UPI TRANSACTIONS 2024

Project Title

Analysis of UPI Transaction Patterns and user Behaviour in Digital Payments(2024)

Project Overview

Unified Payments Interface (UPI) has changed the way people in India transfer money to one another by allowing transfers on the go — quick, easy and secure. As a result of its fast adoption, UPI is now being used for a vast array of payments such as for grocery, bills, shopping and travel. Analysis of these transaction patterns is crucial to study the user spending preference, user behavior and system efficiency.

This work is centered around analysis of UPI transaction data using Python (NumPy, Pandas, Matplotlib, Seaborn). The breakdown will cover types of transactions, number of transactions, time-of-day use, success/failure rates, devices of choice and bank trends. The primary goal is to find some key nuggets of information that speak to how digital payments are changing and driving consumers' preferences. Ultimately, the project seeks to present a detailed understanding of UPI usage behavior to support studies on digital finance and data-driven decision-making.

Domain

Finance

Aim

To analyse UPI transaction data to determine key user behavior trends,peak period of transaction,fraud occurrence patterns and usage patterns with the aim of providing insights to enable digital payment platforms to improve the experience for users and the efficiency of services

Objectives

‣ **To clean and prepare the UPI transaction dataset for analysis**
‣ **To explore the data and understand overall transaction patterns**
‣ **To identify user behavior trends in digital payments**
‣ **To find peak times and days when most transactions occur**
‣ **To analyze how users interact with different types of transactions**
‣ **To detect and analyze fradulent transaction patterns(if a fraud flag is present)**
‣ **To create visualizations that clearly show important patterns and insights**
‣ **To provide useful findings that can help improve digital payment platforms**

Dataset Description

Source : Kaggle(UPI Dataset Generator)
Size : 250,152 Rows and 17 Columns
Features : user ID, Transaction Type, Amount, Date/Time, Status, etc.

Coding & Implementation Section

Data Loading and Initial Overview

🔺 Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

🔺 Load Dataset

In [2]:
```
df = pd.read_csv("upi_transactions_2024.zip",low_memory=False)
```

🔺 Dataset Overview

In [3]:
```
print("\n🔷 Shape of the dataset :",df.shape,"\n" )
print("🌟" * 50)
print("\n🔷 Number of Rows :",df.shape[0],"\n")
print("🌟" * 50)
print("\n🔷 Number of Columns :",df.shape[1],"\n")
print("🌟" * 50)
```

```
print("\n🔷 Data Types :",df.dtypes,"\n")
print("🌟" * 50)
print("\n🔷 Dataset Info :",df.info(),"\n")
print("🌟" * 50)
print("\n🔷 First 5 Rows :",df.head(),"\n")
print("🌟" * 50)
print("\n🔷 Choose a sample row :",df.sample(),"\n")
print("🌟" * 50)
print("\n🔷 Statistical Summary :",df.describe(),"\n")
```

```
🔷 Shape of the dataset : (250152, 17)

☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️
☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️

🔷 Number of Rows : 250152

☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️
☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️

🔷 Number of Columns : 17

☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️
☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️

🔷 Data Types : transaction id         object
timestamp              object
transaction type       object
merchant_category      object
amount (INR)           object
transaction_status     object
sender_age_group       object
receiver_age_group     object
sender_state           object
sender_bank            object
receiver_bank          object
device_type            object
network_type           object
fraud_flag              int64
hour_of_day           float64
day_of_week            object
is_weekend             object
dtype: object

☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️
☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250152 entries, 0 to 250151
Data columns (total 17 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   transaction id      250152 non-null  object 
 1   timestamp           250152 non-null  object 
 2   transaction type    250151 non-null  object 
 3   merchant_category   250139 non-null  object 
 4   amount (INR)        250137 non-null  object 
 5   transaction_status  250149 non-null  object 
 6   sender_age_group    250152 non-null  object 
 7   receiver_age_group  250150 non-null  object 
 8   sender_state        250150 non-null  object 
 9   sender_bank         250143 non-null  object 
 10  receiver_bank       250147 non-null  object 
 11  device_type         250150 non-null  object 
 12  network_type        250144 non-null  object 
 13  fraud_flag          250152 non-null  int64  
 14  hour_of_day         250148 non-null  float64
 15  day_of_week         250150 non-null  object 
 16  is_weekend          250146 non-null  object 
dtypes: float64(1), int64(1), object(15)
memory usage: 32.4+ MB
```

```
🔷 Dataset Info : None

🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟

🔷 First 5 Rows :    transaction id         timestamp transaction type merchant_c
ategory  \
0  TXN0000000001  08-10-2024 15:17              P2P     Entertainment
1  TXN0000000002  11-04-2024 06:56              P2M           Grocery
2  TXN0000000003  02-04-2024 13:27              P2P           Grocery
3  TXN0000000004  07-01-2024 10:09              P2P              Fuel
4  TXN0000000005  23-01-2024 19:04              P2P          Shopping

  amount (INR) transaction_status sender_age_group receiver_age_group  \
0          868            SUCCESS            26-35              18-25
1         1011            SUCCESS            26-35              26-35
2          477            SUCCESS            26-35              36-45
3         2784            SUCCESS            26-35              26-35
4          990            SUCCESS            26-35              18-25

    sender_state sender_bank receiver_bank device_type network_type  \
0          Delhi        Axis           SBI     Android           4G
1  Uttar Pradesh       ICICI          Axis         iOS           4G
2      Karnataka    Yes Bank           PNB     Android           4G
3          Delhi       ICICI           PNB     Android           5G
4          Delhi        Axis      Yes Bank         iOS         WiFi

   fraud_flag  hour_of_day day_of_week is_weekend
0           0         15.0     Tuesday          0
1           0          6.0    Thursday          0
2           0         13.0     Tuesday          0
3           0         10.0      Sunday          1
4           0         19.0     Tuesday          0

🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟

🔷 Choose a sample row :        transaction id         timestamp transaction type
merchant_category  \
22065  TXN0000022066  07-09-2024 18:57     Bill Payment              Food

      amount (INR) transaction_status sender_age_group receiver_age_group  \
22065          335            SUCCESS            36-45              26-35

        sender_state sender_bank receiver_bank device_type network_type  \
22065  Uttar Pradesh        Axis      IndusInd     Android           4G

       fraud_flag  hour_of_day day_of_week is_weekend
22065           0         18.0    Saturday          1

🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟

🔷 Statistical Summary :           fraud_flag    hour_of_day
count  250152.000000  250148.000000
mean        0.002035      14.680633
std         0.045063       5.188191
min         0.000000       0.000000
25%         0.000000      11.000000
```

```
50%         0.000000     15.000000
75%         0.000000     19.000000
max         1.000000     23.000000
```

Data Pre-processing

🔺 Check Missing Values

In [4]:
```
print("\n🔷 Missing Values :\n",df.isnull().sum(),"\n")
print("🌟" * 50)
print("\nCount of total Misisng Values :\n",df.isnull().sum().sum(),"\n")
```

```
🔷 Missing Values :
 transaction id          0
timestamp              0
transaction type       1
merchant_category     13
amount (INR)          15
transaction_status     3
sender_age_group       0
receiver_age_group     2
sender_state           2
sender_bank            9
receiver_bank          5
device_type            2
network_type           8
fraud_flag             0
hour_of_day            4
day_of_week            2
is_weekend             6
dtype: int64

🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟

Count of total Misisng Values :
 72
```

🔺 Text Cleaning

In [5]:
```
# Remove extra spaces
for col in df.select_dtypes(include="object").columns:
    df[col] = df[col].str.strip()

# proper case
proper_case = [
    "merchant_category","transaction type","sender_state", "sender_bank", "recei
    "device_type", "network_type", "day_of_week"]

for col in proper_case:
    df[col] = df[col].str.title()

# make transaction status upper case
df["transaction_status"] = df["transaction_status"].str.upper()
```

In [6]:
```
# spelling correction in merchant category
print(df["merchant_category"].value_counts())
```

```
merchant_category
Grocery          49977
Food             37484
Shopping         29899
Fuel             25069
Other            24837
Utilities        22351
Transport        20116
Entertainment    20114
Healthcare       12673
Education         7606
Health Care          4
Grocry               3
Groce Ry             2
Electricity          1
                     1
Utilites             1
Entmnt               1
Name: count, dtype: int64
```

In [7]:
```
merchant_categories = {
     "Grocry": "Grocery",
     "Groce Ry": "Grocery",
     "Electricity" :"Utilities",
    "Utilites" :"Utilities",
    "Health Care": "Healthcare",
    "Resturants": "Restaurant",
    "Entmnt": "Entertainment" }

df["merchant_category"] = df["merchant_category"].replace(merchant_categories)
```

In [8]:
```
# value counts for categorical columns
for col in df.select_dtypes(include="object").columns:
    print("\n","🌟"* 5,col.title(),"🌟"* 5,"\n")
    print(df[col].value_counts())
```

```
🌟 🌟 🌟 🌟 🌟 Transaction Id 🌟 🌟 🌟 🌟 🌟

transaction id
TXN0000249998    2
TXN0000249999    2
TXN0000250000    2
TXN0000000001    1
TXN0000166757    1
                ..
TXN0000083388    1
TXN0000083389    1
TXN0000083390    1
TXN0000083391    1
TXN0000250149    1
Name: count, Length: 250149, dtype: int64

🌟 🌟 🌟 🌟 🌟 Timestamp 🌟 🌟 🌟 🌟 🌟

timestamp
21-05-2024 16:38    8
27-05-2024 12:11    7
29-03-2024 20:28    7
24-04-2024 17:12    7
26-11-2024 20:46    6
                   ..
03-01-2024 23:25    1
16-01-2024 09:48    1
24-06-2024 17:05    1
27-09-2024 20:32    1
10-06-2024 21:40    1
Name: count, Length: 184002, dtype: int64

🌟 🌟 🌟 🌟 🌟 Transaction Type 🌟 🌟 🌟 🌟 🌟

transaction type
P2P             112472
P2M              87726
Bill Payment     37392
Recharge         12561
Name: count, dtype: int64

🌟 🌟 🌟 🌟 🌟 Merchant_Category 🌟 🌟 🌟 🌟 🌟

merchant_category
Grocery          49982
Food             37484
Shopping         29899
Fuel             25069
Other            24837
Utilities        22353
Transport        20116
Entertainment    20115
Healthcare       12677
Education         7606
                     1
Name: count, dtype: int64

🌟 🌟 🌟 🌟 🌟 Amount (Inr) 🌟 🌟 🌟 🌟 🌟

amount (INR)
```

```
215      311
203      307
174      302
197      296
227      296
        ...
7695       1
23658      1
10483      1
20523      1
9610       1
Name: count, Length: 10376, dtype: int64

 🌟 🌟 🌟 🌟 🌟  Transaction_Status  🌟 🌟 🌟 🌟 🌟

transaction_status
SUCCESS    237732
FAILED      12417
Name: count, dtype: int64

 🌟 🌟 🌟 🌟 🌟  Sender_Age_Group  🌟 🌟 🌟 🌟 🌟

sender_age_group
26-35    87503
36-45    62909
18-25    62376
46-55    24855
56+      12509
Name: count, dtype: int64

 🌟 🌟 🌟 🌟 🌟  Receiver_Age_Group  🌟 🌟 🌟 🌟 🌟

receiver_age_group
26-35    87924
18-25    62630
36-45    62204
46-55    24839
56+      12553
Name: count, dtype: int64

 🌟 🌟 🌟 🌟 🌟  Sender_State  🌟 🌟 🌟 🌟 🌟

sender_state
Maharashtra       37446
Uttar Pradesh     30135
Karnataka         29780
Tamil Nadu        25392
Delhi             24899
Telangana         22444
Gujarat           20067
Andhra Pradesh    20011
Rajasthan         19993
West Bengal       19983
Name: count, dtype: int64

 🌟 🌟 🌟 🌟 🌟  Sender_Bank  🌟 🌟 🌟 🌟 🌟

sender_bank
Sbi         62733
Hdfc        37534
```

```
Icici       29785
Indusind    25173
Axis        25071
Pnb         24954
Yes Bank    24860
Kotak       20033
Name: count, dtype: int64

 🌟🌟🌟🌟🌟 Receiver_Bank 🌟🌟🌟🌟🌟

receiver_bank
Sbi         62402
Hdfc        37696
Icici       29977
Indusind    25086
Axis        25026
Yes Bank    25010
Pnb         24809
Kotak       20141
Name: count, dtype: int64

 🌟🌟🌟🌟🌟 Device_Type 🌟🌟🌟🌟🌟

device_type
Android    187880
Ios         49651
Web         12619
Name: count, dtype: int64

 🌟🌟🌟🌟🌟 Network_Type 🌟🌟🌟🌟🌟

network_type
4G      149880
5G       62590
Wifi     25173
3G       12501
Name: count, dtype: int64

 🌟🌟🌟🌟🌟 Day_Of_Week 🌟🌟🌟🌟🌟

day_of_week
Monday       36515
Sunday       36021
Wednesday    35722
Tuesday      35582
Friday       35514
Thursday     35450
Saturday     35346
Name: count, dtype: int64

 🌟🌟🌟🌟🌟 Is_Weekend 🌟🌟🌟🌟🌟

is_weekend
0    178780
1     71364
          2
Name: count, dtype: int64
```

🔺 Handling Missing Values

In [9]:
```
#drop the rows have 2 or more than 2 empties
df.dropna(thresh=2,inplace=True)

#if the merchant category is empty fill with 'Other'
df['merchant_category'] = df['merchant_category'].fillna('Other')

#if day_of_week provided then find the 'is_weekend"
df['is_weekend'] = df['day_of_week'].apply(lambda x: 1 if x in['Saturday','Sunda

#remove the negative and non numeric values from amount column
df['amount (INR)'] = pd.to_numeric(df['amount (INR)'],errors='coerce')
df = df.dropna(subset=['amount (INR)'])
df = df[df['amount (INR)']>0]


#remove extreme high value and extreme low value
lower = df['amount (INR)'].quantile(0.01)
higher = df['amount (INR)'].quantile(0.99)
df = df[(df['amount (INR)'] >= lower) & (df['amount (INR)'] <= higher)]


df.dropna(inplace=True)

print("\n🔷After Handling of Missing value :\n",df.head(10),"\n")
```

```
🔷After Handling of Missing value :
   transaction id         timestamp transaction type merchant_category  \
0  TXN0000000001  08-10-2024 15:17              P2P     Entertainment
1  TXN0000000002  11-04-2024 06:56              P2M           Grocery
2  TXN0000000003  02-04-2024 13:27              P2P           Grocery
3  TXN0000000004  07-01-2024 10:09              P2P              Fuel
4  TXN0000000005  23-01-2024 19:04              P2P          Shopping
5  TXN0000000006  07-10-2024 22:32              P2P              Food
6  TXN0000000007  08-02-2024 10:25              P2P             Other
7  TXN0000000008  27-10-2024 18:47              P2P         Utilities
8  TXN0000000009  21-11-2024 09:39              P2P             Other
9  TXN0000000010  11-11-2024 15:58              P2M           Grocery

   amount (INR) transaction_status sender_age_group receiver_age_group  \
0         868.0            SUCCESS            26-35              18-25
1        1011.0            SUCCESS            26-35              26-35
2         477.0            SUCCESS            26-35              36-45
3        2784.0            SUCCESS            26-35              26-35
4         990.0            SUCCESS            26-35              18-25
5          91.0            SUCCESS            36-45              18-25
6         314.0            SUCCESS            36-45              18-25
7         264.0            SUCCESS            46-55              36-45
8         887.0            SUCCESS            46-55              36-45
9        3260.0            SUCCESS            46-55              18-25

    sender_state sender_bank receiver_bank device_type network_type  \
0          Delhi        Axis           Sbi     Android           4G
1  Uttar Pradesh       Icici          Axis         Ios           4G
2      Karnataka    Yes Bank           Pnb     Android           4G
3          Delhi       Icici           Pnb     Android           5G
4          Delhi        Axis      Yes Bank         Ios         Wifi
5      Karnataka    Indusind      Yes Bank     Android           3G
6      Telangana        Hdfc      Indusind     Android           4G
7    Maharashtra    Yes Bank           Sbi     Android           5G
8    Maharashtra       Kotak          Hdfc     Android           4G
9          Delhi         Sbi          Hdfc     Android           4G

   fraud_flag  hour_of_day day_of_week  is_weekend
0           0         15.0     Tuesday           0
1           0          6.0    Thursday           0
2           0         13.0     Tuesday           0
3           0         10.0      Sunday           1
4           0         19.0     Tuesday           0
5           0         22.0      Monday           0
6           0         10.0    Thursday           0
7           0         18.0      Sunday           1
8           0          9.0    Thursday           0
9           0         15.0      Monday           0
```

🔺 Remove Duplicates/Column

In [10]:
```
print("\n🔷 No.of Duplicate rows :",df.duplicated().sum())
print("🌟" * 50)

#removal of Duplicate rows
df.drop_duplicates(inplace=True)
```

```
print("\n🔷 Null Values after cleaning :",df.isnull().sum()
,"\n")
```

```
🔷 No.of Duplicate rows : 3
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟

🔷 Null Values after cleaning : transaction id        0
timestamp             0
transaction type      0
merchant_category     0
amount (INR)          0
transaction_status    0
sender_age_group      0
receiver_age_group    0
sender_state          0
sender_bank           0
receiver_bank         0
device_type           0
network_type          0
fraud_flag            0
hour_of_day           0
day_of_week           0
is_weekend            0
dtype: int64
```

🔺 Correcting Data Types

In [11]:

```
df['timestamp'] = pd.to_datetime(df['timestamp'],dayfirst=True,errors="coerce")
df.dropna(subset=['timestamp'],inplace=True)

df['amount (INR)'] = df['amount (INR)'].astype(float)

df['is_weekend'] = df['is_weekend'].astype(int)

print("\n🔷 Corrected Data Types :\n",df.dtypes,"\n")
```

```
🔷 Corrected Data Types :
 transaction id                object
timestamp             datetime64[ns]
transaction type              object
merchant_category             object
amount (INR)                 float64
transaction_status            object
sender_age_group              object
receiver_age_group            object
sender_state                  object
sender_bank                   object
receiver_bank                 object
device_type                   object
network_type                  object
fraud_flag                     int64
hour_of_day                  float64
day_of_week                   object
is_weekend                     int32
dtype: object
```

🔺 Creating Derived Columns

In [12]:
```
df['month'] = df['timestamp'].dt.month_name()

#function for new derived column time_of_day
def time_of_day(hour):
    if 0 <= hour < 4:
        return 'Midnight'
    elif 4 <= hour < 8:
        return 'Early Morning'
    elif 8 <= hour < 12:
        return 'Morning'
    elif 12 <= hour < 16:
        return 'Noon'
    elif 16 <= hour < 20:
        return 'Evening'
    else:
        return 'Night'

df['time_of_day'] = df['hour_of_day'].apply(time_of_day)

print("\n🔷 New Column and its dta types:\n",df.dtypes,"\n")
```

```
🔷 New Column and its dta types:
 transaction id                  object
timestamp                datetime64[ns]
transaction type                 object
merchant_category                object
amount (INR)                    float64
transaction_status               object
sender_age_group                 object
receiver_age_group               object
sender_state                     object
sender_bank                      object
receiver_bank                    object
device_type                      object
network_type                     object
fraud_flag                        int64
hour_of_day                     float64
day_of_week                      object
is_weekend                        int32
month                            object
time_of_day                      object
dtype: object
```

🔺 Filtering or aggregating data

In [13]:
```
#Filtering
# store success and failure separately
df_sucess = df[df["transaction_status"] =="SUCCESS"]
df_failed = df[df["transaction_status"] =="FAILED"]

print("\n🔷 Successful Transactions :\n",df_sucess.head(),"\n")
print("🌟" * 50)
print("\n🔷 Failed Transactions :\n",df_failed.head(),"\n")
print("🌟" * 50)

#for a specific state
```

```
state_delhi = df[df["sender_state"] == "Delhi"]
print("\n🔷 Filter by state Delhi  :\n",state_delhi,"\n")
print("🌟" * 50)

#for morning transactions
morning_transactions = df[df["time_of_day"].isin(["Morning"])]
print("\n🔷 Filter by Morning transaction :\n",morning_transactions,"\n")
```

```
🔷 Successful Transactions :
  transaction id           timestamp transaction type merchant_category  \
0  TXN0000000001 2024-10-08 15:17:00              P2P     Entertainment
1  TXN0000000002 2024-04-11 06:56:00              P2M           Grocery
2  TXN0000000003 2024-04-02 13:27:00              P2P           Grocery
3  TXN0000000004 2024-01-07 10:09:00              P2P              Fuel
4  TXN0000000005 2024-01-23 19:04:00              P2P          Shopping

   amount (INR) transaction_status sender_age_group receiver_age_group  \
0         868.0            SUCCESS            26-35              18-25
1        1011.0            SUCCESS            26-35              26-35
2         477.0            SUCCESS            26-35              36-45
3        2784.0            SUCCESS            26-35              26-35
4         990.0            SUCCESS            26-35              18-25

    sender_state sender_bank receiver_bank device_type network_type  \
0          Delhi        Axis           Sbi     Android           4G
1  Uttar Pradesh       Icici          Axis         Ios           4G
2      Karnataka    Yes Bank           Pnb     Android           4G
3          Delhi       Icici           Pnb     Android           5G
4          Delhi        Axis      Yes Bank         Ios         Wifi

   fraud_flag  hour_of_day day_of_week  is_weekend    month    time_of_day
0           0         15.0     Tuesday           0  October           Noon
1           0          6.0    Thursday           0    April  Early Morning
2           0         13.0     Tuesday           0    April           Noon
3           0         10.0      Sunday           1  January        Morning
4           0         19.0     Tuesday           0  January        Evening

☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️
☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️

🔷 Failed Transactions :
    transaction id           timestamp transaction type merchant_category  \
60   TXN0000000061 2024-09-11 20:20:00              P2P          Shopping
67   TXN0000000068 2024-08-14 18:30:00              P2M          Shopping
168  TXN0000000169 2024-07-20 20:07:00              P2M              Food
181  TXN0000000182 2024-12-06 15:39:00              P2P           Grocery
196  TXN0000000197 2024-03-23 21:20:00              P2P           Grocery

     amount (INR) transaction_status sender_age_group receiver_age_group  \
60          159.0             FAILED              56+              36-45
67          221.0             FAILED            46-55              26-35
168         232.0             FAILED            26-35              26-35
181        2241.0             FAILED            26-35              18-25
196        1490.0             FAILED            36-45              26-35

       sender_state sender_bank receiver_bank device_type network_type  \
60       Tamil Nadu         Sbi           Pnb     Android           5G
67            Delhi        Hdfc      Indusind         Ios           5G
168  Andhra Pradesh        Axis           Sbi         Ios         Wifi
181      Tamil Nadu         Sbi         Icici     Android           5G
196     Maharashtra         Sbi         Icici     Android           4G

     fraud_flag  hour_of_day day_of_week  is_weekend      month time_of_day
60            0         20.0   Wednesday           0  September       Night
67            0         18.0   Wednesday           0     August     Evening
168           0         20.0    Saturday           1       July       Night
181           0         15.0      Friday           0   December        Noon
196           0         21.0    Saturday           1      March       Night
```

