721454946
3                   6.283849e+12                     NaN
4                   6.283195e+12                     NaN
…                            …                    …
65137               6.281363e+12  '1399600222766463072
65138               6.289561e+13  '1399592028554826494
65139               6.283173e+12  '1399279041844705712
65140               6.283129e+12  '1399273574223187523
65141               6.282118e+12  '1398290097585202774

                                                  Group  Count If  Unnamed: 39
0      MPRNNONB_Telkomsel_Regular_100%_'1491123671126…       1.0          NaN
1                                                   NaN       NaN          NaN
2      MGVWGRANDPRIZESHOP_'1430741364721454946_KOTA B…       1.0          NaN
3                                                   NaN       NaN          NaN
4                                                   NaN       NaN          NaN
…                                                   …       …          …
65137  MACQAFFILIATEKOLSP_'1399600222766463072_KOTA B…       1.0          NaN
65138  MACQAFFILIATEKOLSP_'1399592028554826494_KAB. K…       1.0          NaN
65139  MACQAFFILIATEKOLSP_'1399279041844705712_KAB. C…       1.0          NaN
65140  MACQAFFILIATEKOLSP_'1399273574223187523_KOTA C…       1.0          NaN
65141  MACQAFFILIATEKOLSP_'1398290097585202774_KAB. P…       1.0          NaN

[65142 rows x 40 columns]
```

[9]:
```python
# Inisialisasi Faker
fake = Faker()

# Fungsi untuk menghasilkan data palsu sesuai kolom yang dijelaskan
def generate_mock_data(num_records):
    data = []
    for _ in range(65142):
        order = {
            'order_id': fake.random_number(digits=12, fix_len=True),
            'checkout_id': fake.random_number(digits=12, fix_len=True),
            'order_sn': fake.bothify(text='######??#####'),
            'shipping_traceno': fake.bothify(text='??#########'),
            'grand_total': random.randint(50000, 500000),
            'cogs': random.randint(40000, 400000),
```

```
            'shop_id': fake.random_number(digits=8, fix_len=True),
            'order_fe_status': random.choice(['TO_RECEIVE', 'CANCELLED',␣
↪'SHIPPED']),
            'order_logistics_status': random.choice(['PICKUP DONE',␣
↪'DELIVERED', 'IN TRANSIT']),
            'specific_purchased_time': fake.date_time_this_year(),
            'Device_ID': fake.uuid4(),
            'actual_shipping_carrier': random.choice(['JNE', 'SiCepat', 'J&T',␣
↪'Other']),
            'fulfilment_channel_id': fake.random_number(digits=9, fix_len=True),
            'fulfilment_shipping_carrier': random.choice(['JNE', 'SiCepat',␣
↪'J&T', 'Other']),
            'payment_channel': random.choice(['Credit Card', 'ShopeePay', 'Bank␣
↪Transfer', 'Other']),
            'shopee_voucher_rebate': random.randint(0, 50000),
            'coin_earn': random.randint(0, 10000),
            'pv_voucher_code': fake.bothify(text='??#####'),
            'pv_voucher_activity_name': random.choice(['Discount 10%', 'Free␣
↪Shipping', 'Cashback', 'Other']),
            'buyer_status': random.choice(['Active', 'Inactive', 'Bannerd',␣
↪'Other']),
            'Username_Buyer': fake.user_name(),
            'Username_Seller': fake.user_name(),
            'Buyer_User_ID': fake.random_number(digits=12, fix_len=True),
            'Seller_User_ID': fake.random_number(digits=12, fix_len=True),
            'recipient_phone': fake.phone_number(),
            'recipient_name': fake.name(),
            'shipping_address': fake.address(),
            'shipping_city': fake.city(),
            'zip_code': fake.zipcode(),
            'buyer_longitude': fake.longitude(),
            'buyer_latitude': fake.latitude(),
            'seller_longitude': fake.longitude(),
            'seller_latitude': fake.latitude(),
            'registration_phone_number': fake.random_number(digits=12,␣
↪fix_len=True),
            'SZ': random.choice(['Zone 1', 'Zone 2', 'NaN']),
            'Group': random.choice(['Group A', 'Group B', 'NaN']),
            'Count If': random.randint(1, 10),
        }
        data.append(order)

    return pd.DataFrame(data)

# Hasilkan 100 data tiruan
df = generate_mock_data(100)
```

```
# Tampilkan 5 baris pertama
print(df.head())
```

```
        order_id   checkout_id       order_sn shipping_traceno  grand_total  \
0  910174627603  619866912134  060880TX76120      bd789553140       321731
1  642935775624  988975955969  733236vk89631      kY893312941       132701
2  259659818068  446319292924  115894wZ13605      oZ471146015       293729
3  102806747084  864069039531  880252xa16662      gH490758676       219092
4  681729113379  630462785625  8983951V03100      pi261879644        50701

      cogs   shop_id order_fe_status order_logistics_status  \
0  211800  16071196       CANCELLED              DELIVERED
1  209846  92920323         SHIPPED            PICKUP DONE
2  394816  29872902      TO_RECEIVE              DELIVERED
3  236888  10810202      TO_RECEIVE             IN TRANSIT
4  389607  10974506       CANCELLED            PICKUP DONE

  specific_purchased_time  …  shipping_city zip_code  buyer_longitude  \
0     2024-08-21 22:34:25  …      South Amy    81819      -142.180155
1     2024-06-06 20:56:09  …   Kellychester    06193        67.390467
2     2024-08-29 07:42:51  …   New Lisatown    00510      -165.039721
3     2024-08-13 19:50:00  …    Carterhaven    10749        63.096943
4     2024-01-12 14:52:37  …  South Anthony    99505       133.545406

  buyer_latitude seller_longitude  seller_latitude  registration_phone_number  \
0     -0.5721955        21.659943        68.706604               375125555285
1    -15.7984555      -106.800778        88.647712               886472466165
2     -34.807890       155.648445        58.587768               635301248005
3      82.321833      -165.870976       -84.755036               752508444727
4     -53.454469      -152.538938       -18.653815               924823462446

       SZ    Group Count If
0     NaN  Group B        6
1     NaN  Group A        6
2  Zone 2  Group A        6
3     NaN  Group A        6
4  Zone 2  Group A        3

