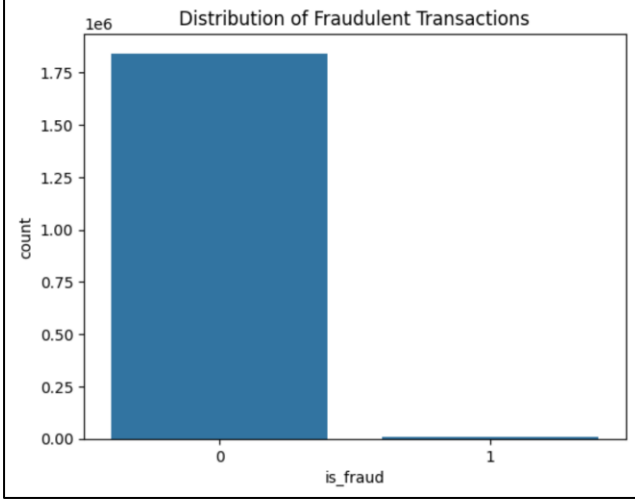
Exploratory Data Analysis

Now that we have cleaned and transformed the DataFrame ‘df’ for credit card fraud detection, we can proceed with the Exploratory Data Analysis (EDA). The goal of EDA is to thoroughly understand the dataset before applying advanced modeling techniques. This step helps identify patterns, trends, anomalies, and relationships within the data, ensuring its quality and integrity.

We begin our EDA by analyzing the target variable, as it reveals critical insights into class distribution—highlighting severe imbalance (e.g., only 0.5% fraud cases). This informs key modeling decisions, such as resampling techniques or prioritizing precision/recall over accuracy.

```
# Analysis of the target variable (is fraud)
# Distribution of the target variable
print(df['is_fraud'].value_counts(normalize=True))

# Visualization
sns.countplot(x='is_fraud', data=df)
plt.title('Distribution of Fraudulent Transactions')
plt.show()
```

```
is_fraud
0    0.99479
1    0.00521
Name: proportion, dtype: float64
```

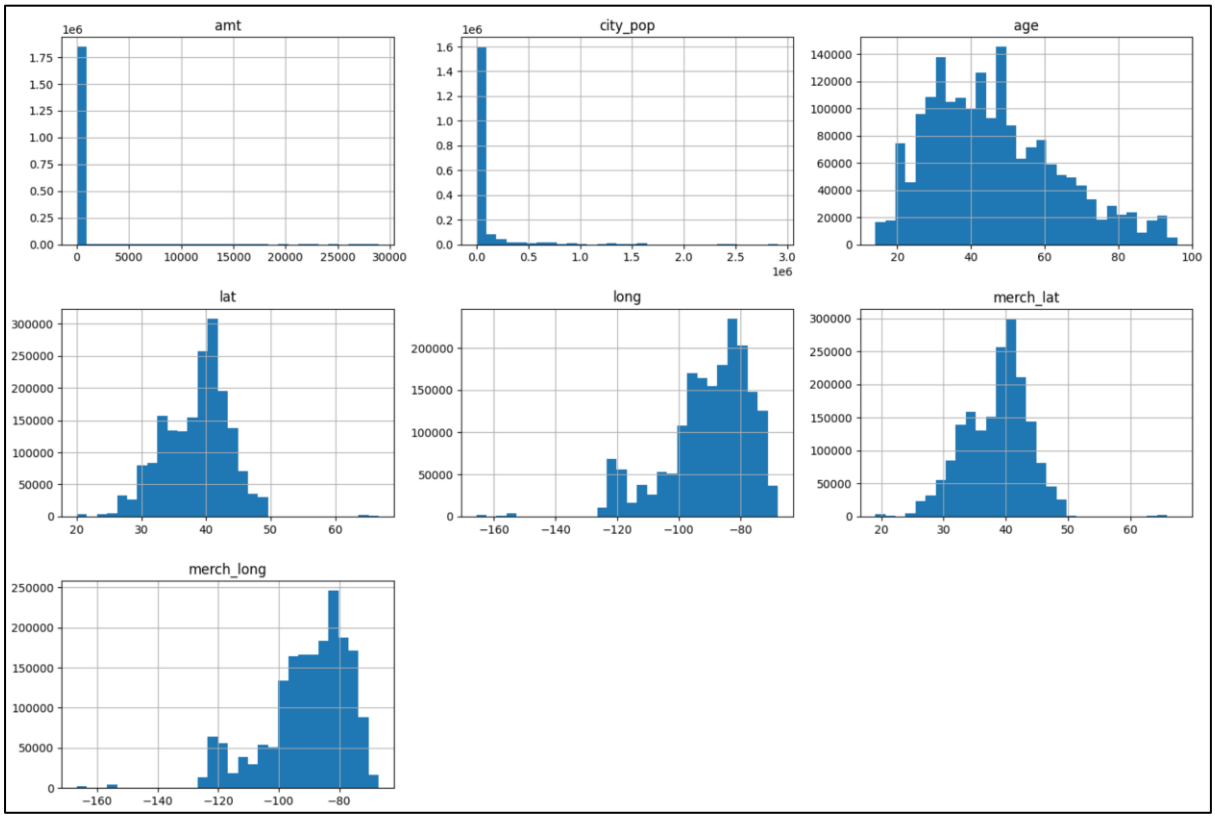


Univariate Analysis

Univariate analysis in EDA serves to examine individual variables, both numerical and categorical, to uncover their inherent patterns and potential issues. For numerical variables, it reveals distributions (e.g., central tendency, spread, skewness, and outliers) and detects data quality concerns like missing values or anomalies.