☀️
☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️

```
🔷 Filter by state Delhi  :
        transaction id           timestamp transaction type merchant_category  \
0       TXN0000000001 2024-10-08 15:17:00              P2P     Entertainment
3       TXN0000000004 2024-01-07 10:09:00              P2P              Fuel
4       TXN0000000005 2024-01-23 19:04:00              P2P          Shopping
9       TXN0000000010 2024-11-11 15:58:00              P2M           Grocery
16      TXN0000000017 2024-01-25 07:27:00              P2P              Food
...               ...                 ...              ...               ...
250117  TXN0000250115 2024-07-05 21:00:00              P2M          Shopping
250127  TXN0000250125 2024-05-22 11:00:00              P2M        Healthcare
250134  TXN0000250132 2024-05-28 08:45:00              P2M              Food
250147  TXN0000250145 2024-06-07 09:20:00              P2P         Transport
250148  TXN0000250146 2024-06-08 12:05:00     Bill Payment              Fuel

        amount (INR) transaction_status sender_age_group receiver_age_group  \
0              868.0            SUCCESS            26-35              18-25
3             2784.0            SUCCESS            26-35              26-35
4              990.0            SUCCESS            26-35              18-25
9             3260.0            SUCCESS            46-55              18-25
16             171.0            SUCCESS            46-55              18-25
...              ...                ...              ...                ...
250117         750.0            SUCCESS            26-35              36-45
250127        1800.0            SUCCESS            26-35              36-45
250134         150.0            SUCCESS            26-35              26-35
250147         700.0            SUCCESS            26-35              36-45
250148        1000.0            SUCCESS            46-55              26-35

       sender_state sender_bank receiver_bank device_type network_type  \
0             Delhi        Axis           Sbi     Android           4G
3             Delhi       Icici           Pnb     Android           5G
4             Delhi        Axis      Yes Bank         Ios         Wifi
9             Delhi         Sbi          Hdfc     Android           4G
16            Delhi        Hdfc           Sbi     Android           5G
...             ...         ...           ...         ...          ...
250117        Delhi         Sbi          Hdfc     Android           5G
250127        Delhi        Hdfc         Icici     Android           4G
250134        Delhi        Hdfc         Icici     Android           4G
250147        Delhi        Hdfc          Hdfc         Ios           4G
250148        Delhi         Pnb         Icici     Android           4G

        fraud_flag  hour_of_day day_of_week  is_weekend     month  \
0                0         15.0     Tuesday           0   October
3                0         10.0      Sunday           1   January
4                0         19.0     Tuesday           0   January
9                0         15.0      Monday           0  November
16               0          7.0    Thursday           0   January
...            ...          ...         ...         ...       ...
250117           0         21.0      Friday           0      July
250127           0         11.0   Wednesday           0       May
250134           0          8.0     Tuesday           0       May
250147           0          9.0      Friday           0      June
250148           0         12.0    Saturday           1      June

          time_of_day
0                Noon
3             Morning
```

```
4             Evening
9                Noon
16      Early Morning
...               ...
250117          Night
250127        Morning
250134        Morning
250147        Morning
250148           Noon

[24368 rows x 19 columns]

☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️
☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️☀️

🔷 Filter by Morning transaction :
        transaction id           timestamp transaction type merchant_category  \
3       TXN0000000004 2024-01-07 10:09:00              P2P              Fuel
6       TXN0000000007 2024-02-08 10:25:00              P2P             Other
8       TXN0000000009 2024-11-21 09:39:00              P2P             Other
11      TXN0000000012 2024-12-24 10:44:00              P2M             Other
20      TXN0000000021 2024-05-19 10:41:00     Bill Payment              Food
...               ...                 ...              ...               ...
250134  TXN0000250132 2024-05-28 08:45:00              P2M              Food
250138  TXN0000250136 2024-05-31 09:00:00     Bill Payment         Utilities
250142  TXN0000250140 2024-06-03 11:45:00              P2M           Grocery
250147  TXN0000250145 2024-06-07 09:20:00              P2P         Transport
250150  TXN0000250148 2024-06-09 10:00:00              P2M          Shopping

        amount (INR) transaction_status sender_age_group receiver_age_group  \
3             2784.0            SUCCESS            26-35              26-35
6              314.0            SUCCESS            36-45              18-25
8              887.0            SUCCESS            46-55              36-45
11             518.0            SUCCESS            26-35              18-25
20            1341.0            SUCCESS            36-45              26-35
...              ...                ...              ...                ...
250134         150.0            SUCCESS            26-35              26-35
250138         950.0             FAILED            18-25              26-35
250142         200.0            SUCCESS            26-35              46-55
250147         700.0            SUCCESS            26-35              36-45
250150         500.0             FAILED            26-35              36-45

       sender_state sender_bank receiver_bank device_type network_type  \
3             Delhi       Icici           Pnb     Android           5G
6         Telangana        Hdfc      Indusind     Android           4G
8       Maharashtra       Kotak          Hdfc     Android           4G
11          Gujarat    Yes Bank           Sbi     Android           4G
20        Karnataka         Sbi         Icici     Android         Wifi
...             ...         ...           ...         ...          ...
250134        Delhi        Hdfc         Icici     Android           4G
250138      Gujarat         Sbi         Icici     Android           4G
250142    Karnataka        Hdfc         Icici     Android         Wifi
250147        Delhi        Hdfc          Hdfc         Ios           4G
250150   Tamil Nadu         Sbi          Axis     Android           3G

        fraud_flag  hour_of_day day_of_week  is_weekend     month time_of_day
3                0         10.0      Sunday           1   January     Morning
6                0         10.0    Thursday           0  February     Morning
8                0          9.0    Thursday           0  November     Morning
11               0         10.0     Tuesday           0  December     Morning
```

```
20              0     10.0     Sunday     1    May   Morning
...           ...      ...        ...   ...    ...       ...
250134          0      8.0    Tuesday     0    May   Morning
250138          1      9.0     Friday     0    May   Morning
250142          0     11.0     Monday     0   June   Morning
250147          0      9.0     Friday     0   June   Morning
250150          1     10.0     Sunday     1   June   Morning

[48074 rows x 19 columns]
```

In [14]:

```
#Aggregations
#Total transactionamount per merchant category
merchant_summary = df.groupby("merchant_category")["amount (INR)"].sum()
print("\n🔷Total transactionamount per merchant category  :\n",merchant_summary
print("🌟" * 50)



#Count the transactions per day of week
day_summary = df.groupby("day_of_week")["transaction id"].count()
print("\n🔷Count the transactions per day of week :\n",day_summary,"\n")
print("🌟" * 50)


#Average transaction amount per sender bank
bank_avg = df.groupby("sender_bank")["amount (INR)"].mean()
print("\n🔷Average transaction amount per sender bank :\n",bank_avg,"\n")
print("🌟" * 50)

# Transaction count by network type and device
network_summary = df.groupby(["network_type", "device_type"])["transaction id"].
print("\n🔷Transaction count by network type and device :\n",network_summary,"\
print("🌟" * 50)

#monthly trend of total transaction value
monthly_summary = df.groupby("month")["amount (INR)"].sum()
print("\n🔷monthly trend of total transaction value :\n",monthly_summary,"\n")
print("🌟" * 50)
```

```
🔷Total transactionamount per merchant category  :
 merchant_category
Education        22651009.75
Entertainment     8310253.00
Food             19893984.50
Fuel             38984925.00
Grocery          58270672.00
Healthcare        6881109.00
Other            20962610.75
Shopping         66881846.05
Transport         6135307.00
Utilities        48225993.00
Name: amount (INR), dtype: float64
```

🌟🌟

```
🔷Count the transactions per day of week :
 day_of_week
Friday       34826
Monday       35731
Saturday     34624
Sunday       35320
Thursday     34716
Tuesday      34871
Wednesday    35001
Name: transaction id, dtype: int64
```

🌟🌟

```
🔷Average transaction amount per sender bank :
 sender_bank
Axis         1203.464742
Hdfc         1216.162294
Icici        1210.855470
Indusind     1199.768660
Kotak        1209.173017
Pnb          1204.439180
Sbi          1220.994061
Yes Bank     1221.393023
Name: amount (INR), dtype: float64
```

🌟🌟

```
🔷Transaction count by network type and device :
 network_type  device_type
3G            Android          9264
              Ios              2337
              Web               624
4G            Android        110248
              Ios             29299
              Web              7363
5G            Android         46097
              Ios             12111
              Web              3113
Wifi          Android         18474
              Ios              4890
              Web              1269
```

```
Name: transaction id, dtype: int64

🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟

🔷monthly trend of total transaction value :
 month
April        24460997.50
August       25580870.75
December     24626826.00
February     23679946.75
January      24894311.25
July         25293026.75
June         24501783.50
March        25073327.75
May          25327349.55
November     24172881.00
October      25252578.00
September    24333811.25
Name: amount (INR), dtype: float64

🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟🌟
```

In [15]:
```
#df.to_csv("upi_transactions_c.csv",index=False)
```

Exploratory Data Analysis(EDA)

🔺 Descrptive Statistics

In [16]:
```
#Statistical summary of Numerical columns
df.describe().T
```

Out[16]:

	count	mean	min	25%	50%	75%	max
timestamp	245089	2024-07-01 15:12:42.165662208	2024-01-01 00:05:00	2024-04-01 15:09:00	2024-07-01 13:22:00	2024-09-30 17:28:00	2024-12-30 23:55:00
amount (INR)	245089.0	1212.611378	48.0	293.0	629.0	1563.0	9012.0
fraud_flag	245089.0	0.001946	0.0	0.0	0.0	0.0	1.0
hour_of_day	245089.0	14.679521	0.0	11.0	15.0	19.0	23.0
is_weekend	245089.0	0.285382	0.0	0.0	0.0	1.0	1.0

In [17]:
```
#Statistical summary of all columns

df.describe(include='object').T
```

Out[17]:

	count	unique	top	freq
transaction id	245089	245089	TXN0000000001	1
transaction type	245089	4	P2P	110224
merchant_category	245089	10	Grocery	49961
transaction_status	245089	2	SUCCESS	232971
sender_age_group	245089	5	26-35	85827
receiver_age_group	245089	5	26-35	86188
sender_state	245089	10	Maharashtra	36714
sender_bank	245089	8	Sbi	61453
receiver_bank	245089	8	Sbi	61126
device_type	245089	3	Android	184083
network_type	245089	4	4G	146910
day_of_week	245089	7	Monday	35731
month	245089	12	May	20939
time_of_day	245089	6	Evening	72158

🔺 Unique Value Counts

In [18]:

```
print("\n🔷 Unique Value in Each Columns :\n")

for col in df.columns:
    print(col,":",df[col].nunique(),"Unique Values\n")
```

🔷 Unique Value in Each Columns :

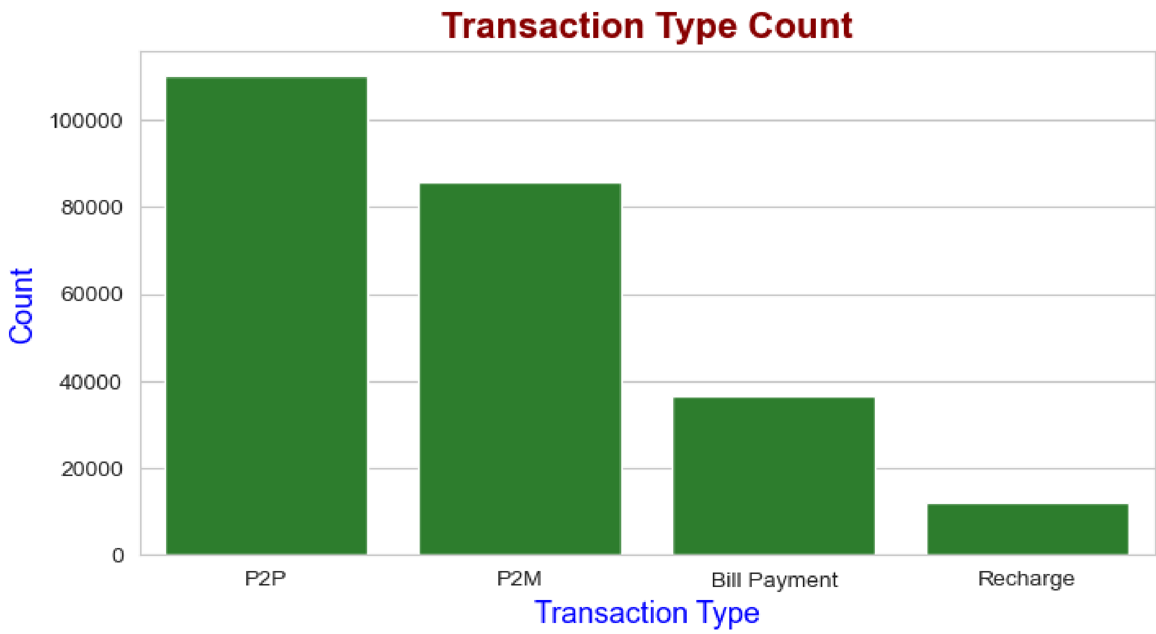
```
transaction id : 245089 Unique Values

timestamp : 181308 Unique Values

transaction type : 4 Unique Values

merchant_category : 10 Unique Values

amount (INR) : 8239 Unique Values

transaction_status : 2 Unique Values

sender_age_group : 5 Unique Values

receiver_age_group : 5 Unique Values

sender_state : 10 Unique Values

sender_bank : 8 Unique Values

receiver_bank : 8 Unique Values

device_type : 3 Unique Values

network_type : 4 Unique Values

fraud_flag : 2 Unique Values

hour_of_day : 24 Unique Values

day_of_week : 7 Unique Values

is_weekend : 2 Unique Values

month : 12 Unique Values

time_of_day : 6 Unique Values
```

🔺 Univariate Analysis

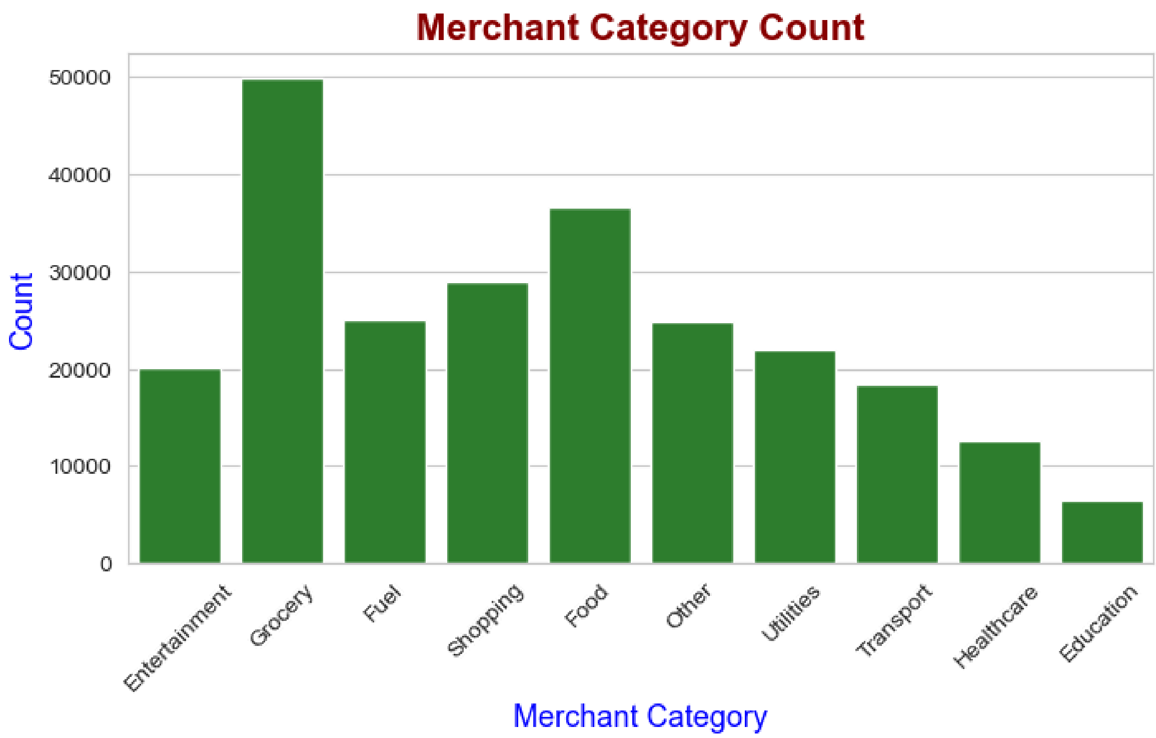
In [19]:
```
#Categorical Columns

sns.set_style("whitegrid")

categorical_cols = [
    "transaction type", "merchant_category", "transaction_status",
    "sender_age_group", "receiver_age_group","sender_state","receiver_bank",
    "device_type", "network_type", "day_of_week","month", "time_of_day" ]


for col in categorical_cols:
    plt.figure(figsize=(8,4))
    sns.countplot(x=col, data=df, color="#228B22")
    plt.title(col.replace("_"," ").title()+" Count", color="darkred", fontsize=1
    plt.xlabel(col.replace("_"," ").title(), color="blue", fontsize=13)
    plt.ylabel("Count", color="blue", fontsize=13)
```

```
    print("☀" * 50)

    if df[col].nunique() > 5:
        plt.xticks(rotation=45)

    plt.show()
```

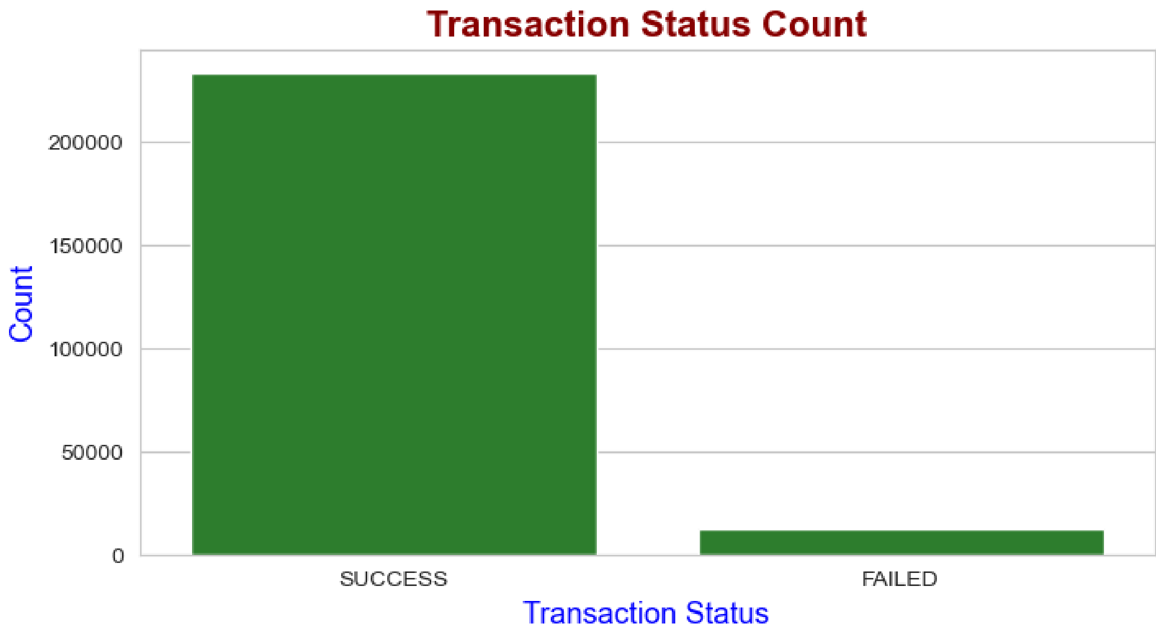
☀☀



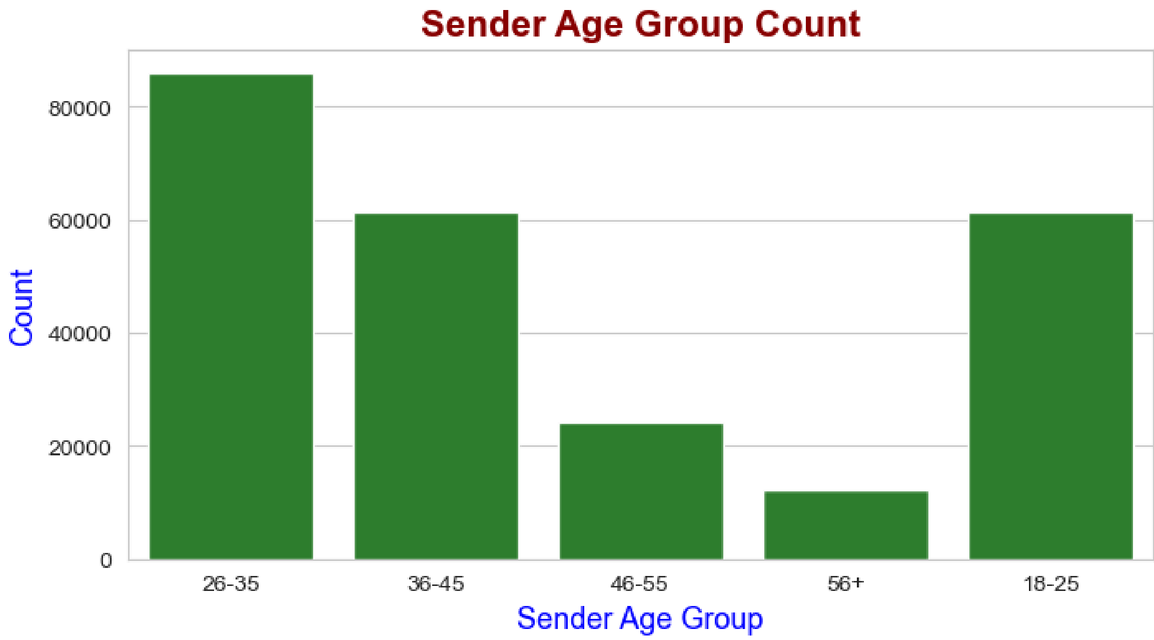
☀☀



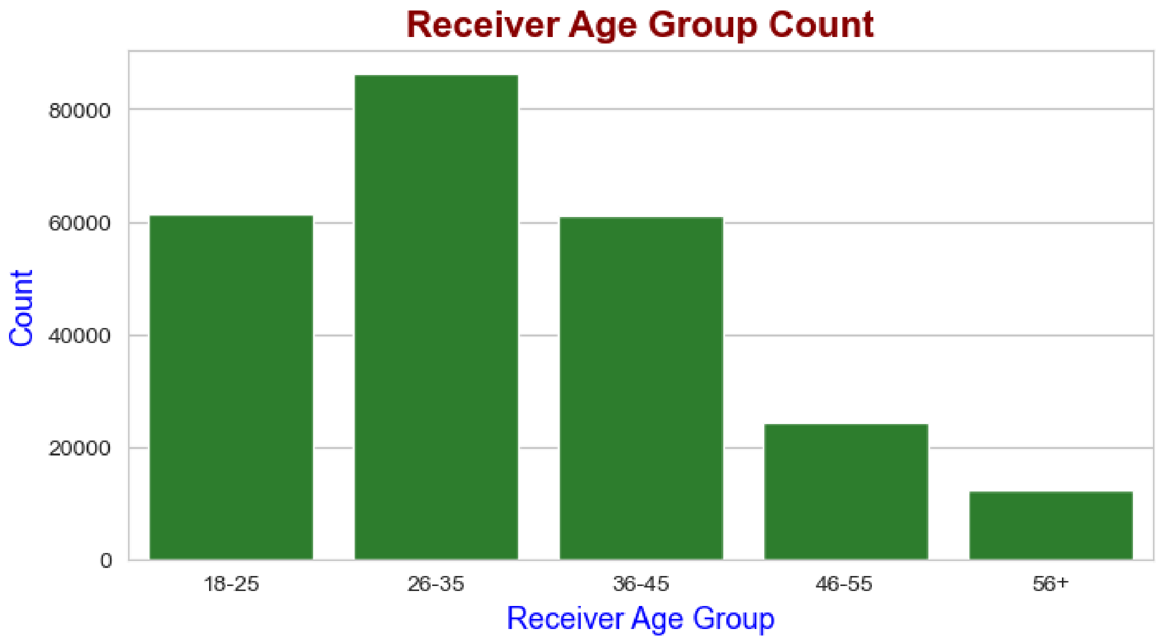
☀☀


Transaction Status Count
Count
200000
150000
100000
50000
0
SUCCESS
FAILED
Transaction Status