[5 rows x 37 columns]
```

3 B.2. Data Exploratory Analysis

3.1 B2.1. Data Types

[11]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 65142 entries, 0 to 65141
Data columns (total 40 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   order_id                     65142 non-null  int64  
 1   checkout_id                  65142 non-null  int64  
 2   order_sn                     65142 non-null  object 
 3   shipping_traceno             36761 non-null  object 
 4   grand_total                  65142 non-null  int64  
 5   cogs                         65142 non-null  int64  
 6   shop_id                      65142 non-null  int64  
 7   order_fe_status              65142 non-null  object 
 8   order_logistics_status       65142 non-null  object 
 9   specific_purchased_time      65142 non-null  object 
 10  Device_ID                    58593 non-null  object 
 11  actual_shipping_carrier      65142 non-null  object 
 12  fulfilment_channel_id        65142 non-null  int64  
 13  fulfilment_shipping_carrier  65142 non-null  object 
 14  payment_channel              65142 non-null  object 
 15  shopee_voucher_rebate        65142 non-null  int64  
 16  coin_earn                    65142 non-null  int64  
 17  pv_voucher_code              65142 non-null  object 
 18  pv_voucher_activity_name     65142 non-null  object 
 19  buyer_status                 65142 non-null  object 
 20  Username_Buyer               65142 non-null  object 
 21  Username_Seller              65142 non-null  object 
 22  Buyer_User_ID                65142 non-null  int64  
 23  Seller_User_ID               65142 non-null  int64  
 24  recipient_phone              65142 non-null  int64  
 25  recipient_name               65142 non-null  object 
 26  shipping_address             65142 non-null  object 
 27  shipping_city                65142 non-null  object 
 28  zip_code                     65120 non-null  float64
 29  seller_longitude             406 non-null    float64
 30  seller_latitude              406 non-null    float64
 31  buyer_longitude              403 non-null    float64
 32  buyer_latitude               403 non-null    float64
 33  sender_user_id               0 non-null      float64
 34  hashed_ba                    0 non-null      float64
 35  registration_phone_number    60219 non-null  float64
 36  SZ                           62538 non-null  object 
 37  Group                        62538 non-null  object 
 38  Count If                     62491 non-null  float64
 39  Unnamed: 39                  8 non-null      object 
dtypes: float64(9), int64(11), object(20)
memory usage: 19.9+ MB
```

[12]:
```
data.describe()
```

[12]:
```
           order_id   checkout_id   grand_total          cogs       shop_id  \
count  6.514200e+04  6.514200e+04  6.514200e+04  6.514200e+04  6.514200e+04
mean   1.032190e+14  1.032190e+14  4.288170e+04  6.429180e+04  1.797163e+08
std    1.383691e+11  1.383686e+11  1.493479e+05  2.454895e+05  1.563214e+08
min    1.029636e+14  1.029636e+14  0.000000e+00  0.000000e+00  1.057000e+04
25%    1.031004e+14  1.031004e+14  7.500000e+03  3.125000e+04  4.351408e+07
50%    1.032174e+14  1.032174e+14  2.200000e+04  4.500000e+04  1.435765e+08
75%    1.033453e+14  1.033453e+14  4.900000e+04  7.000000e+04  2.814130e+08
max    1.034514e+14  1.034514e+14  2.357239e+07  5.000000e+07  7.315251e+08

       fulfilment_channel_id  shopee_voucher_rebate     coin_earn  \
count           65142.000000           65142.000000  65142.000000
mean            76399.849283           24755.629839     25.453670
std             16069.808922           11335.873985    583.911091
min              8003.000000               0.000000      0.000000
25%             80014.000000           15181.000000      0.000000
50%             80014.000000           30000.000000      0.000000
75%             80030.000000           30000.000000      0.000000
max             88020.000000           75000.000000  40000.000000

       Buyer_User_ID  Seller_User_ID  recipient_phone      zip_code  \
count   6.514200e+04    6.514200e+04     6.514200e+04  65120.000000
mean    7.388930e+08    1.797225e+08     9.512709e+12  47183.320224
std     8.524980e+06    1.563280e+08     1.363616e+13  22635.662344
min     7.000054e+08    1.057000e+04     8.216779e+09  10110.000000
25%     7.388269e+08    4.351547e+07     6.282118e+12  28825.000000
50%     7.418412e+08    1.435784e+08     6.283878e+12  45252.000000
75%     7.434966e+08    2.814197e+08     6.285876e+12  62311.500000
max     7.456823e+08    7.315447e+08     6.289976e+13  99962.000000

       seller_longitude  seller_latitude  buyer_longitude  buyer_latitude  \
count        406.000000       406.000000       403.000000      403.000000
mean         114.822660        23.687192       108.146402       -4.215881
std            2.271254         3.355950         5.500421        3.840158
min          105.000000        -7.000000        95.000000       -9.000000
25%          114.000000        23.000000       105.000000       -7.000000
50%          114.000000        23.000000       108.000000       -6.000000
75%          114.000000        23.000000       111.000000       -1.000000
max          121.000000        31.000000       131.000000        6.000000

       sender_user_id  hashed_ba  registration_phone_number      Count If
count             0.0        0.0               6.021900e+04  62491.000000
mean              NaN        NaN               9.717997e+12      1.018179
std               NaN        NaN               1.389865e+13      0.176891
min               NaN        NaN               1.825425e+10      1.000000
```

```
25%                NaN        NaN              6.282115e+12       1.000000
50%                NaN        NaN              6.283892e+12       1.000000
75%                NaN        NaN              6.285893e+12       1.000000
max                NaN        NaN              6.289618e+13       6.000000
```

[13]:
```
data['order_id'] = data['order_id'].astype('Int64')
data['checkout_id'] = data['checkout_id'].astype('Int64')
data['grand_total'] = data['grand_total'].astype('Int64')
data['cogs'] = data['cogs'].astype('Int64')
data['shop_id'] = data['shop_id'].astype('Int64')
data['fulfilment_channel_id'] = data['fulfilment_channel_id'].astype('Int64')
data['Buyer_User_ID'] = data['Buyer_User_ID'].astype('Int64')
data['Seller_User_ID'] = data['Seller_User_ID'].astype('Int64')
data['recipient_phone'] = data['recipient_phone'].astype('Int64')
data['Count If'] = data['Count If'].astype('Int64')
# Tangani NaN sebelum mengubah tipe data
data['registration_phone_number'] = pd.
 ↪to_numeric(data['registration_phone_number'], errors='coerce').
 ↪astype('Int64')

# Ubah kolom waktu menjadi datetime
data['specific_purchased_time'] = pd.
 ↪to_datetime(data['specific_purchased_time'], format='%Y-%m-%d %H:%M:%S')

# Ubah kolom float menjadi integer jika tidak ada nilai pecahan
data['zip_code'] = pd.to_numeric(data['zip_code'], errors='coerce').
 ↪astype('Int64')
```