For categorical variables, it assesses frequency distributions, identifies class imbalances, and flags rare or dominant categories. This foundational step ensures data integrity, guides feature preprocessing, and informs subsequent multivariate analysis, ultimately leading to more robust modeling decisions.

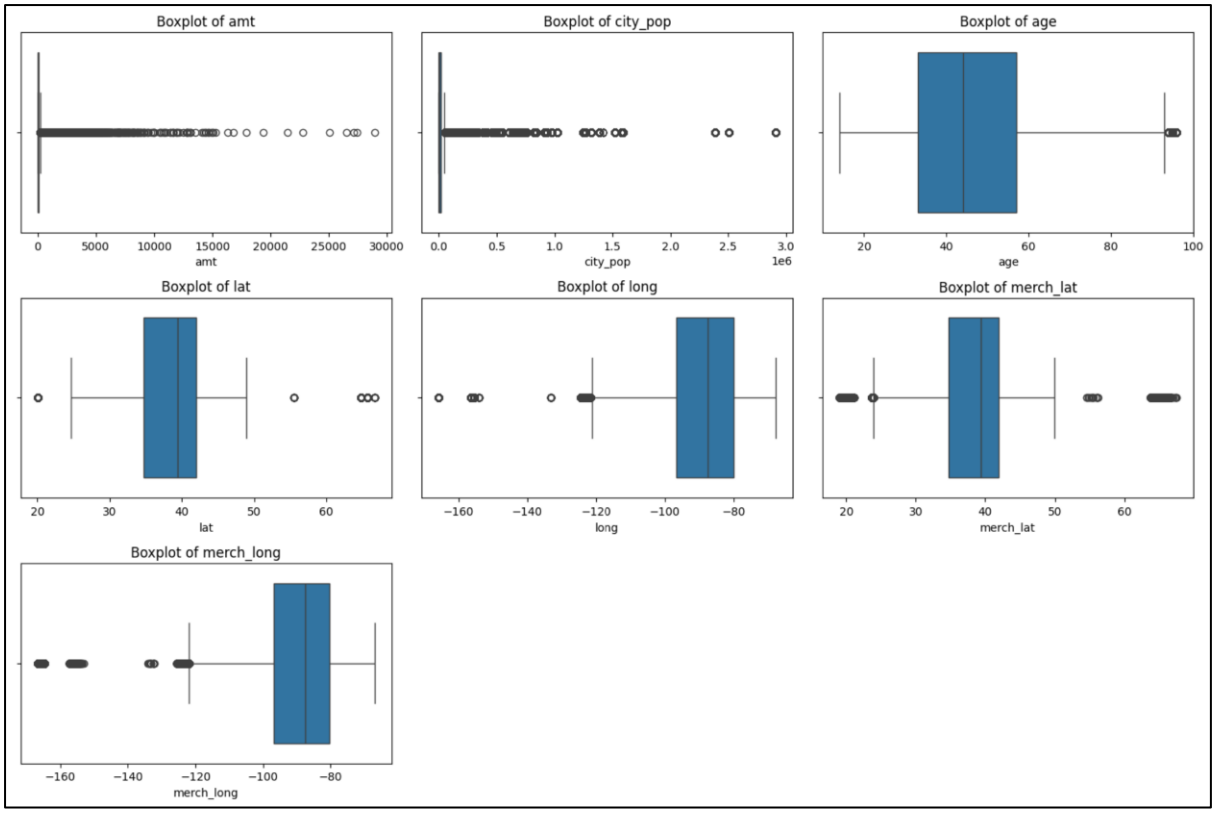
```
# Histograms for numerical variables
num_cols = ['amt','city_pop','age', 'lat', 'long', 'merch_lat', 'merch_long']
df[num_cols].hist(bins=30, figsize=(15, 10))
plt.tight_layout()
plt.show()
```



To effectively analyze the numerical data, create boxplots for all variables. This visualization will serve two key purposes: clearly identifying outliers that may distort analysis while simultaneously revealing the underlying distribution patterns of each feature.

Boxplots provide a concise yet comprehensive view of the data's spread, central tendency, and potential anomalies making them an essential diagnostic tool during exploratory analysis.

```
# Boxplots to detect outliers
plt.figure(figsize=(15, 10))  # Adjust size as needed

# Loop through each column and plot in subplots
for i, col in enumerate(num_cols, 1):  # Start index at 1 for subplot numbering
    plt.subplot((len(num_cols) // 3) + 1, 3, i)  # (rows, cols, position)
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')

plt.tight_layout()  # Prevent overlapping
plt.show()
```