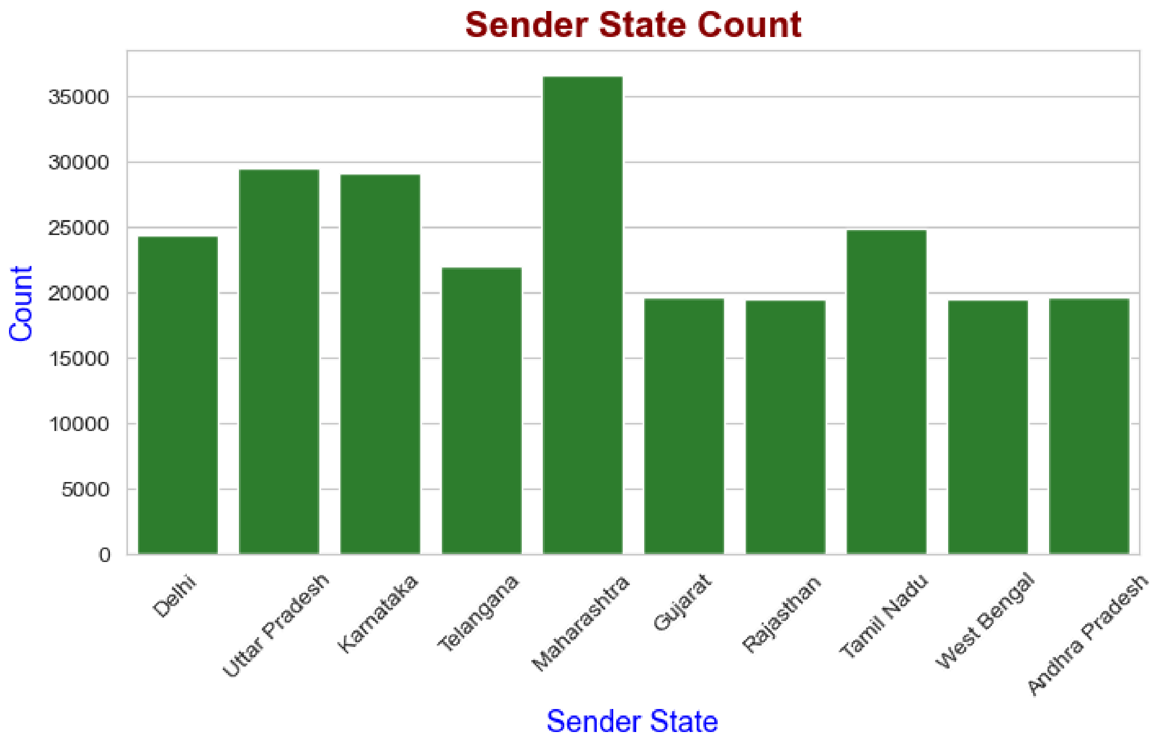
🌟🌟


Sender Age Group Count
Count
80000
60000
40000
20000
0
26-35
36-45
46-55
56+
18-25
Sender Age Group

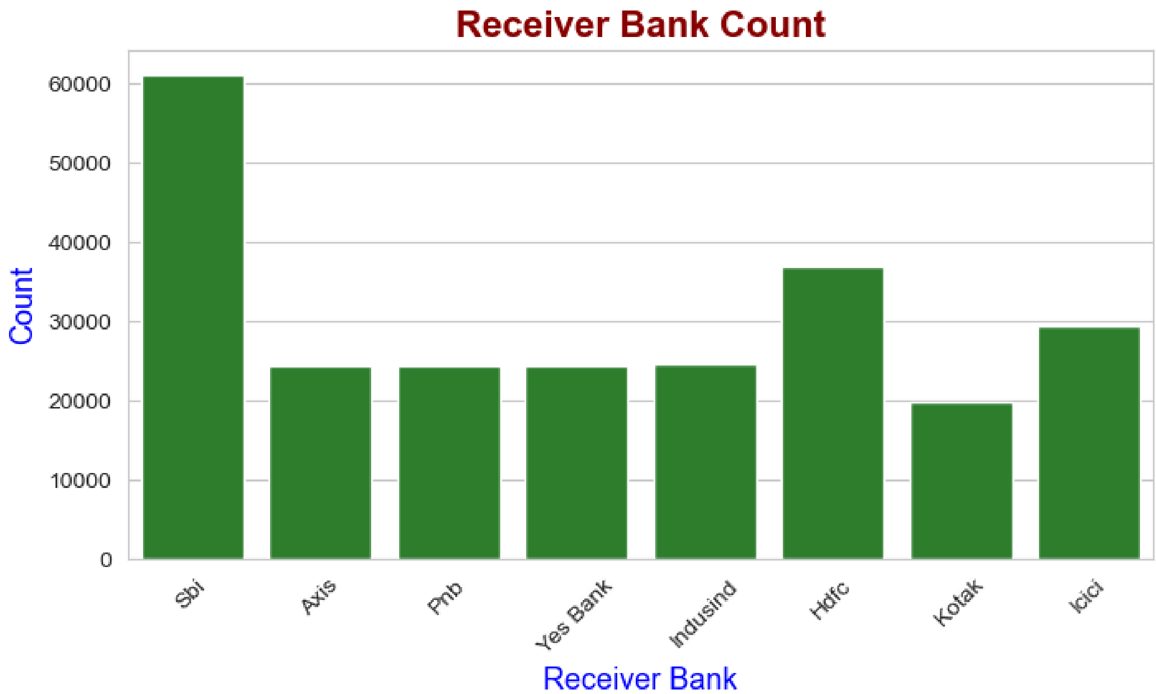
🌟🌟


Receiver Age Group Count
Count
80000
60000
40000
20000
0
18-25
26-35
36-45
46-55
56+
Receiver Age Group

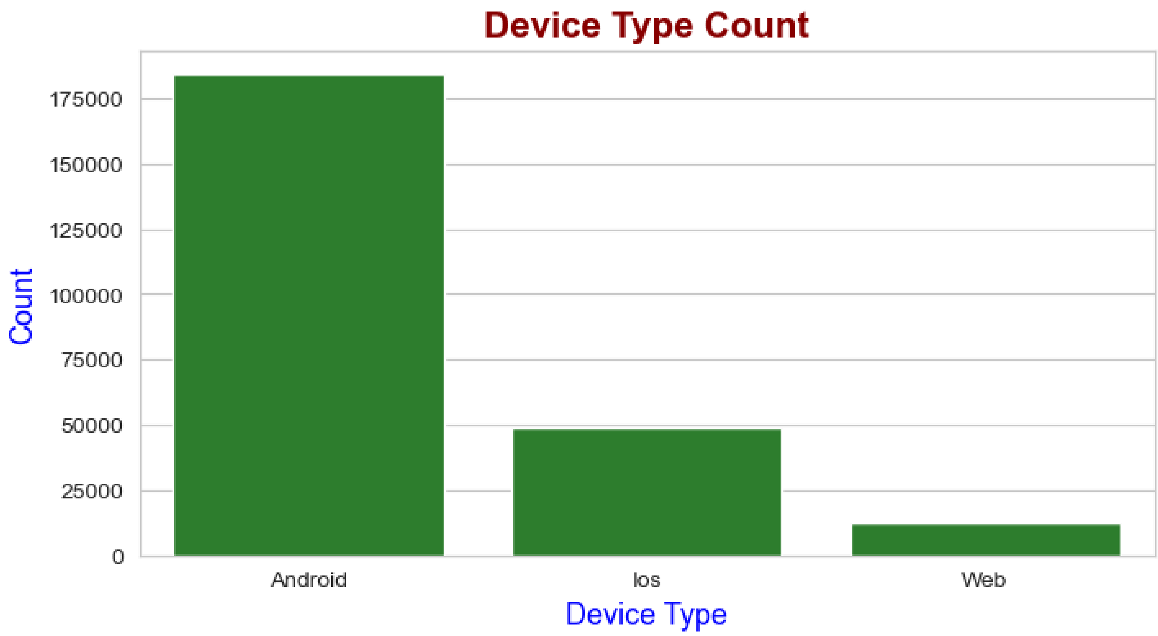
🌟🌟


Sender State Count
Count
35000
30000
25000
20000
15000
10000
5000
0
Delhi
Uttar Pradesh
Karnataka
Telangana
Maharashtra
Gujarat
Rajasthan
Tamil Nadu
West Bengal
Andhra Pradesh
Sender State

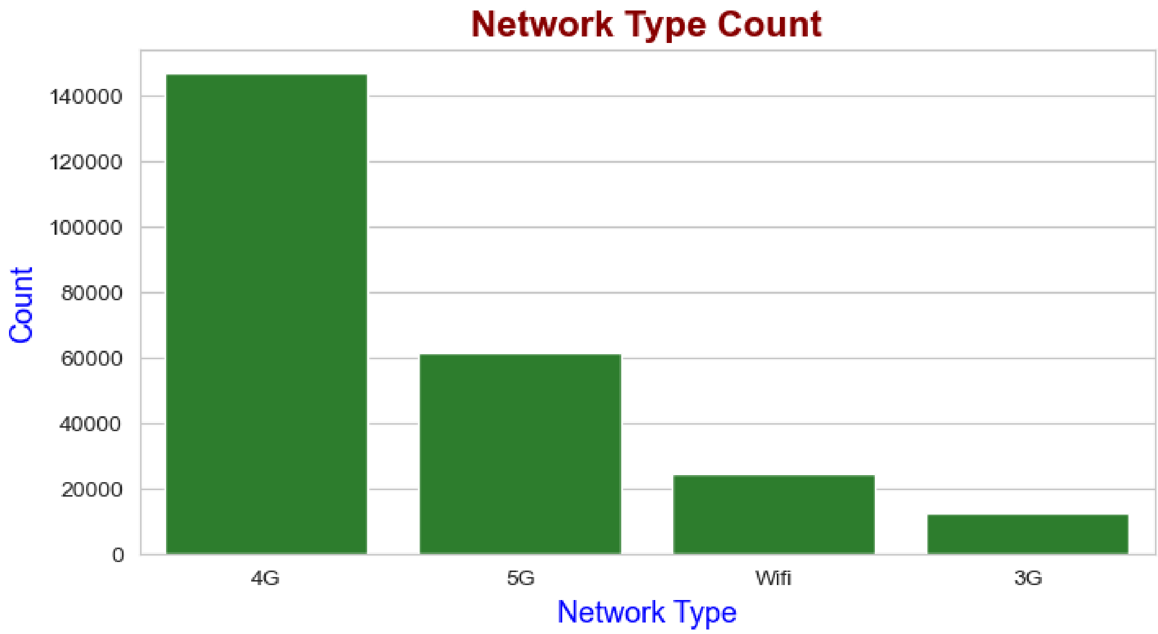
🌟🌟

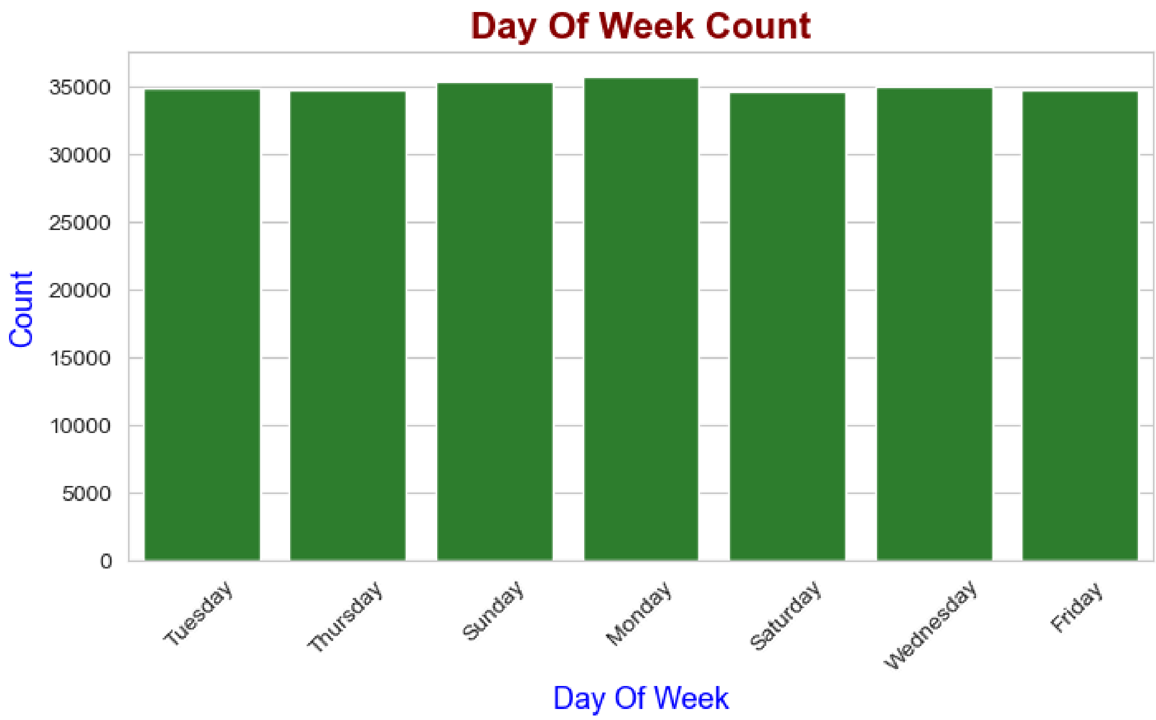

Receiver Bank Count
Count
60000
50000
40000
30000
20000
10000
0
Sbi
Axis
Pnb
Yes Bank
Indusind
Hdfc
Kotak
Icici
Receiver Bank

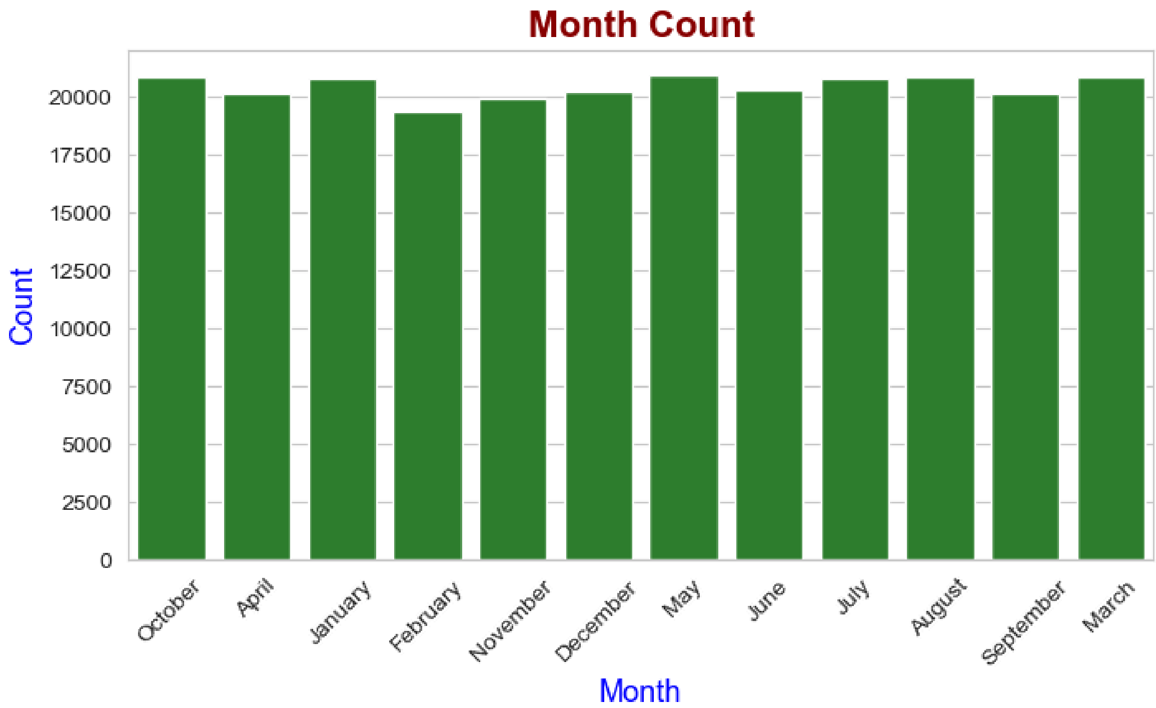
🌟🌟


Device Type Count
Count
175000
150000
125000
100000
75000
50000
25000
0
Android
Ios
Web
Device Type

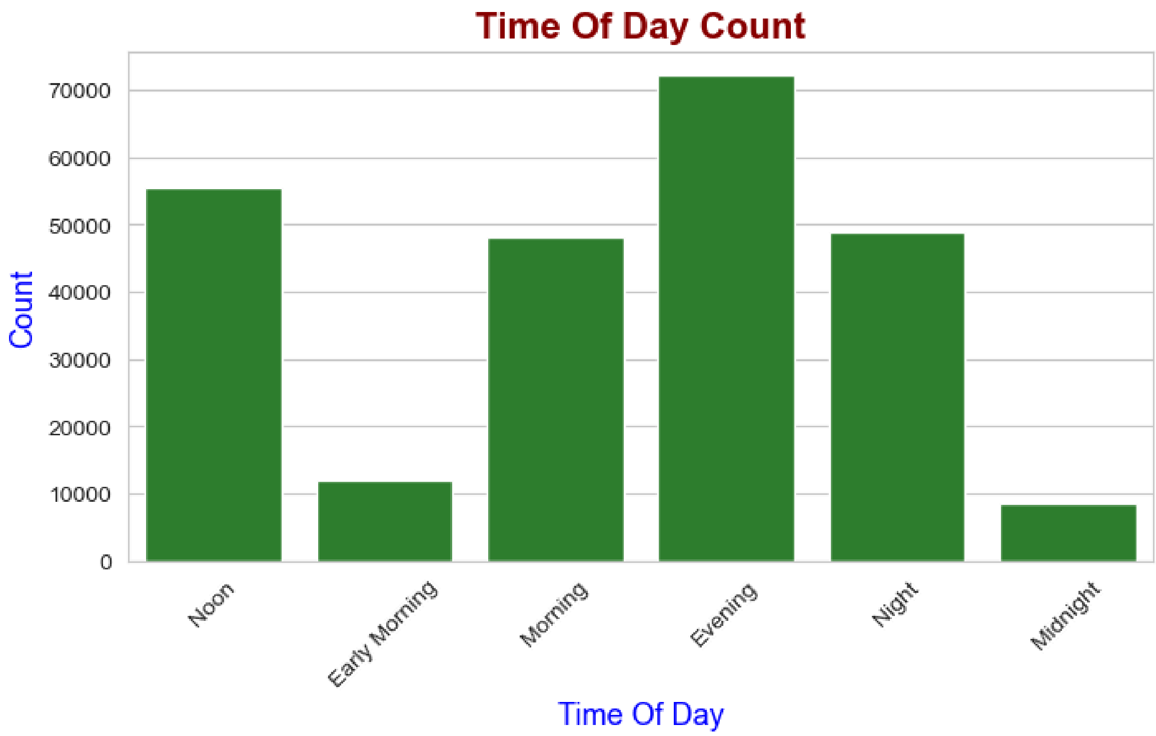
🌟🌟


Network Type Count
Count
140000
120000
100000
80000
60000
40000
20000
0
4G
5G
Wifi
3G
Network Type


Day Of Week Count
Count
35000
30000
25000
20000
15000
10000
5000
0
Tuesday
Thursday
Sunday
Monday
Saturday
Wednesday
Friday
Day Of Week

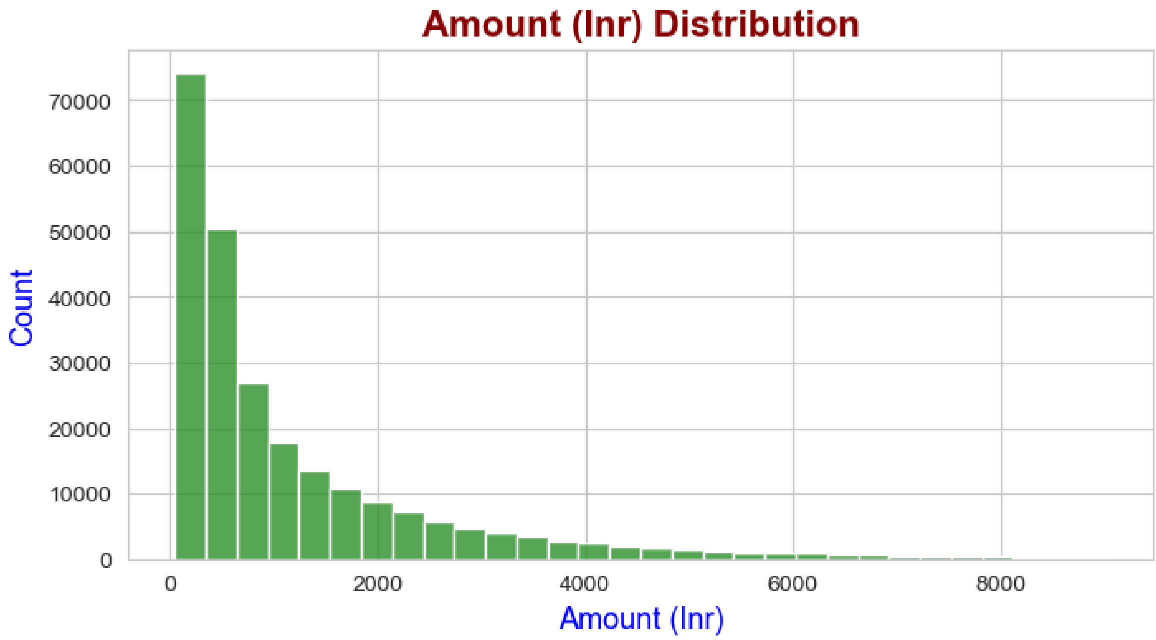


🌞🌞

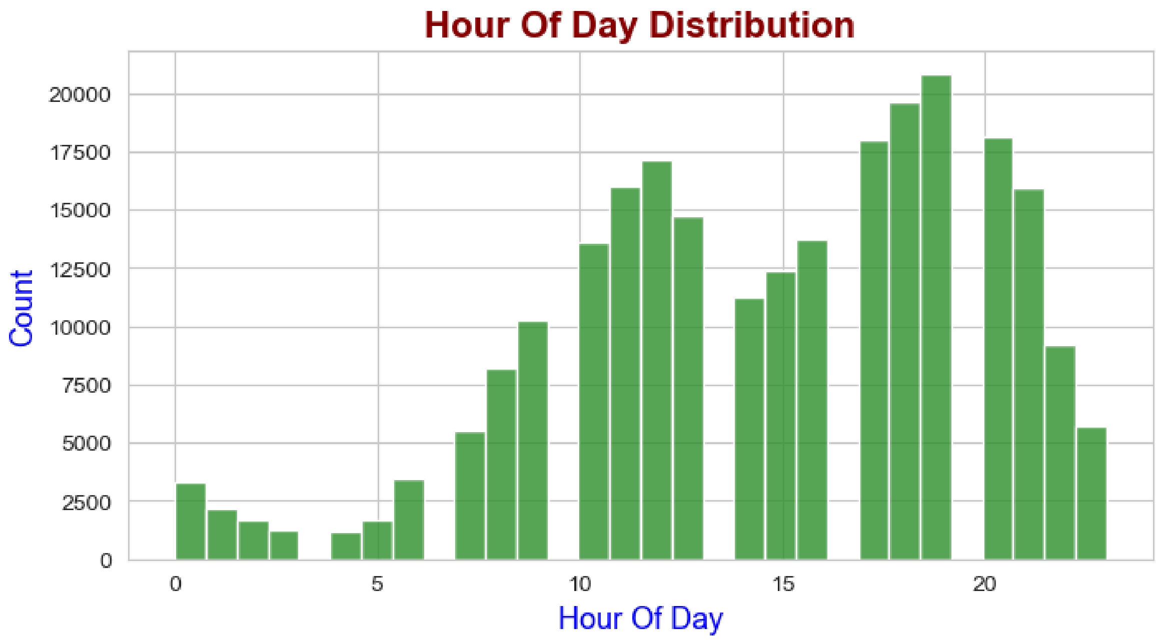


In [20]:
```
# Numerical Columns
numerical_cols = ["amount (INR)", "hour_of_day"]

for col in numerical_cols:
    plt.figure(figsize=(8,4))
    sns.histplot(df[col], color="#228B22", bins=30, kde=False)
    plt.title(col.replace("_", " ").title() + " Distribution", color="darkred",
    plt.xlabel(col.replace("_", " ").title(), color="blue", fontsize=13)
    plt.ylabel("Count", color="blue", fontsize=13)
    print("🌞" * 50)

    plt.show()
```