3.2 B2.2. Missing Value

[14]:
```
data.isna().sum()
```

[14]:
```
order_id                         0
checkout_id                      0
order_sn                         0
shipping_traceno             28381
grand_total                      0
cogs                             0
shop_id                          0
order_fe_status                  0
order_logistics_status           0
specific_purchased_time          0
Device_ID                     6549
actual_shipping_carrier          0
fulfilment_channel_id            0
fulfilment_shipping_carrier      0
payment_channel                  0
```

```
shopee_voucher_rebate              0
coin_earn                          0
pv_voucher_code                    0
pv_voucher_activity_name           0
buyer_status                       0
Username_Buyer                     0
Username_Seller                    0
Buyer_User_ID                      0
Seller_User_ID                     0
recipient_phone                    0
recipient_name                     0
shipping_address                   0
shipping_city                      0
zip_code                          22
seller_longitude               64736
seller_latitude                64736
buyer_longitude                64739
buyer_latitude                 64739
sender_user_id                 65142
hashed_ba                      65142
registration_phone_number       4923
SZ                              2604
Group                           2604
Count If                        2651
Unnamed: 39                    65134
dtype: int64
```

[15]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65142 entries, 0 to 65141
Data columns (total 40 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   order_id                    65142 non-null  Int64
 1   checkout_id                 65142 non-null  Int64
 2   order_sn                    65142 non-null  object
 3   shipping_traceno            36761 non-null  object
 4   grand_total                 65142 non-null  Int64
 5   cogs                        65142 non-null  Int64
 6   shop_id                     65142 non-null  Int64
 7   order_fe_status             65142 non-null  object
 8   order_logistics_status      65142 non-null  object
 9   specific_purchased_time     65142 non-null  datetime64[ns]
 10  Device_ID                   58593 non-null  object
 11  actual_shipping_carrier     65142 non-null  object
 12  fulfilment_channel_id       65142 non-null  Int64
 13  fulfilment_shipping_carrier 65142 non-null  object
```

```
 14   payment_channel             65142 non-null  object
 15   shopee_voucher_rebate       65142 non-null  int64
 16   coin_earn                   65142 non-null  int64
 17   pv_voucher_code             65142 non-null  object
 18   pv_voucher_activity_name    65142 non-null  object
 19   buyer_status                65142 non-null  object
 20   Username_Buyer              65142 non-null  object
 21   Username_Seller             65142 non-null  object
 22   Buyer_User_ID               65142 non-null  Int64
 23   Seller_User_ID              65142 non-null  Int64
 24   recipient_phone             65142 non-null  Int64
 25   recipient_name              65142 non-null  object
 26   shipping_address            65142 non-null  object
 27   shipping_city               65142 non-null  object
 28   zip_code                    65120 non-null  Int64
 29   seller_longitude            406 non-null    float64
 30   seller_latitude             406 non-null    float64
 31   buyer_longitude             403 non-null    float64
 32   buyer_latitude              403 non-null    float64
 33   sender_user_id              0 non-null      float64
 34   hashed_ba                   0 non-null      float64
 35   registration_phone_number   60219 non-null  Int64
 36   SZ                          62538 non-null  object
 37   Group                       62538 non-null  object
 38   Count If                    62491 non-null  Int64
 39   Unnamed: 39                 8 non-null      object
dtypes: Int64(12), datetime64[ns](1), float64(6), int64(2), object(19)
memory usage: 20.6+ MB
```

3.3 B2.3. Data Wrangling

[16]:
```python
data.drop(columns=['seller_latitude', 'seller_longitude', 'buyer_longitude',
 'buyer_latitude', 'hashed_ba', 'Unnamed: 39', 'sender_user_id' ],
 inplace=True)
```

[16]:
```
          order_id      checkout_id         order_sn shipping_traceno  \
0  103297621228623  103297621284800  2204108N5SQ32F     JP5929909549
1  103451378688478  103451378625388  220412D4C48CEX              NaN
2  103074208315718  103074208361763  220408253JJ2A6     JP8423988280
3  103451371680118  103451371670424  220412D4BWJGBP              NaN
4  103451368345982  103451368318277  220412D4BTCRBX              NaN

   grand_total      cogs    shop_id order_fe_status order_logistics_status  \
0       112000    104000   33255986      TO_RECEIVE            PICKUP DONE
1         9900     39900  270505142          TO_PAY                INVALID
2            0  50000000   72176016      TO_RECEIVE            PICKUP DONE
3         3250     33250   96014510          TO_PAY                  READY
```

```
4         1125      4500     5149374          TO_PAY                 READY

  specific_purchased_time   … Seller_User_ID recipient_phone  recipient_name  \
0     2022-04-10 20:47:02   …       33257370   6282333044826          Aldiinr
1     2022-04-12 15:29:38   …      270509125  62882016723744     Nanda gustii
2     2022-04-08 06:43:29   …       72177477   6281240268046     Sabina/Hasan
3     2022-04-12 15:29:31   …       96015988   6283849084556         Rizal Lm
4     2022-04-12 15:29:29   …        5150668   6283195435211            Yulia

                                 shipping_address    shipping_city  \
0  Btn Lepo-Lepo Indah, RT.1/RW.1, Kel Wundudopi,…     KOTA KENDARI
1  Jl Marelan VI,psr 2 timur depan kolam renang T…       KOTA MEDAN
2  Jalan Leynan Umar Baki No. 137 ( Depan SIT Alf…      KOTA BINJAI
3                     Kedai Lombok (Rumah tingkat)  KOTA YOGYAKARTA
4  Toko pakaian KHANZA FASHION STORE deretan lari…    KAB. SEMARANG

  zip_code  registration_phone_number                    SZ  \
0    93116              6282333044826  '1491123671126906758
1    20256             62882016723744                   NaN
2    20718             62895401889327  '1430741364721454946
3    55253              6283849084556                   NaN
4    50611              6283195435211                   NaN

                                               Group Count If
0  MPRNNONB_Telkomsel_Regular_100%_'1491123671126…        1
1                                                NaN     <NA>
2  MGVWGRANDPRIZESHOP_'1430741364721454946_KOTA B…        1
3                                                NaN     <NA>
4                                                NaN     <NA>

[5 rows x 33 columns]
```

3.4 B2.4. Masking Data

[17]:
```python
def mask_value(val):
    if pd.isna(val):  # Periksa jika nilai NaN
        return val
    val = str(val)  # Pastikan semua nilai diubah ke string
    if len(val) > 4:
        return '****' + val[4:]  # Ganti 4 karakter pertama dengan '****'
    else:
        return '****'

# Kolom yang akan dimasking
cols_to_mask = [
    'order_id', 'checkout_id', 'order_sn', 'shipping_traceno', 'Device_ID',
 ↪'Username_Buyer',
```

```python
    'Username_Seller', 'Buyer_User_ID', 'Seller_User_ID', 'recipient_phone',␣
↪'recipient_name',
    'shipping_address', 'shipping_city', 'zip_code', 'buyer_longitude',␣
↪'buyer_latitude',
    'seller_longitude', 'seller_latitude', 'registration_phone_number', 'cogs',␣
↪'grand_total', 'SZ', 'shop_id'
]

# Terapkan masking pada kolom yang membutuhkan
for col in cols_to_mask:
    if col in data.columns:  # Periksa apakah kolom ada di DataFrame
        if data[col].dtype == 'object' or data[col].dtype == 'Int64':  # Tipe␣
↪data object atau Int64
            data[col] = data[col].apply(mask_value)
        elif data[col].dtype == 'float64':  # Tipe data float64
            data[col] = data[col].astype('str').apply(mask_value)
```

