Project: Fastage Fraud Detection

In [ ]:

```
# Import libraries
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [ ]:

```
# Read Data
data = pd.read_csv("../Data/raw/fastag-data.csv")

# View
data.head()
```

Out[ ]:

	Transaction_ID	Timestamp	Vehicle_Type	FastagID	TollBoothID	Lane_Type	Vehicle_
0	1	1/6/2023 11:20	Bus	FTG-001-ABC-121	A-101	Express	
1	2	1/7/2023 14:55	Car	FTG-002-XYZ-451	B-102	Regular	
2	3	1/8/2023 18:25	Motorcycle	NaN	D-104	Regular	
3	4	1/9/2023 2:05	Truck	FTG-044-LMN-322	C-103	Regular	
4	5	1/10/2023 6:35	Van	FTG-505-DEF-652	B-102	Express	

In [ ]:

```
# Shape
data.shape
```

Out[ ]:

```
(5000, 13)
```

In [ ]:

```
# Check for duplicates
data.duplicated().sum()
```

Out[ ]:

```
0
```

In [ ]:

```
# Metainformation
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Transaction_ID        5000 non-null   int64 
 1   Timestamp             5000 non-null   object
 2   Vehicle_Type          5000 non-null   object
 3   FastagID              4451 non-null   object
 4   TollBoothID           5000 non-null   object
 5   Lane_Type             5000 non-null   object
 6   Vehicle_Dimensions    5000 non-null   object
 7   Transaction_Amount    5000 non-null   int64 
 8   Amount_paid           5000 non-null   int64 
 9   Geographical_Location 5000 non-null   object
 10  Vehicle_Speed         5000 non-null   int64 
 11  Vehicle_Plate_Number  5000 non-null   object
 12  Fraud_indicator       5000 non-null   object
dtypes: int64(4), object(9)
memory usage: 507.9+ KB
```

- *Dataset contains 5000 records for fastag transactions, there are no missing values in dataset except for the column 'FastageID'.*

In [ ]:

```
# Analyze the missing FastagID
data[data.FastagID.isnull()]
```

Out[ ]:

	Transaction_ID	Timestamp	Vehicle_Type	FastagID	TollBoothID	Lane_Type	Vehi
2	3	1/8/2023 18:25	Motorcycle	NaN	D-104	Regular	
9	10	1/15/2023 7:30	Motorcycle	NaN	D-104	Regular	
16	17	1/22/2023 16:45	Motorcycle	NaN	D-104	Regular	
23	24	1/29/2023 3:05	Motorcycle	NaN	D-104	Regular	
30	31	2/5/2023 13:20	Motorcycle	NaN	D-104	Regular	
...	...	...	...	...	...	...	
4966	4967	8/31/2023 6:08	Motorcycle	NaN	D-106	Regular	
4973	4974	12/27/2023 19:04	Motorcycle	NaN	D-106	Regular	
4980	4981	4/20/2023 6:01	Motorcycle	NaN	D-106	Regular	
4987	4988	8/19/2023 18:57	Motorcycle	NaN	D-106	Regular	
4994	4995	12/14/2023 6:53	Motorcycle	NaN	D-106	Regular	

549 rows × 13 columns

In [ ]:

```
# Transaction amount with no FastagID
transaction_amt = data[data.FastagID.isnull()]['Transaction_Amount'].unique()
amt_paid = data[data.FastagID.isnull()]['Amount_paid'].unique()

transaction_amt, amt_paid
```

Out[ ]:

```
(array([0], dtype=int64), array([0], dtype=int64))
```

- *All of these transactions without FastagID has zero for amount paid.*
- *Meaning, there is no record for amount paid for these transactions. so, lets drop all rows with missing FastagID.*

In [ ]:

```
# Drop missing FastagId rows.
data = data.dropna(axis=0).copy()
```

In [ ]:

```
# Unique values in IDs, and categorical data.
data.nunique()
```

Out[ ]:
```
Transaction_ID          4451
Timestamp               4008
Vehicle_Type               7
FastagID                4451
TollBoothID                4
Lane_Type                  2
Vehicle_Dimensions         3
Transaction_Amount        20
Amount_paid               23
Geographical_Location      5
Vehicle_Speed             85
Vehicle_Plate_Number    4451
Fraud_indicator            2
dtype: int64
```

In [ ]:
```python
for col in data.select_dtypes(include='object').columns:
    print(col)
    print(data[col].unique())
    print('--' * 10)
```

```
Timestamp
['1/6/2023 11:20' '1/7/2023 14:55' '1/9/2023 2:05' ... '2/5/2023 5:08'
 '2/20/2023 20:34' '3/10/2023 0:59']
--------------------
Vehicle_Type
['Bus ' 'Car' 'Truck' 'Van' 'Sedan' 'SUV' 'Motorcycle']
--------------------
FastagID
['FTG-001-ABC-121' 'FTG-002-XYZ-451' 'FTG-044-LMN-322' ...
 'FTG-447-PLN-109' 'FTG-458-VFR-876' 'FTG-459-WSX-543']
--------------------
TollBoothID
['A-101' 'B-102' 'C-103' 'D-106']
--------------------
Lane_Type
['Express' 'Regular']
--------------------
Vehicle_Dimensions
['Large' 'Small' 'Medium']
--------------------
Geographical_Location
['13.059816123454882, 77.77068662374292'
 '13.042660878688794, 77.47580097259879'
 '12.84197701525119, 77.67547528176169'
 '12.936687032945434, 77.53113977439017'
 '13.21331620748757, 77.55413526894684']
--------------------
Vehicle_Plate_Number
['KA11AB1234' 'KA66CD5678' 'KA11GH3456' ... 'KA33WX6789' 'KA35YZ0123'
 'KA37AB3456']
--------------------
Fraud_indicator
['Fraud' 'Not Fraud']
--------------------
```

In [ ]:
```python
# Convert date into datetime type
data['Timestamp'] = pd.to_datetime(data.Timestamp)
```

In [ ]:
```python
# Convert columns
data.rename(columns={col: col.lower() for col in data.columns.tolist()}, inplace
# rename
data.rename(columns={
    'fastagid': 'fastag_id',
    'tollboothid': 'tollbooth_id',
    'vehicle_dimensions': 'vehicle_size'
}, inplace=True)
```

In [ ]:

```
# Check data types
data.dtypes
```

Out[ ]:

```
transaction_id                    int64
timestamp                datetime64[ns]
vehicle_type                     object
fastag_id                        object
tollbooth_id                     object
lane_type                        object
vehicle_size                     object
transaction_amount                int64
amount_paid                       int64
geographical_location            object
vehicle_speed                     int64
vehicle_plate_number             object
fraud_indicator                  object
dtype: object
```

In [ ]:

```
# Summary descriptions
columns = ['transaction_amount', 'amount_paid', 'vehicle_speed']
fraud_desc = data[data.fraud_indicator == 'Fraud'][columns].describe()
notfraud_desc = data[data.fraud_indicator == 'Not Fraud'][columns].describe()

pd.concat([fraud_desc, notfraud_desc], axis=1, keys=['Fraudulent', 'Non-Fraudule
```

Out[ ]:

		Fraudulent			
	transaction_amount	**amount_paid**	**vehicle_speed**	**transaction_amount**	**amount_pa**
count	983.000000	983.000000	983.000000	3468.000000	3468.0000
mean	193.555443	92.838250	68.340793	177.348616	177.3486
std	97.465586	35.230277	16.832977	104.256672	104.2566
min	60.000000	0.000000	20.000000	0.000000	0.0000
25%	120.000000	90.000000	55.000000	110.000000	110.0000
50%	145.000000	100.000000	68.000000	130.000000	130.0000
75%	300.000000	110.000000	82.000000	300.000000	300.0000
max	350.000000	190.000000	118.000000	350.000000	350.0000

In [ ]:

```
# Datetime
data['timestamp'].dt.year.unique() # 2023 data
```

Out[ ]:

```
array([2023])
```

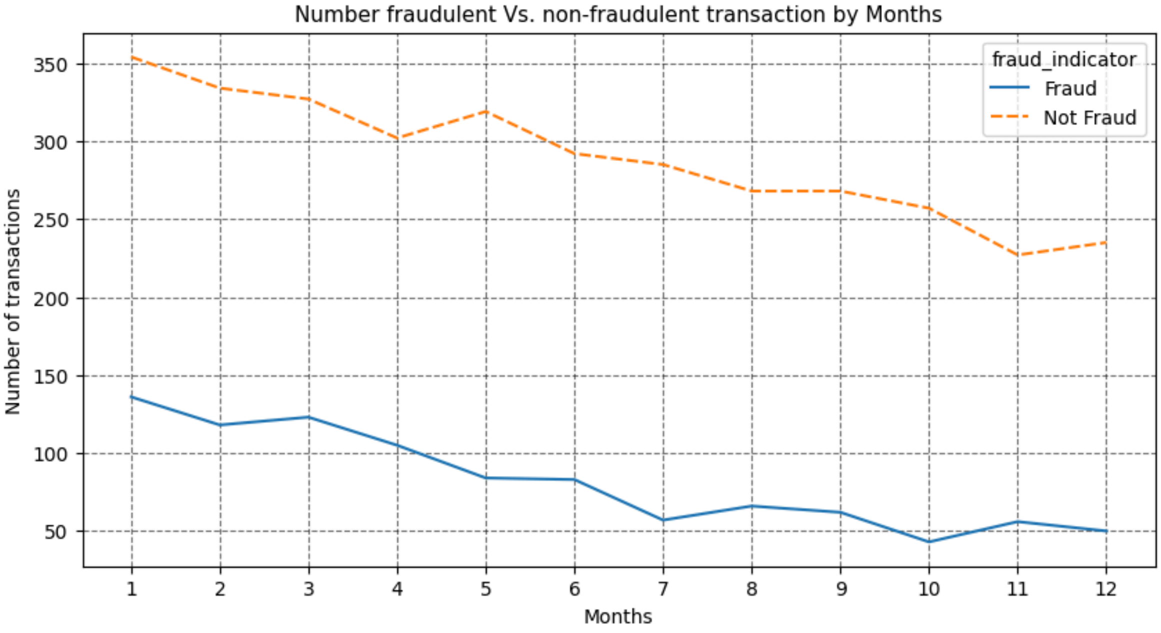
```
In [ ]: # Total days of records
        data['timestamp'].dt.date.nunique()
```

```
Out[ ]: 365
```

```
In [ ]: # Create new columns
        data['month'] = data['timestamp'].dt.month
        data['weekday'] = data['timestamp'].dt.weekday
        data['hour'] = data['timestamp'].dt.hour
        data['is_month_end'] = data['timestamp'].dt.is_month_end.astype('int')
        data['is_month_start'] = data['timestamp'].dt.is_month_start.astype('int')
        data['quarter'] = data['timestamp'].dt.quarter
        data['day_month'] = data['timestamp'].dt.strftime(date_format='%d-%m')
```

```
In [ ]: # Datetime analysis - Number of fraudulent transaction in each month
        # data preparation
        monthly_n_transactions = pd.pivot_table(data, values='transaction_id', index='mc

        # for label ticks
        hours = sorted(data.hour.unique().tolist())
        months = sorted(data.month.unique().tolist())
```

```
In [ ]: # plot
        plt.figure(figsize=(10, 5))
        sns.lineplot(monthly_n_transactions)

        plt.title('Number fraudulent Vs. non-fraudulent transaction by Months', size=11)
        plt.xlabel('Months')
        plt.ylabel('Number of transactions')

        plt.grid(ls='--', c="#181818", alpha=0.6)
        plt.xticks(ticks=months);
```



- *The lineplot compares the count of total fraudulent and non-fraudulent transactions over months(from 1 to 12) in year 2023.*
- *The line plot represents the decreasing trend in overall fastag transactions.*

- *The line plot shows, the highest number of fraudulent as well as non-fraudulent transaction in January, while the octobar has lowest fraudulent transaction.*

In [ ]:

```
# What is fraction of monthly contribution in total fraudulent transactions?
pct_fraud = (monthly_n_transactions['Fraud'] * 100 / monthly_n_transactions['Fra
pct_fraud.columns = ['Month', 'Fraud_Percent']

pct_fraud.sort_values(by='Fraud_Percent', ascending=False)
```

Out[ ]:

	Month	Fraud_Percent
0	1	13.835198
2	3	12.512716
1	2	12.004069
3	4	10.681587
4	5	8.545270
5	6	8.443540
7	8	6.714140
8	9	6.307223
6	7	5.798576
10	11	5.696846
11	12	5.086470
9	10	4.374364

- *The first quarter (the first four months of the year-January, February, March and April) seems to have hightest fraudelent transaction over the year 2023.*

In [ ]:

```
# What happened in January?
jan_23 = data[data.month == 1]

jan_23[jan_23.fraud_indicator=='Fraud'].groupby(by='day_month').count()['transac
```

Out[ ]:

```
day_month
15-01    9
03-01    7
24-01    7
23-01    7
07-01    7
Name: transaction_id, dtype: int64
```

In [ ]:

```
# day-name: 0 -> Monday, 1 -> Tuesday, 2 -> Wednesday, 3 -> Thursday, 4 -> Frida
jan_23.loc[jan_23.day_month == '15-01', 'weekday'].head(1)
```

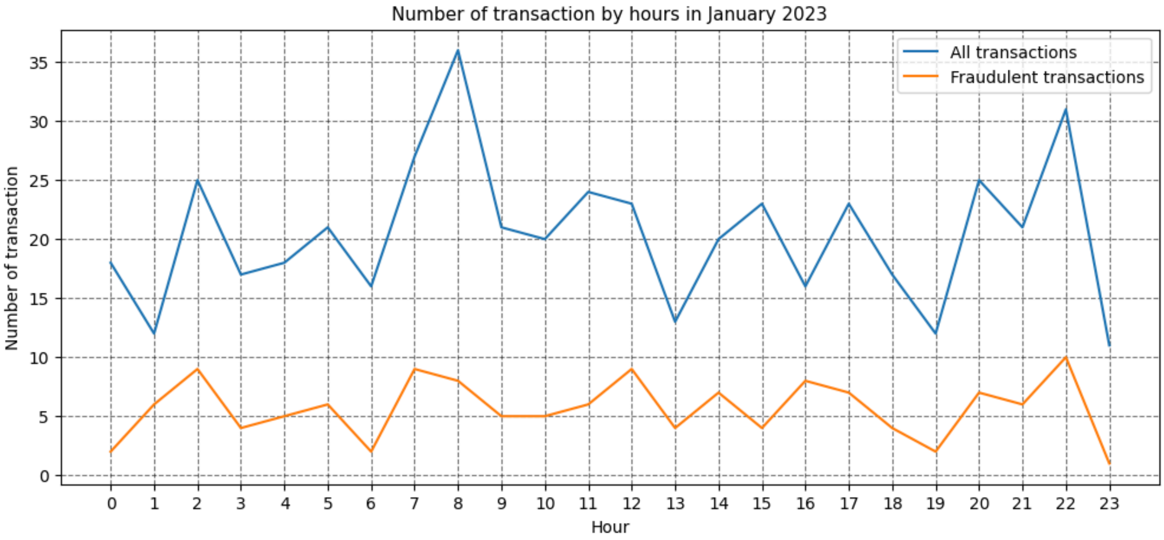
Out[ ]:

```
126    6
Name: weekday, dtype: int32
```

- *The highest fraudulent transaction observed on Sunday, 15th January.*

In [ ]:

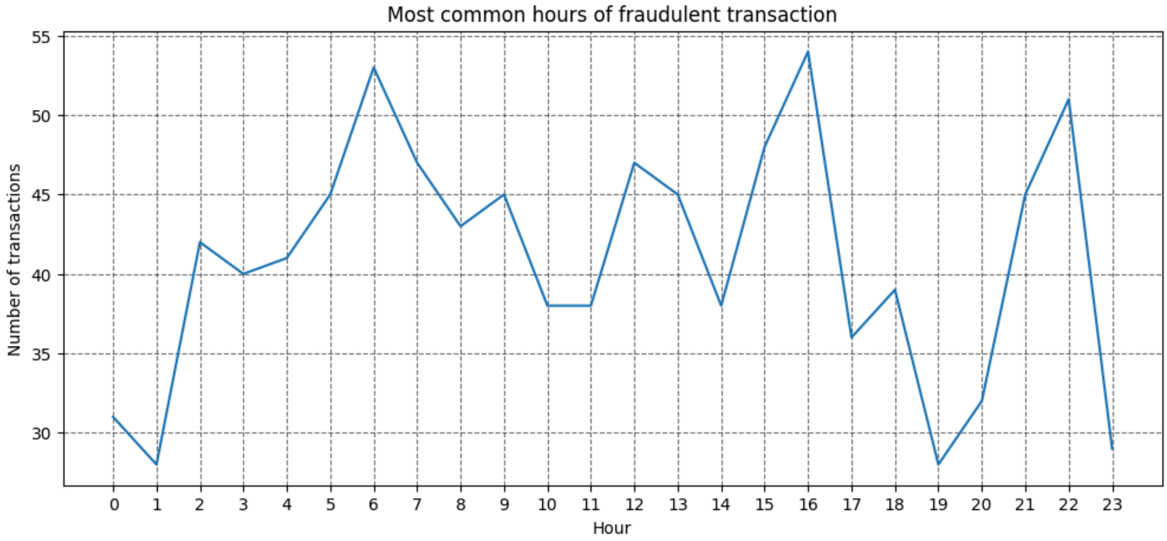
```
# What are the most common hours for fraudulent transaction in January?

# Data preparation
pivot_nt = pd.pivot_table(jan_23, values='transaction_id', columns='fraud_indica
pivot_nt['total_transactions'] = pivot_nt['Fraud'] + pivot_nt['Not Fraud']

# plot
plt.figure(figsize=(12, 5))
sns.lineplot(pivot_nt['total_transactions'], label='All transactions')
sns.lineplot(pivot_nt['Fraud'], label='Fraudulent transactions')

plt.title('Number of transaction by hours in January 2023', size=11)
plt.xlabel('Hour')
plt.ylabel('Number of transaction')
plt.grid(ls='--', c="#181818", alpha=0.6)
plt.xticks(ticks=hours, color='#000')
plt.show()
```



- *The above lineplot shows the total number of hourly transaction in January 2023.*
- *The highest fraudulent transactions are observed at the peakest hours in January.*

In [ ]:

```
# What is the most common hour for fraudulent transaction in
pivot_nt = pd.pivot_table(data, values='transaction_id', columns='fraud_indicato

plt.figure(figsize=(12, 5))
sns.lineplot(pivot_nt['Fraud'])

plt.title('Most common hours of fraudulent transaction')
plt.xlabel('Hour')
plt.ylabel('Number of transactions')

plt.grid(ls='--', c="#181818", alpha=0.6)
plt.xticks(ticks=hours);
```

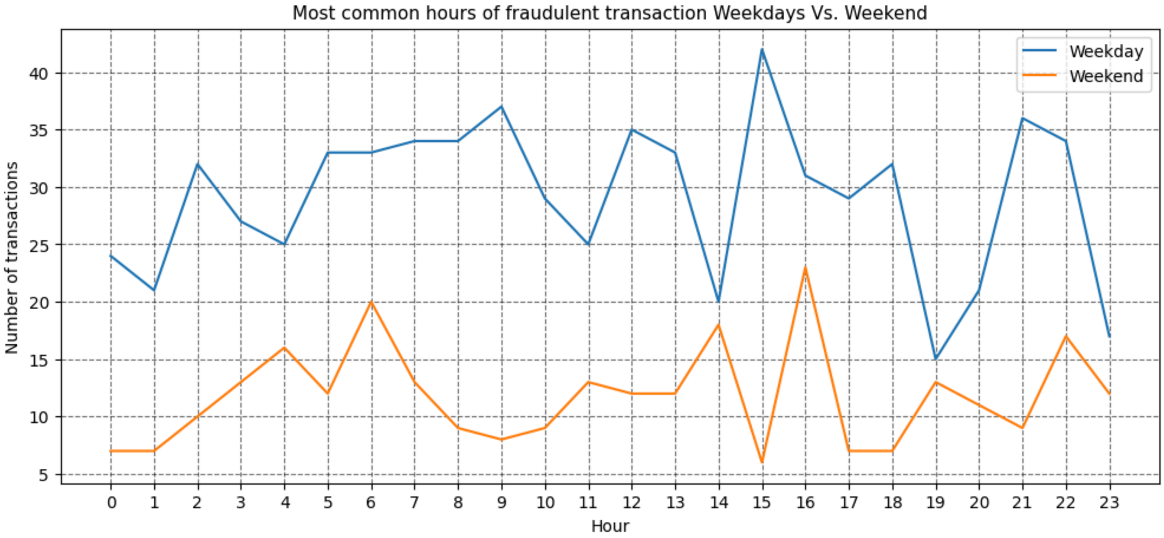


- *The line plot has some noticeable peaks at around Early mornings (around 2nd and 6th hour), and late evening (around 16th and 22th hour).*
- *Lunchtime (around 12th hour) could be another peak.*
- *This might happend because these are the peak traffic hours.*

There might be difference in patterns for weeekdays and weekends. Let's analysis further for days of the week.

In [ ]:

```
# Is there any difference in fraud patterns for weekdays and weekends.

# Data preparation
# filter data
fraud_data = data[data.fraud_indicator=='Fraud']
weekdays = fraud_data[fraud_data.weekday <= 4]
weekends = fraud_data[fraud_data.weekday > 4]

# Get the count of transaction by hour
weekdays_grp = weekdays.groupby(by='hour')['transaction_id'].count()
weekends_grp = weekends.groupby(by='hour')['transaction_id'].count()


plt.figure(figsize=(12, 5))
sns.lineplot(weekdays_grp, label='Weekday')
sns.lineplot(weekends_grp, label='Weekend')

plt.title('Most common hours of fraudulent transaction Weekdays Vs. Weekend', si
plt.xlabel('Hour')
plt.ylabel('Number of transactions')

plt.grid(ls='--', c="#181818", alpha=0.6)
plt.xticks(ticks=hours);
```



- *The above line graph shows the fraud transactions patterns for weekdays and weekends.*
- *The number of transactions on weekends are less compare to weekdays.*
- *The main difference can observed at the 15th hour, which has highest peak for weekdays however, lowest for the weekends, this might happend becuase it is not busy traffic hour on weekends.*
- *Similary, the hours that has lowest peak on weekdays has highest peak for weekends (around 14th, and 19th hour)*
- *Overall pattern of early mornings and late evening highest fraudulent transaction can observed in both*

In [ ]:
```
# Earlier we saw that the first four months have higher rate of fraudulent trans
data.groupby(by=['fraud_indicator', 'quarter'])['transaction_id'].count()
```

Out[ ]:
```
fraud_indicator  quarter
Fraud            1           377
                 2           272
                 3           185
                 4           149
Not Fraud        1          1015
                 2           913
                 3           821
                 4           719
Name: transaction_id, dtype: int64
```

In [ ]:
```
# Total transaction amount and acutal amount paid by quarter.
grp_data = data.groupby(by=['fraud_indicator', 'quarter'])[['transaction_amount'
```

In [ ]:
```
grp_data['amt_paid_percent'] = grp_data['amount_paid'] * 100 / grp_data['transac
grp_data = grp_data.reset_index()
```

In [ ]:
```
grp_data
```

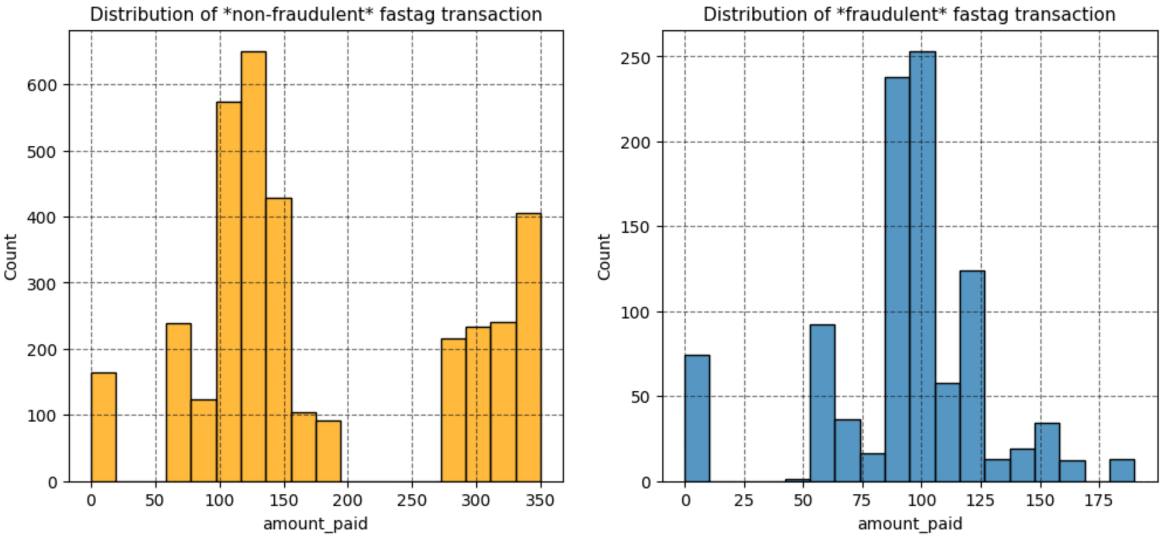
Out[ ]:

	fraud_indicator	quarter	transaction_amount	amount_paid	amt_paid_percent
0	Fraud	1	75305	35755	47.480247
1	Fraud	2	51880	24900	47.995374
2	Fraud	3	35275	16985	48.150248
3	Fraud	4	27805	13620	48.983996
4	Not Fraud	1	183010	183010	100.000000
5	Not Fraud	2	160625	160625	100.000000
6	Not Fraud	3	142865	142865	100.000000
7	Not Fraud	4	128545	128545	100.000000

The rate of actual amount paid in fraudulent transaction is always less than 50% for all the quarters, whereas in guinue transactions is 100%.

In [ ]:

```python
# Distribution of amount paid
non_fraudulent_amount = data[data['fraud_indicator'] == 'Not Fraud']['amount_pai
fraudulent_amount = data[data['fraud_indicator'] == 'Fraud']['amount_paid']

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
sns.histplot(non_fraudulent_amount, bins=18, color='orange', ax=ax1)
sns.histplot(fraudulent_amount, bins=18, ax=ax2)

ax1.grid(ls='--', c="#181818", alpha=0.6)
ax2.grid(ls='--', c="#181818", alpha=0.6)

ax1.set_title('Distribution of *non-fraudulent* fastag transaction', size=11)
ax2.set_title('Distribution of *fraudulent* fastag transaction', size=11);
```



- *Histogram plot shows the distribution of amount paid in non-fraudulent transaction(left) and fraudulent transaction(right).*
- *The graph shows multiple groups for both transaction, which represent the multi-model distribution.*

To be Continue...