🌟🌟

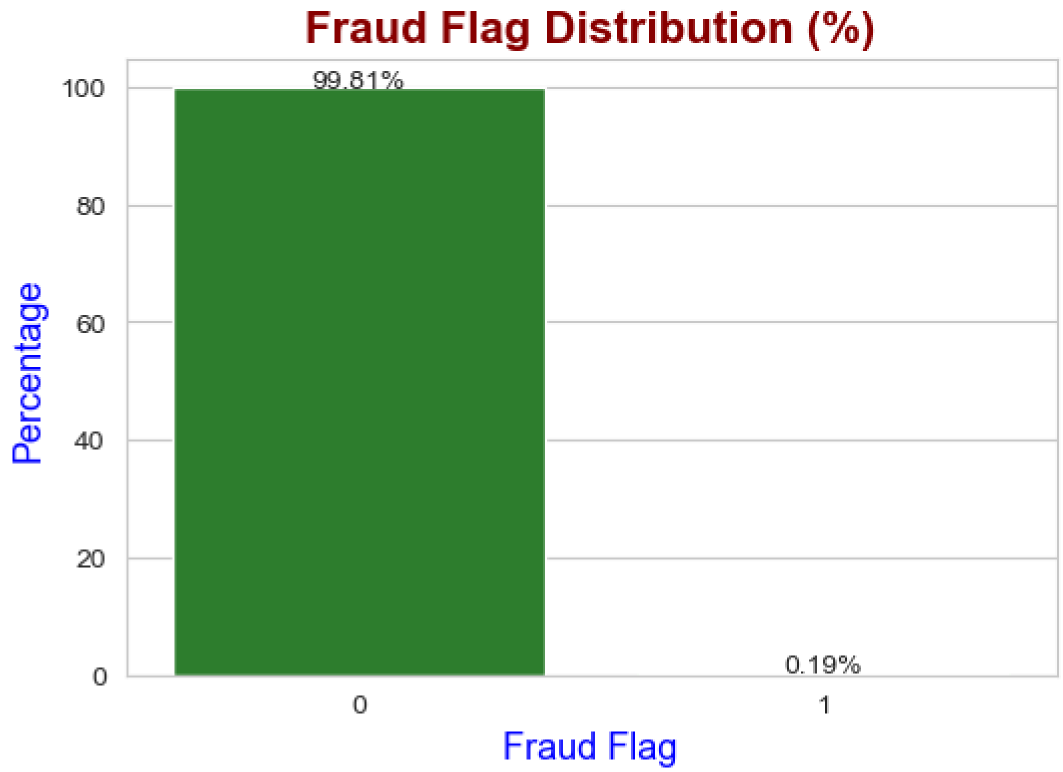


🌟🌟

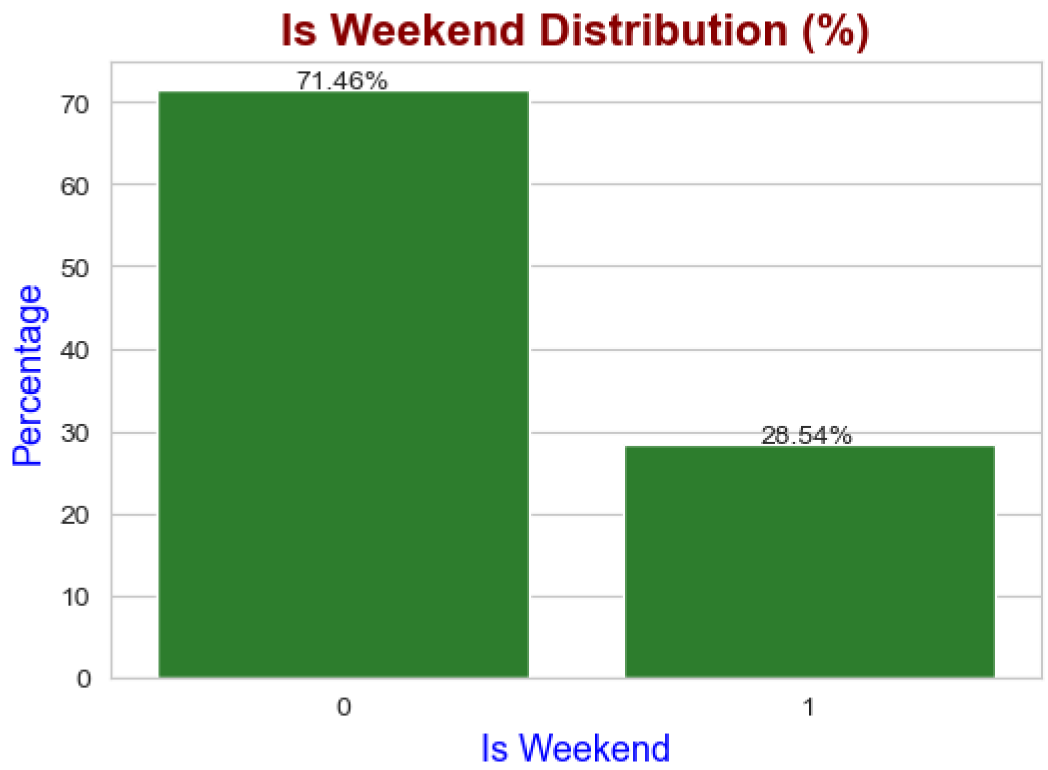


In [21]:
```
#Numerical binary values
binary_cols = ["fraud_flag", "is_weekend"]

for col in binary_cols:
    plt.figure(figsize=(6,4))

    percent = df[col].value_counts(normalize=True) * 100

    sns.barplot(x=percent.index, y=percent.values, color="#228B22")
    plt.title(col.replace("_", " ").title() + " Distribution (%)", color="darkre
    plt.xlabel(col.replace("_", " ").title(), color="blue", fontsize=13)
    plt.ylabel("Percentage", color="blue", fontsize=13)


    for i, val in enumerate(percent.values):
        plt.text(i, val + 0.2, "{0:.2f}%".format(val), ha='center')
```

```
plt.show()
print("🌟" * 50)
```



🌟🌟

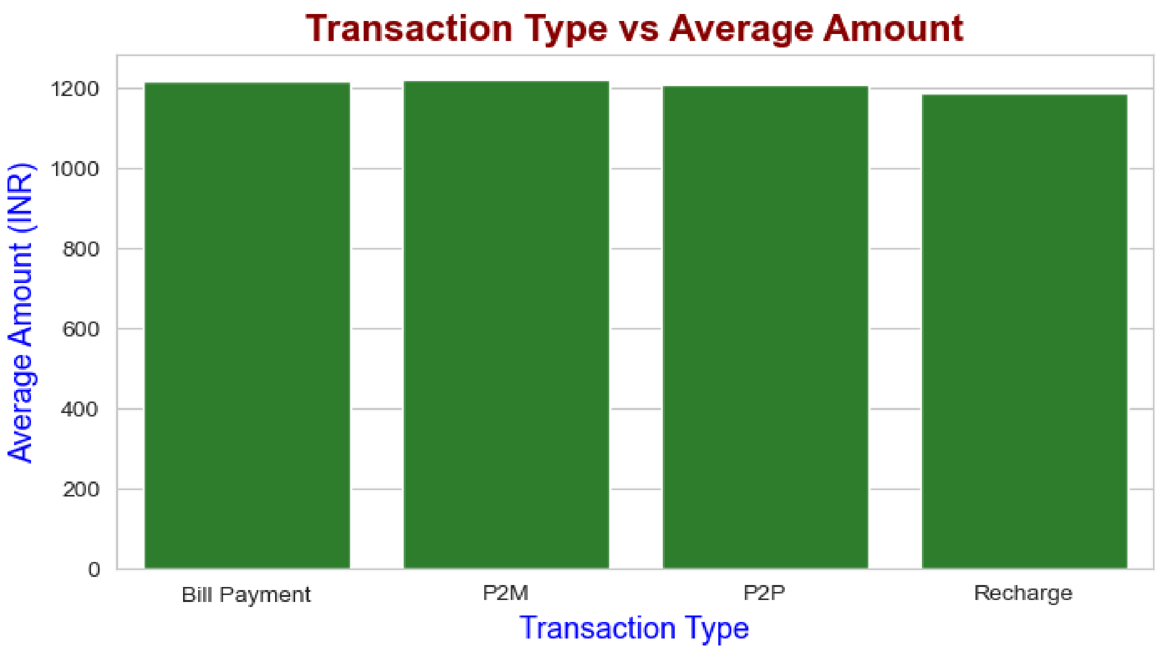


🌟🌟

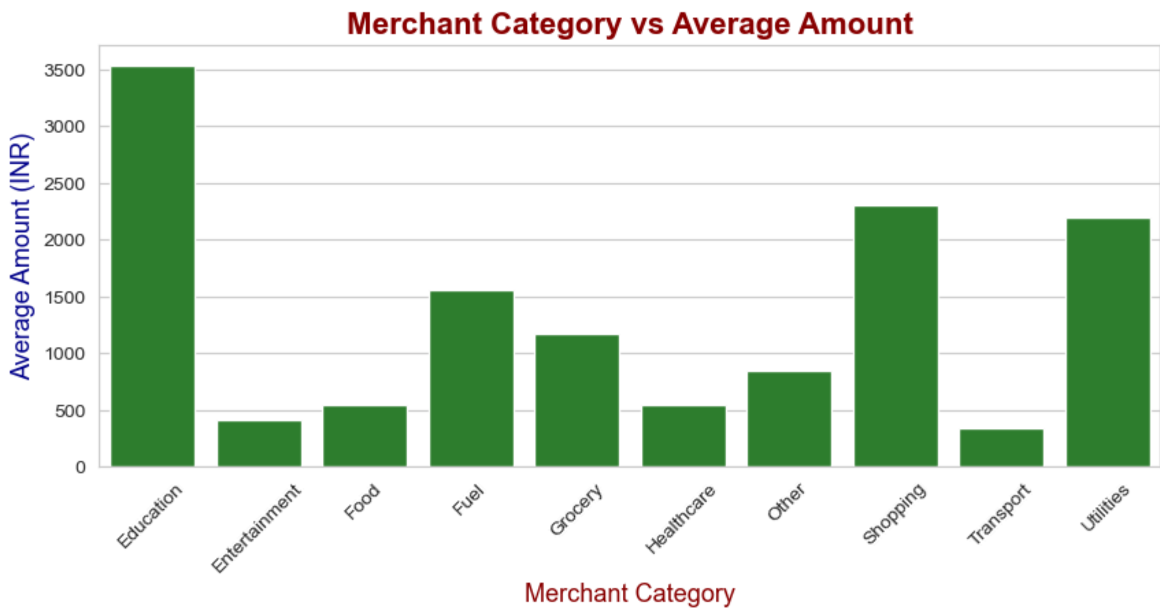
▲ Bivariate Analysis

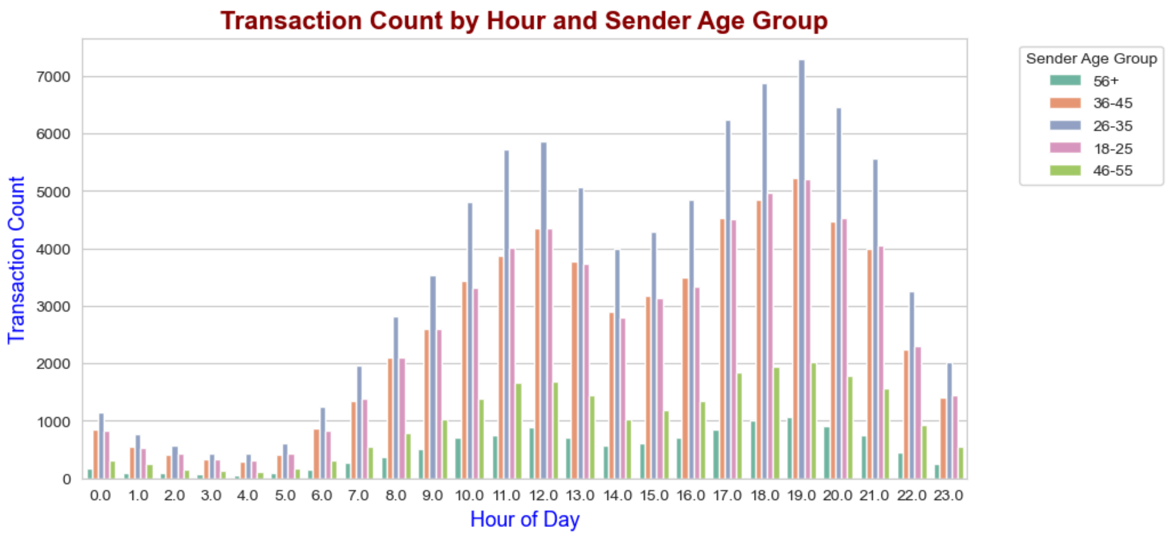
In [22]:
```
#Transaction Type vs Amount
mean_amount = df.groupby("transaction type")["amount (INR)"].mean()
```

```
plt.figure(figsize=(8,4))
sns.barplot(x=mean_amount.index, y=mean_amount.values, color="#228B22")
plt.title("Transaction Type vs Average Amount", color="darkred", fontsize=16, fc
plt.xlabel("Transaction Type", color="blue", fontsize=13)
plt.ylabel("Average Amount (INR)", color="blue", fontsize=13)
plt.show()
print("🌟" * 50)

#Merchant Category vs Amount
mean_merchant = df.groupby("merchant_category")["amount (INR)"].mean()

plt.figure(figsize=(10,4))
sns.barplot(x=mean_merchant.index, y=mean_merchant.values, color="#228B22")
plt.title("Merchant Category vs Average Amount", color="darkred", fontsize=16, f
plt.xlabel("Merchant Category", color="#8B0000", fontsize=13)
plt.ylabel("Average Amount (INR)", color="#00008B", fontsize=13)
plt.xticks(rotation=45)
plt.show()
print("🌟" * 50)

#Sender Age Group vs Hour of Day

plt.figure(figsize=(10,5))
sns.countplot(x="hour_of_day", hue="sender_age_group", data=df, palette="Set2")
plt.title("Transaction Count by Hour and Sender Age Group", color="darkred", fon
plt.xlabel("Hour of Day", color="blue", fontsize=13)
plt.ylabel("Transaction Count", color="blue", fontsize=13)
plt.legend(title="Sender Age Group", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
print("🌟" * 50)
```



🌟🌟

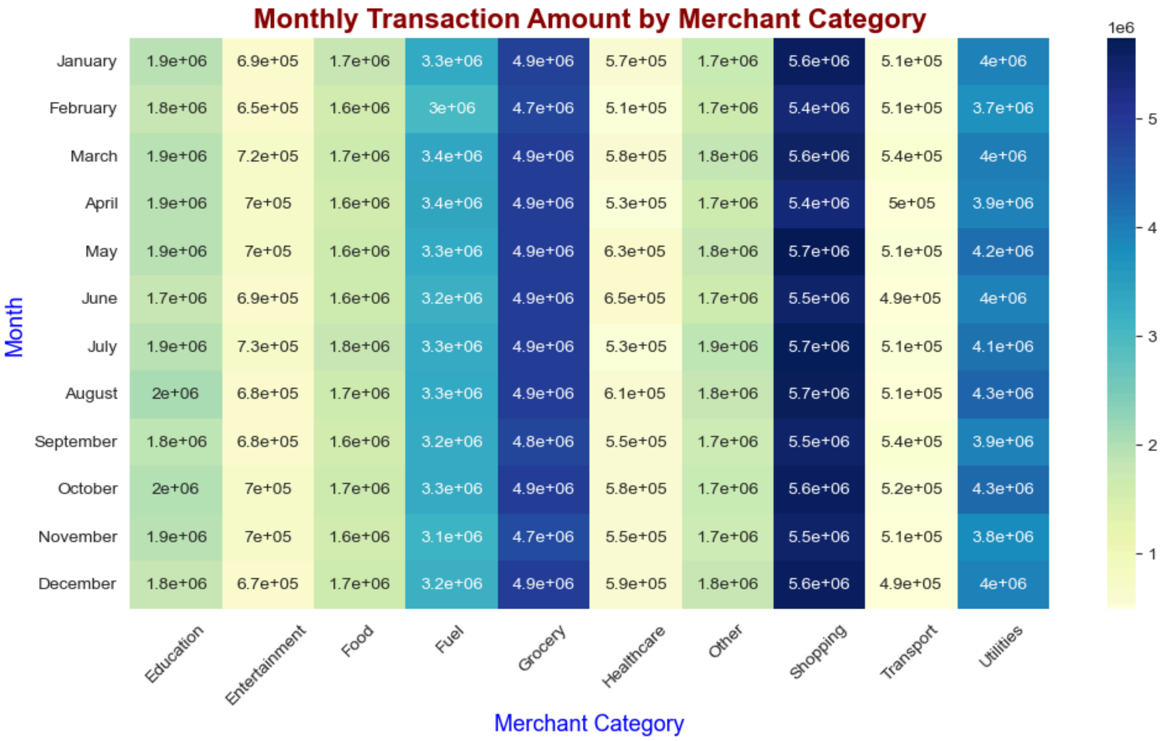




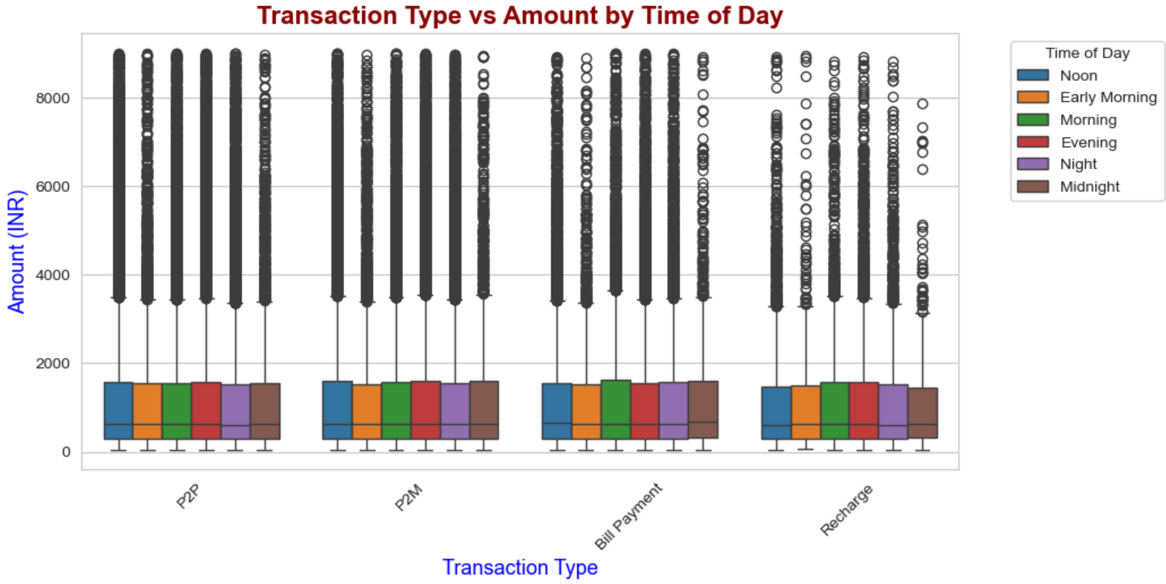
▲ Multivariate Analysis

In [23]:
```
#Monthly Transaction Amount by Merchant Category#
pivot = pd.pivot_table(df,
                       values='amount (INR)',
                       index='month',
                       columns='merchant_category',
                       aggfunc='sum')

# Ensure months are ordered Jan → Dec
month_order = ["January", "February", "March", "April", "May", "June",
               "July", "August", "September", "October", "November", "December"]

pivot = pivot.reindex(month_order)

plt.figure(figsize=(12,6))
sns.heatmap(pivot, cmap="YlGnBu", annot=True)
plt.title("Monthly Transaction Amount by Merchant Category", color="darkred", fc
plt.xlabel("Merchant Category", color="blue", fontsize=13)
plt.ylabel("Month", color="blue", fontsize=13)
```

```
plt.xticks(rotation=45)
plt.show()
```



In [24]:
```
#Transaction Type vs Amount by Time of Day
plt.figure(figsize=(10,5))
sns.boxplot(x="transaction type", y="amount (INR)", hue="time_of_day", data=df)
plt.title("Transaction Type vs Amount by Time of Day", color="darkred", fontsize
plt.xlabel("Transaction Type", color="blue", fontsize=13)
plt.ylabel("Amount (INR)", color="blue", fontsize=13)
plt.xticks(rotation=45)

plt.legend(title="Time of Day", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



Groupby and Pivot Table Analysis

In [25]:
```
#Groupby : Total transaction value bybank
bank_tran = df.groupby('receiver_bank')['amount (INR)'].sum().sort_values(ascend
```

```
print(bank_tran.head())
```

```
receiver_bank
Sbi          74396497.75
Hdfc         44953335.75
Icici        35380010.75
Axis         29896200.80
Indusind     29830405.00
Name: amount (INR), dtype: float64
```

In [26]:
```
#Groupby :Average transaction amount per merchant category
avg_tran_merchant = df.groupby('merchant_category')['amount (INR)'].mean().sort_

print(avg_tran_merchant.head(10))
```

```
merchant_category
Education        3538.667357
Shopping         2306.668255
Utilities        2196.383522
Fuel             1555.291032
Grocery          1166.323172
Other             844.857760
Healthcare        543.145394
Food              542.262504
Entertainment     413.486566
Transport         332.663178
Name: amount (INR), dtype: float64
```

In [27]:
```
# Pivot table: Transaction trends by merchant & month
pivot_table = pd.pivot_table(df, values='amount (INR)',
                             index='merchant_category',
                             columns='month',
                             aggfunc='sum')
print(pivot_table)
```

```
month                  April      August   December    February     January  \
merchant_category                                                            
Education          1902232.0  2041487.00  1777016.0  1776474.75  1889211.00   
Entertainment       695801.0   678329.00   667442.0   654571.00   693161.00   
Food               1610082.0  1694035.00  1652750.0  1603736.00  1747537.25   
Fuel               3378771.0  3332782.00  3171524.0  3036091.00  3329627.00   
Grocery            4904982.0  4916841.00  4901921.0  4748529.00  4855922.00   
Healthcare          529019.0   605371.00   586400.0   512463.00   569188.00   
Other              1666191.0  1799557.00  1779387.0  1689428.00  1730268.00   
Shopping           5350664.5  5671120.75  5636526.0  5410893.00  5600715.00   
Transport           497944.0   507860.00   488939.0   509887.00   510107.00   
Utilities          3925311.0  4333488.00  3964921.0  3737874.00  3968575.00   

month                    July       June       March         May   November  \
merchant_category                                                            
Education          1905645.00  1747391.0  1914665.00  1933089.00  1943268.0   
Entertainment       726798.00   687134.0   724867.00   703154.00   698944.0   
Food               1759773.50  1625742.0  1695308.00  1642789.75  1593132.0   
Fuel               3253395.00  3185297.0  3353334.00  3296524.00  3146485.0   
Grocery            4901294.00  4946283.0  4880023.00  4905442.00  4658830.0   
Healthcare          528521.00   654041.0   575007.00   633180.00   550513.0   
Other              1857266.25  1679673.0  1814158.50  1794221.00  1712110.0   
Shopping           5705714.00  5519105.0  5559696.00  5734273.80  5549107.0   
Transport           510685.00   489505.0   536260.00   512567.00   508027.0   
Utilities          4143935.00  3967612.5  4020009.25  4172109.00  3812465.0   

month                October   September  
merchant_category                         
Education          1975797.0  1844734.00  
Entertainment       700275.0   679777.00  
Food               1688751.0  1580348.00  
Fuel               3255874.0  3245221.00  
Grocery            4879369.0  4771236.00  
Healthcare          584772.0   552634.00  
Other              1730408.0  1709943.00  
Shopping           5641474.0  5502557.00  
Transport           522184.0   541342.00  
Utilities          4273674.0  3906019.25  
```

In [28]:

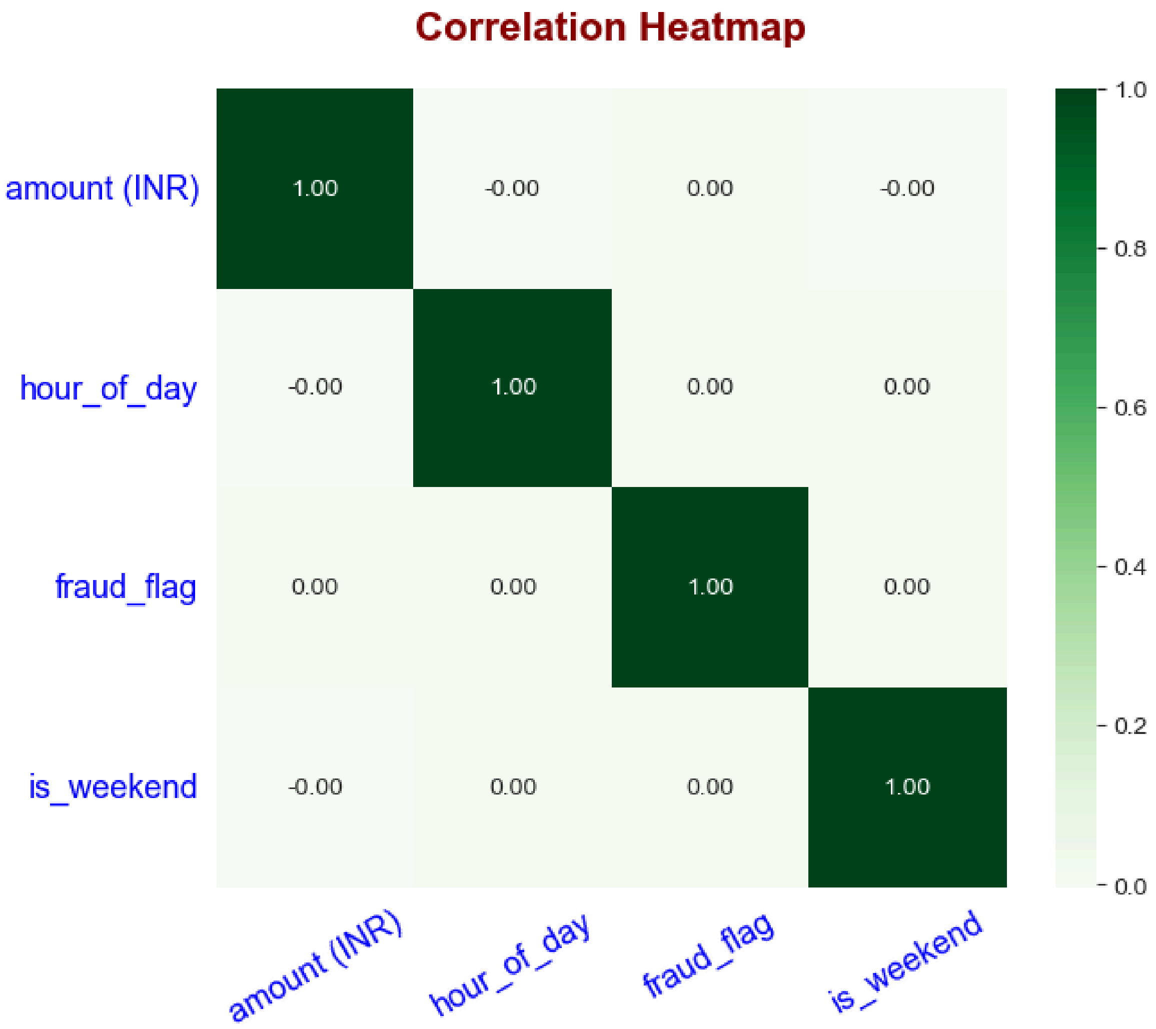
```python
#Pivot table : Transaction count per month
pivot_table2 = pd.pivot_table(df,
                              values='amount (INR)',
                              index='receiver_bank',
                              columns='month',
                              aggfunc='count')