```
             order_id      checkout_id        order_sn shipping_traceno  \
0      ****97621228623  ****97621284800  ****108N5SQ32F     ****29909549
1      ****51378688478  ****51378625388  ****12D4C48CEX              NaN
2      ****74208315718  ****74208361763  ****08253JJ2A6     ****23988280
3      ****51371680118  ****51371670424  ****12D4BWJGBP              NaN
4      ****51368345982  ****51368318277  ****12D4BTCRBX              NaN
…                  …                …               …                …
65137  ****91844356054  ****91844318773  ****095JND13XP              NaN
65138  ****02777287765  ****02777238703  ****108SYEWH2N              NaN
65139  ****93040397746  ****93040359774  ****095KS1NDDJ              NaN
65140  ****54992227224  ****54992291538  ****11AAKJY0VR     ****68283178
65141  ****97359333463  ****97359395117  ****108MWYXP2Q              NaN

      grand_total      cogs    shop_id order_fe_status order_logistics_status  \
0          ****00    ****00    ****5986      TO_RECEIVE            PICKUP DONE
1            ****     ****0   ****05142          TO_PAY                INVALID
2            ****  ****0000    ****6016      TO_RECEIVE            PICKUP DONE
3            ****     ****0    ****4510          TO_PAY                  READY
4            ****      ****     ****374          TO_PAY                  READY
…             …        …          …               …                     …
65137       ****4     ****0    ****2819         TO_SHIP                  READY
65138       ****3     ****0   ****51117         TO_SHIP                INVALID
65139       ****8     ****0   ****29176         TO_SHIP                  READY
65140        ****     ****0    ****2819         TO_SHIP        REQUEST CREATED
65141        ****     ****0     ****910         TO_SHIP                  READY

      specific_purchased_time  … Seller_User_ID recipient_phone  \
0         2022-04-10 20:47:02  …       ****7370    ****333044826
1         2022-04-12 15:29:38  …      ****09125   ****2016723744
2         2022-04-08 06:43:29  …       ****7477    ****240268046
```

```
3        2022-04-12 15:29:31  …       ****5988   ****849084556
4        2022-04-12 15:29:29  …        ****668   ****195435211
…                        …  …          …             …
65137    2022-04-09 15:24:05  …       ****4207   ****363369279
65138    2022-04-10 22:12:59  …      ****54239   ****275818071
65139    2022-04-09 15:44:00  …      ****31118   ****173341631
65140    2022-04-11 12:43:13  …       ****4207   ****129013761
65141    2022-04-10 20:42:40  …        ****204   ****118254547

                  recipient_name  \
0                        ****inr
1                   ****a gustii
2                   ****na/Hasan
3                       ****l Lm
4                          ****a
…                            …
65137                      ****i
65138            **** Nurhatijah
65139          **** DURIYA/RAJAM
65140      ****fah Noor Fitirani
65141             ****ri Apriliani

                                                   shipping_address  \
0      ****Lepo-Lepo Indah, RT.1/RW.1, Kel Wundudopi,…
1      ****arelan VI,psr 2 timur depan kolam renang T…
2      ****n Leynan Umar Baki No. 137 ( Depan SIT Alf…
3                           ****i Lombok (Rumah tingkat)
4      **** pakaian KHANZA FASHION STORE deretan lari…
…                                                   …
65137  ****ing Saguba Asri, Blok d No.132, RT.2/RW.17…
65138  ****esmas Alah Air, Jalan Puskesmas, Desa Alah…
65139                      **** KAPETAKAN KIDUL DEWI CELL
65140  ****Encep Kartawiria, Gg. Amil, Rt 02/Rw 07, C…
65141  ****n Pangolahan, RT.8/RW.4, Desa Karangmulya,…

                  shipping_city  zip_code  registration_phone_number  \
0                  **** KENDARI    ****6.0            ****333044826.0
1                    **** MEDAN    ****6.0           ****2016723744.0
2                   **** BINJAI    ****8.0           ****5401889327.0
3               **** YOGYAKARTA    ****3.0            ****849084556.0
4                 **** SEMARANG    ****1.0            ****195435211.0
…                           …        …                        …
65137                **** BATAM    ****9.0            ****363369279.0
65138  **** KEPULAUAN MERANTI    ****3.0           ****5614031787.0
65139              **** CIREBON    ****2.0            ****173341631.0
65140               **** CIMAHI    ****2.0            ****129013761.0
65141          **** PANGANDARAN    ****7.0            ****118254547.0
```

```
                           SZ  \
0      ****1123671126906758
1                       NaN
2      ****0741364721454946
3                       NaN
4                       NaN
…                       …
65137  ****9600222766463072
65138  ****9592028554826494
65139  ****9279041844705712
65140  ****9273574223187523
65141  ****8290097585202774

                                                  Group Count If
0      MPRNNONB_Telkomsel_Regular_100%_'1491123671126…        1
1                                                   NaN     <NA>
2      MGVWGRANDPRIZESHOP_'1430741364721454946_KOTA B…        1
3                                                   NaN     <NA>
4                                                   NaN     <NA>
…                                                   …       …
65137  MACQAFFILIATEKOLSP_'1399600222766463072_KOTA B…        1
65138  MACQAFFILIATEKOLSP_'1399592028554826494_KAB. K…        1
65139  MACQAFFILIATEKOLSP_'1399279041844705712_KAB. C…        1
65140  MACQAFFILIATEKOLSP_'1399273574223187523_KOTA C…        1
65141  MACQAFFILIATEKOLSP_'1398290097585202774_KAB. P…        1