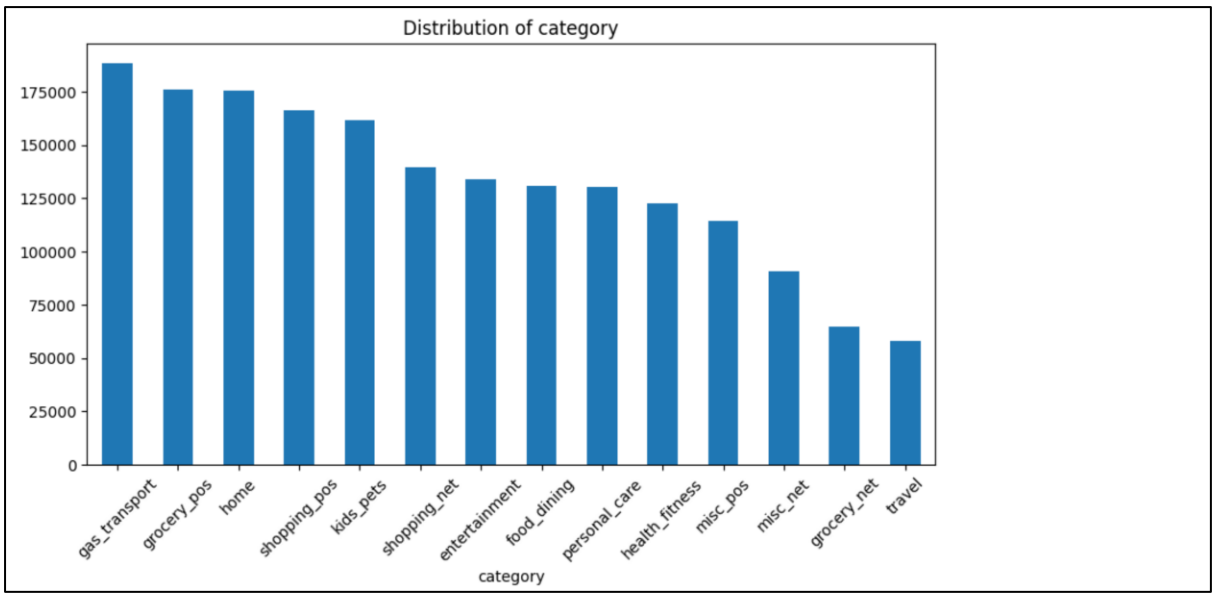


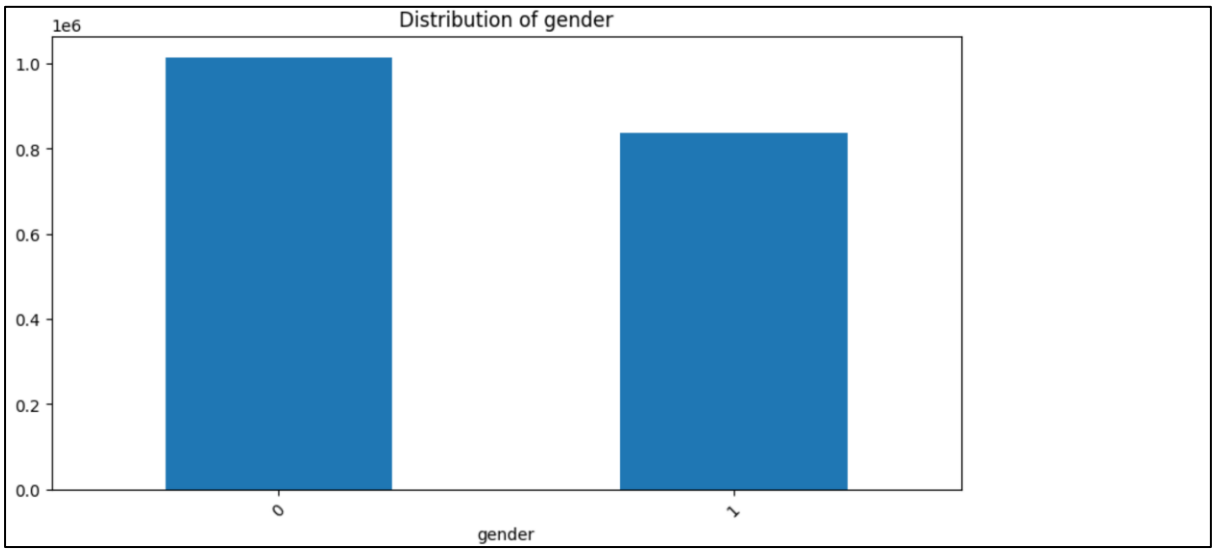
Key Takeaways:

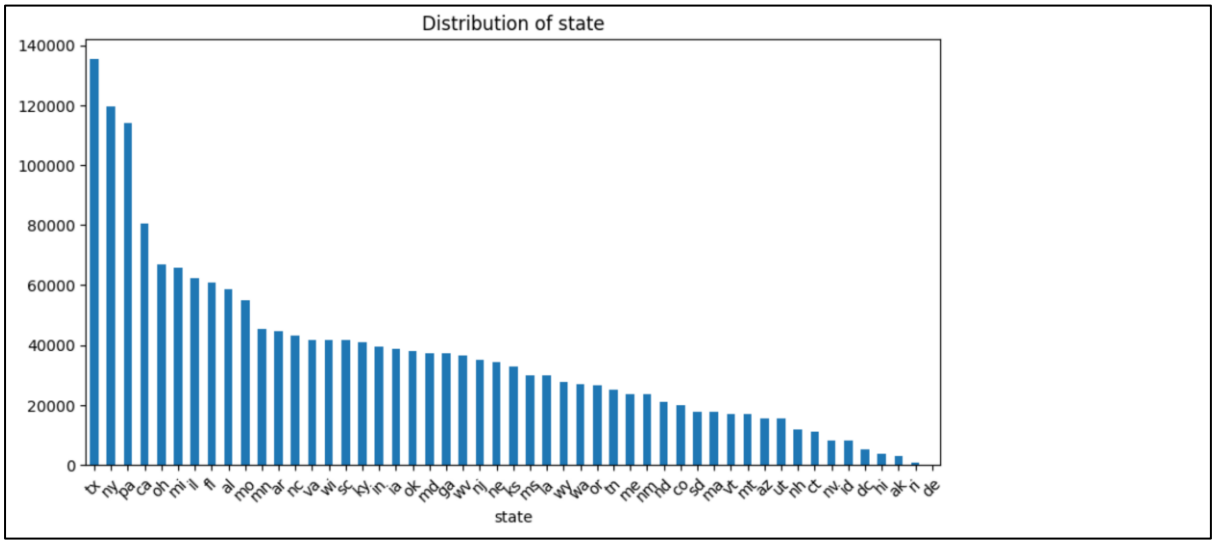
- Variable **amt** is highly **right skewed**, with most transactions below **$5,000** but extreme values up to **$30,000**. Thus, it is necessary to apply log transformation to reduce skewness.
- Variable **city_pop** has been normalized for the analysis. The graph suggests most of the transactions occurred in low-population areas.
- Variables **age**, **merch_lat** and **long** reveal correct statistical distributions based on its characteristics.