print(pivot_table2.head())
```

```
month          April  August  December  February  January  July  June  March  \
receiver_bank
Axis            2043    2058      1957      1923     2089  2083  2008   2132
Hdfc            2960    3104      3098      3000     3149  3140  3076   3127
Icici           2424    2542      2424      2336     2518  2480  2402   2554
Indusind        1961    2072      2055      1989     2023  2119  1990   2080
Kotak           1662    1694      1696      1547     1642  1662  1630   1599

month           May  November  October  September
receiver_bank
Axis           2042      1979     2117       2053
Hdfc           3160      3044     3165       2941
Icici          2439      2345     2446       2437
Indusind       2157      2069     2102       2011
Kotak          1697      1597     1644       1662
```

🔺 Correlation Analysis

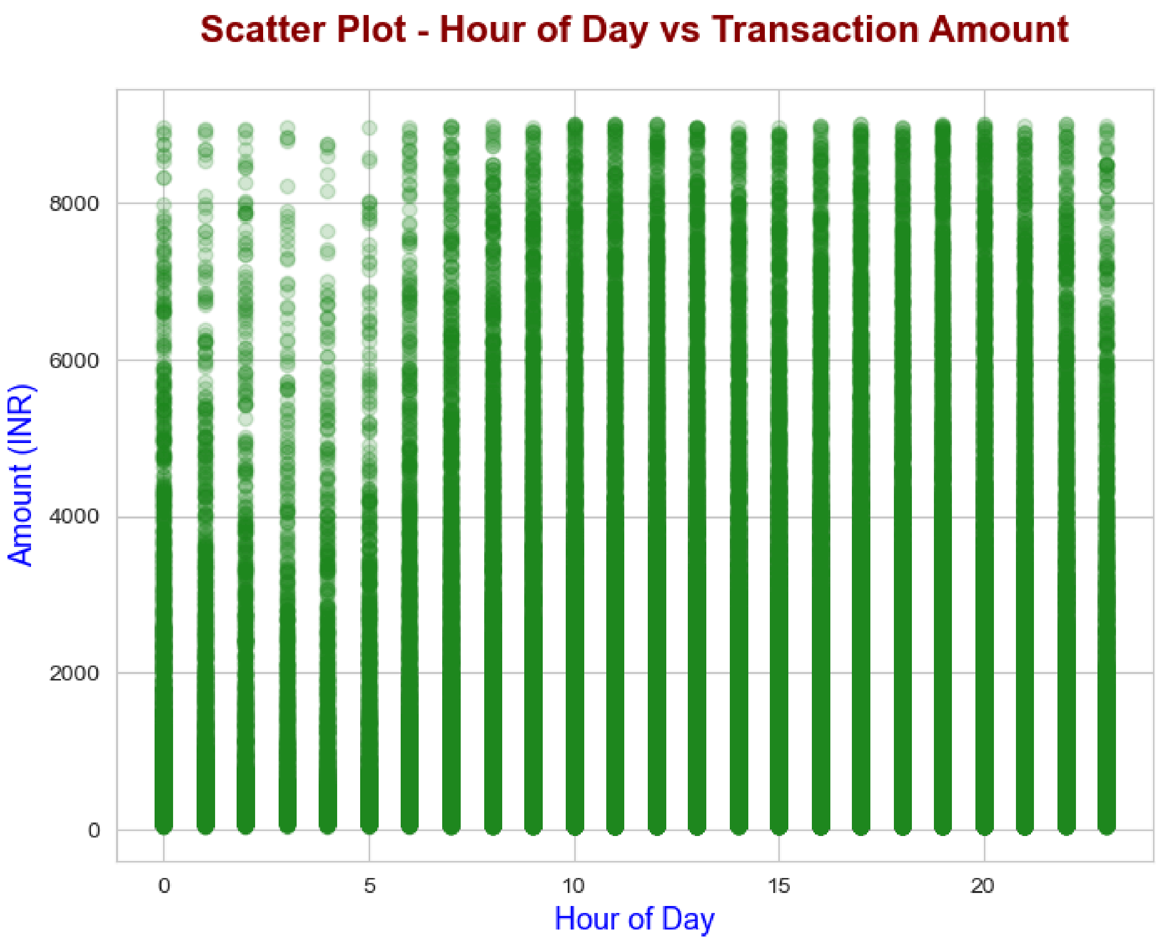
In [29]:
```python
numeric_cols = ['amount (INR)', 'hour_of_day', 'fraud_flag', 'is_weekend']
df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric)
corr_matrix = df[numeric_cols].corr()
print(corr_matrix)

plt.figure(figsize=(7,6))
sns.heatmap(corr_matrix, annot=True, cmap="Greens", fmt=".2f")

plt.title("Correlation Heatmap", color="darkred", fontsize=16, fontweight='bold'
plt.xticks(rotation=30, color="blue", fontsize=13)
plt.yticks(rotation=0, color="blue", fontsize=13)

plt.tight_layout()
plt.show()
```

```
              amount (INR)  hour_of_day  fraud_flag  is_weekend
amount (INR)      1.000000    -0.003357    0.002626   -0.000587
hour_of_day      -0.003357     1.000000    0.000604    0.001356
fraud_flag        0.002626     0.000604    1.000000    0.000794
is_weekend       -0.000587     0.001356    0.000794    1.000000
```



Outlier Analysis

In [30]:
```
plt.figure(figsize=(8,6))
plt.scatter(df['hour_of_day'], df['amount (INR)'],color="#228B22", alpha=0.2)

plt.title("Scatter Plot - Hour of Day vs Transaction Amount", color='darkred', f
plt.xlabel("Hour of Day", color='blue', fontsize=13)
plt.ylabel("Amount (INR)", color='blue', fontsize=13)

plt.show()
```



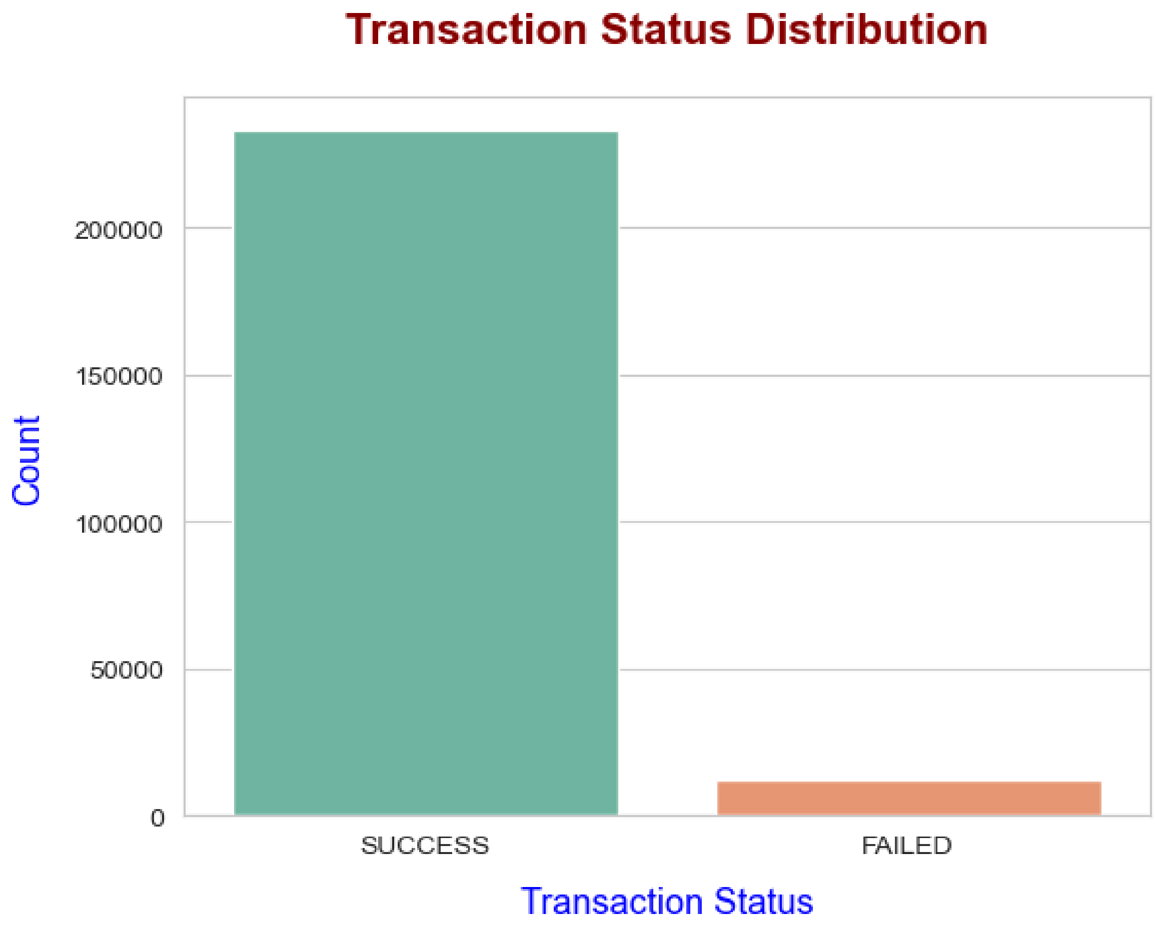
Finding and Insight of EDA

- They are mostly **P2P and grocery-related** and they tend to be mostly successful, **young adults (26–35)** are their largest remitting and receiving segment.
- Sender bank and receiver state are most frequent **Maharashtra and SBI** and medium is **Android + 4G**.
- The **evening and weekdays** are when the most transactions are occurring and the average transaction amount is that little bit higher.
- The amount of **fraudulent transactions is very low (0.19%)** and there is little correlation between the numeric features.
- Grocery and P2P dominate small-to-medium transactions, while **shopping** show higher-value spends.
- Festive months show spikes in both transaction count and amount, especially in **shopping and entertainment**.
- Majority of transactions are successful, while failures are mostly linked to **network/bank issues**.

Visualization

Transaction Status Distribution

In [31]:
```
sns.countplot(data=df, x="transaction_status", hue="transaction_status", palette
plt.title("Transaction Status Distribution", color='darkred', fontweight='bold',
plt.xlabel("Transaction Status", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Count", color='blue', fontsize=13, labelpad=10)
plt.show()
```

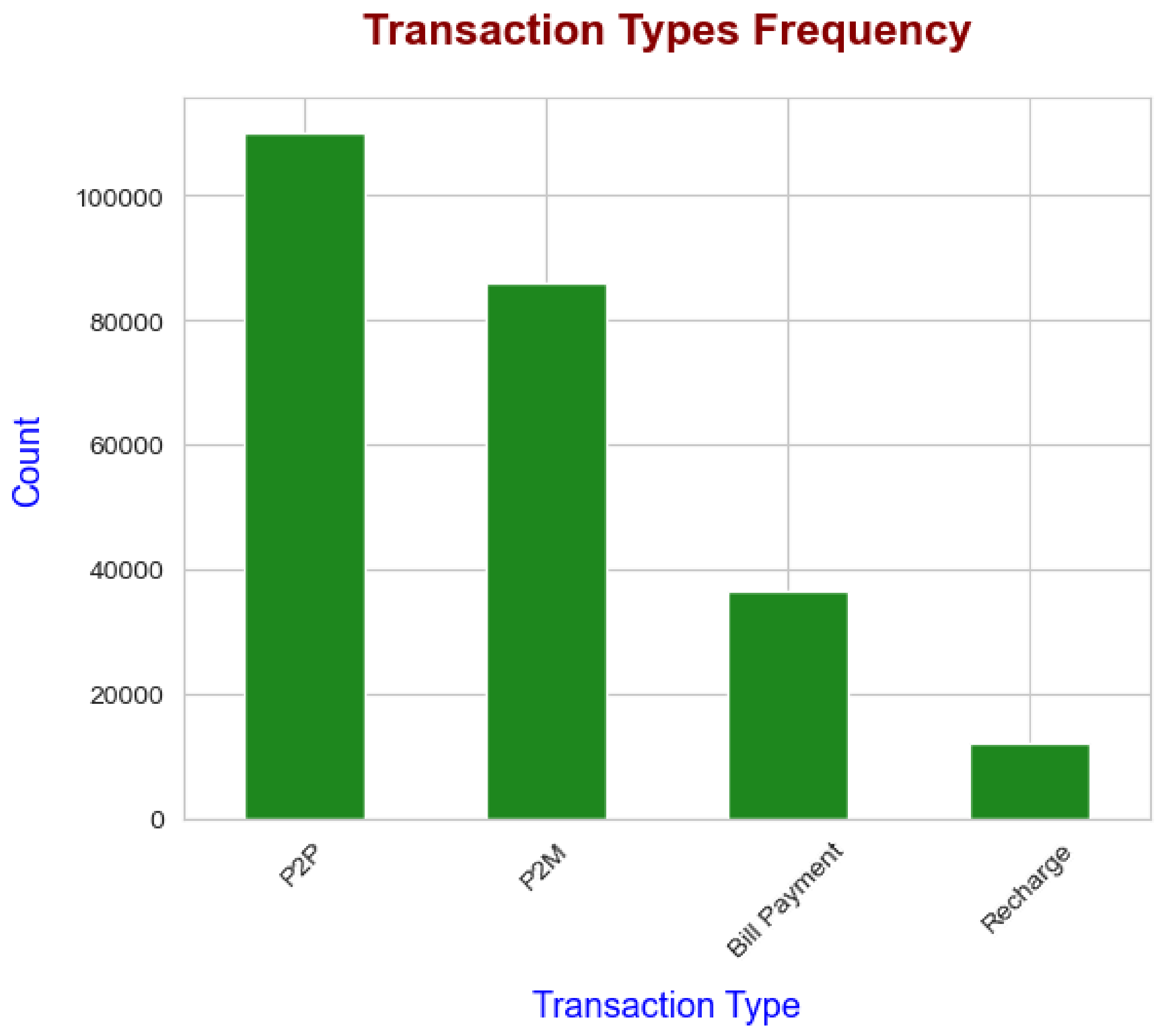


Interpretation – Transaction Status Distribution

- **Most UPI transactions are successful**, showing that the system is highly reliable.
- **Only a small portion of transactions fail**, usually due to network issues, bank downtime, or user mistakes.
- **The high success rate builds user confidence and trust** in using UPI for daily payments.

▲ Transaction Type Count

In [32]:
```
df['transaction type'].value_counts().plot(kind="bar", color="#228B22")
plt.title("Transaction Types Frequency",color='darkred',  fontweight='bold',font
plt.xlabel("Transaction Type", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Count", color='blue', fontsize=13, labelpad=10)
plt.xticks(rotation=45)
plt.show()
```

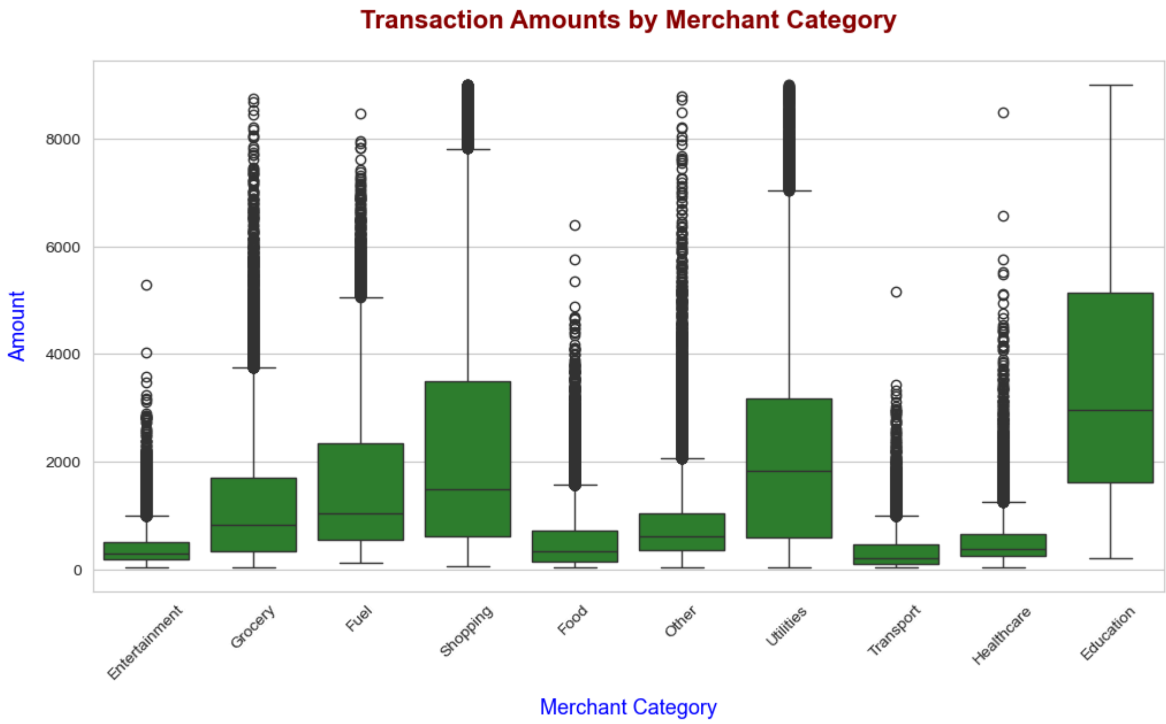


Interpretation – Transaction Types Frequency

- **P2P (Person-to-Person) transfers are the most common**, showing UPI's primary role as a money transfer system between individuals.
- **P2M (Person-to-Merchant) transactions are also high**, reflecting UPI's growing adoption for retail, shopping, and service payments.
- **Bill payments are moderately frequent**, showing users rely on UPI for essential recurring expenses.
- **Recharges form the smallest share**, as many users prefer direct telecom apps or wallets for quick top-ups.

Amount by Merchant Category

In [33]:
```
plt.figure(figsize=(12,6))
sns.boxplot(x="merchant_category", y="amount (INR)", data=df,color="#228B22")
plt.xticks(rotation=45)
plt.title("Transaction Amounts by Merchant Category",color='darkred', fontweight
plt.xlabel("Merchant Category", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Amount", color='blue', fontsize=13, labelpad=10)

plt.show()
```

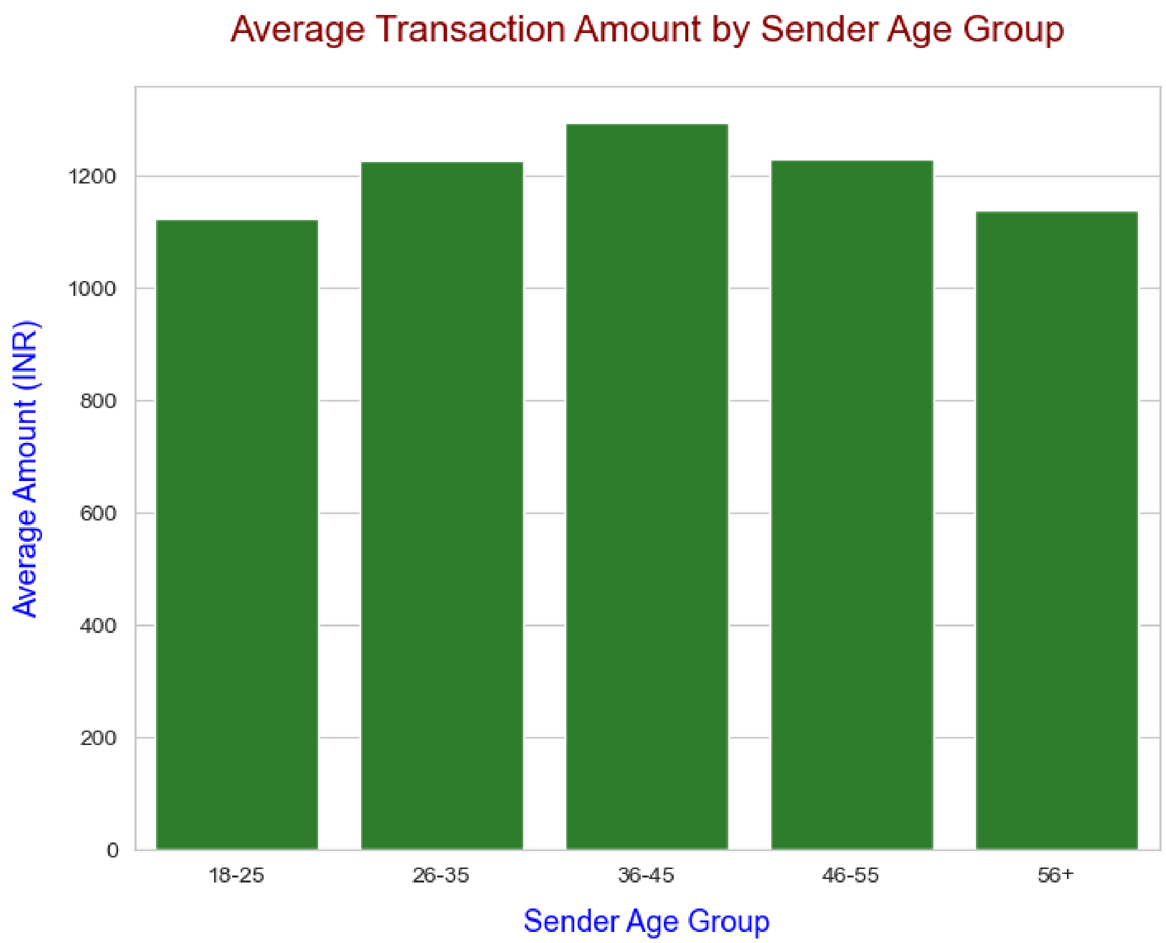


Interpretation – Transaction Amounts by Merchant Category

- **Shopping and Education have the highest transaction amounts**, indicating that users spend more on big purchases and educational fees.
- **Grocery, Fuel, and Utilities show moderate transaction amounts**, reflecting routine daily or monthly expenses.
- **Entertainment, Transport, Food, and Healthcare have lower amounts**, meaning these payments are usually smaller and more frequent.
- **Most categories have many outliers**, especially Shopping, Utilities, and Other, suggesting occasional high-value transactions.
- **Overall, UPI is used for both small daily payments and occasional large payments**, showing its versatility.

🔺 Sender Age Group vs Average Amount

In [34]:
```
plt.figure(figsize=(8,6))
sns.barplot(
    x=df.groupby("sender_age_group")["amount (INR)"].mean().index,
    y=df.groupby("sender_age_group")["amount (INR)"].mean().values,
    color="#228B22"
)

plt.title("Average Transaction Amount by Sender Age Group", color='darkred', fon
plt.xlabel("Sender Age Group", color="blue", fontsize=13, labelpad=10)
plt.ylabel("Average Amount (INR)", color="blue", fontsize=13, labelpad=10)
plt.show()
```



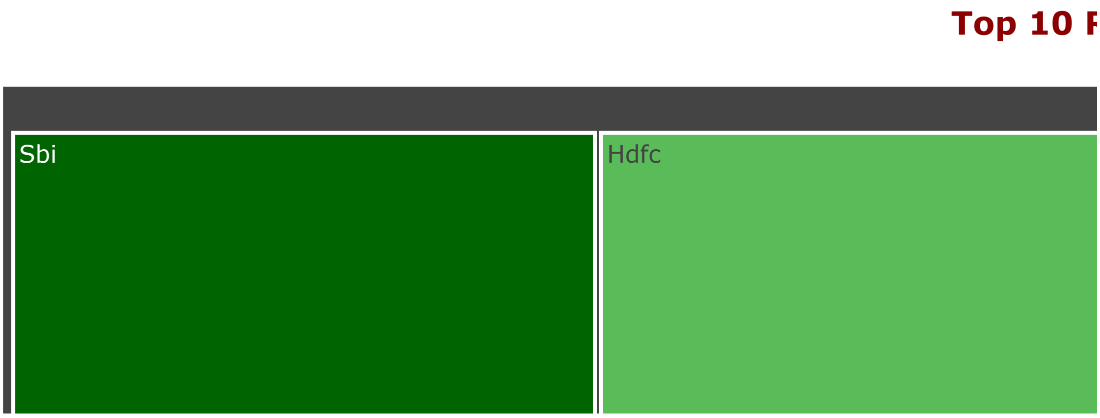
Interpretation – Average Transaction Amount by Sender Age Group

- **Senders aged 36-45 have the highest average transaction amount**, indicating that middle-aged users tend to make larger payments compared to other age groups.
- **Senders aged 18-25 and 56+** have the lowest average amounts, suggesting younger and older users make smaller payments.
- **Overall, transaction amounts rise from 18-25 to 36-45**, then slightly decrease for 46-55 and 56+, showing a peak in the middle age range.
- UPI usage for larger payments is more common among middle-aged adults, possibly due to higher income and financial responsibilities.

🔺 Receiver Bank vs Number of Transactions

In [35]:
```
bank_counts = df['receiver_bank'].value_counts().head(10).reset_index()
bank_counts.columns = ['Receiver Bank', 'Transaction Count']

fig = px.treemap(
    bank_counts,
    path=['Receiver Bank'],
    values='Transaction Count',
    color='Transaction Count',
    color_continuous_scale=['#98FB98', '#006400']
```