[65142 rows x 33 columns]
```

[18]:
```
data.head()
```

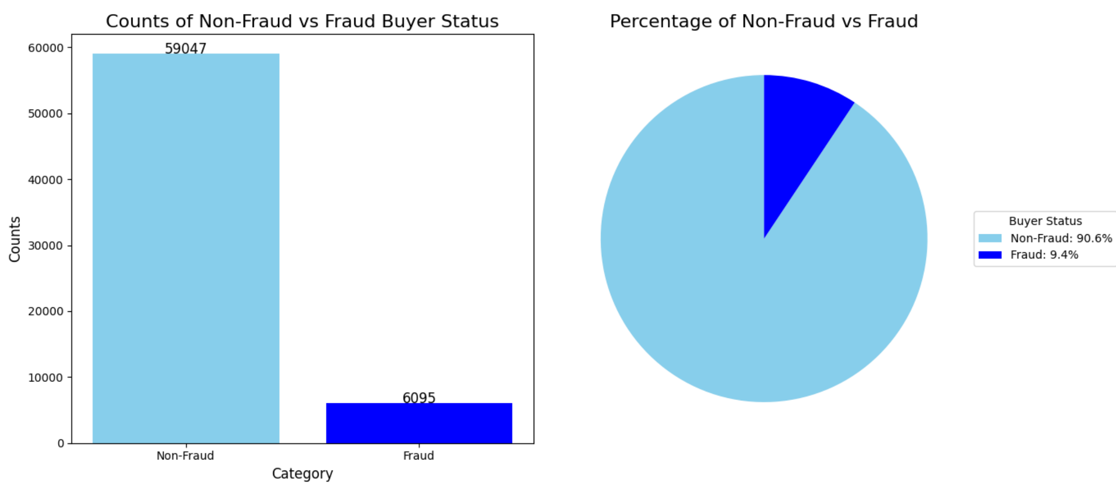
[18]:
```
          order_id      checkout_id        order_sn shipping_traceno  \
0  ****97621228623  ****97621284800  ****108N5SQ32F     ****29909549
1  ****51378688478  ****51378625388  ****12D4C48CEX              NaN
2  ****74208315718  ****74208361763  ****08253JJ2A6     ****23988280
3  ****51371680118  ****51371670424  ****12D4BWJGBP              NaN
4  ****51368345982  ****51368318277  ****12D4BTCRBX              NaN

  grand_total      cogs    shop_id order_fe_status order_logistics_status  \
0      ****00    ****00  ****5986      TO_RECEIVE            PICKUP DONE
1        ****     ****0 ****05142          TO_PAY                INVALID
2        ****  ****0000  ****6016      TO_RECEIVE            PICKUP DONE
3        ****     ****0  ****4510          TO_PAY                  READY
4        ****      ****   ****374          TO_PAY                  READY

  specific_purchased_time  … Seller_User_ID recipient_phone recipient_name  \
0     2022-04-10 20:47:02  …       ****7370    ****333044826        ****inr
1     2022-04-12 15:29:38  …      ****09125   ****2016723744    ****a gustii
```

```
2     2022-04-08 06:43:29  …       ****7477   ****240268046    ****na/Hasan
3     2022-04-12 15:29:31  …       ****5988   ****849084556        ****l Lm
4     2022-04-12 15:29:29  …        ****668   ****195435211           ****a

                                    shipping_address    shipping_city  \
0  ****Lepo-Lepo Indah, RT.1/RW.1, Kel Wundudopi,…     **** KENDARI
1  ****arelan VI,psr 2 timur depan kolam renang T…       **** MEDAN
2  ****n Leynan Umar Baki No. 137 ( Depan SIT Alf…      **** BINJAI
3                     ****i Lombok (Rumah tingkat)   **** YOGYAKARTA
4  **** pakaian KHANZA FASHION STORE deretan lari…    **** SEMARANG

   zip_code  registration_phone_number                    SZ  \
0   ****6.0            ****333044826.0  ****1123671126906758
1   ****6.0           ****2016723744.0                   NaN
2   ****8.0           ****5401889327.0  ****0741364721454946
3   ****3.0            ****849084556.0                   NaN
4   ****1.0            ****195435211.0                   NaN

                                           Group Count If
0  MPRNNONB_Telkomsel_Regular_100%_'1491123671126…        1
1                                               NaN     <NA>
2  MGVWGRANDPRIZESHOP_'1430741364721454946_KOTA B…        1
3                                               NaN     <NA>
4                                               NaN     <NA>

[5 rows x 33 columns]
```

[118]:

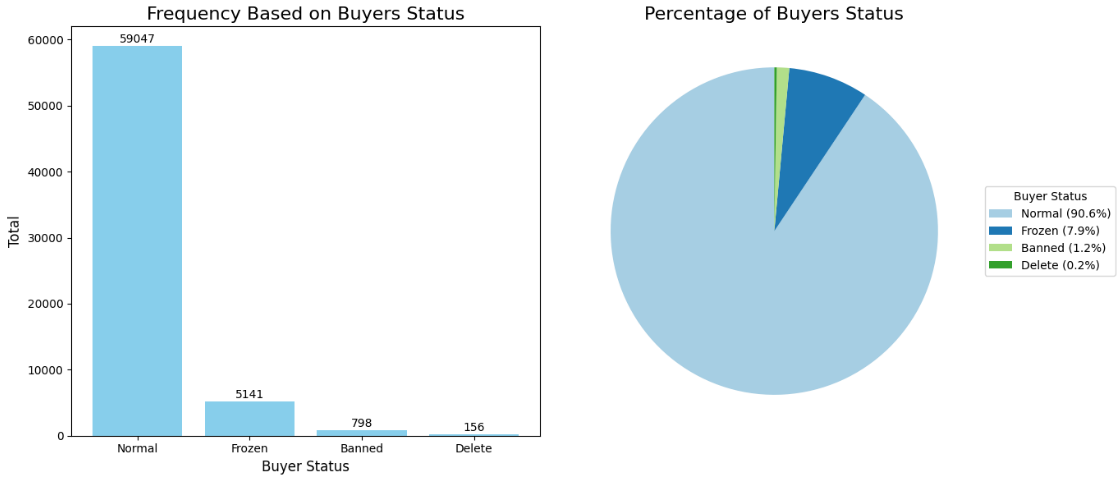
```python
# Menghitung jumlah kemunculan setiap nilai unik di kolom 'buyer_status'
unique_values = data['buyer_status'].value_counts()

# Mengkategorikan 'buyer_status' menjadi 'Non-Fraud' jika 'Normal', dan 'Fraud'␣
↪jika tidak
data['Fraud_Status'] = data['buyer_status'].apply(lambda x: 'Non-Fraud' if x ==␣
↪'Normal' else 'Fraud')

# Menghitung jumlah masing-masing kategori
fraud_status_counts = data['Fraud_Status'].value_counts()

# Menghitung persentase untuk legend
fraud_status_percentages = (fraud_status_counts / fraud_status_counts.sum() *␣
↪100).round(1).astype(str) + '%'

# Membuat Bar Plot
plt.figure(figsize=(14, 6))

# Plot Bar di subplot 1
plt.subplot(1, 2, 1)
```

```python
plt.bar(fraud_status_counts.index, fraud_status_counts.values,
    color=['skyblue', 'blue'])
plt.title('Counts of Non-Fraud vs Fraud Buyer Status', fontsize=16)
plt.xlabel('Category', fontsize=12)
plt.ylabel('Counts', fontsize=12)

# Menambahkan angka di atas setiap bar
for index, value in enumerate(fraud_status_counts.values):
    plt.text(index, value + 0.5, str(value), ha='center', fontsize=12)

# Membuat Pie Chart di subplot 2
plt.subplot(1, 2, 2)
# Menghilangkan angka persentase dari pie chart
plt.pie(fraud_status_counts.values, startangle=90, colors=['skyblue', 'blue'])

# Menambahkan legend dengan kategori dan persentase
legend_labels = [f'{status}: {percentage}' for status, percentage in
    zip(fraud_status_counts.index, fraud_status_percentages)]
plt.legend(legend_labels, title="Buyer Status", loc="center left",
    bbox_to_anchor=(1, 0, 0.5, 1))
plt.title('Percentage of Non-Fraud vs Fraud', fontsize=16)

# Menampilkan plot
plt.tight_layout()
plt.show()
```



There are significantly more non-fraud cases (59,047) compared to fraud cases (6,095).90.6% of the cases are non-fraud, while 9.4% are fraud.The data indicates that the majority of the analyzed transactions or activities are non-fraudulent. Although the number of fraud cases is lower, the 9.4% percentage indicates that the risk of fraud remains significant.

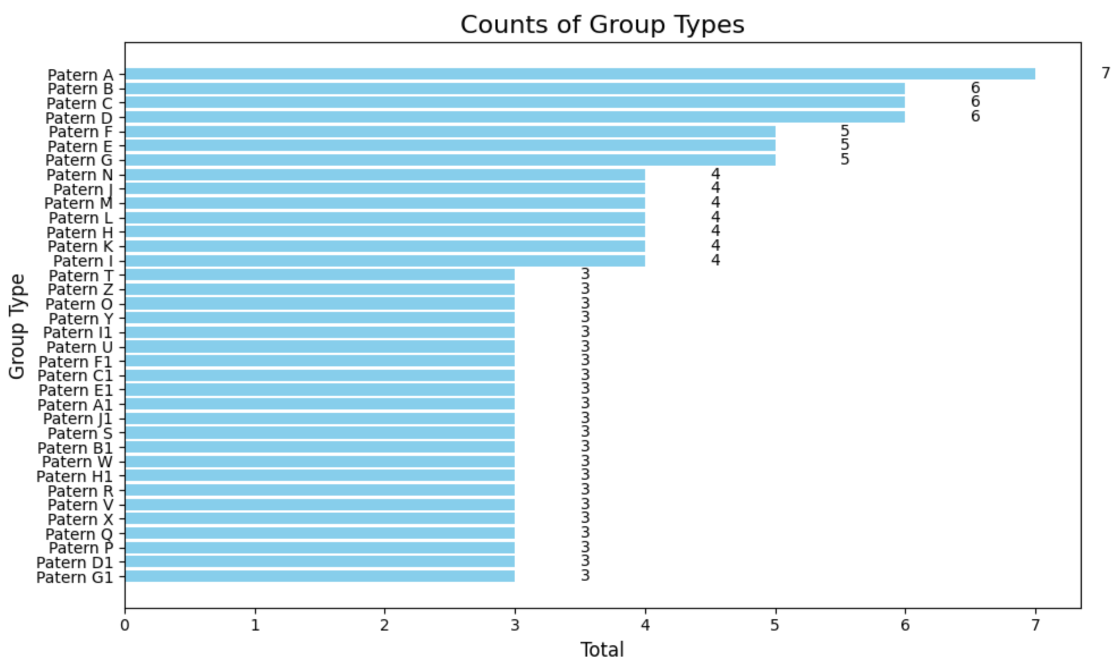
[120]:
```python
# Membuat figure dengan dua subplot: satu untuk bar plot, satu untuk pie chart
plt.figure(figsize=(14, 6))

# Membuat bar plot di subplot pertama
plt.subplot(1, 2, 1)
plt.bar(unique_values.index, unique_values.values, color='skyblue')
plt.title('Frequency Based on Buyers Status', fontsize=16)
plt.xlabel('Buyer Status', fontsize=12)
plt.ylabel('Total', fontsize=12)

# Menambahkan angka di atas setiap bar
for i, value in enumerate(unique_values.values):
    plt.text(i, value + 500, str(value), ha='center', fontsize=10)

# Membuat pie chart di subplot kedua
plt.subplot(1, 2, 2)
# Menghilangkan label di pie chart
wedges = plt.pie(unique_values.values, startangle=90, colors=plt.cm.Paired.
 ↪colors)

# Menambahkan legend dengan persentase
plt.legend(wedges[0],
           [f'{label} ({count / sum(unique_values.values) * 100:.1f}%)' for␣
 ↪label, count in zip(unique_values.index, unique_values.values)],
           title="Buyer Status", loc="center left", bbox_to_anchor=(1, 0, 0.5,␣
 ↪1))

plt.title('Percentage of Buyers Status', fontsize=16)

# Menampilkan kedua plot
plt.tight_layout()
plt.show()
```



The analysis shows that the majority of buyers maintain a Normal status, totaling 59,047, which constitutes 90.6% of all accounts, followed by Frozen at 5,141 (7.9%), Banned at 798 (1.2%), and Delete at 156 (0.2%). This indicates that most purchasing activities are running smoothly; however, the presence of Frozen and Banned cases highlights significant security risks, while the Delete status reflects that some buyers have chosen to remove their accounts. These findings suggest that the company must enhance its security measures to mitigate risks and address the reasons behind account deletions.

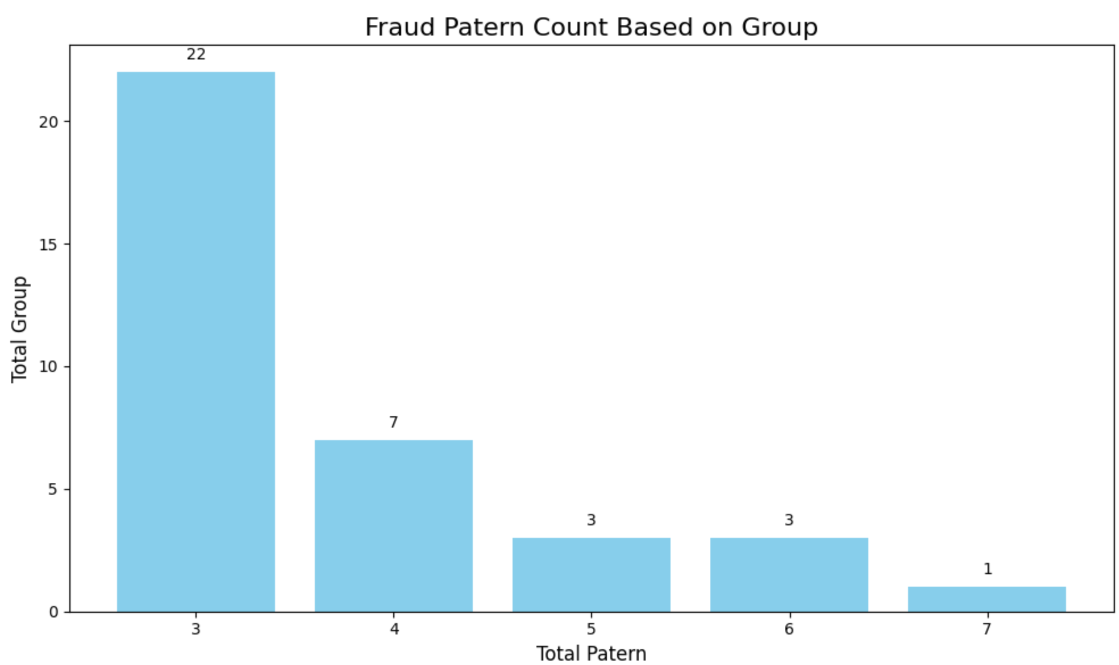
[123]:
```
# Menghitung jumlah kemunculan setiap nilai di kolom 'Group'
group_counts = data['Group'].value_counts()