Now let's proceed with the univariate analysis on low-cardinality categorical variables (*category, gender, state*) because their limited distinct values enable clear visualization and interpretable frequency distributions. In contrast, high-cardinality variables (*merchant, job, city*) are unsuitable for direct visualization due to excessive unique values, which cause cluttered plots, fragmented data interpretation, and computational inefficiency.

```
# Univariate analysis for categorical variables
cat_cols = ['category', 'gender', 'state']
for col in cat_cols:
    plt.figure(figsize=(10, 5))
    df[col].value_counts().plot(kind='bar')
    plt.title(f'Distribución de {col}')
    plt.xticks(rotation=45)
    plt.show()
```







Key Takeaways:

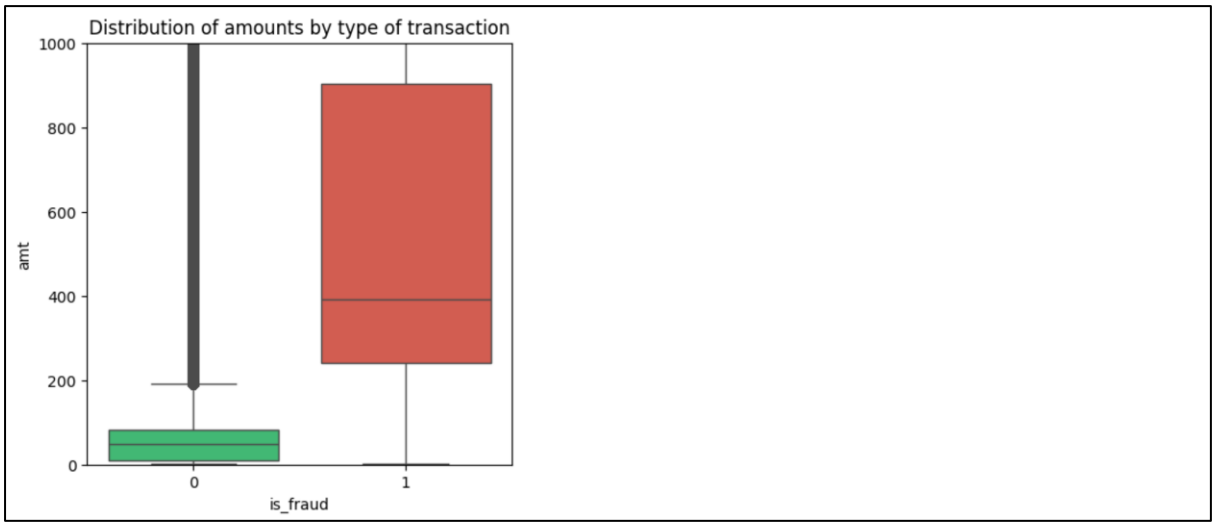
- **grocery_pos** (likely "grocery point-of-sale") and **shopping_pos** have the highest transaction volumes, suggesting frequent consumer spending in these sectors.
- One gender, **female**, dominates transactions significantly. This result might follow cultural spending patterns (e.g., primary shoppers in households).
- The distribution of transactions across states reveals a **highly concentrated pattern**, with **80% of activity occurring in just 10 states.** Fraud may cluster in high-volume states.

Bivariate Analysis

Bivariate analysis is critical in EDA as it reveals relationships between variables, guiding feature selection and model development. By examining pairs of variables, it uncovers meaningful patterns like higher fraud rates in specific categories or time periods—that univariate analysis misses. This directly informs risk modeling, helping prioritize high-impact features and detect interaction effects.

Fraud by Amount

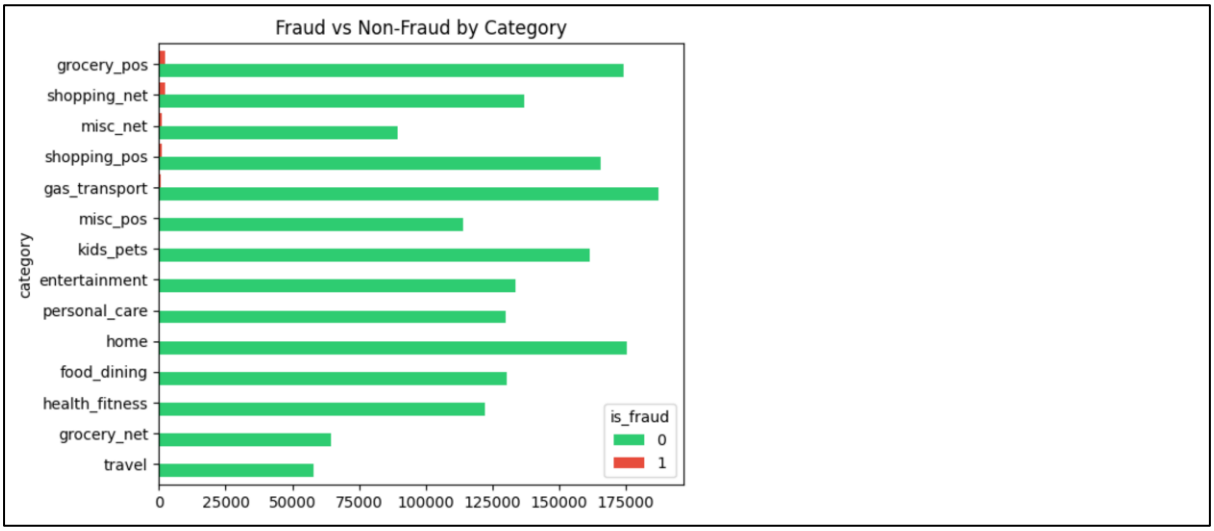
```
# Fraud by Amount
plt.figure(figsize=(5, 5))
sns.boxplot(x='is_fraud', y='amt', hue='is_fraud', palette=['#2ecc71','#e74c3c'], data=df, legend=False)  # Suppress duplicate legend
plt.title('Fraud vs Non-Fraud by Amount')
plt.ylim(0, 1000)
plt.show()
```



The boxplot reveals that fraudulent transactions (is_fraud=1) have significantly higher median amounts (~$400) compared to legitimate ones (~$50), with greater variability suggesting diverse fraud tactics. Non-fraud transactions show tighter clustering below $200. This clear monetary distinction makes transaction amount (amt) a critical fraud indicator.

Fraud by Category

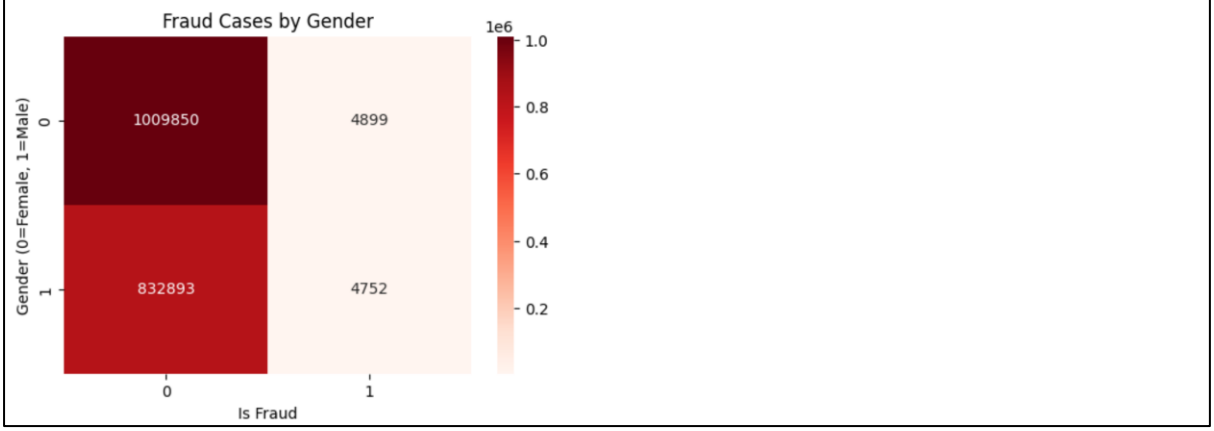
```
# Fraud by Category
pd.crosstab(df['category'], df['is_fraud'])\
  .sort_values(1)\
  .plot(kind='barh',width=0.85,color=['#2ecc71', '#e74c3c'])
plt.title('Fraud vs Non-Fraud by Category')
plt.tight_layout()
plt.show()
```



The fraud by category graph reveals clear concentration patterns. High-volume sectors like grocery POS and online shopping (shopping_net) show the most fraud instances, likely due to frequent transactions and card-skimming opportunities, while low-frequency categories (travel, health/fitness) exhibit minimal fraud. This suggests fraudsters target everyday spending channels rather than specialized purchases.

Fraud by Gender

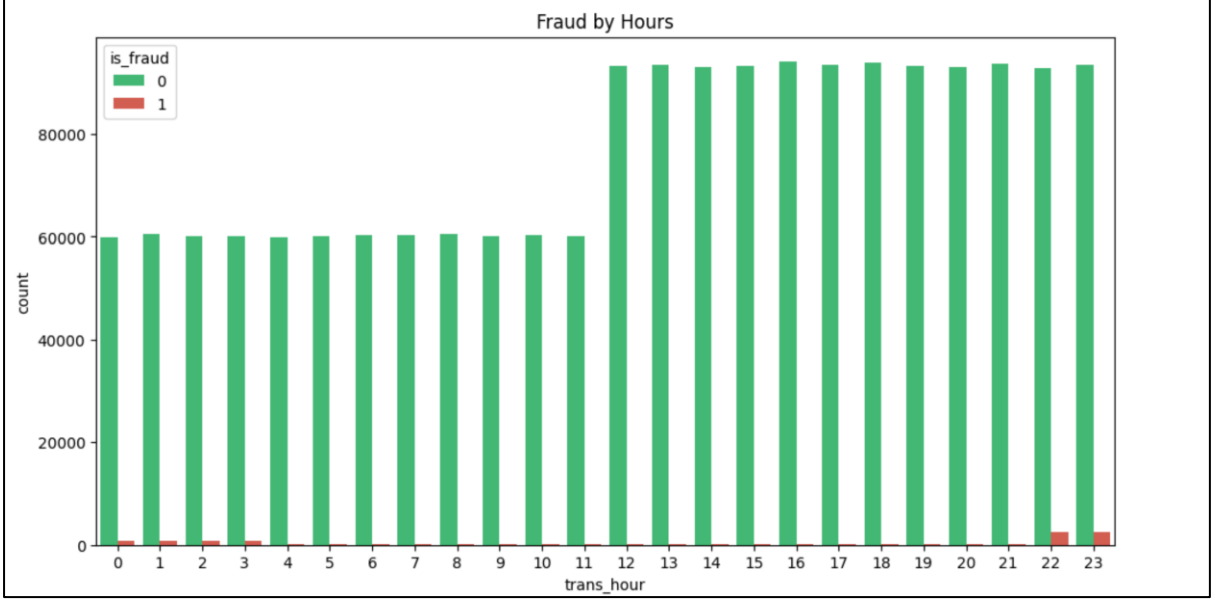
```
# Fraud by Gender
# Create a crosstab
ct = pd.crosstab(df['gender'], df['is_fraud'])

# Plot heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(ct, annot=True, fmt='d', cmap='Reds')
plt.title('Fraud Cases by Gender')
plt.xlabel('Is Fraud')
plt.ylabel('Gender (0=Female, 1=Male)')
plt.show()
```



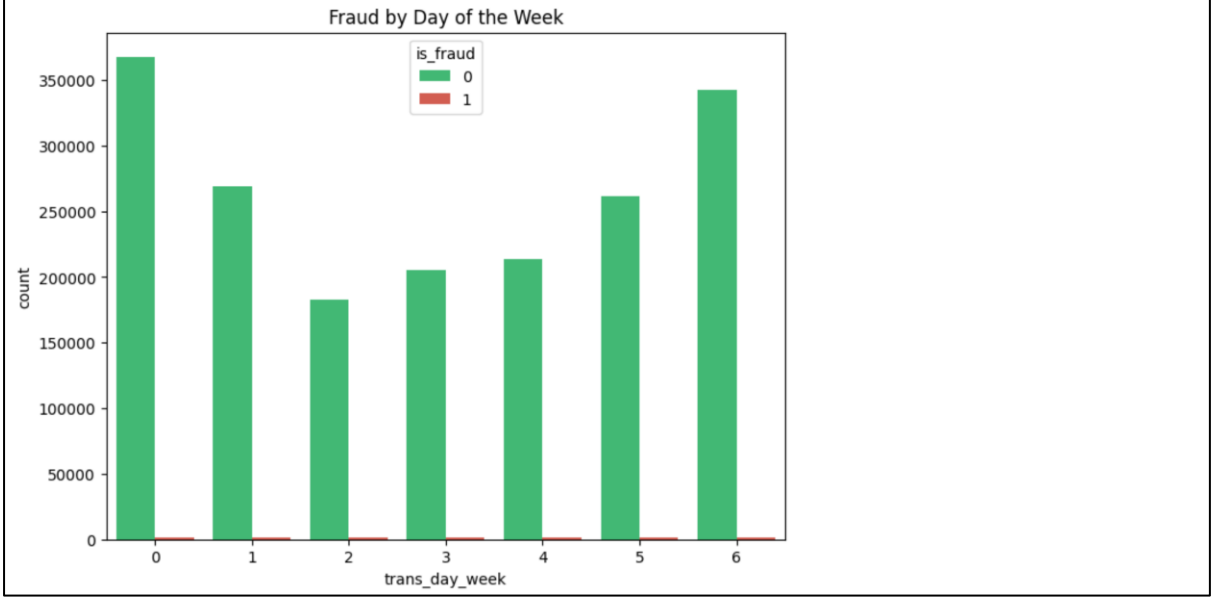
The fraud by gender analysis reveals a marginal difference, with females (4,899 cases) slightly exceeding males (4,752 cases) in absolute fraud counts. However, when normalized by transaction volume, fraud rates are close (0.49% vs. 0.57%), indicating no meaningful gender-based risk disparity. Thus, for fraud detection, gender is a weak predictor—priority should instead of focusing on stronger behavioral and transactional patterns like time, amount, or location.

Temporal Analysis

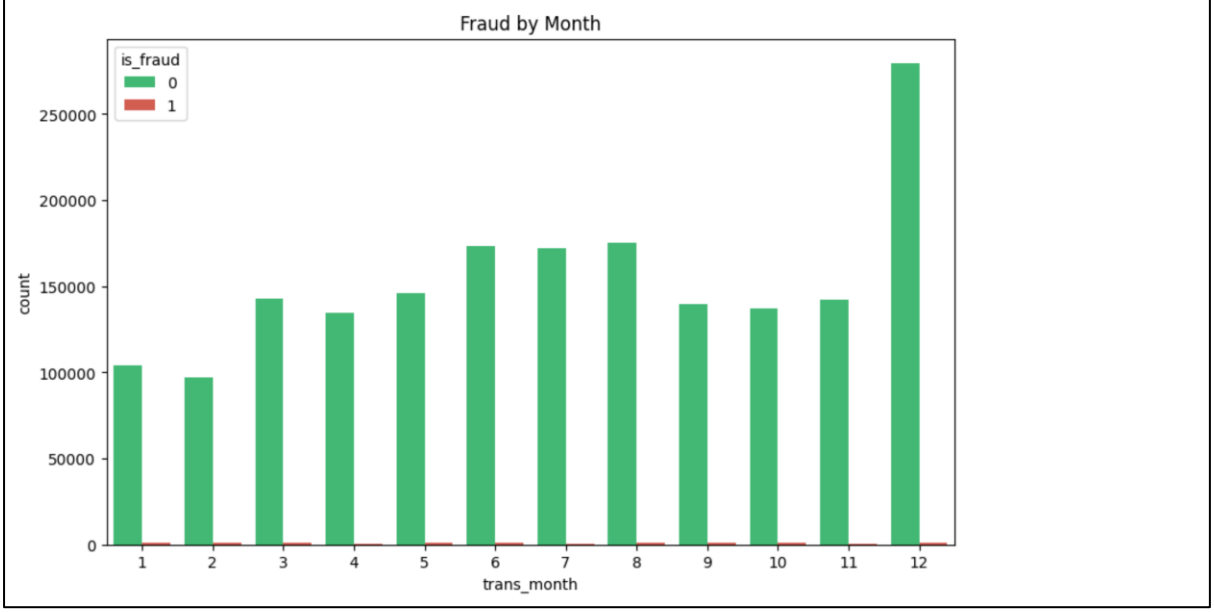
```
# Fraud by hours
plt.figure(figsize=(12, 6))
sns.countplot(x='trans_hour', hue='is_fraud',palette=['#2ecc71','#e74c3c'], data=df)
plt.title('Fraud by Hours')
plt.show()
```




```
# Fraud by Day of the Week
plt.figure(figsize=(8, 6))
sns.countplot(x='trans_day_week', hue='is_fraud',palette=['#2ecc71','#e74c3c'], data=df)
plt.title('Fraud by Day of the Week')
plt.show()
```



```
# Fraud by month
plt.figure(figsize=(10, 6))
sns.countplot(x='trans_month', hue='is_fraud',palette=['#2ecc71','#e74c3c'], data=df)
plt.title('Fraud by Month')
plt.show()
```

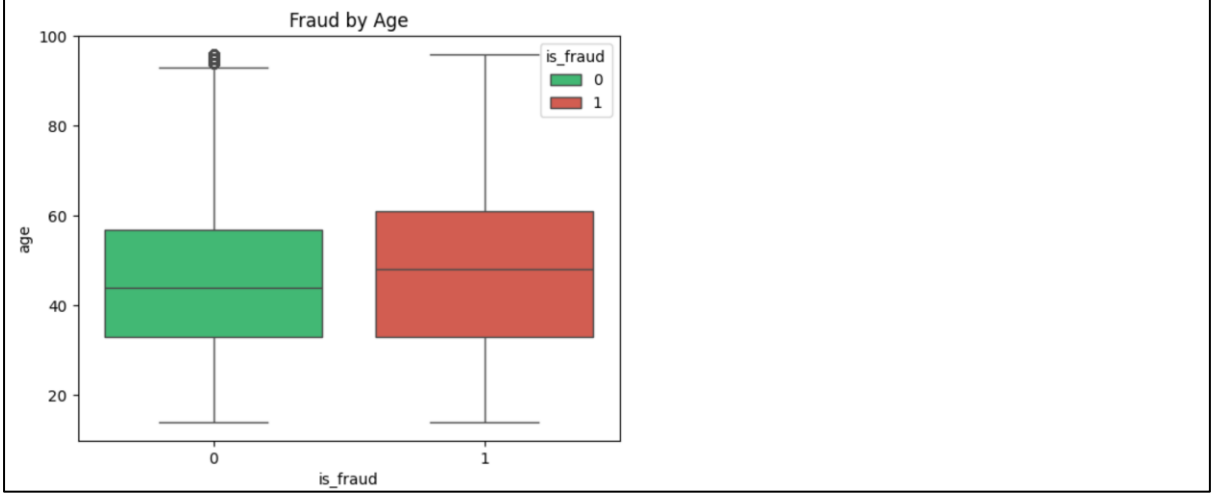


Key Takeaways:

- Fraud activity peaks between midnight and 6 a.m., a period typically marked by reduced monitoring and oversight.
- Fraud shows no significant weekday/weekend variation - remains consistently proportional. Fraudsters operate uniformly across all days.
- Summer months (Jun-Aug) show the highest fraud volume. December shows disproportionate fraud despite lower overall transactions.

Fraud by Age

```
# Fraud by Age
sns.boxplot(x='is_fraud', y='age', hue='is_fraud', palette=['#2ecc71','#e74c3c'], data=df)
plt.title('Fraud by Age')
plt.show()
```

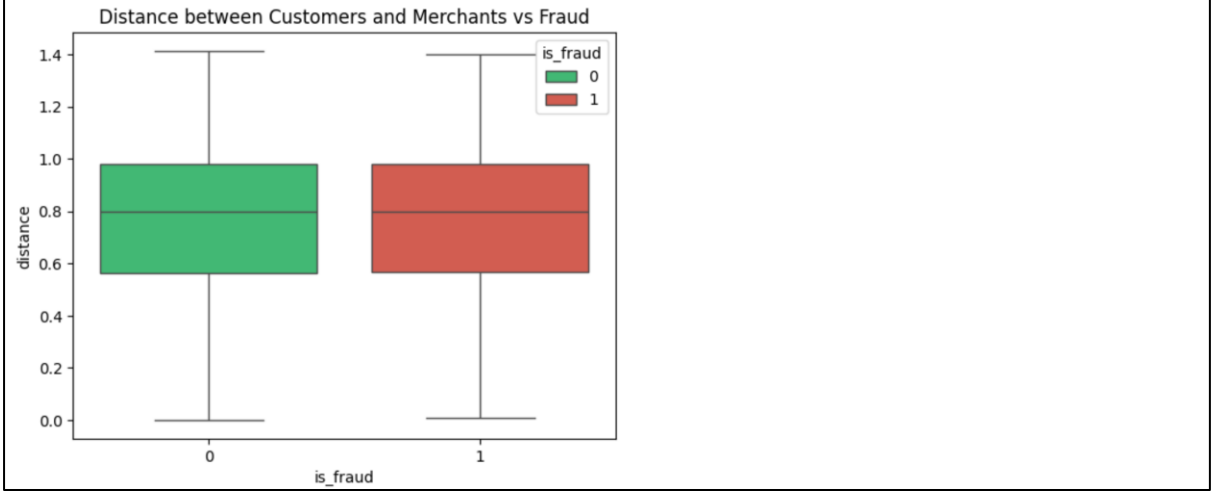


The age distribution analysis through these boxplots reveals that fraudulent transactions (is_fraud = 1) are slightly more common among older clients, with a median age around 50 compared to 44 for non-fraudulent ones. The age range of fraudsters is wider, indicating greater variability, though it lacks extreme outliers seen in the non-fraud group. Overall, fraud tends to occur more frequently among individuals aged 35 to 65.

Geographic Analysis

To perform this geographic analysis, a distance variable must be created within the DataFrame 'df'. This variable will represent the distance between the client's residence and the merchant location where the transaction occurred.

```
# Distance between the customer's location and the merchants
df['distance'] = np.sqrt((df['lat'] - df['merch_lat'])**2 + (df['long'] - df['merch_long'])**2)

# Fraud by Distance
sns.boxplot(x='is_fraud', y='distance', hue='is_fraud', palette=['#2ecc71','#e74c3c'], data=df)
plt.title('Distance between Customers and Merchants vs Fraud')
plt.show()
```



The boxplot shows that the median distance between customers and merchants is similar for both fraudulent and non-fraudulent transactions. Fraudulent transactions (is_fraud = 1) display slightly higher variability, but no significant difference is observed overall. This suggests that distance alone is not a strong indicator of fraud.

Fraud by Credit Card

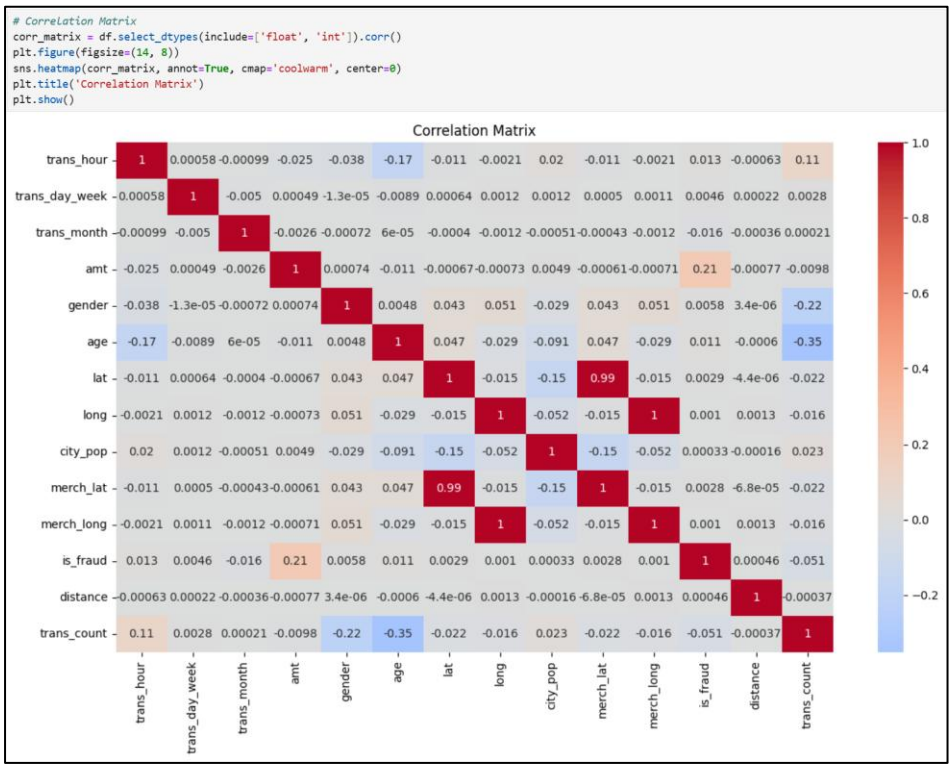
The goal is to identify how frequently each credit card has been used for transactions. To achieve this, a new variable, **trans_count**, will be derived from **cc_num** to capture the total number of transactions associated with each card.

```
# Analysis of repeated transactions
# Frequency of card transactions
trans_per_card = df['cc_num'].value_counts()
df['trans_count'] = df['cc_num'].map(trans_per_card)

# Relationship between frequency of use and fraud
sns.boxplot(x='is_fraud', y='trans_count',hue='is_fraud', palette=['#2ecc71','#e74c3c'], data=df)
plt.title('Number of card transactions vs. fraud')
plt.show()
```

The boxplot shows that fraudulent transactions (is_fraud = 1) are more common on credit cards with fewer total transactions, indicating a lower usage history. In contrast, non-fraudulent transactions are associated with cards that have higher and more consistent activity, with a median near 3,000 transactions. This suggests that fraud tends to occur on less frequently used or potentially newer cards.

Correlation Analysis

We are going to conclude this EDA with a correlation analysis. Correlation analysis is essential to identify significant relationships between variables, prioritize impactful features for modeling, and detect multicollinearity that could skew results. It reveals actionable patterns while flagging data quirks. This step ensures models are built on robust, non-redundant predictors and avoids wasted effort on irrelevant variables.



Key Takeaways:

- Most correlated variable with target variables is **amt (transaction amount)**, with a moderate positive correlation of **0.21**, suggesting that higher transaction amounts are associated with fraud.
- All other variables show very weak or negligible correlation with **is_fraud** (absolute values < 0.05), including **trans_hour**, **distance**, **trans_count**, and location-related features.
- **age (-0.35)** and **gender (-0.22)** show moderate negative correlation with **trans_count**, indicating that older users and a specific gender group may transact less frequently, but not linked to fraud.
- A strong positive correlation (**0.99**) between **lat and merch_lat** suggests potential multicollinearity, which should be addressed if using both in modeling.