```
)

fig.update_layout(
    title=dict(
        text="<b>Top 10 Receiver Banks by Transaction Count</b>",
        font=dict(size=16, color="darkred"),
        x=0.5,
        xanchor="center",
        pad=dict(t=20)
    ),
    margin=dict(t=50, l=20, r=20, b=20)
)

fig.update_traces(marker_line_color='white', marker_line_width=2)

fig.show()
```



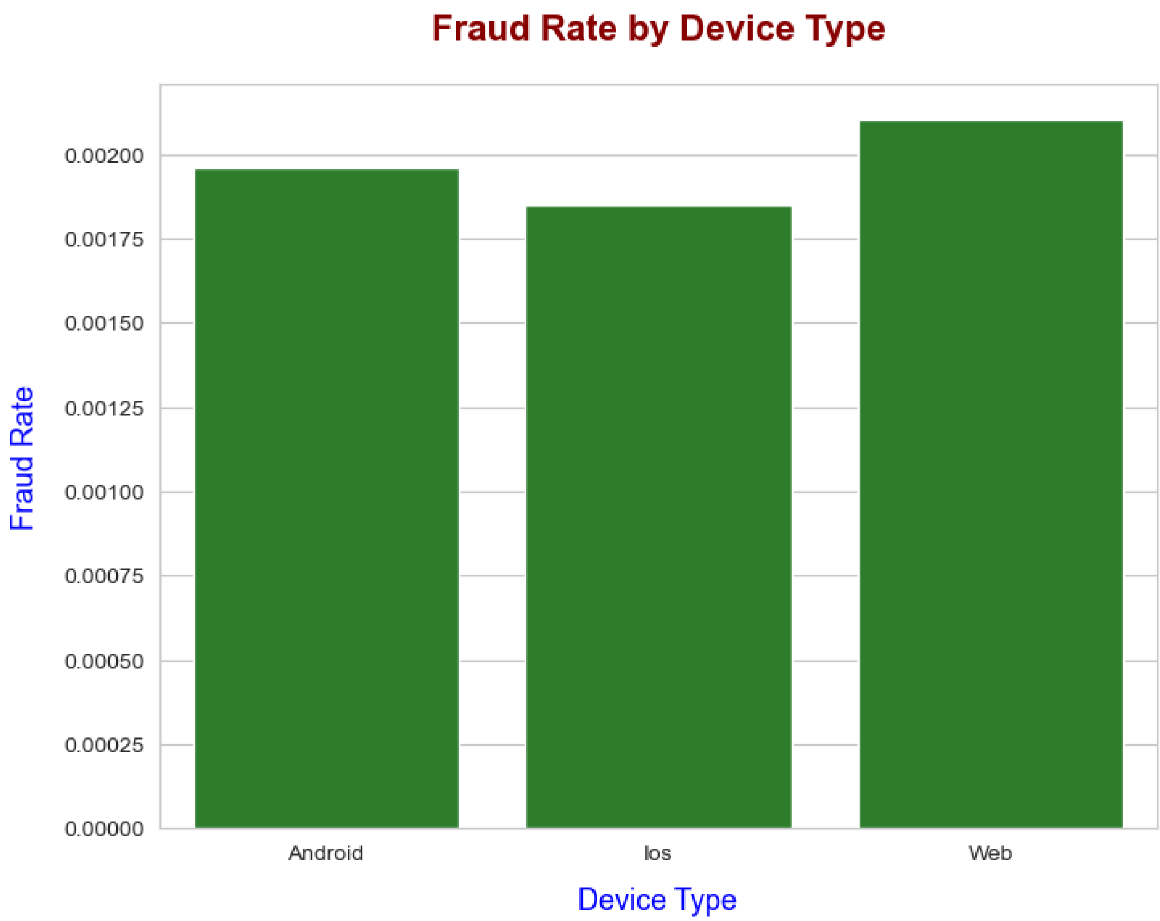
Interpretation – Top 10 Receiver Banks by Transaction Count

- **SBI receives the highest number of UPI transactions**, making it the most preferred bank for money transfers.
- **HDFC and ICICI** also record a large number of transactions, but their counts are much lower compared to SBI.
- **Kotak, PNB, and Yes Bank** are also in the top 10 but handle fewer transactions than the leading banks.

- **Overall, SBI stands out with a clear lead**, while the other banks share smaller portions of UPI inflow.
- UPI users trust and rely more on SBI for receiving payments, while private banks play a secondary role

Device Type vs Fraud Rate

In [36]:
```
fraud_rate = df.groupby("device_type")["fraud_flag"].mean()

plt.figure(figsize=(8,6))
sns.barplot(
    x=fraud_rate.index,
    y=fraud_rate.values,
    color="#228B22"
)

plt.title("Fraud Rate by Device Type", color='darkred', fontweight='bold', fonts
plt.xlabel("Device Type", color="blue", fontsize=13, labelpad=10)
plt.ylabel("Fraud Rate", color="blue", fontsize=13, labelpad=10)
plt.show()
```



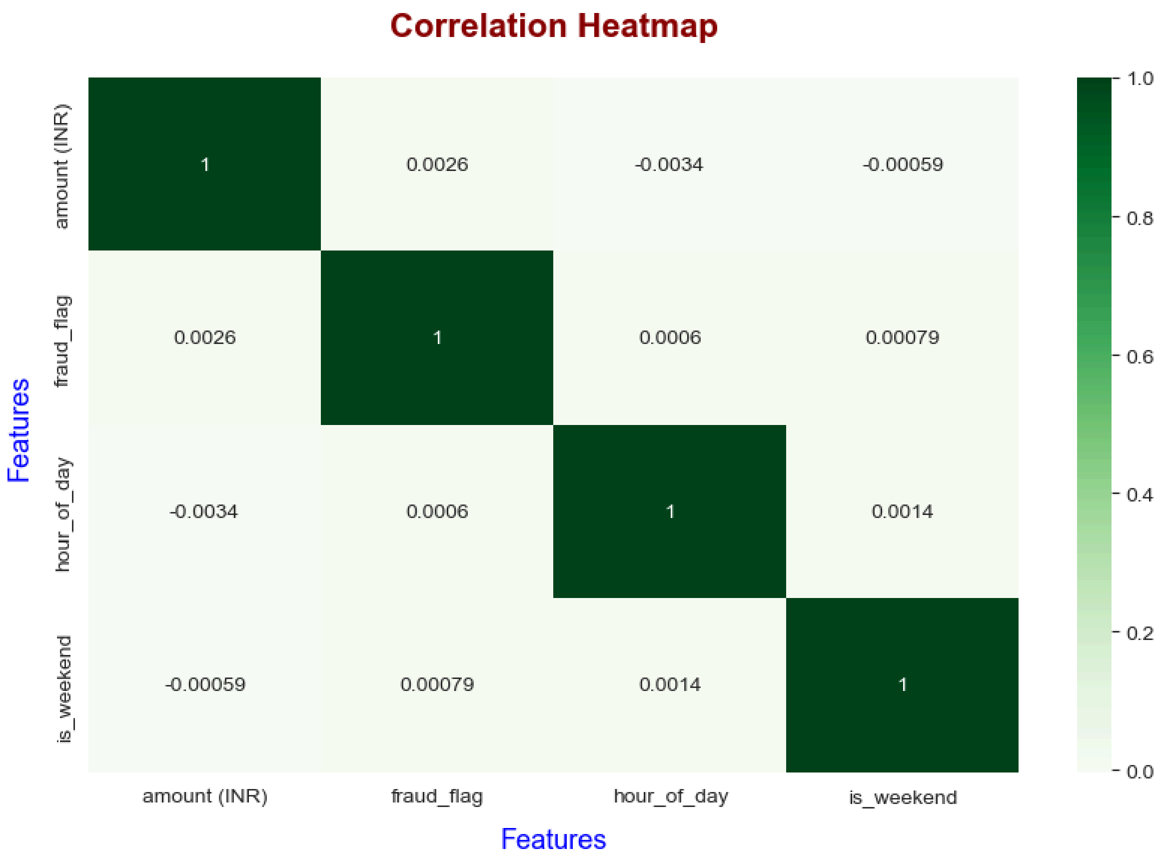
Interpretation – Fraud Rate by Device Type

- **Web transactions show the highest fraud rate**, indicating they are more vulnerable to fraudulent transactions compared to other device types.

- **Android devicesalso have a considerable fraud rate**, showing that UPI usage carries some risk.
- **IOS devices record the lowest fraud rate**, suggesting better security or lower risk among Apple users.
- **Overall, fraud rates are low across all devices**, but Web stands out with slightly higher risks than iOS and Android.
- Extra fraud monitoring and security measures may be required for Android and Web users to ensure safer UPI transactions.

Heatmap of Correlation

In [37]:
```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="Greens")

plt.title("Correlation Heatmap", color='darkred', fontweight='bold', fontsize=16
plt.xlabel("Features", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Features", color='blue', fontsize=13, labelpad=10)

plt.show()
```



Interpretation – Correlation Heatmap

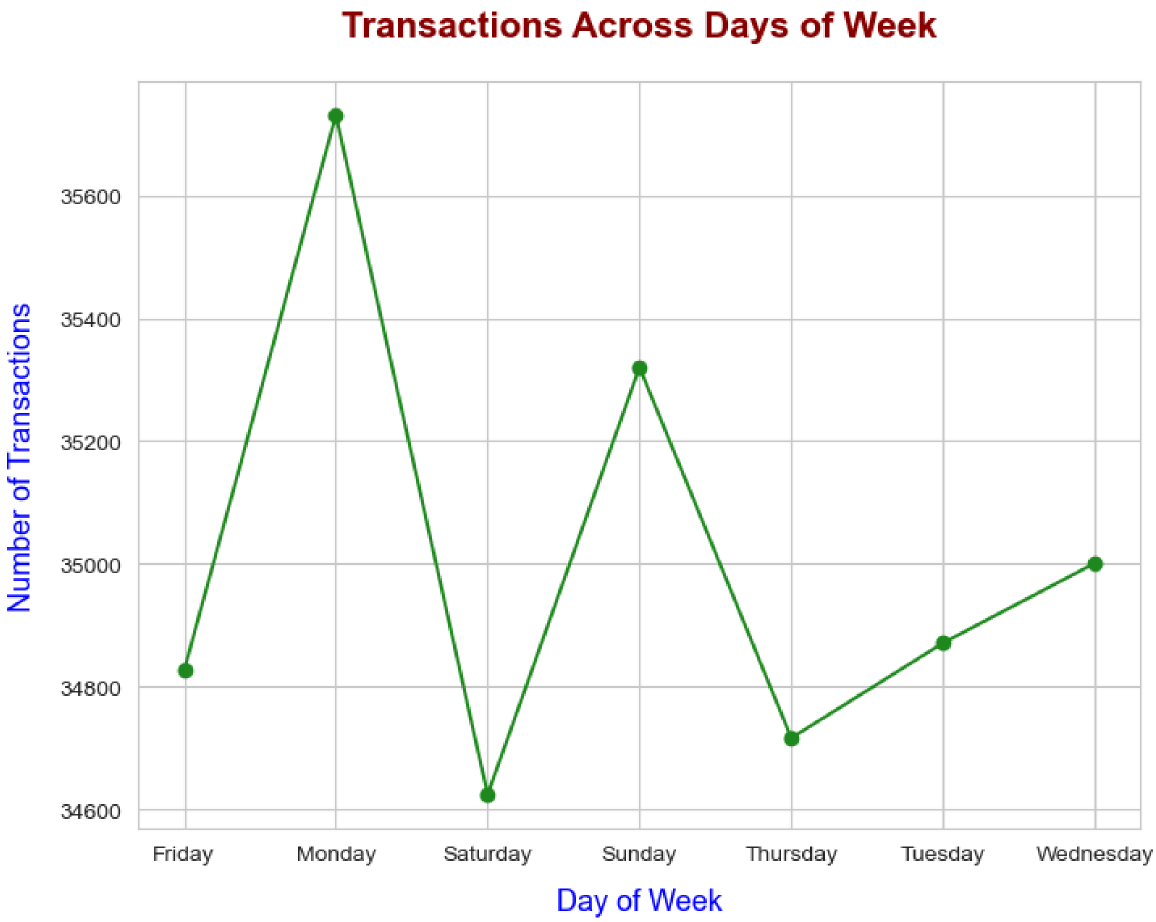
- **The correlations between features are very weak**, as most values are close to 0.
- **Amount (INR) has almost no strong link with fraud_flag, hour_of_day, or is_weekend**, meaning fraud is not directly related to

transaction size or time.

- Since no strong correlations exist, fraud detection requires more advanced methods beyond simple correlation (e.g., machine learning models).

Transactions per Day of Week

In [38]:
```
day_counts = df.groupby("day_of_week")["transaction id"].count()

plt.figure(figsize=(8,6))
day_counts.plot(kind="line", marker="o", color="#228B22")

plt.title("Transactions Across Days of Week", color='darkred', fontweight='bold'
          fontsize=16, pad=20)
plt.xlabel("Day of Week", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Number of Transactions", color='blue', fontsize=13, labelpad=10)

plt.show()
```



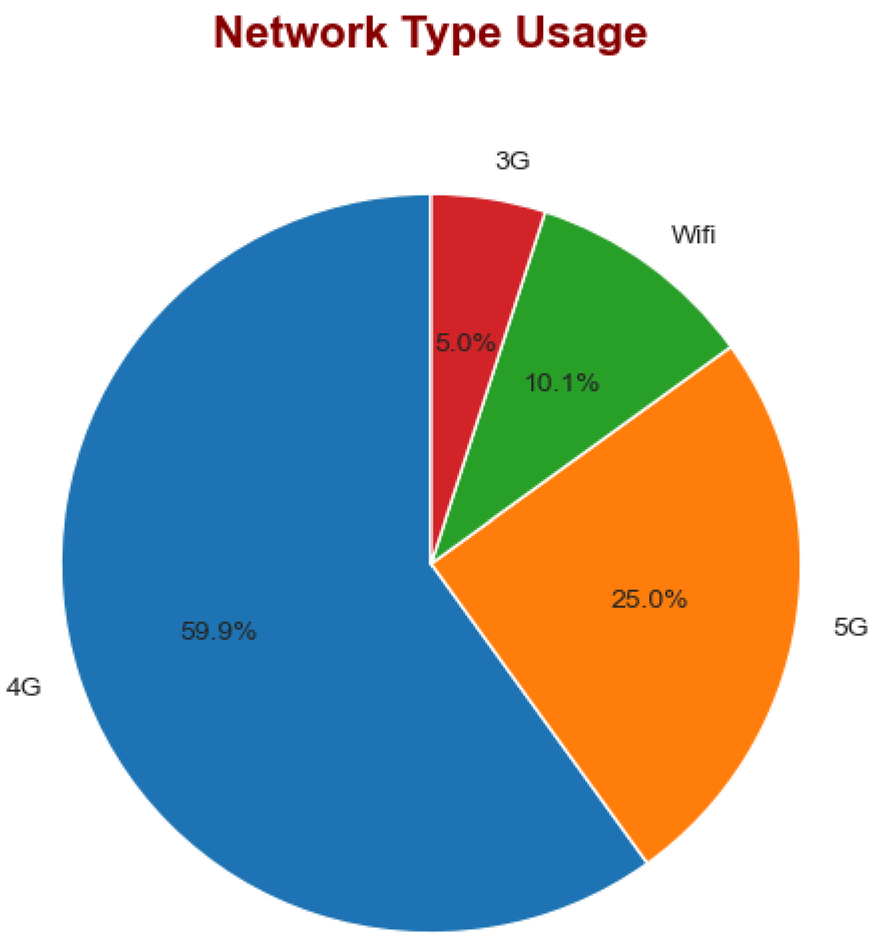
Interpretation – Transactions Across Days of Week

- **Monday records the highest number of transactions**, showing peak activity at the start of the week.
- **Saturday has the lowest number of transactions**, reflecting reduced usage on weekends.
- Transactions slightly dip mid-week but rise again towards Friday.

- Overall, **weekday transactions are higher than weekend transactions**.

▲ Network Type Distribution

In [39]:
```
plt.figure(figsize=(8,6))
df['network_type'].value_counts().plot(
    kind="pie",
    autopct='%1.1f%%',
    startangle=90
)

plt.title("Network Type Usage", color='darkred', fontweight='bold',
          fontsize=16, pad=20)
plt.ylabel("")

plt.show()
```



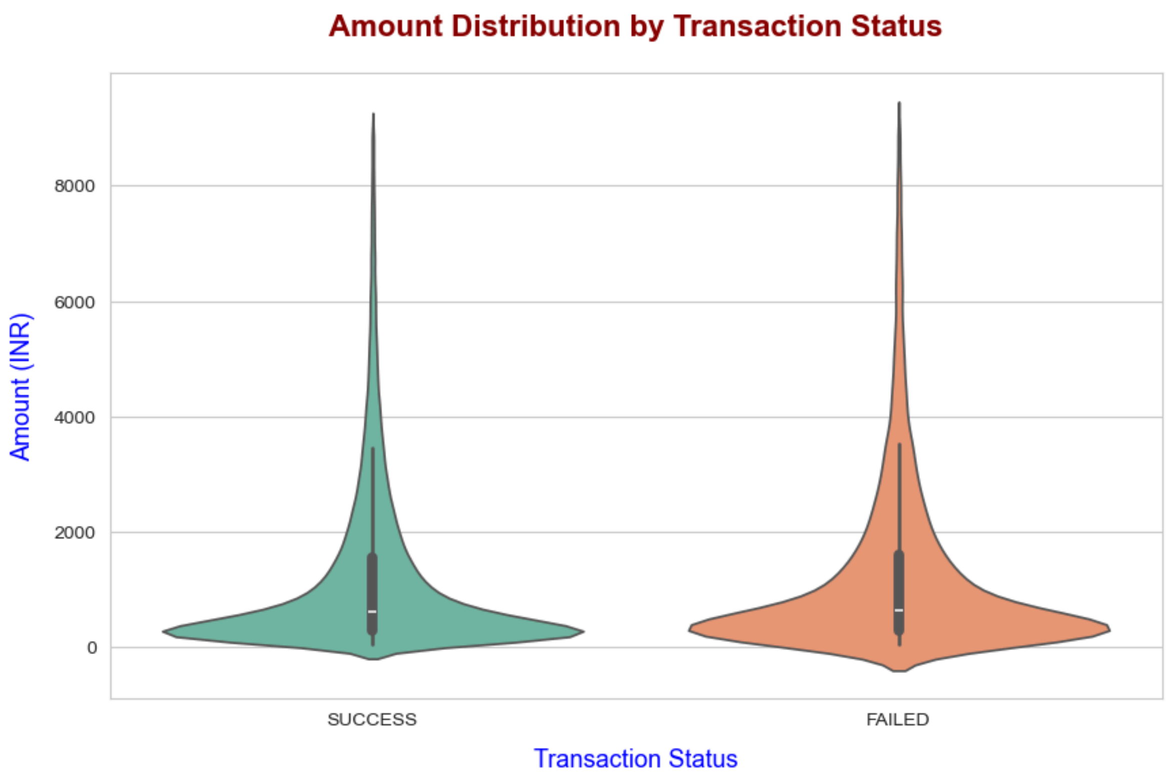
Interpretation – Network Type Usage

- **4G has the highest share of transactions (59.9%)**, indicating that nearly half of all UPI transactions are carried out using 4G networks.
- **5G accounts for 25% of transactions**, showing that adoption of 5G is growing but still trails behind 4G usage.

- **WiFi contributes 10.1% of transactions**, suggesting a significant portion of users prefer stable broadband connections for UPI payments.
- **3G has the lowest share (5%)**, reflecting a decline in its usage as users migrate to faster networks.
- UPI transactions are dominated by mobile networks, with 4G leading, while WiFi and 5G provide alternative channels. This trend highlights the reliance on mobile internet for digital payments.

🔺 Amount vs Transaction Status

In [40]:
```
plt.figure(figsize=(10,6))
sns.violinplot(
    x="transaction_status",
    y="amount (INR)",
    hue="transaction_status",
    data=df,
    palette="Set2",
    legend=False
)

plt.title("Amount Distribution by Transaction Status",
          color='darkred', fontweight='bold', fontsize=16, pad=20)
plt.xlabel("Transaction Status", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Amount (INR)", color='blue', fontsize=13, labelpad=10)