# Membuat mapping dari indeks unique ke alfabet dengan menambahkan kata 'Group'
def index_to_alphabet(index):
    if index < 26:
        return f"Patern {string.ascii_uppercase[index]}"  # Mengambil huruf␣
 ↪sesuai urutan alfabet
    else:
        return f"Patern {string.ascii_uppercase[index % 26]}{index // 26}"

# Membuat mapping dari nilai group ke huruf berdasarkan urutan munculnya
group_mapping = {}
index = 0
for group, count in group_counts.items():
    # Hanya buat mapping jika count lebih dari 2, jika tidak kelompokkan␣
 ↪sebagai 'Normal'
    if count > 2:
        group_mapping[group] = index_to_alphabet(index)
        index += 1
    else:
        group_mapping[group] = 'Normal'

# Membuat kolom baru 'Group Type' berdasarkan mapping
data['Group Type'] = data['Group'].map(group_mapping)

# Menghitung jumlah kemunculan setiap nilai di kolom 'Group Type', kecuali␣
 ↪'Normal'
group_type_counts = data['Group Type'].value_counts()
group_type_counts = group_type_counts[group_type_counts.index != 'Normal']
group_type_counts = group_type_counts.sort_values(ascending=True)

# Membuat DataFrame baru dari group_type_counts
group_type_df = group_type_counts.reset_index()
group_type_df.columns = ['Group Type', 'Total']
# Membuat bar plot vertikal dari DataFrame group_type_df
plt.figure(figsize=(10, 6))
```

```python
plt.barh(group_type_df['Group Type'], group_type_df['Total'], color='skyblue')

# Menambahkan judul dan label
plt.title('Counts of Group Types', fontsize=16)
plt.xlabel('Total', fontsize=12)
plt.ylabel('Group Type', fontsize=12)

# Menambahkan angka di samping setiap bar
for index, value in enumerate(group_type_df['Total']):
    plt.text(value + 0.5, index, str(value), va='center', fontsize=10)

# Menampilkan plot
plt.tight_layout()
plt.show()
```



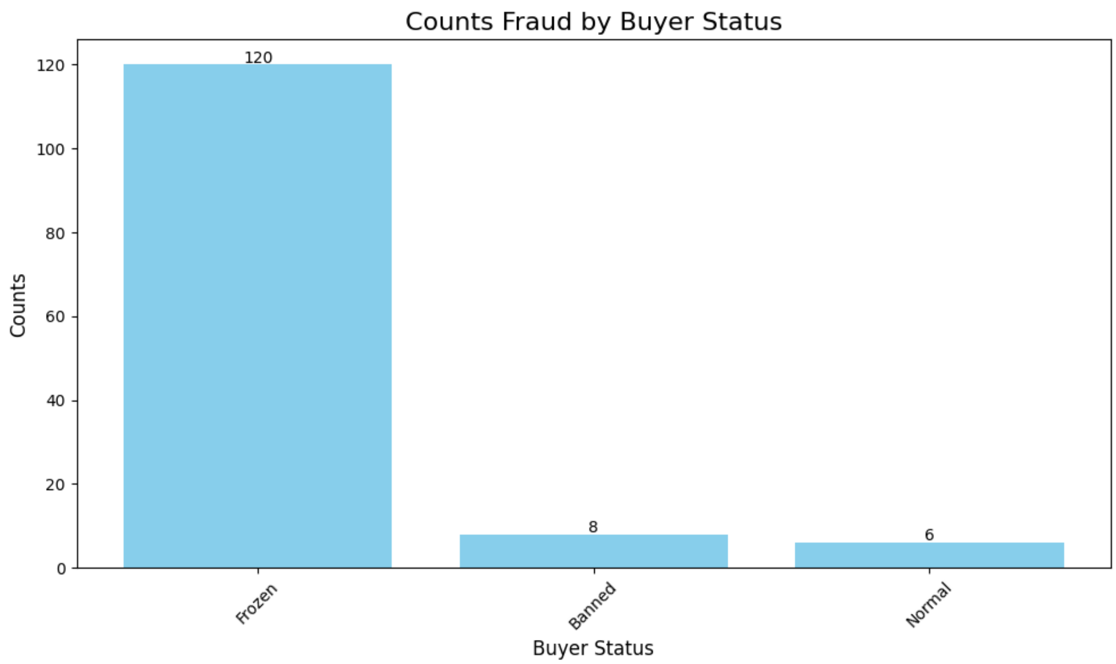
The analysis of fraud patterns indicates that the “A” and “B” types significantly dominate in frequency compared to other types, with most remaining group types showing low frequencies, often below five. This asymmetric distribution, characterized by a longer right tail, suggests that “A” and “B” may represent larger populations with more common characteristics, while the other types are minority groups. These findings underscore the need for further investigation into the dominance of “A” and “B,” as well as an exploration of the unique traits of minority groups. Additionally, segmenting the data by group type could enhance understanding of specific patterns and inform decisions in marketing, product development, and resource management.

[129]:
```python
# Menghitung jumlah grup berdasarkan total
total_group_counts = group_type_df['Total'].value_counts().sort_index()

# Membuat DataFrame dari total_group_counts
total_group_df = total_group_counts.reset_index()
total_group_df.columns = ['Total', 'Group Count']

# Membuat bar plot
plt.figure(figsize=(10, 6))
plt.bar(total_group_df['Total'], total_group_df['Group Count'], color='skyblue')

# Menambahkan judul dan label sumbu
plt.title('Fraud Patern Count Based on Group', fontsize=16)
plt.xlabel('Total Patern', fontsize=12)
plt.ylabel('Total Group', fontsize=12)
plt.xticks(total_group_df['Total'])  # Menampilkan semua total pada sumbu x

# Menambahkan angka di atas setiap bar
for index, value in enumerate(total_group_df['Group Count']):
    plt.text(total_group_df['Total'][index], value + 0.5, str(value),␣
 ↪ha='center', fontsize=10)

plt.tight_layout()  # Menyesuaikan layout agar tidak terpotong
plt.show()
```



The fraud pattern analysis reveals that pattern 3 is the most prevalent, occurring more frequently than other patterns, while most incidents are concentrated in specific patterns (3, 4, 5, and 6). This suggests that pattern 3 likely represents the most common or easiest type of fraud to commit. The variety of patterns indicates that fraudsters use different strategies, with the less frequent patterns possibly representing more complex or specific fraud types.

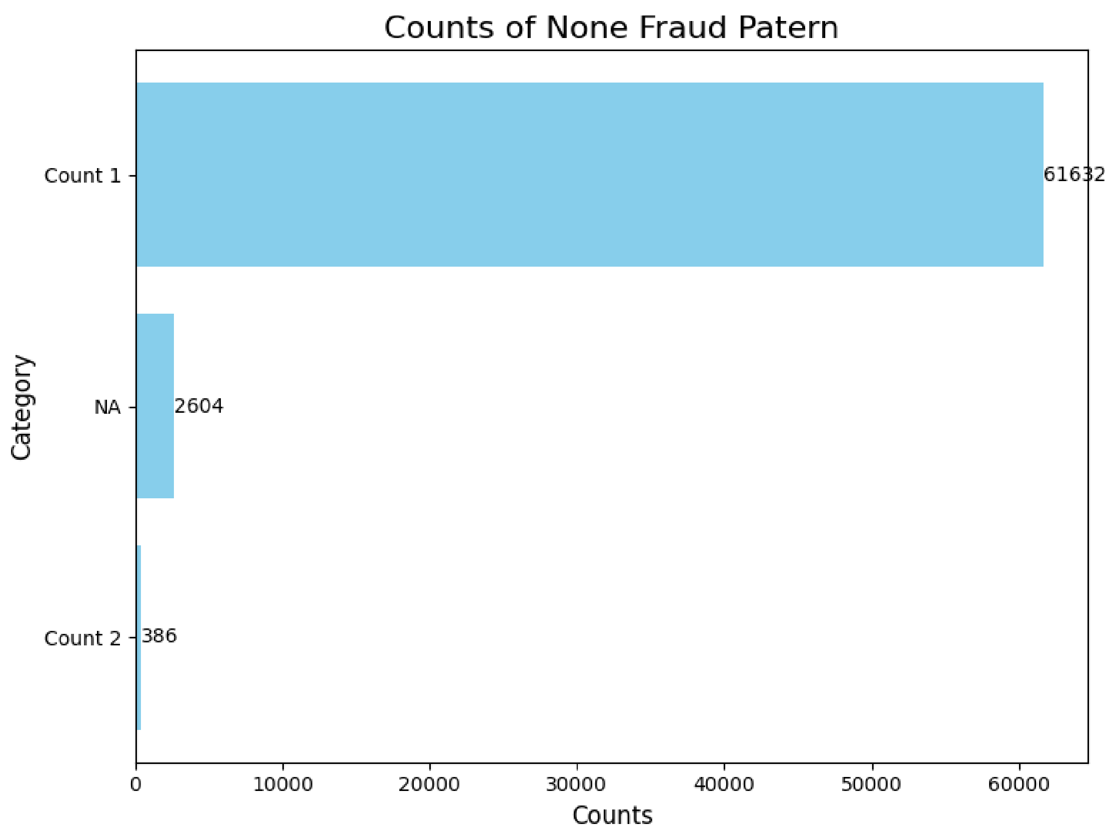
[130]:
```python
# Memfilter data hanya untuk 'Group' dengan value counts lebih dari 2
filtered_groups = group_counts[group_counts > 2].index

# Memfilter data asli berdasarkan hasil di atas
filtered_data = data[data['Group'].isin(filtered_groups)]

# Menghitung data berdasarkan 'buyer_status' setelah memfilter 'Group'
buyer_status_counts = filtered_data['buyer_status'].value_counts()

# Membuat bar plot untuk hasil
plt.figure(figsize=(10, 6))
plt.bar(buyer_status_counts.index, buyer_status_counts.values, color='skyblue')

# Menambahkan judul dan label sumbu
plt.title('Counts Fraud by Buyer Status', fontsize=16)
plt.xlabel('Buyer Status', fontsize=12)
plt.ylabel('Counts', fontsize=12)

# Menambahkan angka di atas setiap bar
for index, value in enumerate(buyer_status_counts.values):
    plt.text(index, value + 0.5, str(value), ha='center', fontsize=10)

# Menampilkan plot
plt.xticks(rotation=45)  # Rotasi label x untuk visibilitas lebih baik
plt.tight_layout()
plt.show()
```



The analysis of fraud counts by buyer status shows that most fraud cases occur in Frozen accounts, indicating a higher risk for fraudulent activity. While there are fewer cases in Banned accounts, this suggests that blocking measures may not be entirely effective. Additionally, some fraud cases in Normal accounts point to weaknesses in the fraud detection system. These findings highlight vulnerabilities in Frozen accounts and the need to reevaluate account blocking effectiveness, while also calling for enhancements in fraud detection systems to better identify fraud across all buyer statuses.

[126]:
```
# Menghitung jumlah NA di kolom 'Group'
na_count = data['Group'].isna().sum()

# Menghitung value counts untuk kolom 'Group' dan filter untuk count 1 dan 2
value_counts = data['Group'].value_counts()
count_1 = value_counts[value_counts == 1].count()  # Menghitung jumlah grup␣
 ↪dengan count 1
count_2 = value_counts[value_counts == 2].count()  # Menghitung jumlah grup␣
 ↪dengan count 2

# Membuat DataFrame untuk visualisasi
counts_df = pd.DataFrame({
    'Category': ['NA', 'Count 1', 'Count 2'],
    'Counts': [na_count, count_1, count_2]
})

# Mengurutkan DataFrame berdasarkan 'Counts' dari yang tertinggi ke terendah
```

```
counts_df = counts_df.sort_values(by='Counts', ascending=True)

# Membuat bar plot
plt.figure(figsize=(8, 6))
plt.barh(counts_df['Category'], counts_df['Counts'], color='skyblue')

# Menambahkan judul dan label sumbu
plt.title('Counts of None Fraud Patern', fontsize=16)
plt.xlabel('Counts', fontsize=12)
plt.ylabel('Category', fontsize=12)

# Menambahkan nilai di samping setiap bar
for index, value in enumerate(counts_df['Counts']):
    plt.text(value + 0.5, index, str(value), va='center', fontsize=10)

# Menampilkan plot
plt.tight_layout()
plt.show()
```



The analysis of non-fraudulent patterns shows that most data is classified as (Count 1), indicating it is the most common pattern among legitimate activities. A significant portion is labeled as (NA), suggesting that some data could not be accurately categorized, while (Count 2) is rarely observed. This implies that (Count 1) represents expected normal transaction patterns, and the presence of (NA) highlights issues with data completeness. The rarity of (Count 2) may point to unusual behaviors. Therefore, further investigation is needed to understand (Count 1) characteristics, improve data quality to reduce (NA) instances, and explore (Count 2) as a potential indicator of anomalies.