plt.show()
```

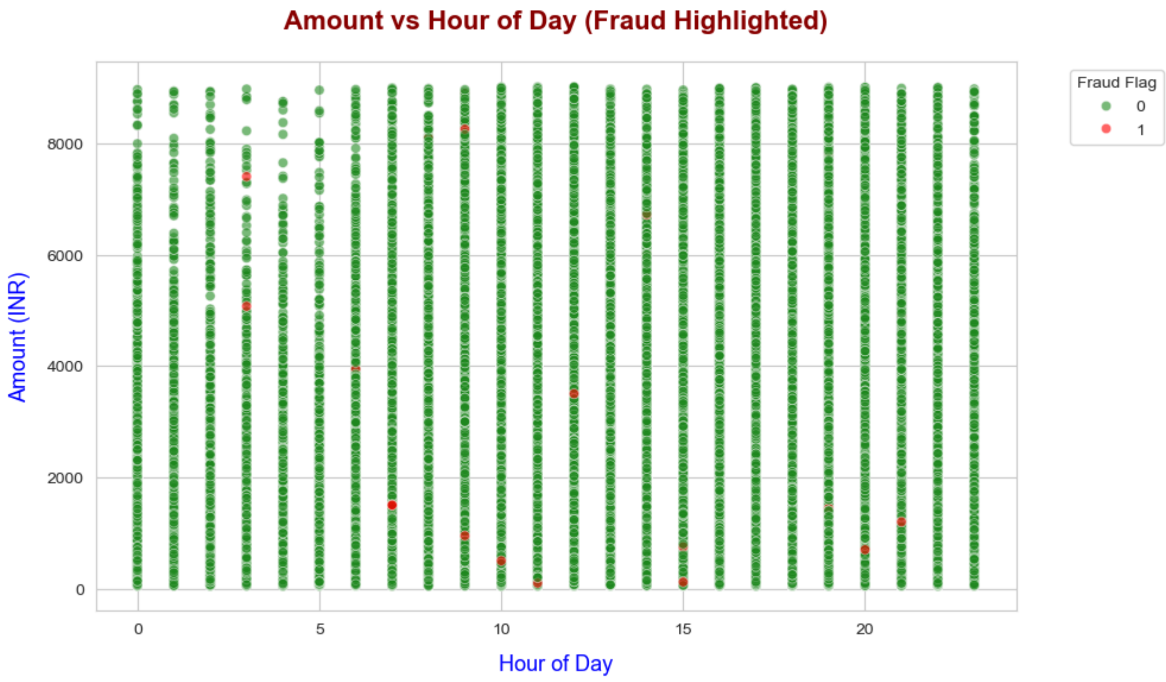


Interpretation – Amount Distribution by Transaction Status

- **Successful transactions show a wider spread of amounts**, indicating that they occur across both lower and higher ranges of INR values.
- **Failed transactions are more concentrated around lower amounts**, suggesting that failures are more frequent in smaller-value payments.
- **Both success and failure distributions extend to higher values**, but the density is much lower at the extreme ends, showing fewer high-value transactions overall.
- **The median amount is slightly higher for successful transactions** compared to failed ones, reflecting greater stability in mid-range transaction values.
- Transaction patterns suggest that success rates are better sustained across varied amounts, while failures cluster more in lower-value ranges.

▲ Amount vs Hour of Day

In [41]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(
    x="hour_of_day",
    y="amount (INR)",
    hue="fraud_flag",
    data=df,
    palette={0: "#228B22", 1: "red"},
    alpha=0.6
)

plt.title("Amount vs Hour of Day (Fraud Highlighted)",
          color='darkred', fontweight='bold', fontsize=16, pad=20)
plt.xlabel("Hour of Day", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Amount (INR)", color='blue', fontsize=13, labelpad=10)

plt.legend(title="Fraud Flag", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

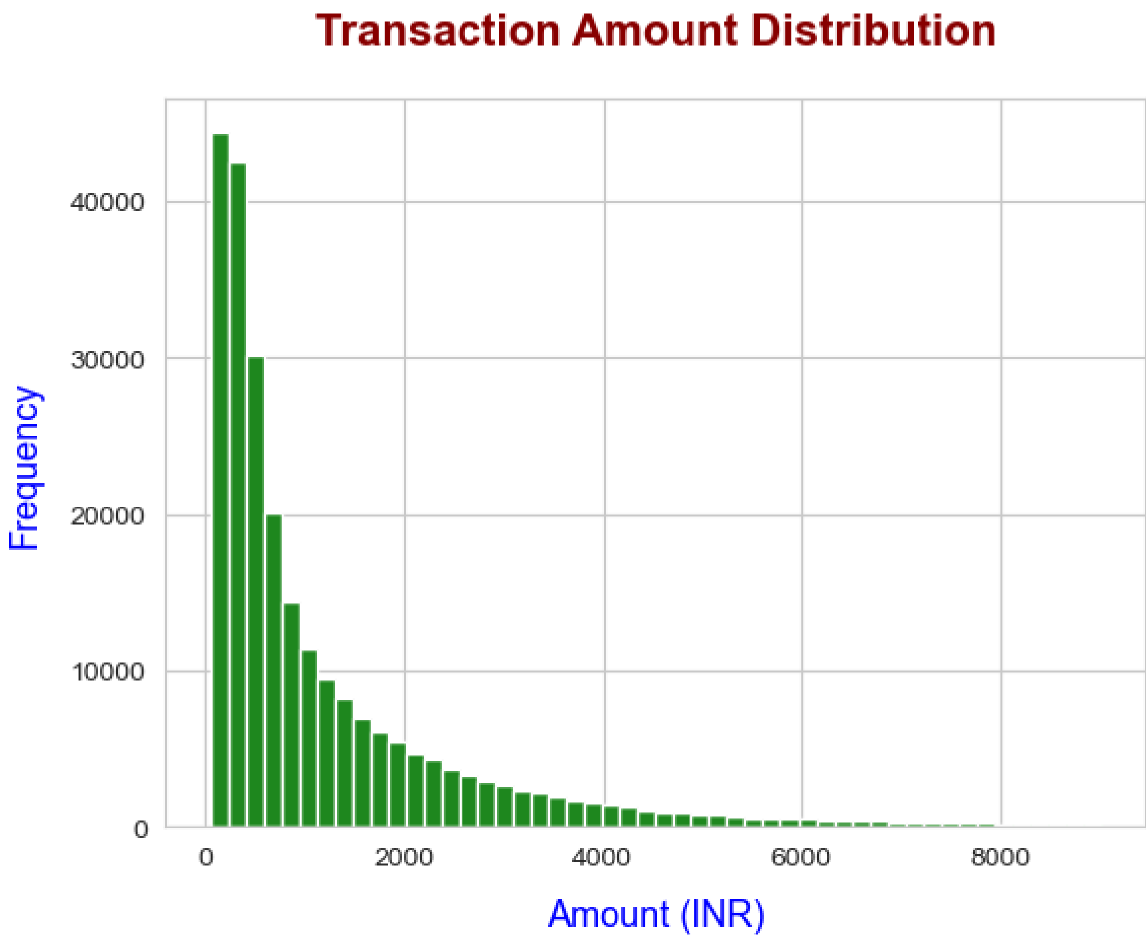


Interpretation – Amount vs Hour of Day (Fraud Highlighted)

- **Transactions are distributed across all 24 hours**, showing that UPI payments occur consistently throughout the day.
- **Most transactions cluster at lower to mid-value ranges**, with fewer high-value payments above INR 60,000.
- **Fraudulent transactions (red points) are scattered across hours**, but they are relatively rare compared to legitimate ones.
- **Fraud cases appear both in small and high-value ranges**, indicating that fraud attempts are not limited to any specific transaction size.
- The overall pattern suggests that while transaction activity is steady across the day, fraud risks exist at varying times and amounts, requiring continuous monitoring.

▲ Transaction Amount Distribution

In [42]:
```
df['amount (INR)'].plot(kind="hist", bins=50, color="#228B22")

plt.title("Transaction Amount Distribution",
          color='darkred', fontweight='bold', fontsize=16, pad=20)
plt.xlabel("Amount (INR)", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Frequency", color='blue', fontsize=13, labelpad=10)

plt.show()
```



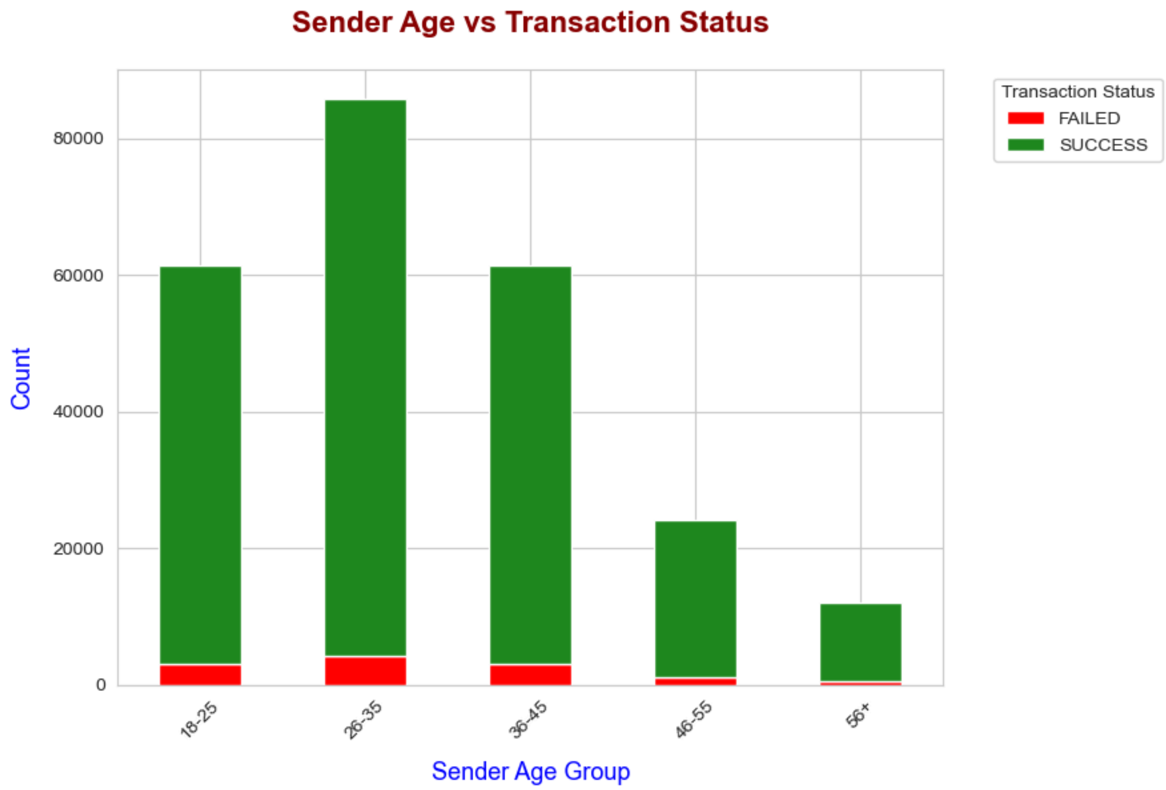
Interpretation – Transaction Amount Distribution

- **The majority of transactions are concentrated in the lower amount range**, mostly below INR 2,000.
- **As the transaction amount increases, the frequency decreases sharply**, indicating fewer high-value transactions.
- **Very high-value transactions (above INR 5,000)** are extremely rare compared to smaller payments.
- **The distribution is highly right-skewed**, showing that most UPI payments are for low amounts, while only a small portion involves large sums.
- The pattern reflects the everyday use of UPI for small, frequent payments rather than high-value transfers.

▲ Sender Age vs Transaction Status

In [43]:
```
cross_tab = pd.crosstab(df['sender_age_group'], df['transaction_status'])
cross_tab.plot(
    kind="bar",
    stacked=True,
    color=["red", "#228B22"],
    figsize=(8,6)
)
```

```
plt.title("Sender Age vs Transaction Status",
          color='darkred', fontweight='bold', fontsize=16, pad=20)
plt.xlabel("Sender Age Group", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Count", color='blue', fontsize=13, labelpad=10)
plt.xticks(rotation=45)
plt.legend(title="Transaction Status", bbox_to_anchor=(1.05, 1), loc='upper left
plt.show()
```



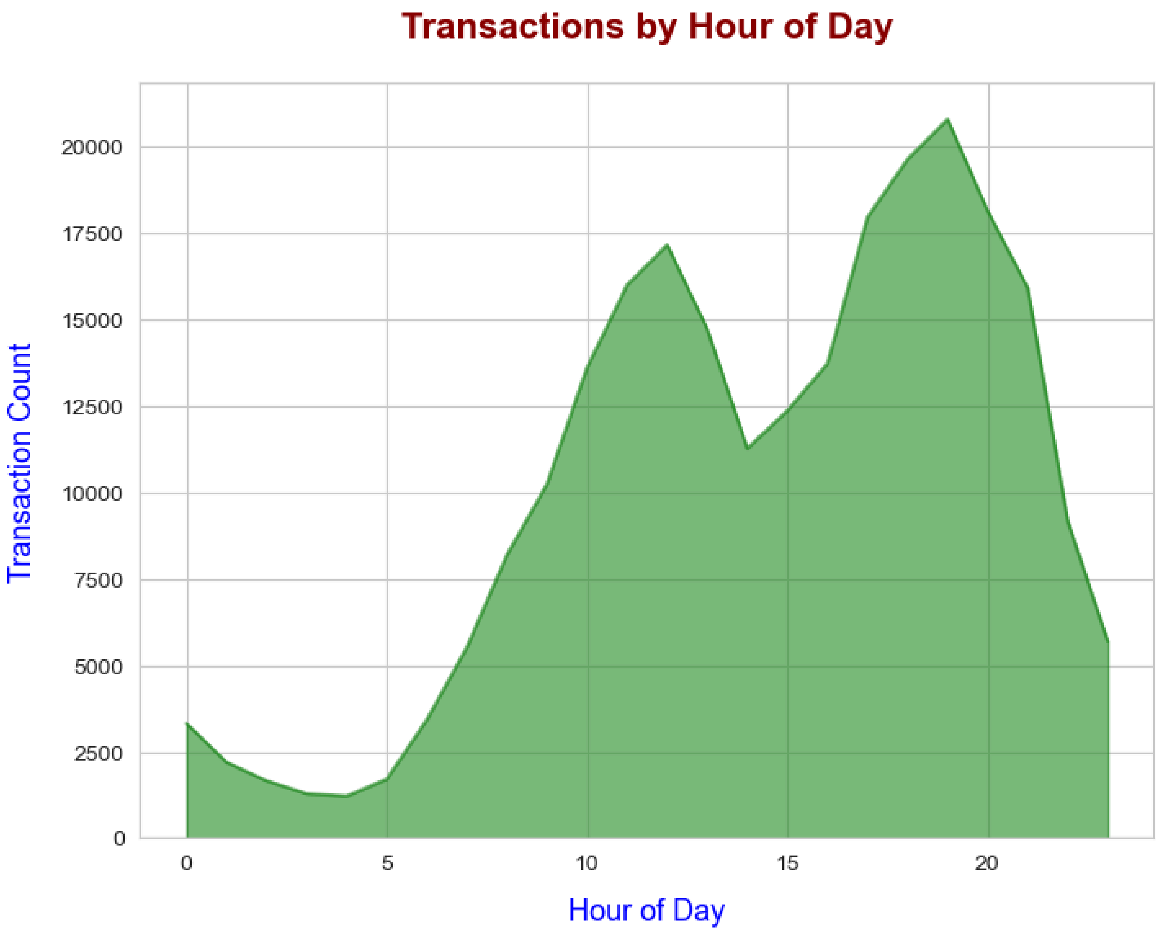
Interpretation – Sender Age vs Transaction Status

- **The 26–35 age group has the highest number of transactions**, showing that young working professionals are the most active UPI users.
- **The 18–25 and 36–45 groups also show significant activity**, together contributing a large share of transactions.
- **Transaction volume declines steadily for age groups above 45**, with the 56+ category recording the lowest participation.
- **Across all age groups, successful transactions (green) dominate**, while failures (red) form a very small fraction.
- The trend highlights that UPI adoption is strongest among younger and middle-aged users, while older groups show relatively lower engagement.

▲ Transactions per Hour

In [44]:
```
hourly = df.groupby("hour_of_day")["transaction id"].count()
hourly.plot(kind="area", alpha=0.6, color="#228B22", figsize=(8,6))

plt.title("Transactions by Hour of Day",
          color='darkred', fontweight='bold', fontsize=16, pad=20)
```

```
plt.xlabel("Hour of Day", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Transaction Count", color='blue', fontsize=13, labelpad=10)

plt.show()
```



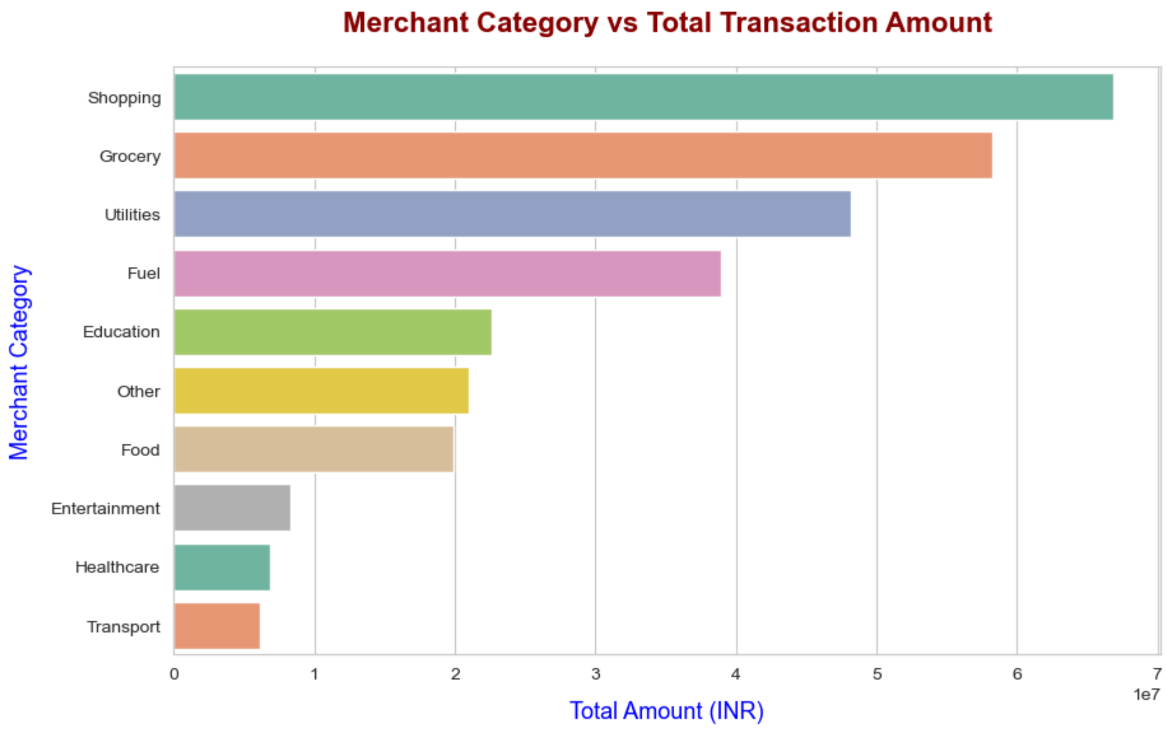
Interpretation – Transactions by Hour of Day

- **Transaction volume starts increasing from early morning (around 6–7 AM)**, showing the beginning of daily UPI activity.
- **A steady rise continues through the morning and midday**, indicating high activity during working hours.
- **There are two major peaks in the evening hours**, reflecting higher transaction activity after work and during shopping/dining times.
- **Late-night transactions drop sharply**, with very low activity observed after midnight.
- The pattern reflects typical daily behavior, where UPI usage is aligned with work schedules, shopping times, and personal expenses across the day.

▲ Merchant Category vs Total Amount

In [45]:
```
merchant_amount = df.groupby("merchant_category", as_index=False)["amount (INR)"
merchant_amount = merchant_amount.sort_values(by="amount (INR)", ascending=False
```

```
plt.figure(figsize=(10,6))
sns.barplot(
    x='amount (INR)',
    y='merchant_category',
    data=merchant_amount,
    hue="merchant_category",
    legend=False,
    palette="Set2"
)

plt.title("Merchant Category vs Total Transaction Amount",
          color='darkred', fontweight='bold', fontsize=16, pad=20)
plt.xlabel("Total Amount (INR)", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Merchant Category", color='blue', fontsize=13, labelpad=10)
plt.show()
```



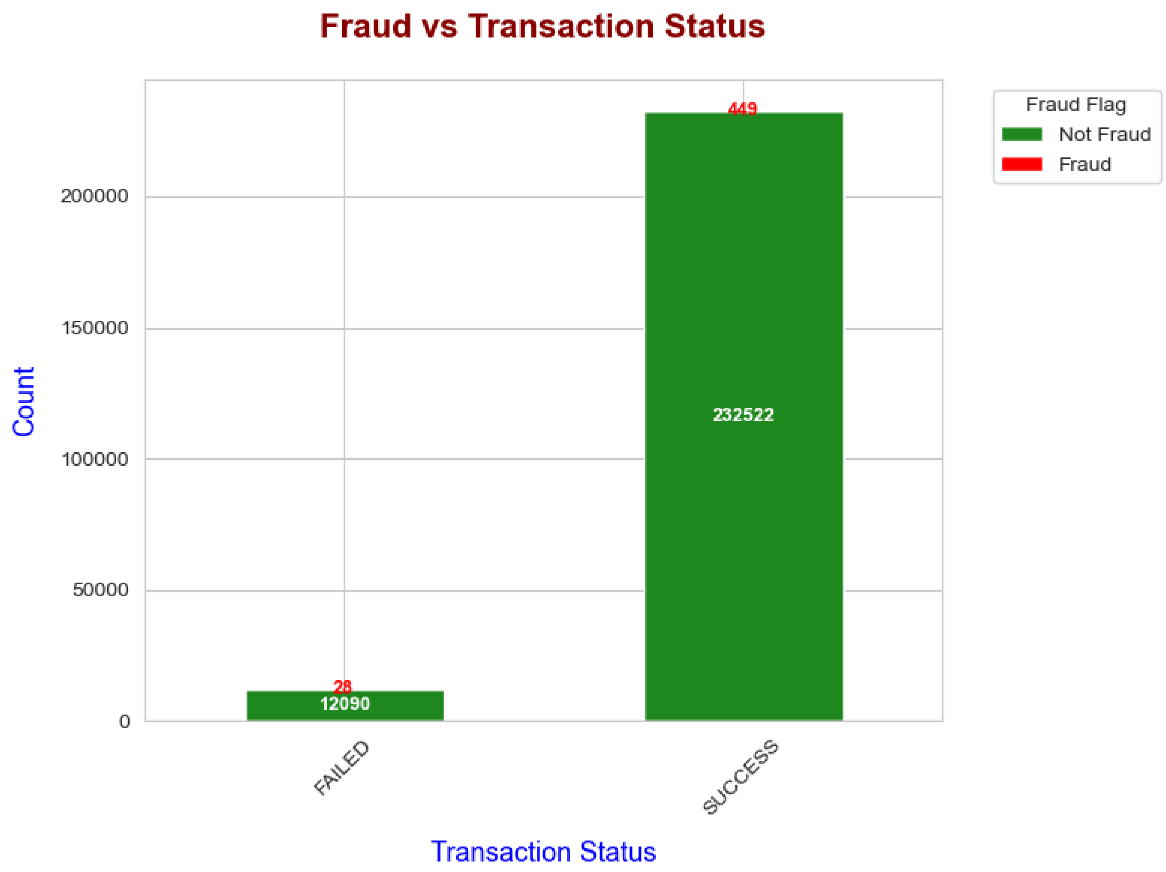
Interpretation – Merchant Category vs Total Transaction Amount

- **Shopping records the highest total transaction amount**, highlighting consumer preference for retail and e-commerce payments.
- **Grocery is the second-largest category**, showing frequent spending on daily essentials through UPI.
- **Utilities and Fuel contribute significantly**, reflecting the growing trend of paying bills and fuel expenses digitally.
- **Categories like Education and Food show moderate levels**, indicating regular but smaller-scale payments.
- **Healthcare, Entertainment, and Transport are among the lowest**, suggesting limited but niche usage in these sectors.

- Overall, UPI usage is dominated by lifestyle and essential needs, with shopping and groceries leading digital spending habits.

Fraud vs Transaction Status

In [46]:
```
fraud_status = pd.crosstab(df['transaction_status'], df['fraud_flag'])

ax = fraud_status.plot(
    kind="bar",
    stacked=True,
    color=["#228B22", "red"],  # green = not fraud, red = fraud
    figsize=(8,6)
)

# Add labels with different text colors
for container in ax.containers:
    for bar in container:
        height = bar.get_height()
        if height > 0:  # only label non-zero bars
            # Check if this bar is red (fraud) or green (non-fraud)
            if bar.get_facecolor() == (1.0, 0.0, 0.0, 1.0):  # red RGBA
                color = "red"   # fraud count label in red
            else:
                color = "white" # non-fraud label in white

            ax.text(
                bar.get_x() + bar.get_width()/2,
                bar.get_y() + height/2,
                f"{int(height)}",
                ha="center", va="center",
                color=color, fontsize=9, fontweight="bold"
            )

plt.title("Fraud vs Transaction Status", color='darkred', fontsize=16, pad=20, f
plt.xlabel("Transaction Status", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Count", color='blue', fontsize=13, labelpad=10)
plt.xticks(rotation=45)
plt.legend(["Not Fraud", "Fraud"], title="Fraud Flag", bbox_to_anchor=(1.05, 1),
plt.tight_layout()
plt.show()
```



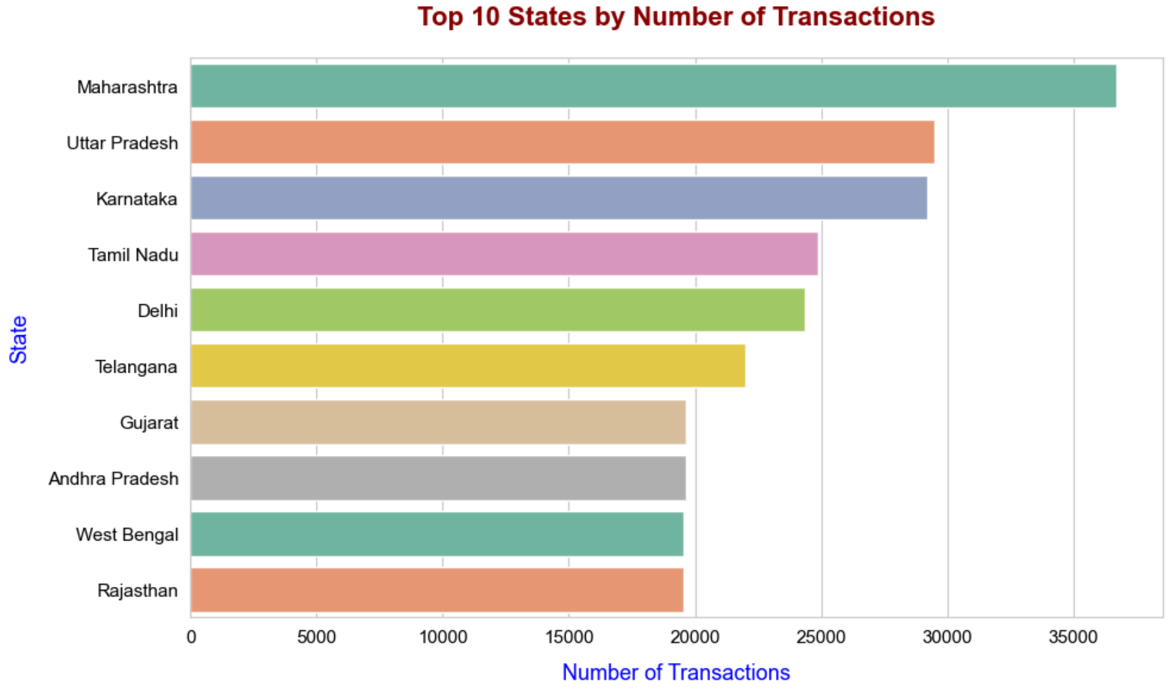
Interpretation – Fraud vs Transaction Status

- **Successful transactions dominate the dataset**, showing the majority of UPI payments are completed without issues.
- **Failed transactions form a smaller share**, but they are still visible in the distribution.
- **Fraudulent transactions (red) are very few compared to non-fraudulent ones**, indicating that most activity is genuine.
- **Both successful and failed transactions contain some fraud cases**, meaning fraudulent attempts occur in different transaction outcomes.
- Overall, the dataset highlights that while fraud exists, it represents only a small fraction of total UPI activity.

🔺 Top 10 States by Number of Transactions

In [47]:
```
state_txn = df.groupby("sender_state", as_index=False)["transaction id"].count()

state_txn.rename(columns={"sender_state": "State", "transaction id": "Transactic

state_txn = state_txn.sort_values(by="Transactions", ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(
    data=state_txn,
    y="State",
    x="Transactions",
```

```
    hue="State",
    dodge=False,
    palette="Set2",
    legend=False
)

plt.title("Top 10 States by Number of Transactions", color='darkred', fontsize=1
plt.xlabel("Number of Transactions", color='blue', fontsize=13, labelpad=10)
plt.ylabel("State", color='blue', fontsize=13, labelpad=10)
plt.xticks(fontsize=11, color='black')
plt.yticks(fontsize=11, color='black')
plt.tight_layout()
plt.show()
```

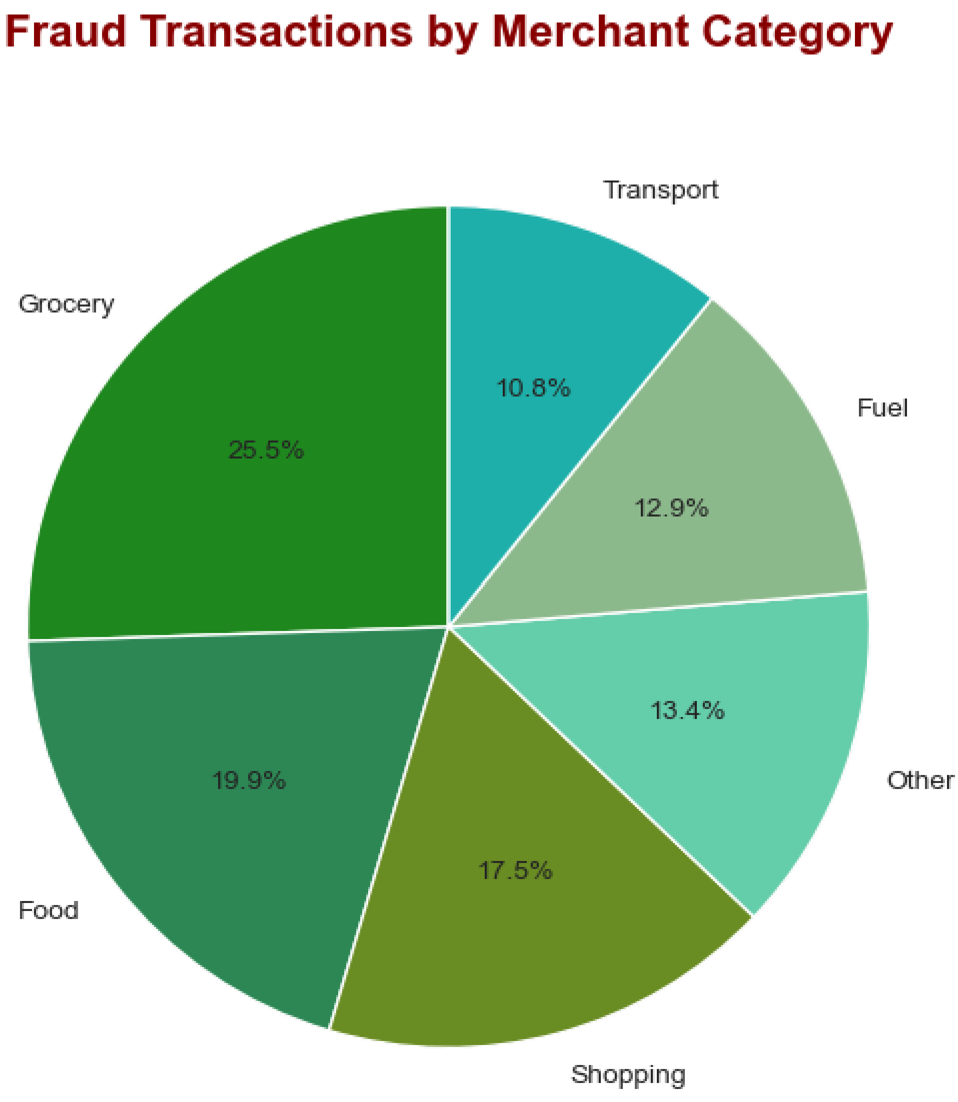


Interpretation – Top 10 States by Number of Transactions

- **Maharashtra records the highest number of transactions**, making it the leading state in UPI activity.
- **Uttar Pradesh and Karnataka follow closely behind**, showing strong adoption of digital payments in both northern and southern regions.
- **Other states like Tamil Nadu, Delhi, Telangana, Gujarat, Andhra Pradesh, West Bengal, and Rajasthan** also contribute significantly, though with comparatively lower volumes.
- **The presence of states from different regions** highlights that UPI adoption is widespread across the country, not limited to metropolitan hubs.
- The overall pattern reflects that digital financial activity is concentrated in economically strong and highly populated states, while steadily expanding across diverse regions in India.

▲ Fraud Transactions by Merchant Category

In [48]:
```
fraud_merchant = df[df["fraud_flag"] == 1]["merchant_category"].value_counts().h

fraud_merchant.plot(
    kind="pie",
    autopct="%1.1f%%",
    startangle=90,
    colors=["#228B22","#2E8B57","#6B8E23","#66CDAA","#8FBC8F","#20B2AA"],
    figsize=(8,6)
)

plt.title("Fraud Transactions by Merchant Category", color='darkred', fontsize=1
plt.ylabel("")  # remove default y-label
plt.tight_layout()
plt.show()
```



Interpretation – Fraud Transactions by Merchant Category

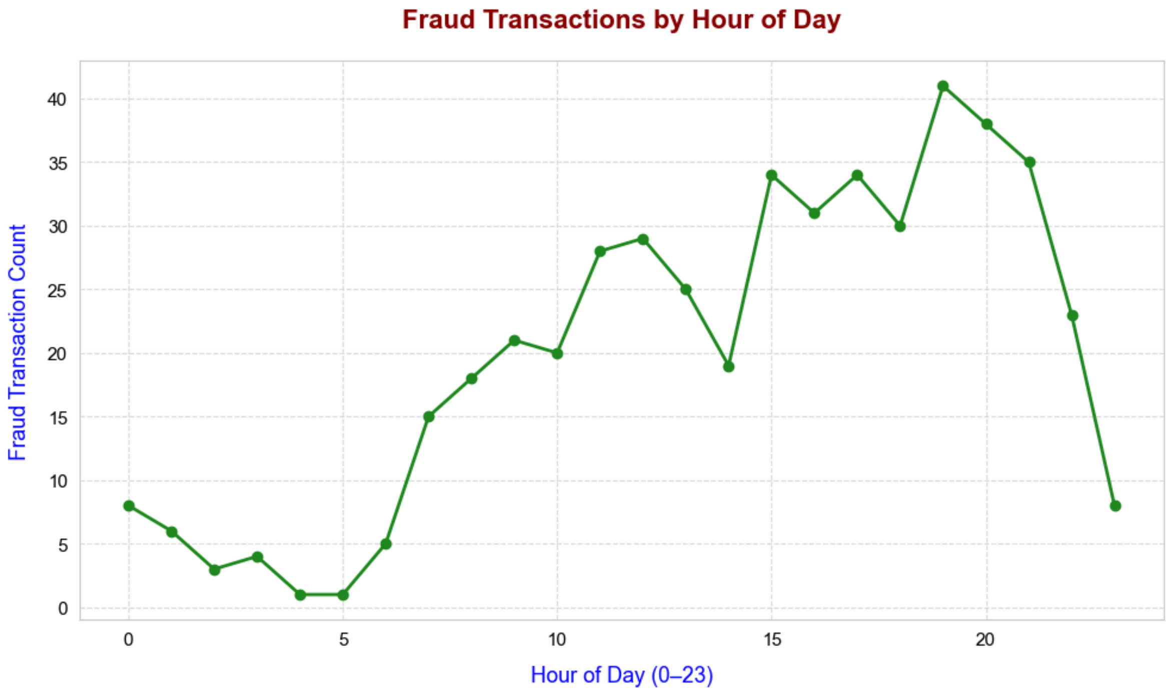
- **Grocery holds the largest share of fraud cases (25.5%)**, making it the most targeted merchant category.
- **Food-related transactions (19.3%) also face significant fraud attempts**, showing vulnerability in daily spending.

- **Shopping and Other categories each account for around 17%–13%**, highlighting fraud risks in discretionary purchases.
- **Fuel (12.9%) and Transport (10.8%) categories also experience fraud**, though at relatively lower levels compared to Grocery and Food.
- Overall, **essential and high-frequency spending categories (like Grocery and Food) attract the majority of fraud**, reflecting fraudsters' focus on common transaction types.

▲ Fraud Transactions by Hour of Day

In [49]:

```
fraud_hour = df[df["fraud_flag"] == 1]["hour_of_day"].value_counts().sort_index(

fraud_hour.plot(
    kind="line",
    color="#228B22",
    marker="o",
    linewidth=2,
    figsize=(10,6)
)

plt.title("Fraud Transactions by Hour of Day", color='darkred', fontsize=16, for
plt.xlabel("Hour of Day (0–23)", color='blue', fontsize=13, labelpad=10)
plt.ylabel("Fraud Transaction Count", color='blue', fontsize=13, labelpad=10)
plt.xticks(fontsize=11, color='black')
plt.yticks(fontsize=11, color='black')
plt.grid(True, linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```

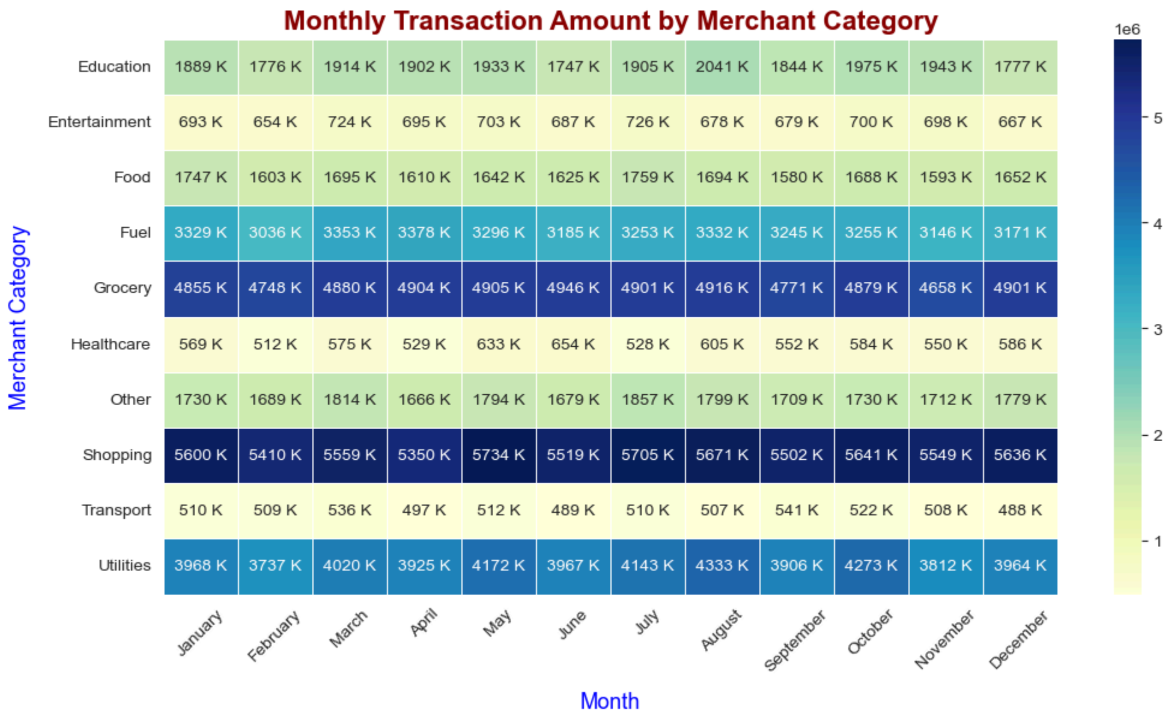


Interpretation – Fraud Transactions by Hour of Day

- **Fraud attempts occur throughout the day**, but their frequency is uneven across different hours.
- **Fraud cases are relatively low in the early morning (0–6 hours)**, suggesting fraudsters are less active at night.
- **The number of fraud transactions increases after 10 AM**, showing more activity during regular business and online shopping hours.
- **Peaks are observed in the late evening (around 20–22 hours)**, which may indicate fraudsters target times when users are more engaged in digital payments.
- Overall, the pattern highlights that fraud activity aligns with general user activity, with fraudsters taking advantage of busy transaction periods.

🔺 Monthly Transaction Amount by Merchant Category

In [50]:
```python
month_merchant = df.groupby(['month', 'merchant_category'], as_index=False)['amo

heatmap_data = month_merchant.pivot(index='merchant_category', columns='month',


month_order = ['January', 'February', 'March', 'April', 'May', 'June',
               'July', 'August', 'September', 'October', 'November', 'December']
heatmap_data = heatmap_data[month_order]

def format_k(x):
    if pd.isna(x):
        return ""
    return f"{int(x/1000)} K"

annot_data = heatmap_data.copy()
for col in annot_data.columns:
    annot_data[col] = annot_data[col].map(format_k)

plt.figure(figsize=(12,6))
sns.heatmap(
    heatmap_data,
    annot=annot_data,
    fmt="",
    cmap="YlGnBu",
    linewidths=0.5
)

plt.title("Monthly Transaction Amount by Merchant Category", fontsize=16, fontwe
plt.xlabel("Month", color="blue",fontsize=13, labelpad=10)
plt.ylabel("Merchant Category", color="blue",fontsize=13, labelpad=10)
plt.xticks(rotation=45)
plt.show()
```



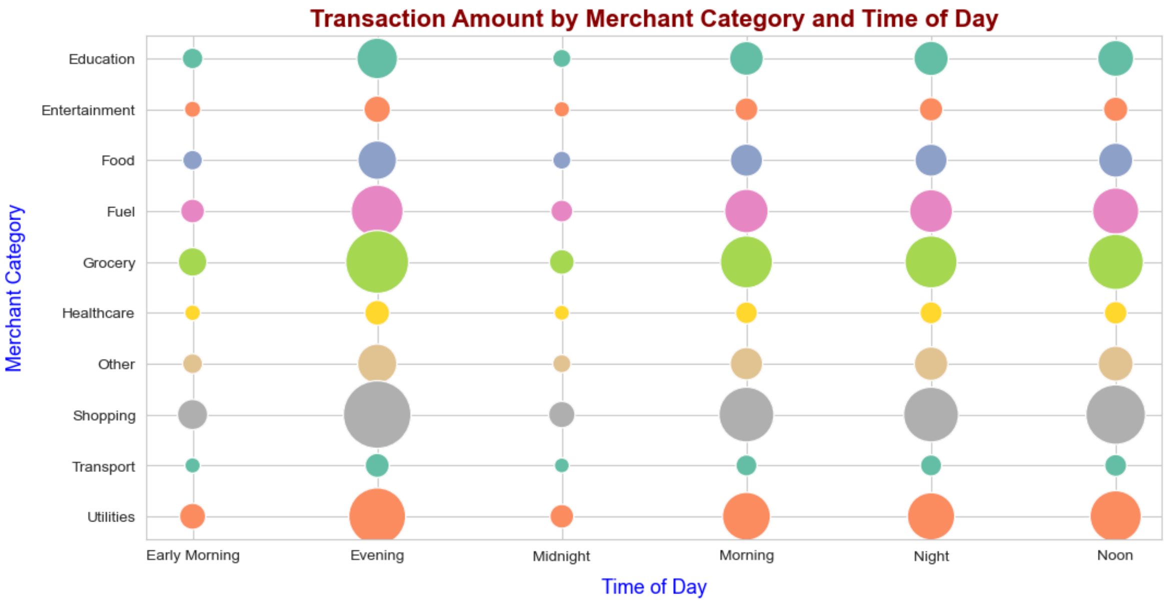
Interpretation – Monthly Transaction Amount by Merchant Category

- **Shopping consistently records the highest transaction amounts**, shown by the darkest heatmap cells and largest K annotations across most months (with noticeable mid-year and year-end peaks).
- **Grocery is the second-highest contributor**, maintaining high and steady monthly volumes just below Shopping.
- **Utilities rank third**, showing consistently strong monthly totals indicative of recurring bill payments.
- **Fuel ranks fourth**, with moderate-to-high amounts that vary seasonally (higher during travel peaks).
- **Education comes after Fuel**, recording moderate but stable totals that are lower than Utilities and Fuel.
- The remaining categories (Food, Healthcare, Transport, Entertainment) show lower totals and more month-to-month variation, while the color-intensity and annotated K-values on the heatmap confirm the overall ranking: **Shopping > Grocery > Utilities > Fuel > Education**.

▲ Transaction Amount by Merchant Category and Time of Day

In [51]:
```
plt.figure(figsize=(12,6))
sns.scatterplot(
    x='time_of_day',
    y='merchant_category',
    size='amount (INR)',
    sizes=(100, 2000),  # adjust min/max bubble size
    data=df.groupby(['time_of_day','merchant_category'])['amount (INR)'].sum().r
```

```
    hue='merchant_category',
    palette='Set2',
    legend=False
)
plt.title("Transaction Amount by Merchant Category and Time of Day", fontsize=16
plt.xlabel("Time of Day", color="blue",labelpad=10,fontsize=13)
plt.ylabel("Merchant Category", color="blue",labelpad=10,fontsize=13)
plt.show()
```



Interpretation – Transaction Amount by Merchant Category and Time of Day

- **Shopping and Grocery dominate the transaction amounts**, with the largest bubbles, showing they are the most preferred spending categories across different times of day.
- **Utilities and Fuel also hold significant shares**, especially during the evening and morning, reflecting essential daily payments.
- **Education and Food transactions are moderate**, occurring steadily throughout the day but with smaller amounts compared to shopping and grocery.
- **Entertainment and Healthcare show smaller bubbles**, suggesting they contribute less to overall transaction value across all time slots.
- **Evening is the peak time for high-value transactions**, where shopping, grocery, and utilities bubbles appear largest, followed by morning activity.
- Overall, **spending behavior is time-sensitive**: essential categories (grocery, fuel, utilities) peak in morning/evening, while discretionary categories (shopping, entertainment) peak in the evening.

Insight Generation and Report

In order to investigate **user behavior, spending trends, and fraud risks,** this project examined UPI transactions. The results provide a better understanding of digital payment usage and areas for improvement by highlighting distinct trends across **time, age groups, merchants, and transaction outcomes.**

🔺 Data Understanding

- 🔷 Key fields include **Transaction details** (ID, Date, Time, Amount, Payment Mode, Status), **User attributes** (Age Group, Device Type, Network), **Context** (Bank, Merchant Category, Location), and **Fraud flag**.
- 🔷 Covers large-scale usage patterns, reflecting real-world adoption, merchant diversity, and fraud risks.

🔺 Key Insights

- 🔷 **UPI usage patterns –** The majority of UPI payments take place in the morning and evening. Although they are less frequent, big-value transactions typically take place in the afternoon and evening.
- 🔷 **Top merchant categories –** A few categories like Groceries, Utilities, and E-commerce take the biggest share of payments.
- 🔷 **User age groups –** Young users spend more on digital and lifestyle services, while older groups spend more on bills and essentials.
- 🔷 **Fraud insights –** Although fraud is rare overall, it is more prevalent late at night and with specific networks and device types.
- 🔷 **Transaction success –** The majority of payments go through, but occasionally there are issues during busy periods or with specific banks.
- 🔷 **Weekend vs. weekday –** Weekends are spent more on entertainment, dining, and shopping, while weekdays are used for bills and transfers.

🔺 Correlation & Multivariate Patterns

- 🔷 **Transaction Amount × Fraud –** High-value late-night transactions show slightly higher fraud likelihood.
- 🔷 **Bank × Success Rate –** Some banks show more failures under peak load compared to others.
- 🔷 **Age Group × Merchant Category –** Young users are strongly correlated with **e-commerce & lifestyle**, older groups with **utilities & bills**.
- 🔷 **Time of Day × Success Rate –** Evening peak hours slightly reduce success rates due to heavy load.

🔺 Recommendations

- 🔷 **Improve peak-hour reliability –** Strengthen banking infrastructure and server capacity to minimize transaction failures during busy times.

- 🔷 **Targeted fraud prevention –** Implement stricter monitoring for late-night transactions, risky device types, and unusual usage spikes.
- 🔷 **Customer segmentation –** Design age-specific offers and merchant tie-ups, such as lifestyle deals for younger users and bill-payment benefits for older groups.
- 🔷 **Merchant diversification –** Encourage growth in underrepresented categories like travel, dining, and entertainment to balance spending distribution.
- 🔷 **Awareness campaigns –** Educate users on safe payment practices, especially during weekends and late-night high-risk periods.

🔺 Conclusion

- 🔷 **UPI adoption –** The ecosystem is deeply integrated into daily life, with peak usage during mornings, evenings, and weekends across all age groups.
- 🔷 **User and merchant dynamics –** Younger users drive lifestyle spending, while older users focus on essentials. Groceries, utilities, and e-commerce remain the top categories.
- 🔷 **System performance –** Transaction success rates are high overall, but peak-hour failures show the need for stronger infrastructure and backend support.
- 🔷 **Risk landscape –** Fraud remains uncommon but concentrated in late-night usage and specific networks, highlighting the need for targeted risk strategies.
- 🔷 **Behavioral trends –** Weekdays are dominated by bill payments and transfers, while weekends highlight discretionary spending in entertainment and shopping.

Overall, this study shows how data analysis of UPI transactions can support smarter decisions, improve user experience, and strengthen fraud detection.

In [ ]: