Credit Card Fraud Detection

Dipanta

2023-08-18

Problem statement

- Banks have a big issue with credit card fraud, where people try to cheat by using fake transactions. So, we want to create a computer program that can look at past customer transactions and figure out if they're fake or real.
- we'll show the bank people how much money the program could save them and give them ideas on how to stop the cheating.

Understanding the data set

```
#importing the packages

library(tidyverse)
library(knitr)
library(gridExtra)
library(corrplot)
library(caTools)
library(caret)
library(e1071)
library(ROCR)
library(ROSE)
```

```
#Loading the data set

data1= read.csv("C:/Users/DIPANTA MISTRY/OneDrive/Documents/R_dataset/fraudTrain.csv")
data2 = read.csv("C:/Users/DIPANTA MISTRY/OneDrive/Documents/R_dataset/fraudTest.csv")

#combining both data sets

df = rbind(data1,data2)
glimpse(df)
```

```
## Rows: 1,852,394
## Columns: 23
## $ X                     <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14~
## $ trans_date_trans_time <chr> "2019-01-01 00:00:18", "2019-01-01 00:00:44", "2~
## $ cc_num                <dbl> 2.703186e+15, 6.304233e+11, 3.885949e+13, 3.5340~
## $ merchant              <chr> "fraud_Rippin, Kub and Mann", "fraud_Heller, Gut~
```

```
## $ category              <chr> "misc_net", "grocery_pos", "entertainment", "gas~
## $ amt                   <dbl> 4.97, 107.23, 220.11, 45.00, 41.96, 94.63, 44.54~
## $ first                 <chr> "Jennifer", "Stephanie", "Edward", "Jeremy", "Ty~
## $ last                  <chr> "Banks", "Gill", "Sanchez", "White", "Garcia", "~
## $ gender                <chr> "F", "F", "M", "M", "M", "F", "F", "M", "F", "F"~
## $ street                <chr> "561 Perry Cove", "43039 Riley Greens Suite 393"~
## $ city                  <chr> "Moravian Falls", "Orient", "Malad City", "Bould~
## $ state                 <chr> "NC", "WA", "ID", "MT", "VA", "PA", "KS", "VA", ~
## $ zip                   <int> 28654, 99160, 83252, 59632, 24433, 18917, 67851,~
## $ lat                   <dbl> 36.0788, 48.8878, 42.1808, 46.2306, 38.4207, 40.~
## $ long                  <dbl> -81.1781, -118.2105, -112.2620, -112.1138, -79.4~
## $ city_pop              <int> 3495, 149, 4154, 1939, 99, 2158, 2691, 6018, 147~
## $ job                   <chr> "Psychologist, counselling", "Special educationa~
## $ dob                   <chr> "1988-03-09", "1978-06-21", "1962-01-19", "1967-~
## $ trans_num             <chr> "0b242abb623afc578575680df30655b9", "1f76529f857~
## $ unix_time             <int> 1325376018, 1325376044, 1325376051, 1325376076, ~
## $ merch_lat             <dbl> 36.01129, 49.15905, 43.15070, 47.03433, 38.67500~
## $ merch_long            <dbl> -82.04832, -118.18646, -112.15448, -112.56107, -~
## $ is_fraud              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
```

Data Preprocessing

```
#Converting trans_date_trans_time as date time
df$trans_date_trans_time= as_datetime(df$trans_date_trans_time)
```

- Lets check the data balance in your data set for target variable, 'is_fraud'.

```
unique_counts= sapply(df, function(col) length(unique(col)))
print(unique_counts)
```

```
##                     X trans_date_trans_time                cc_num
##               1296675               1819551                   999
##              merchant              category                   amt
##                   693                    14                 60616
##                 first                  last                gender
##                   355                   486                     2
##                street                  city                 state
##                   999                   906                    51
##                   zip                   lat                  long
##                   985                   983                   983
##              city_pop                   job                   dob
##                   891                   497                   984
##             trans_num             unix_time             merch_lat
##               1852394               1819583               1754157
##            merch_long              is_fraud
##               1809753                     2
```

- Splitting the trans_date_trans_time column and making different column say hour, day, month-year to get more valuable information.

```r
#making hour col
df$trans_hour = hour(df$trans_date_trans_time)

#making weeks days column
df$trans_day_of_week = weekdays(df$trans_date_trans_time)

#making year_month column
df$trans_year_month = format(df$trans_date_trans_time, '%y-%m')

glimpse(df)
```

```
## Rows: 1,852,394
## Columns: 26
## $ X                     <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14~
## $ trans_date_trans_time <dttm> 2019-01-01 00:00:18, 2019-01-01 00:00:44, 2019-~
## $ cc_num                <dbl> 2.703186e+15, 6.304233e+11, 3.885949e+13, 3.5340~
## $ merchant              <chr> "fraud_Rippin, Kub and Mann", "fraud_Heller, Gut~
## $ category              <chr> "misc_net", "grocery_pos", "entertainment", "gas~
## $ amt                   <dbl> 4.97, 107.23, 220.11, 45.00, 41.96, 94.63, 44.54~
## $ first                 <chr> "Jennifer", "Stephanie", "Edward", "Jeremy", "Ty~
## $ last                  <chr> "Banks", "Gill", "Sanchez", "White", "Garcia", "~
## $ gender                <chr> "F", "F", "M", "M", "M", "F", "F", "M", "F", "F"~
## $ street                <chr> "561 Perry Cove", "43039 Riley Greens Suite 393"~
## $ city                  <chr> "Moravian Falls", "Orient", "Malad City", "Bould~
## $ state                 <chr> "NC", "WA", "ID", "MT", "VA", "PA", "KS", "VA", ~
## $ zip                   <int> 28654, 99160, 83252, 59632, 24433, 18917, 67851,~
## $ lat                   <dbl> 36.0788, 48.8878, 42.1808, 46.2306, 38.4207, 40.~
## $ long                  <dbl> -81.1781, -118.2105, -112.2620, -112.1138, -79.4~
## $ city_pop              <int> 3495, 149, 4154, 1939, 99, 2158, 2691, 6018, 147~
## $ job                   <chr> "Psychologist, counselling", "Special educationa~
## $ dob                   <chr> "1988-03-09", "1978-06-21", "1962-01-19", "1967-~
## $ trans_num             <chr> "0b242abb623afc578575680df30655b9", "1f76529f857~
## $ unix_time             <int> 1325376018, 1325376044, 1325376051, 1325376076, ~
## $ merch_lat             <dbl> 36.01129, 49.15905, 43.15070, 47.03433, 38.67500~
## $ merch_long            <dbl> -82.04832, -118.18646, -112.15448, -112.56107, -~
## $ is_fraud              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ trans_hour            <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ trans_day_of_week     <chr> "Tuesday", "Tuesday", "Tuesday", "Tuesday", "Tue~
## $ trans_year_month      <chr> "19-01", "19-01", "19-01", "19-01", "19-01", "19~
```

- Let us find the age of the customer

```r
#converting dob col as date

df$dob = as.Date(df$dob)

# Calculate age based on date of birth
df$age = year(df$trans_date_trans_time) - year(df$dob)

glimpse(df)
```

```
## Rows: 1,852,394
```

```
## Columns: 27
## $ X                     <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14~
## $ trans_date_trans_time <dttm> 2019-01-01 00:00:18, 2019-01-01 00:00:44, 2019-~
## $ cc_num                <dbl> 2.703186e+15, 6.304233e+11, 3.885949e+13, 3.5340~
## $ merchant              <chr> "fraud_Rippin, Kub and Mann", "fraud_Heller, Gut~
## $ category              <chr> "misc_net", "grocery_pos", "entertainment", "gas~
## $ amt                   <dbl> 4.97, 107.23, 220.11, 45.00, 41.96, 94.63, 44.54~
## $ first                 <chr> "Jennifer", "Stephanie", "Edward", "Jeremy", "Ty~
## $ last                  <chr> "Banks", "Gill", "Sanchez", "White", "Garcia", "~
## $ gender                <chr> "F", "F", "M", "M", "M", "F", "F", "M", "F", "F"~
## $ street                <chr> "561 Perry Cove", "43039 Riley Greens Suite 393"~
## $ city                  <chr> "Moravian Falls", "Orient", "Malad City", "Bould~
## $ state                 <chr> "NC", "WA", "ID", "MT", "VA", "PA", "KS", "VA", ~
## $ zip                   <int> 28654, 99160, 83252, 59632, 24433, 18917, 67851,~
## $ lat                   <dbl> 36.0788, 48.8878, 42.1808, 46.2306, 38.4207, 40.~
## $ long                  <dbl> -81.1781, -118.2105, -112.2620, -112.1138, -79.4~
## $ city_pop              <int> 3495, 149, 4154, 1939, 99, 2158, 2691, 6018, 147~
## $ job                   <chr> "Psychologist, counselling", "Special educationa~
## $ dob                   <date> 1988-03-09, 1978-06-21, 1962-01-19, 1967-01-12,~
## $ trans_num             <chr> "0b242abb623afc578575680df30655b9", "1f76529f857~
## $ unix_time             <int> 1325376018, 1325376044, 1325376051, 1325376076, ~
## $ merch_lat             <dbl> 36.01129, 49.15905, 43.15070, 47.03433, 38.67500~
## $ merch_long            <dbl> -82.04832, -118.18646, -112.15448, -112.56107, -~
## $ is_fraud              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ trans_hour            <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ trans_day_of_week     <chr> "Tuesday", "Tuesday", "Tuesday", "Tuesday", "Tue~
## $ trans_year_month      <chr> "19-01", "19-01", "19-01", "19-01", "19-01", "19~
## $ age                   <dbl> 31, 41, 57, 52, 33, 58, 26, 72, 78, 45, 29, 53, ~
```

- Now we can remove unneccessary columns

```r
#Removing cols
df = df %>%
  select(-trans_date_trans_time,-first,-last,-dob)
```

- Now the data set has only needed info, now we can proceed with other process

```r
#Take a look
summary(df)
```

```
##        X               cc_num             merchant            category        
##  Min.   :      0   Min.   :6.042e+10   Length:1852394     Length:1852394    
##  1st Qu.: 231549   1st Qu.:1.800e+14   Class :character   Class :character  
##  Median : 463098   Median :3.521e+15   Mode  :character   Mode  :character  
##  Mean   : 537193   Mean   :4.174e+17                                        
##  3rd Qu.: 833576   3rd Qu.:4.642e+15                                        
##  Max.   :1296674   Max.   :4.992e+18                                        
##       amt              gender             street              city          
##  Min.   :    1.00   Length:1852394     Length:1852394     Length:1852394    
##  1st Qu.:    9.64   Class :character   Class :character   Class :character  
##  Median :   47.45   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :   70.06
```

```
##  3rd Qu.:   83.10
##  Max.   :28948.90
##     state                zip             lat             long
##  Length:1852394     Min.   : 1257   Min.   :20.03   Min.   :-165.67
##  Class :character   1st Qu.:26237   1st Qu.:34.67   1st Qu.: -96.80
##  Mode  :character   Median :48174   Median :39.35   Median : -87.48
##                     Mean   :48813   Mean   :38.54   Mean   : -90.23
##                     3rd Qu.:72042   3rd Qu.:41.94   3rd Qu.: -80.16
##                     Max.   :99921   Max.   :66.69   Max.   : -67.95
##     city_pop           job             trans_num           unix_time
##  Min.   :      23   Length:1852394     Length:1852394     Min.   :1.325e+09
##  1st Qu.:     741   Class :character   Class :character   1st Qu.:1.343e+09
##  Median :    2443   Mode  :character   Mode  :character   Median :1.357e+09
##  Mean   :   88644                                         Mean   :1.359e+09
##  3rd Qu.:   20328                                         3rd Qu.:1.375e+09
##  Max.   :2906700                                          Max.   :1.389e+09
##    merch_lat       merch_long         is_fraud         trans_hour
##  Min.   :19.03   Min.   :-166.67   Min.   :0.00000   Min.   : 0.00
##  1st Qu.:34.74   1st Qu.: -96.90   1st Qu.:0.00000   1st Qu.: 7.00
##  Median :39.37   Median : -87.44   Median :0.00000   Median :14.00
##  Mean   :38.54   Mean   : -90.23   Mean   :0.00521   Mean   :12.81
##  3rd Qu.:41.96   3rd Qu.: -80.25   3rd Qu.:0.00000   3rd Qu.:19.00
##  Max.   :67.51   Max.   : -66.95   Max.   :1.00000   Max.   :23.00
##  trans_day_of_week  trans_year_month        age
##  Length:1852394     Length:1852394     Min.   :14.00
##  Class :character   Class :character   1st Qu.:33.00
##  Mode  :character   Mode  :character   Median :44.00
##                                        Mean   :46.21
##                                        3rd Qu.:57.00
##                                        Max.   :96.00
```

- From the above summarization we can see that there is no missing value in our dataset.

```
#Store a copy
df_copy = df
```

Explaratory Data Analysis

- Let us check the percentage of fraud transaction

```
value=table(df$is_fraud)
print(prop.table(value)*100)
```

```
##
##          0          1
## 99.4789985  0.5210015
```

- From the above section we can clearly see the presence of data imbalance. So we have to balance the data to avoid any biases.

Exploring the Amount data

- overall summary

```
summary(df$amt)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     1.00     9.64    47.45    70.06    83.10 28948.90
```

- Non-fraud transaction summary

```
summary(df$amt[df$is_fraud==0])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     1.00     9.61    47.24    67.65    82.56 28948.90
```

- Fraud transaction summary

```
summary(df$amt[df$is_fraud==1])
```

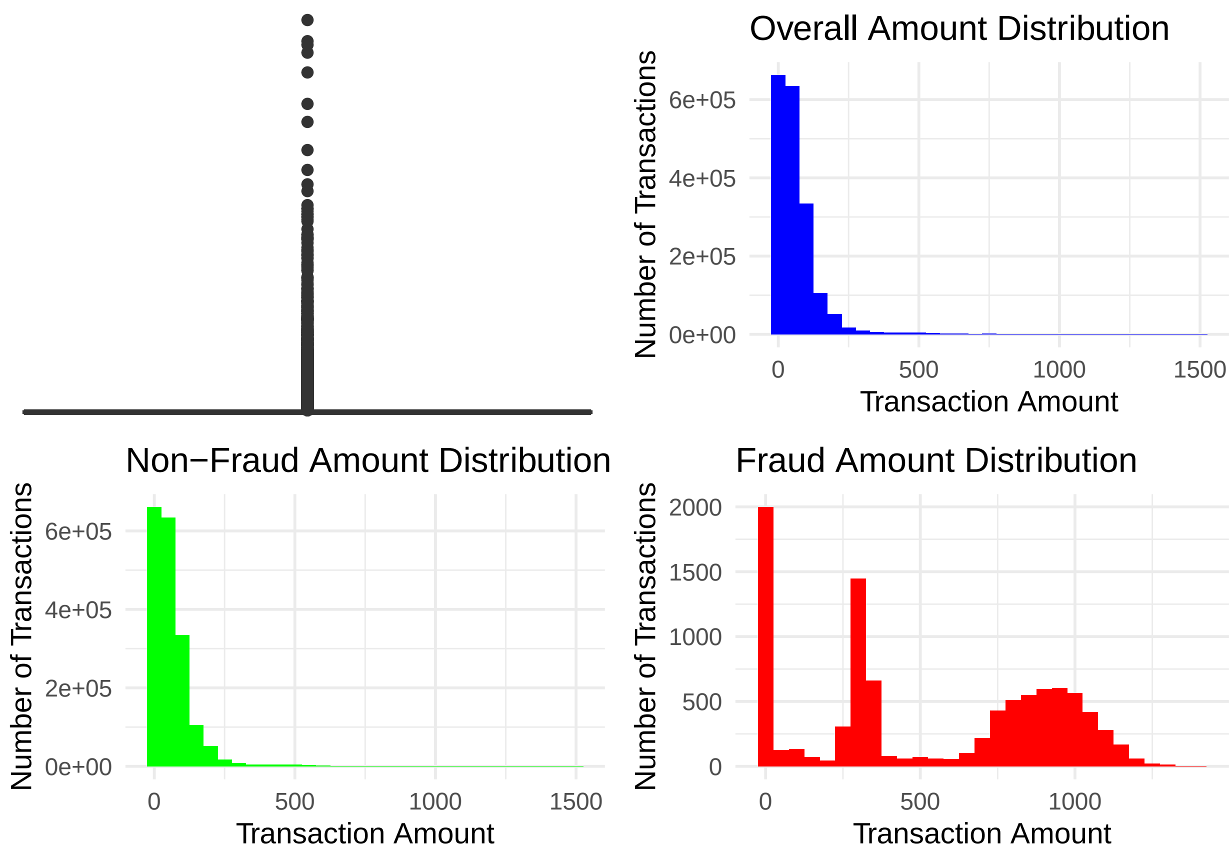
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.06  240.07  390.00  530.66  902.37 1376.04
```

- From the above analysis we can see that the mean transaction in fraud case is high compare to non-fraud case.

```
# Create a list to store plots
plots = list()

# Create a boxplot
plots[[1]] = ggplot(df, aes(x = 1, y = amt)) +
  geom_boxplot() +
  labs(x = NULL, y = "Transaction Amount") +
  theme_void()

# Create distribution plots
plots[[2]] = ggplot(df[df$amt <= 1500, ], aes(x = amt)) +
  geom_histogram(binwidth = 50, fill = "blue") +
  labs(title = "Overall Amount Distribution",
       x = "Transaction Amount",
       y = "Number of Transactions") +
  theme_minimal()

plots[[3]] = ggplot(subset(df, is_fraud == 0 & amt <= 1500), aes(x = amt)) +
  geom_histogram(binwidth = 50, fill = "green") +
  labs(title = "Non-Fraud Amount Distribution",
       x = "Transaction Amount",
       y = "Number of Transactions") +
```

```
  theme_minimal()

plots[[4]] = ggplot(subset(df, is_fraud == 1 & amt <= 1500), aes(x = amt)) +
  geom_histogram(binwidth = 50, fill = "red") +
  labs(title = "Fraud Amount Distribution",
       x = "Transaction Amount",
       y = "Number of Transactions") +
  theme_minimal()

# Arrange and print the plots

grid.arrange(grobs = plots, ncol = 2, nrow = 2)
```

Plot the above distribution

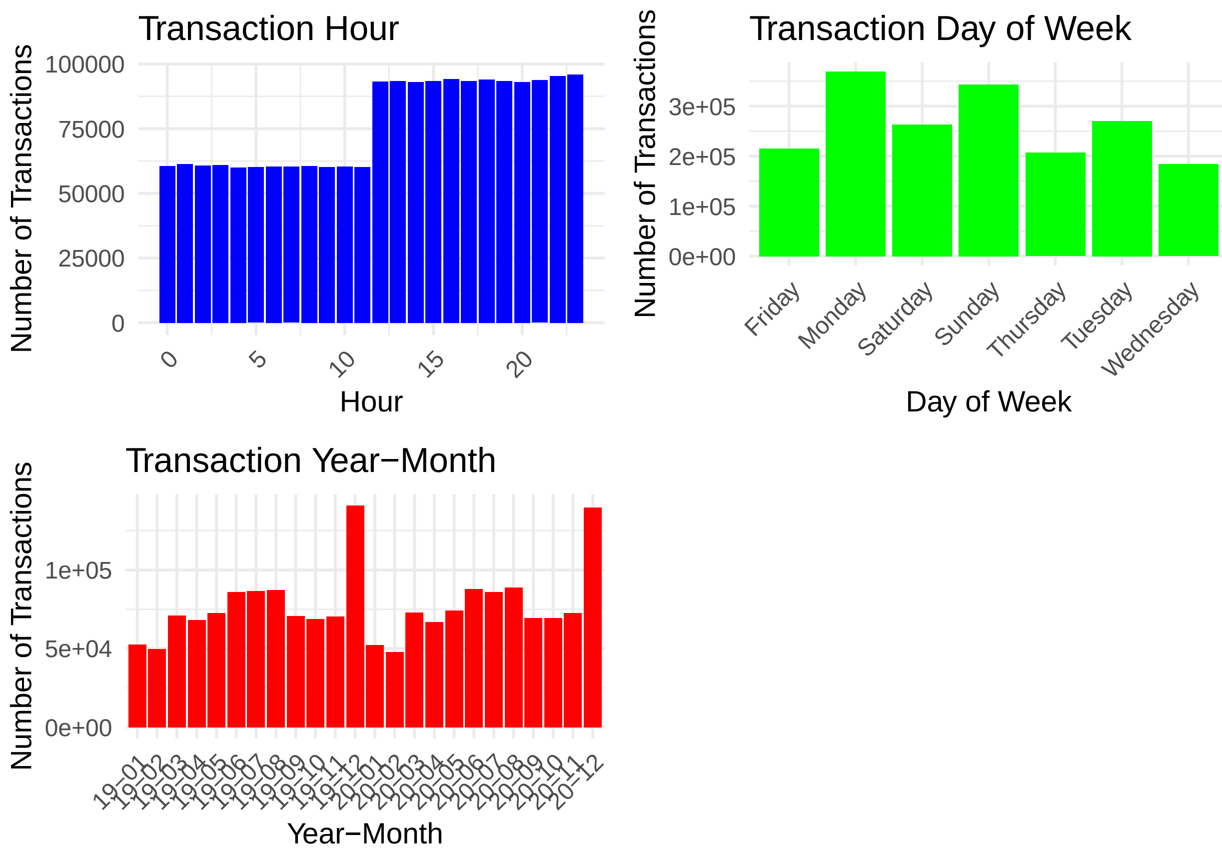


- From the above plots we can see that: The 'amt' feature has lots of outliers in the data. The distribution of the overall amount is and non fraud amount is similar. The skewness of the data distribution can be seen.

Exploring the Time data

```
# Plotting 'trans_hour' feature
plot_trans_hour = ggplot(df, aes(x = trans_hour)) +
```

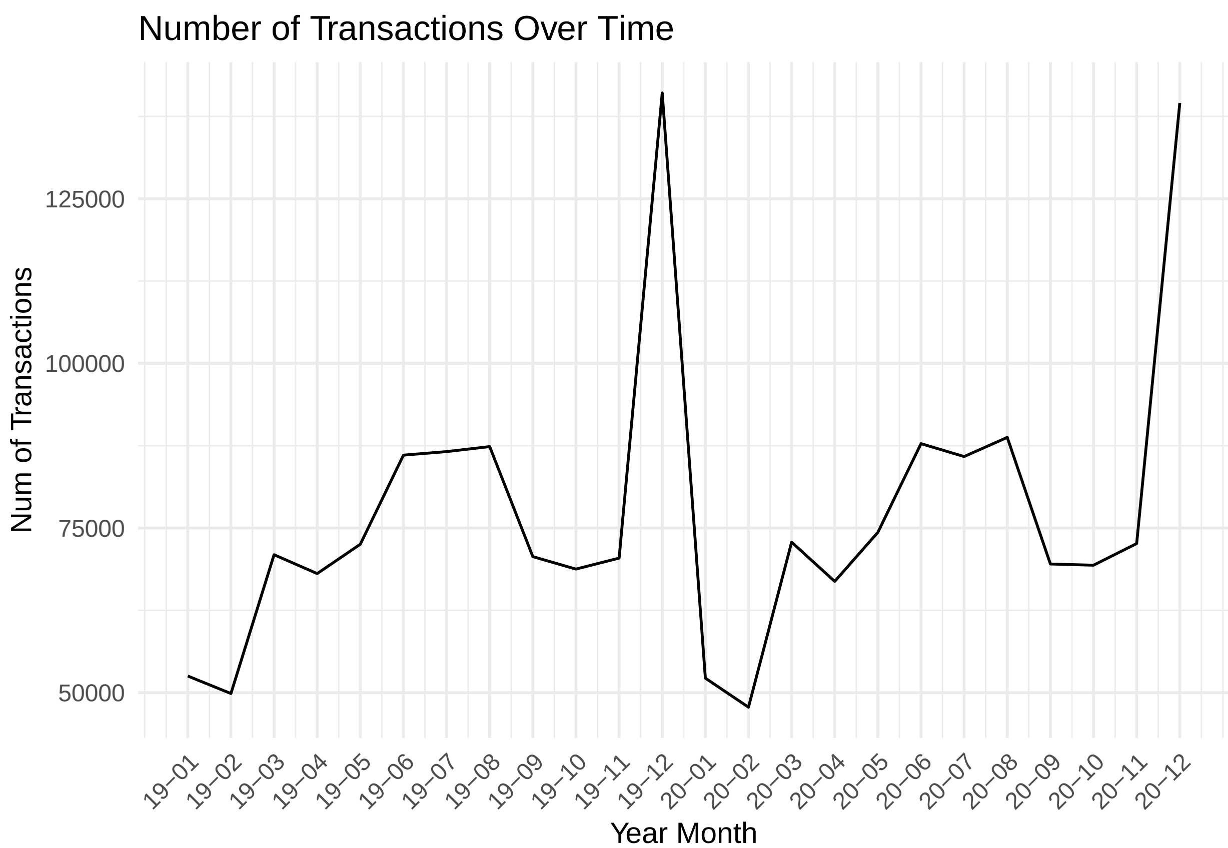
```r
  geom_bar(fill = "blue") +
  labs(title = "Transaction Hour",
       x = "Hour",
       y = "Number of Transactions") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Plotting 'trans_day_of_week' feature
plot_trans_day = ggplot(df, aes(x = trans_day_of_week)) +
  geom_bar(fill = "green") +
  labs(title = "Transaction Day of Week",
       x = "Day of Week",
       y = "Number of Transactions") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Plotting 'trans_year_month' feature
plot_trans_year_month = ggplot(df, aes(x = trans_year_month)) +
  geom_bar(fill = "red") +
  labs(title = "Transaction Year-Month",
       x = "Year-Month",
       y = "Number of Transactions") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Arrange and print the plots
grid.arrange(plot_trans_hour, plot_trans_day, plot_trans_year_month, ncol = 2, nrow = 2)
```




```
# Group by 'trans_year_month' and calculate number of unique transactions and customers
df_timeline01 = df %>%
  group_by(trans_year_month) %>%
  summarise(num_of_transactions = n_distinct(trans_num),
            customers = n_distinct(cc_num)) %>%
  ungroup() %>%
  rename(year_month = trans_year_month)
```

- Now plot the above distribution

```
# Create a sequence for x-axis
x = seq(1, nrow(df_timeline01), 1)

# Create the plot using ggplot
ggplot(df_timeline01, aes(x = x, y = num_of_transactions)) +
  geom_line() +
  scale_x_continuous(breaks = x, labels = df_timeline01$year_month) +
  labs(title = "Number of Transactions Over Time",
       x = "Year Month",
       y = "Num of Transactions") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



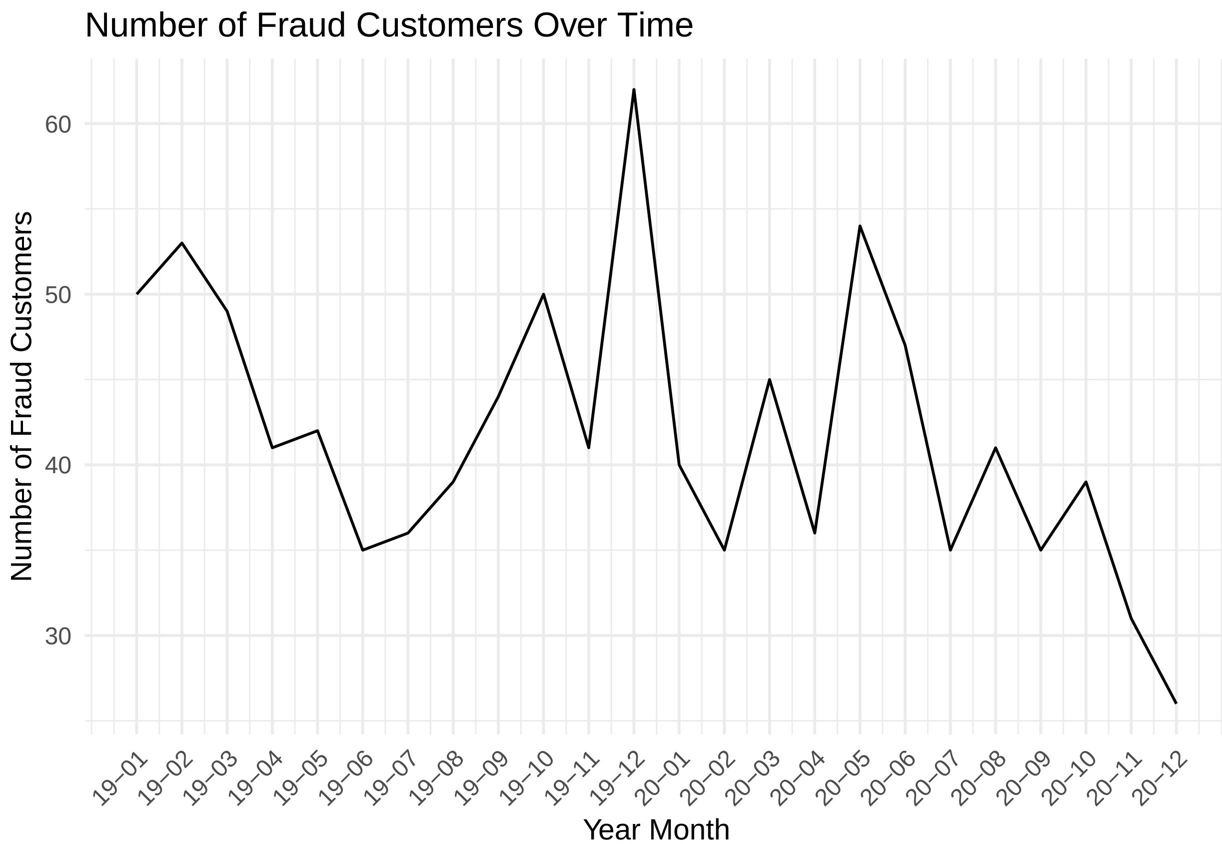
- year_month vs fraud customers and fraud transaction

```
# Filter for fraud transactions
df_fraud_transactions = df %>%
  filter(is_fraud == 1)

# Group by 'trans_year_month' for fraud transactions and calculate number of unique transactions and cu
df_timeline02 = df_fraud_transactions %>%
  group_by(trans_year_month) %>%
  summarise(num_of_fraud_transactions = n_distinct(trans_num),
            fraud_customers = n_distinct(cc_num)) %>%
  ungroup() %>%
  rename(year_month = trans_year_month)
```

- Now plot the above distribution

```
# Create a sequence for x-axis
x = seq(1, nrow(df_timeline02), 1)

# Create the plot using ggplot
ggplot(df_timeline02, aes(x = x, y = fraud_customers)) +
  geom_line() +
  scale_x_continuous(breaks = x, labels = df_timeline02$year_month) +
  labs(title = "Number of Fraud Customers Over Time",
       x = "Year Month",
```

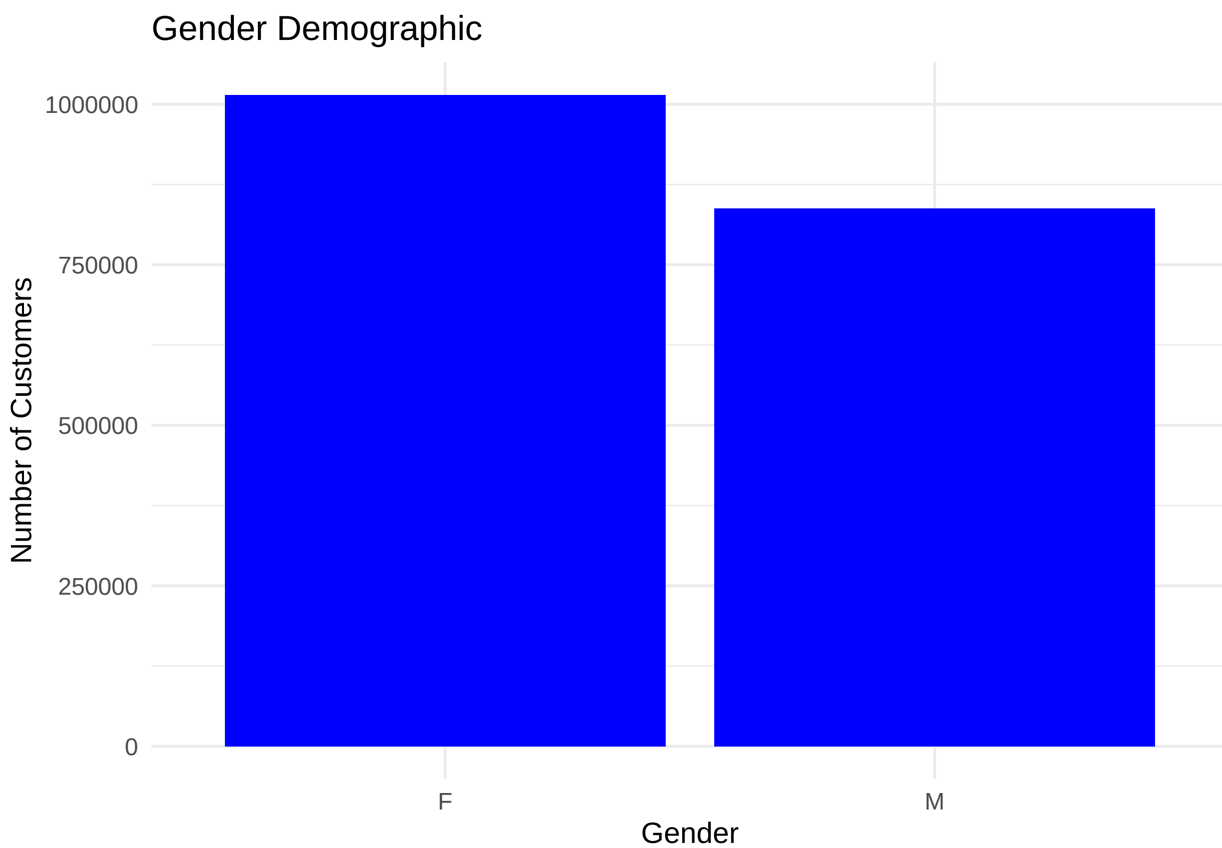
```r
       y = "Number of Fraud Customers") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



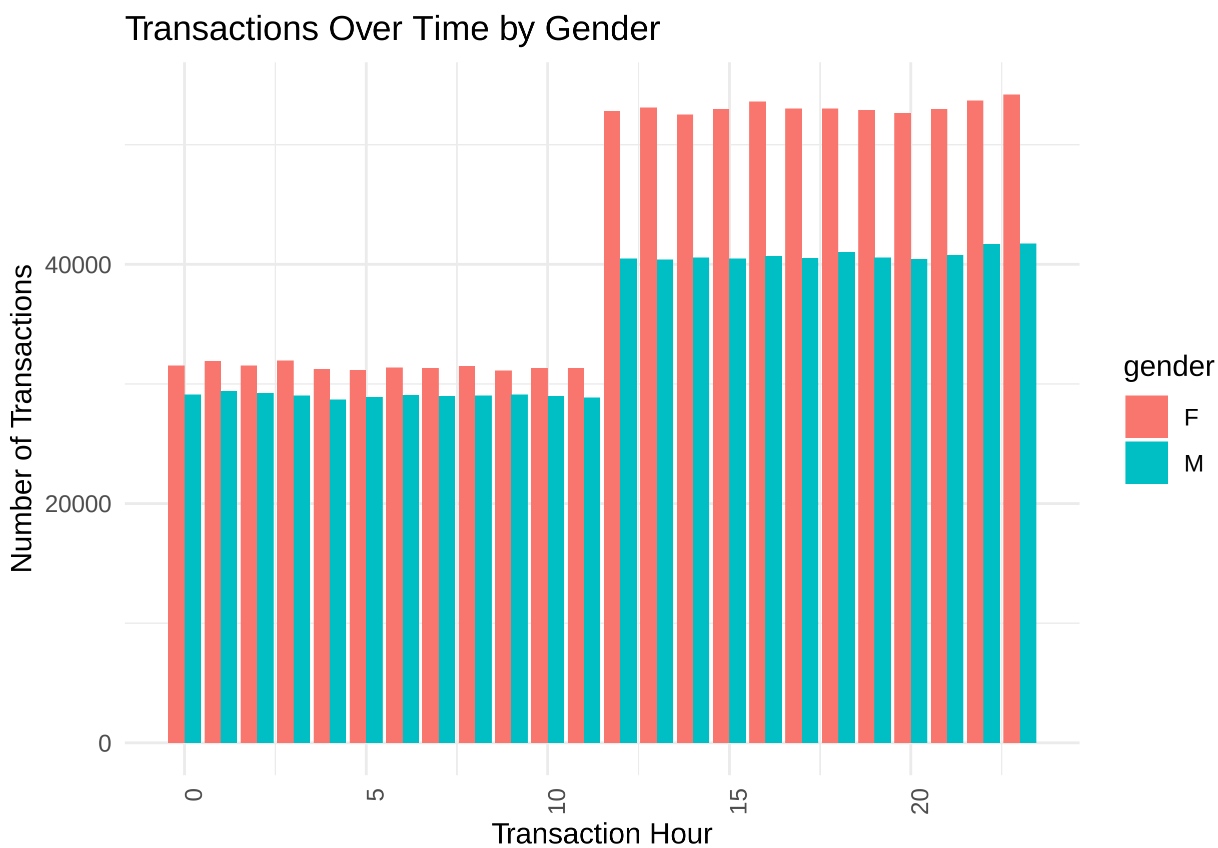
- From the above graphs it can be seen that the most of the transaction happen after the noon. So, security can be increased at that time.
- Also the overall transaction and the fraud transaction is increased during the 12 month, i.e, in the December. so such times can be watched closely.
- Also in the holidays people mostly uses their cards. So Surveilance can be increased on those days.

Exploring Gender data

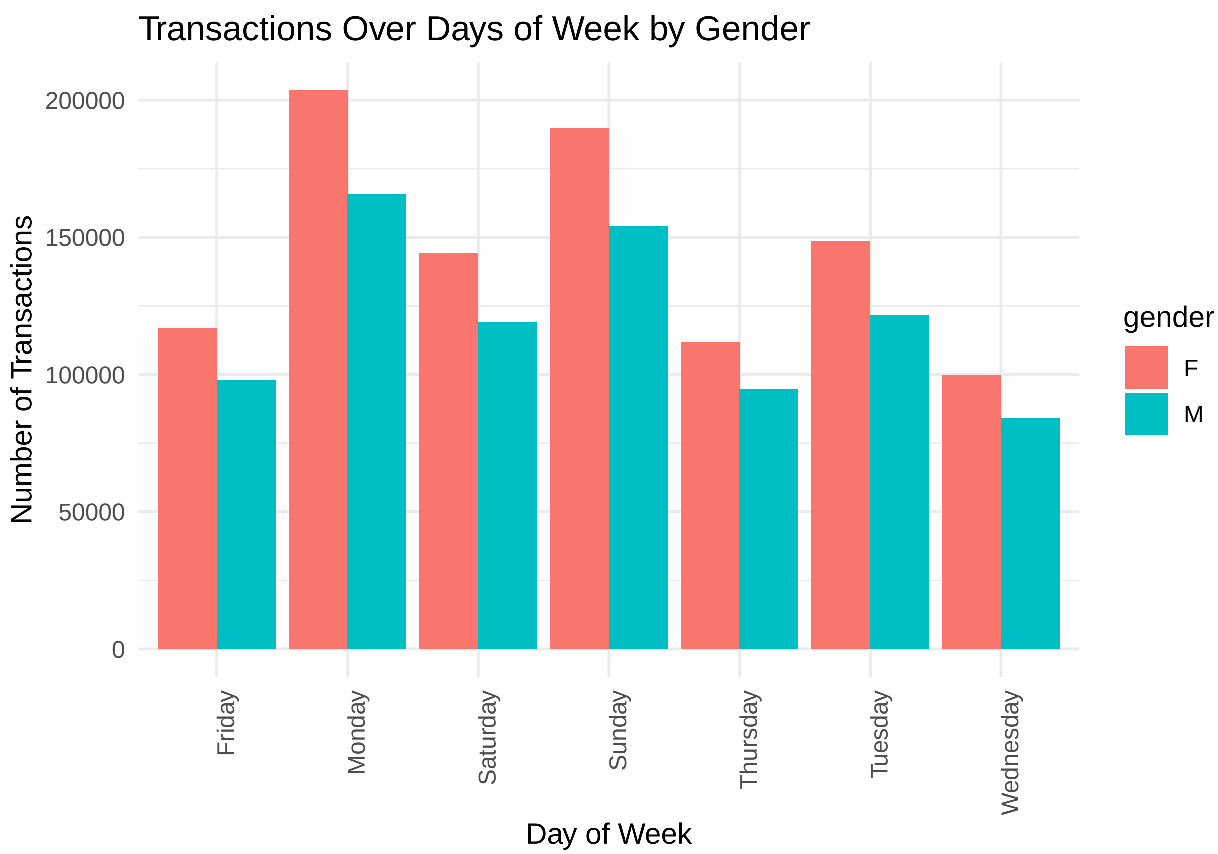
```r
# Plotting gender demographic
ggplot(df, aes(x = gender)) +
  geom_bar(fill = "blue") +
  labs(title = "Gender Demographic",
       x = "Gender",
       y = "Number of Customers") +
  theme_minimal()
```



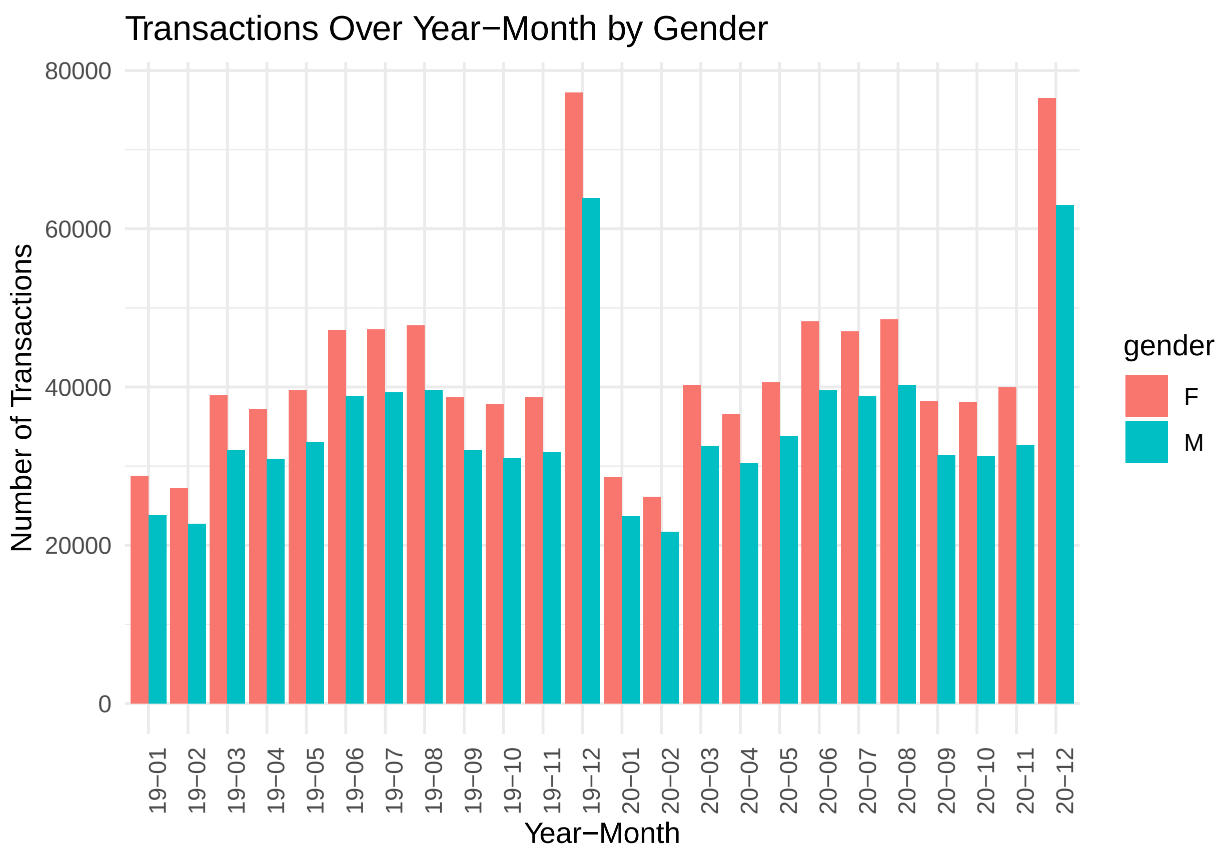

```
# Plotting transactions over time with respect to gender
ggplot(df, aes(x = trans_hour, fill = gender)) +
  geom_bar(position = "dodge") +
  labs(title = "Transactions Over Time by Gender",
       x = "Transaction Hour",
       y = "Number of Transactions") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
ggplot(df, aes(x = trans_day_of_week, fill = gender)) +
  geom_bar(position = "dodge") +
  labs(title = "Transactions Over Days of Week by Gender",
       x = "Day of Week",
       y = "Number of Transactions") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
ggplot(df, aes(x = trans_year_month, fill = gender)) +
  geom_bar(position = "dodge") +
  labs(title = "Transactions Over Year-Month by Gender",
       x = "Year-Month",
       y = "Number of Transactions") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Create the 'gender' distributed data frame
df_gender = df %>%
  group_by(gender) %>%
  summarise(gender_count = n()) %>%
  ungroup() %>%
  rename(Gender = gender)

# Create the gender-fraud distribution data frame
df_fraud_gender = df %>%
  group_by(gender, is_fraud) %>%
  summarise(Transaction_Count = n()) %>%
  ungroup() %>%
  rename(Gender = gender, Is_Fraud = is_fraud)
```

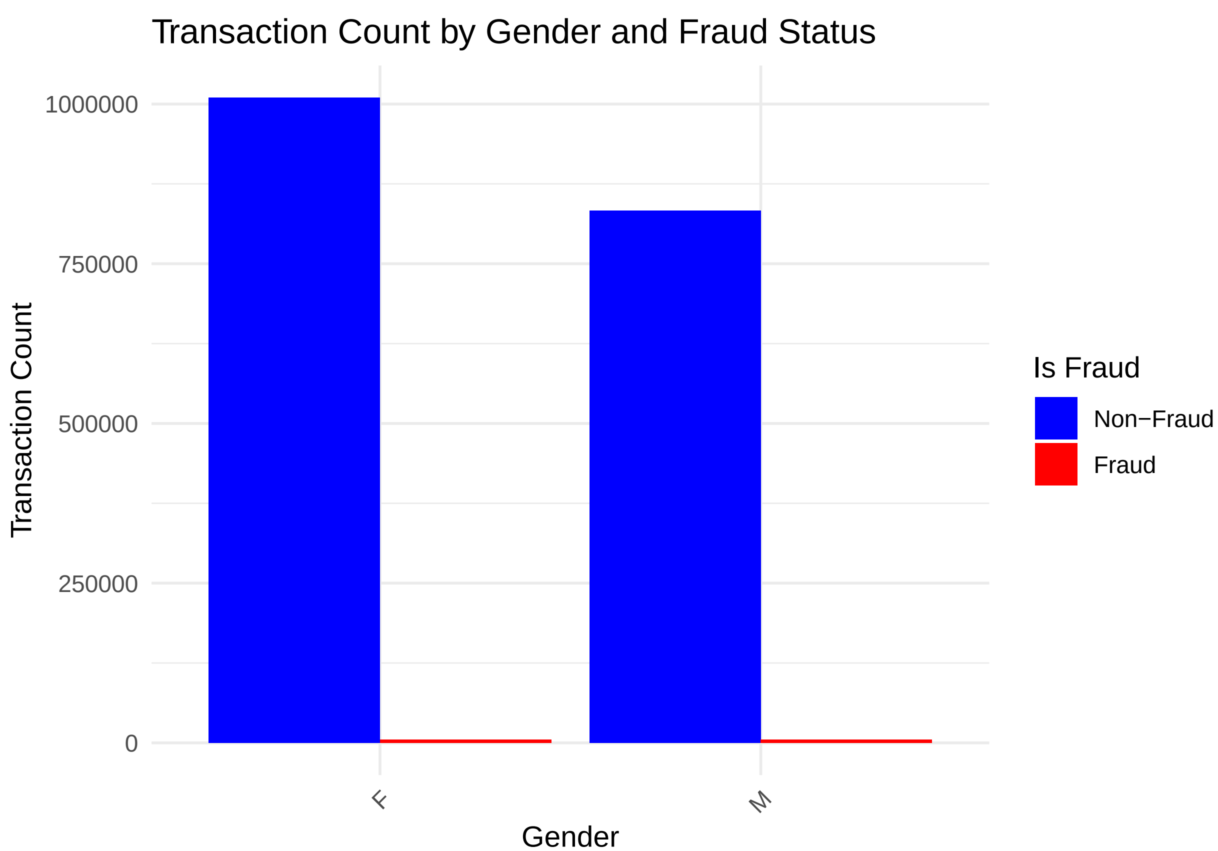
```
## `summarise()` has grouped output by 'gender'. You can override using the
## `.groups` argument.
```

```r
# Merge the data frames
df_fraud_gender = df_fraud_gender %>%
  left_join(df_gender, by = "Gender") %>%
  mutate(Transaction_Percentage = (Transaction_Count / gender_count) * 100)

head(df_fraud_gender)
```

```
## # A tibble: 4 x 5
```

```
##   Gender Is_Fraud Transaction_Count gender_count Transaction_Percentage
##   <chr>     <int>             <int>        <int>                  <dbl>
## 1 F             0           1009850      1014749                 99.5
## 2 F             1              4899      1014749                  0.483
## 3 M             0            832893       837645                 99.4
## 4 M             1              4752       837645                  0.567
```

```r
# Create the bar plot using ggplot
ggplot(df_fraud_gender, aes(x = Gender, y = Transaction_Count, fill = factor(Is_Fraud))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Transaction Count by Gender and Fraud Status",
       x = "Gender",
       y = "Transaction Count",
       fill = "Is Fraud") +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"), labels = c("0" = "Non-Fraud", "1" = "Fraud"))
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



- women are involved in most of the transactions and hence, they be more prone to frauds.
- Therefore, while there is a need for all sexes in the data to be knowledgeable about the frauds and their methods happening due to credit cards, in order to reduce the amount of frauds women should be educated and trained to be a bit more vigilant since they are much more prone to frauds.
- It can be concluded that men are a bit more inclined to be involved in fraud although both the sexes appear to be almost equally involved in all fraudulent transactions

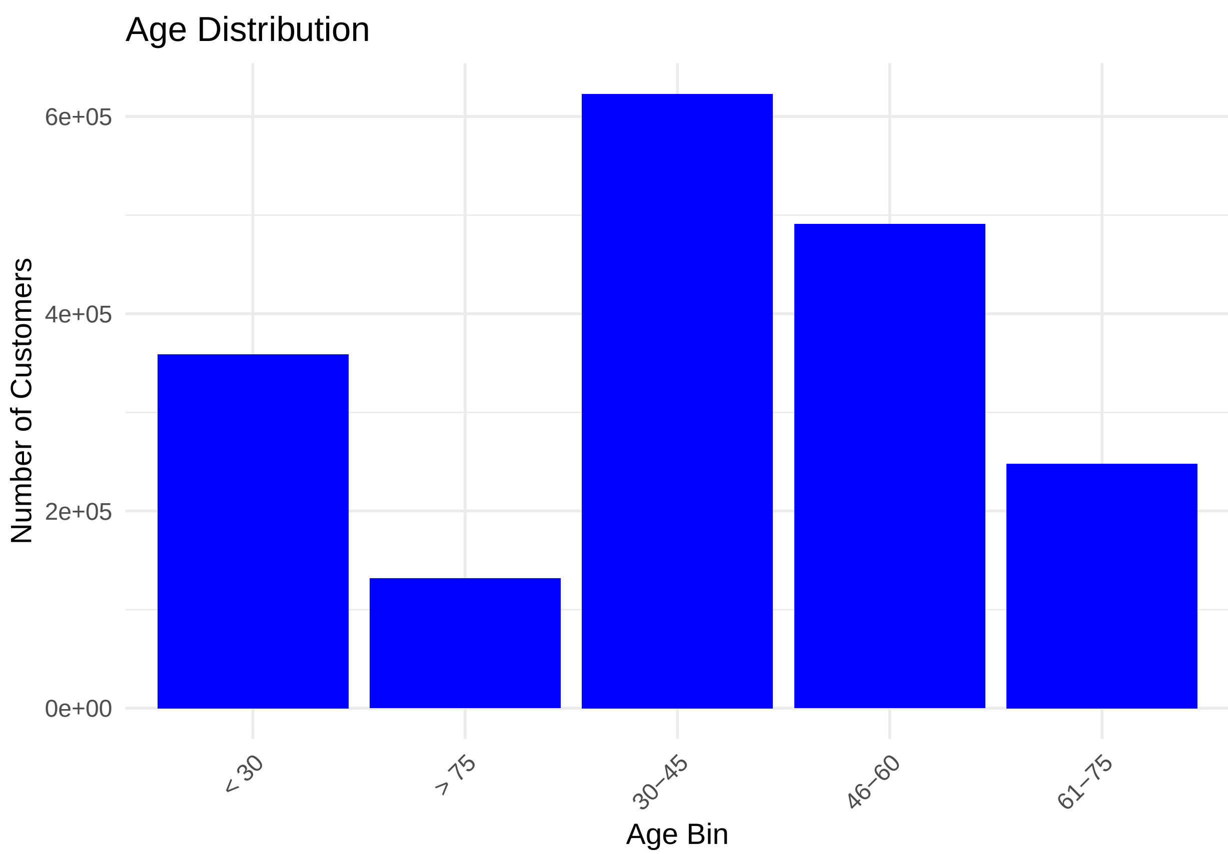
Exploring age data

```
# Create a new column for age bins
df = df %>%
  mutate(age_bin = case_when(
    age <= 30 ~ "< 30",
    age > 30 & age <= 45 ~ "30-45",
    age > 45 & age <= 60 ~ "46-60",
    age > 60 & age <= 75 ~ "61-75",
    TRUE ~ "> 75"
  ))
```

```
head(df$age_bin)
```

```
## [1] "30-45" "30-45" "46-60" "46-60" "30-45" "46-60"
```

```
# Create the count plot using ggplot
ggplot(df, aes(x = age_bin)) +
  geom_bar(fill = "blue") +
  labs(title = "Age Distribution",
       x = "Age Bin",
       y = "Number of Customers") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



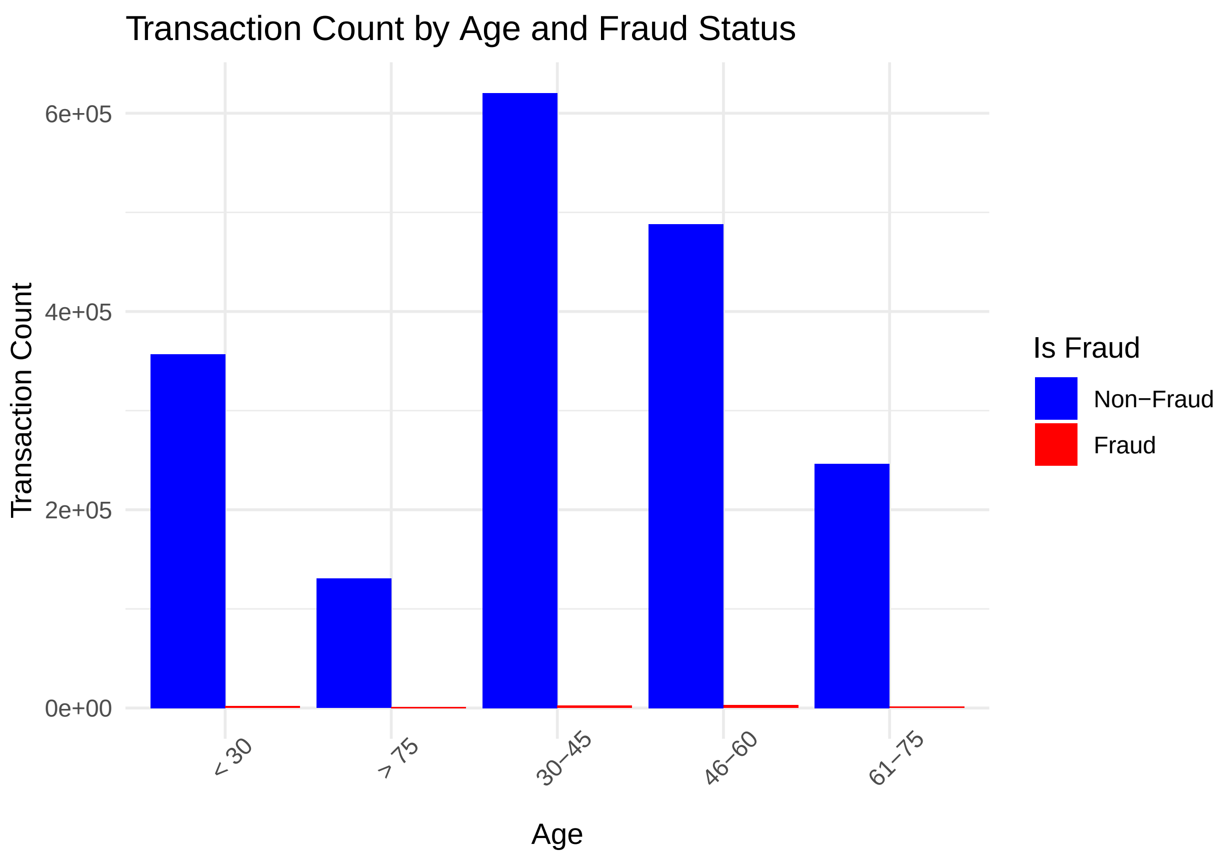
```r
# Create the age-transaction count distribution data frame
df_age = df %>%
  group_by(age_bin) %>%
  summarise(age_count = n()) %>%
  ungroup()

# Create the age-fraud distribution data frame
df_fraud_age = df %>%
  group_by(age_bin, is_fraud) %>%
  summarise(Transaction_Count = n()) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'age_bin'. You can override using the
## `.groups` argument.
```

```r
# Merge the data frames
df_fraud_age = df_fraud_age %>%
  left_join(df_age, by = "age_bin") %>%
  mutate(Transaction_Percentage = (Transaction_Count / age_count) * 100)

head(df_fraud_age)
```

```
## # A tibble: 6 x 5
##   age_bin is_fraud Transaction_Count age_count Transaction_Percentage
##   <chr>      <int>             <int>     <int>                  <dbl>
## 1 30-45          0            620404    622888                 99.6
## 2 30-45          1              2484    622888                  0.399
## 3 46-60          0            488201    490980                 99.4
## 4 46-60          1              2779    490980                  0.566
## 5 61-75          0            246418    247923                 99.4
## 6 61-75          1              1505    247923                  0.607
```

```r
ggplot(df_fraud_age, aes(x = age_bin, y = Transaction_Count, fill = factor(is_fraud))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Transaction Count by Age and Fraud Status",
       x = "Age",
       y = "Transaction Count",
       fill = "Is Fraud") +
  scale_fill_manual(values = c("0" = "blue", "1" = "red"), labels = c("0" = "Non-Fraud", "1" = "Fraud"))
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45))
```



Explore state data

```r
length(unique(df$state))
```

```
## [1] 51
```

```r
names(head(sort(table(df$state), decreasing = TRUE), 20))
```

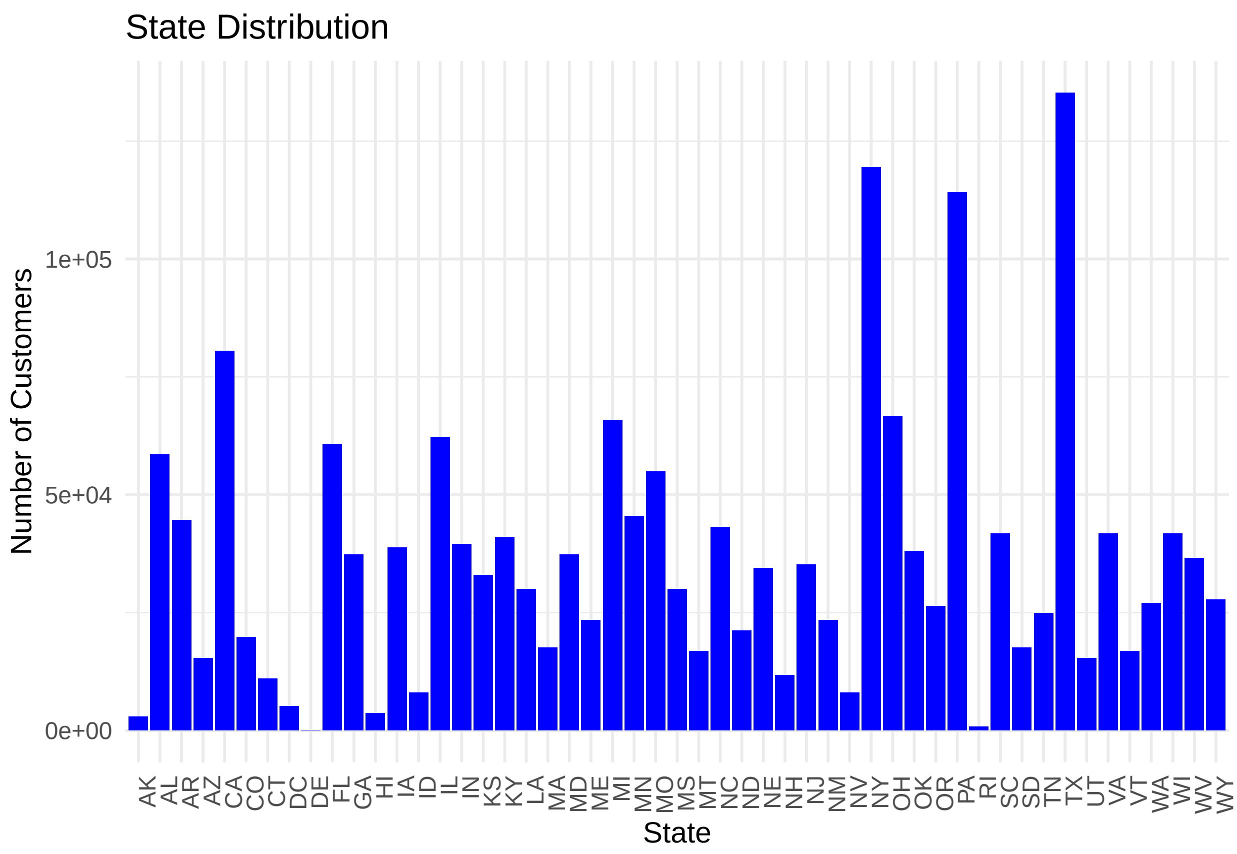
```
##  [1] "TX" "NY" "PA" "CA" "OH" "MI" "IL" "FL" "AL" "MO" "MN" "AR" "NC" "VA" "WI"
## [16] "SC" "KY" "IN" "IA" "OK"
```

```r
# Fetch the top 20 states with the highest transaction frequency
high_trans_states = names(head(sort(table(df$state), decreasing = TRUE), 20))

# Calculate the percentage distribution
percentage_distribution = prop.table(table(df$state[df$state %in% high_trans_states])) * 100

# Print the percentage distribution
print(percentage_distribution)
```

```
##
##        AL        AR        CA        FL        IA        IL        IN        KY
```

```
## 4.742394  3.615163  6.523111  4.925052  3.144578  5.041503  3.204141  3.320997
##        MI        MN        MO        NC        NY        OH        OK        PA
## 5.334292  3.681776  4.449281  3.495470  9.677414  5.399284  3.083476  9.252292
##        SC        TX        VA        WI
## 3.381775 10.961858  3.383801  3.382342
```

```r
# Create the count plot using ggplot
ggplot(df, aes(x = state)) +
  geom_bar(fill = "blue") +
  labs(title = "State Distribution",
       x = "State",
       y = "Number of Customers") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



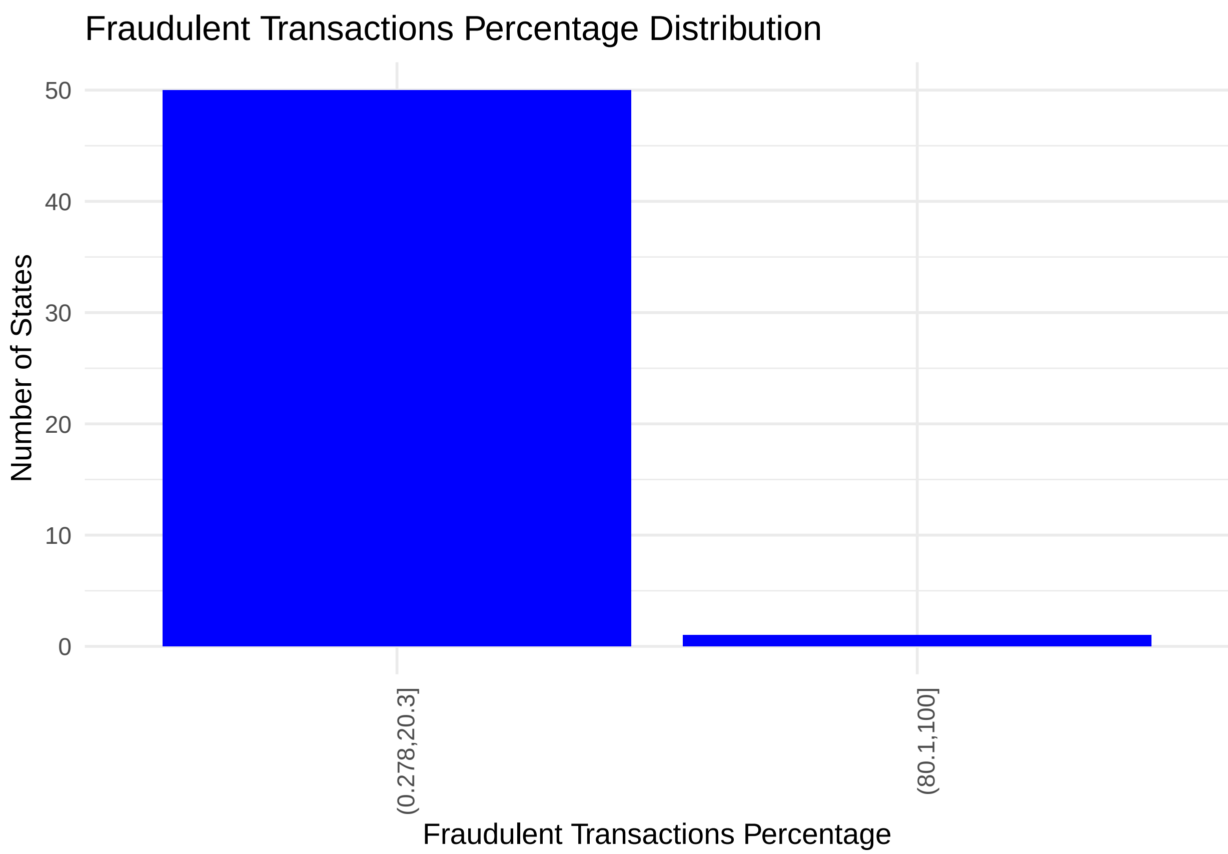

```r
# Create the state-transaction count distribution data frame
df_state = df %>%
  group_by(state) %>%
  summarise(state_count = n()) %>%
  ungroup()

# Create the state-fraud distribution data frame
df_fraud_state = df %>%
  group_by(state, is_fraud) %>%
  summarise(Transaction_Count = n()) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'state'. You can override using the
## `.groups` argument.
```

```r
# Merge the data frames
df_fraud_state = df_fraud_state %>%
  left_join(df_state, by = "state") %>%
  mutate(Transaction_Percentage = (Transaction_Count / state_count) * 100)

# View the top 20 states with high fraudulent transactions
top_fraud_states = df_fraud_state %>%
  filter(is_fraud == 1) %>%
  arrange(desc(Transaction_Percentage)) %>%
  head(20)

# Print the resulting data frame
head(top_fraud_states)
```

```
## # A tibble: 6 x 5
##   state is_fraud Transaction_Count state_count Transaction_Percentage
##   <chr>    <int>             <int>       <int>                  <dbl>
## 1 DE           1                 9           9                100
## 2 RI           1                15         745                  2.01
## 3 AK           1                50        2963                  1.69
## 4 OR           1               197       26408                  0.746
## 5 NH           1                79       11727                  0.674
## 6 VA           1               273       41756                  0.654
```

```r
# Filter the data for fraudulent transactions
fraudulent_data = df_fraud_state %>%
  filter(is_fraud == 1)

# Create the count plot using ggplot
ggplot(fraudulent_data, aes(x = cut(Transaction_Percentage, breaks = 5))) +
  geom_bar(fill = "blue") +
  labs(title = "Fraudulent Transactions Percentage Distribution",
       x = "Fraudulent Transactions Percentage",
       y = "Number of States") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Filter and print states with more than 75% fraudulent transactions
fraudulent_states = df_fraud_state %>%
  filter(is_fraud == 1, Transaction_Percentage >= 75) %>%
  select(state)

# Print the list of states
cat("States with more than 75% fraudulent transactions:\n")
```

```
## States with more than 75% fraudulent transactions:
```

```r
print(fraudulent_states$state)
```

```
## [1] "DE"
```

- In view of the above observations, it can be concluded that in order to reduce the number of fraudulent transactions overall, it is necessary that the monitoring of transactions in areas where in the most number of transaction must be increased.

Exploring city and zip

```r
# Print the number of unique cities and zip codes
cat("Number of cities:", length(unique(df$city)), "\n")
```

```
## Number of cities: 906
```

```r
cat("Number of zip codes:", length(unique(df$zip)), "\n")
```

```
## Number of zip codes: 985
```

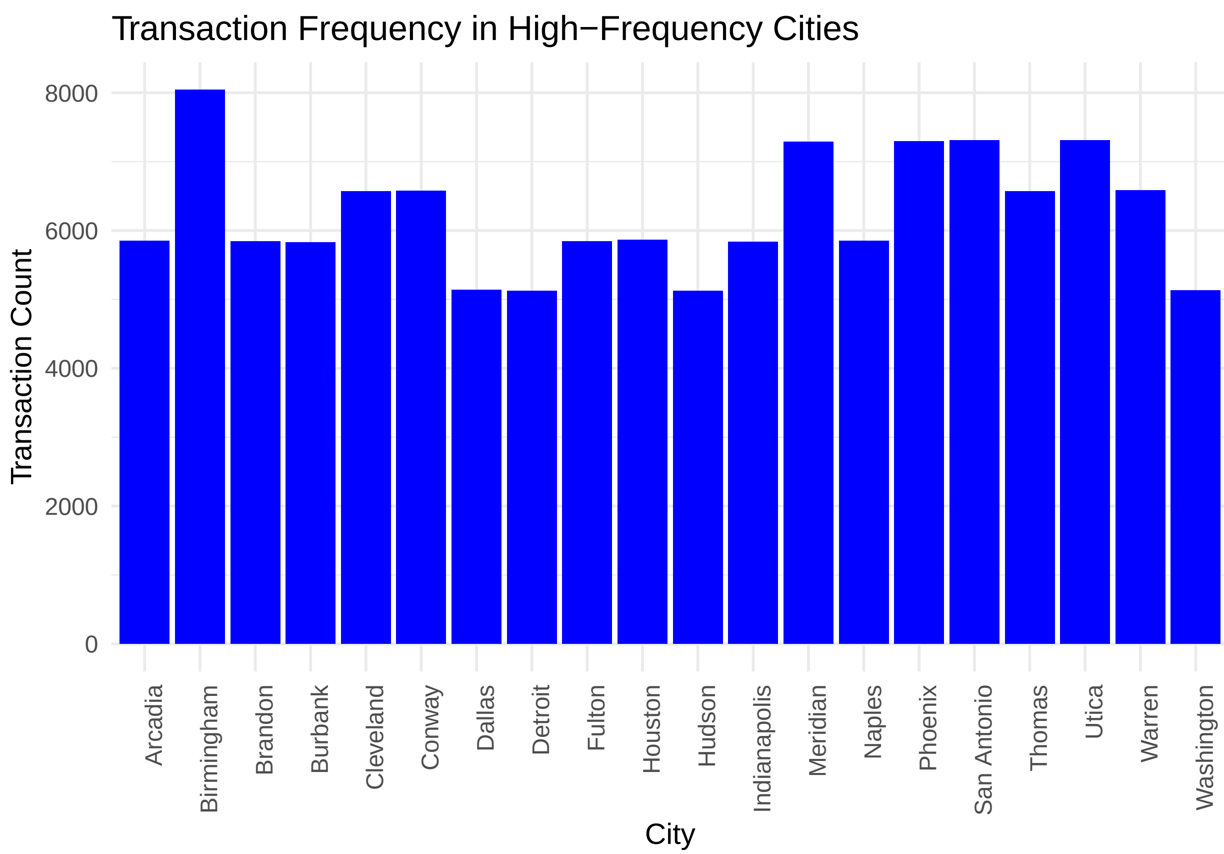
```r
# Fetch the top 20 high-frequency cities and zip codes
high_trans_cities = names(head(sort(table(df$city), decreasing = TRUE), 20))
high_trans_zips = names(head(sort(table(df$zip), decreasing = TRUE), 20))

# Print the high-frequency cities and zip codes
cat("High-frequency cities:", paste(high_trans_cities, collapse = ", "), "\n")
```

```
## High-frequency cities: Birmingham, San Antonio, Utica, Phoenix, Meridian, Warren, Conway, Cleveland,
```

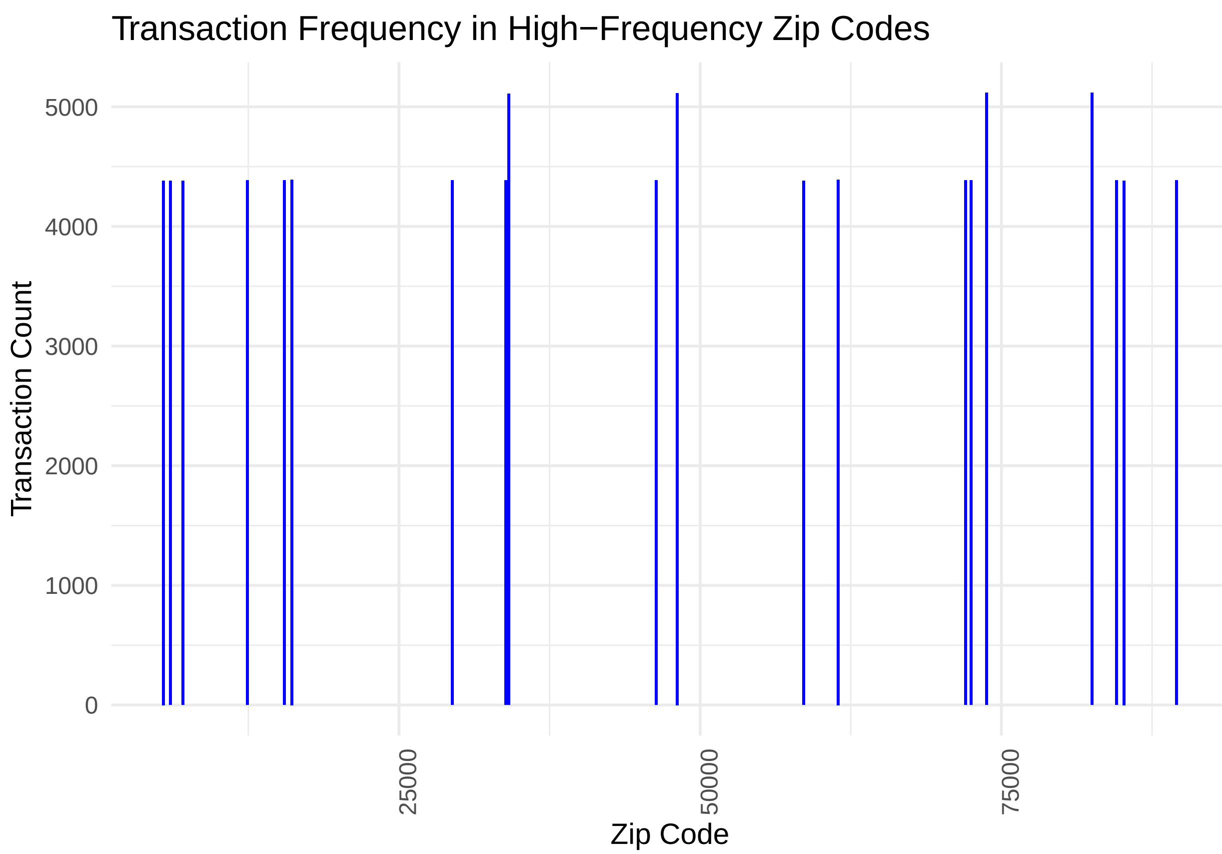
```r
cat("High-frequency zip codes:", paste(high_trans_zips, collapse = ", "), "\n")
```

```
## High-frequency zip codes: 73754, 82514, 48088, 34112, 16114, 61454, 72476, 84540, 89512, 33872, 7204
```

```r
# Filter the data for high-frequency cities
high_freq_cities_data = df %>%
  filter(city %in% high_trans_cities)



# Create the plots using ggplot
ggplot(high_freq_cities_data, aes(x = city)) +
  geom_bar(fill = "blue") +
  labs(title = "Transaction Frequency in High-Frequency Cities",
       x = "City",
       y = "Transaction Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Filter the data for high-frequency  zips

high_freq_zips_data = df %>%
  filter(zip %in% high_trans_zips)

# Create the plots using ggplot
ggplot(high_freq_zips_data, aes(x = zip)) +
  geom_bar(fill = "blue") +
  labs(title = "Transaction Frequency in High-Frequency Zip Codes",
       x = "Zip Code",
       y = "Transaction Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Create the city-transaction count distribution data frame
df_city = df %>%
  group_by(city) %>%
  summarise(city_count = n()) %>%
  ungroup()

# Create the city-fraud distribution data frame
df_fraud_city = df %>%
  group_by(city, is_fraud) %>%
  summarise(Transaction_Count = n()) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'city'. You can override using the
## `.groups` argument.
```

```r
# Merge the data frames
df_fraud_city = df_fraud_city %>%
  left_join(df_city, by = "city") %>%
  mutate(Transaction_Percentage = (Transaction_Count / city_count) * 100)

# View the top 20 cities with high fraudulent transaction volumes
top_fraud_cities = df_fraud_city %>%
  filter(is_fraud == 1) %>%
  arrange(desc(Transaction_Percentage)) %>%
  head(20)
```

```r
# Print the resulting data frame
print(top_fraud_cities)
```

```
## # A tibble: 20 x 5
##    city        is_fraud Transaction_Count city_count Transaction_Percentage
##    <chr>          <int>             <int>      <int>                  <dbl>
##  1 Angwin             1                10         10                    100
##  2 Ashland            1                10         10                    100
##  3 Beacon             1                11         11                    100
##  4 Brookfield         1                 9          9                    100
##  5 Bruce              1                 7          7                    100
##  6 Buellton           1                 8          8                    100
##  7 Byesville          1                12         12                    100
##  8 Chattanooga        1                 7          7                    100
##  9 Clarion            1                 9          9                    100
## 10 Claypool           1                 7          7                    100
## 11 Clinton            1                12         12                    100
## 12 Coulee Dam         1                15         15                    100
## 13 Craig              1                14         14                    100
## 14 Crouse             1                 8          8                    100
## 15 Downey             1                10         10                    100
## 16 East China         1                 9          9                    100
## 17 Freeport           1                 9          9                    100
## 18 Gaines             1                 8          8                    100
## 19 Granbury           1                12         12                    100
## 20 Greenport          1                10         10                    100
```

```r
# Create the zip-transaction count distribution data frame
df_zip = df %>%
  group_by(zip) %>%
  summarise(zip_count = n()) %>%
  ungroup()

# Create the zip-fraud distribution data frame
df_fraud_zip = df %>%
  group_by(zip, is_fraud) %>%
  summarise(Transaction_Count = n()) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'zip'. You can override using the `.groups`
## argument.
```

```r
# Merge the data frames
df_fraud_zip = df_fraud_zip %>%
  left_join(df_zip, by = "zip") %>%
  mutate(Transaction_Percentage = (Transaction_Count / zip_count) * 100)

# View the top 20 zip codes with high fraudulent transaction volumes
top_fraud_zips = df_fraud_zip %>%
  filter(is_fraud == 1) %>%
  arrange(desc(Transaction_Percentage)) %>%
  head(20)
```

```r
# Print the resulting data frame
print(top_fraud_zips)
```

```
## # A tibble: 20 x 5
##      zip is_fraud Transaction_Count zip_count Transaction_Percentage
##    <int>    <int>             <int>     <int>                  <dbl>
##  1  4032        1                 9         9                    100
##  2 10018        1                 7         7                    100
##  3 10533        1                 8         8                    100
##  4 10553        1                11        11                    100
##  5 10954        1                10        10                    100
##  6 11747        1                15        15                    100
##  7 11763        1                 9         9                    100
##  8 11944        1                10        10                    100
##  9 12207        1                11        11                    100
## 10 12508        1                11        11                    100
## 11 13795        1                12        12                    100
## 12 14141        1                12        12                    100
## 13 14532        1                11        11                    100
## 14 16041        1                 7         7                    100
## 15 16214        1                 9         9                    100
## 16 16428        1                 9         9                    100
## 17 18446        1                 9         9                    100
## 18 19947        1                 9         9                    100
## 19 21657        1                13        13                    100
## 20 22124        1                 9         9                    100
```

Exploring job feature

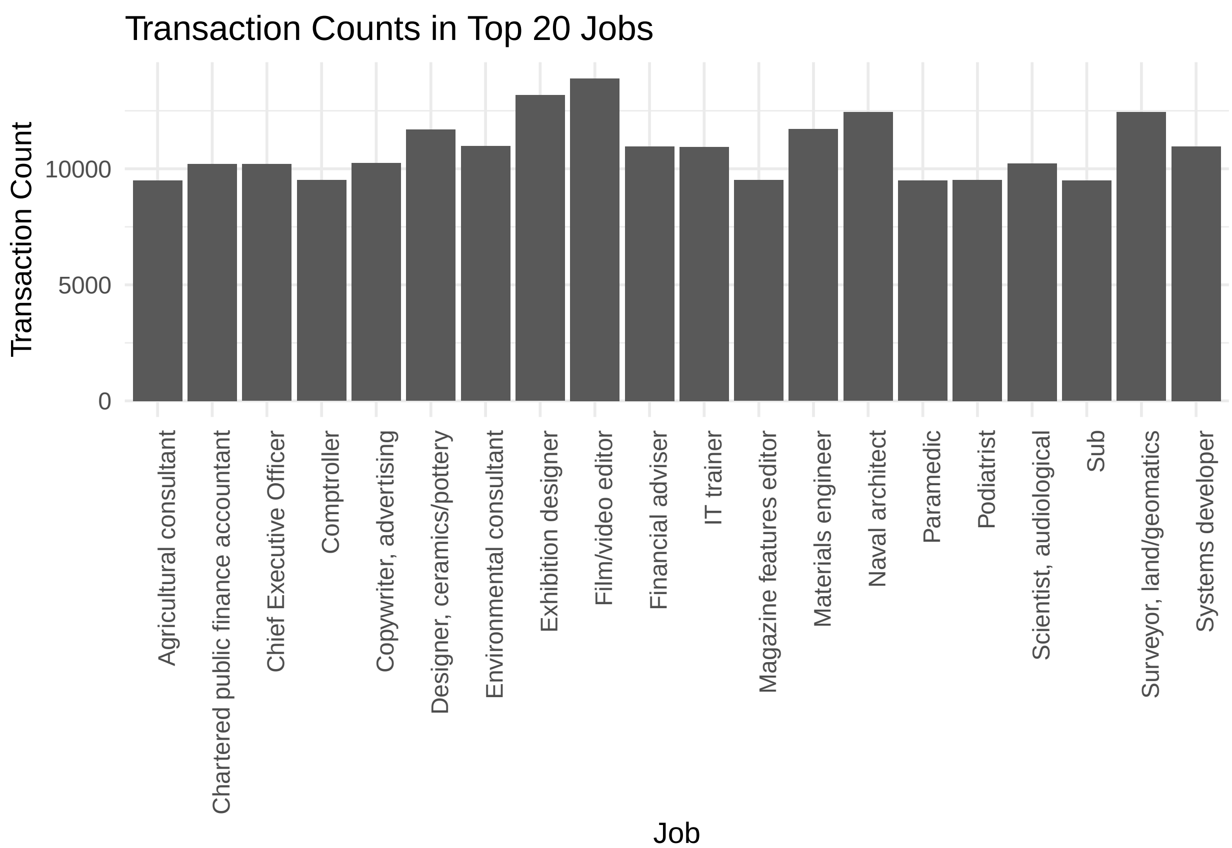
```r
cat("Number of unique job values:",length(unique(df$job)),"\n")
```

```
## Number of unique job values: 497
```

```r
high_trans_jobs <- names(head(sort(table(df$job), decreasing = TRUE), 20))
cat("Top 20 jobs with high transaction frequencies:", names(head(sort(table(df$job), decreasing = TRUE)
```

```
## Top 20 jobs with high transaction frequencies: Film/video editor Exhibition designer Surveyor, land/g
```

```r
# Create the plot using ggplot
ggplot(subset(df, job %in% high_trans_jobs), aes(x = job)) +
  geom_bar() +
  labs(title = "Transaction Counts in Top 20 Jobs",
       x = "Job",
       y = "Transaction Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

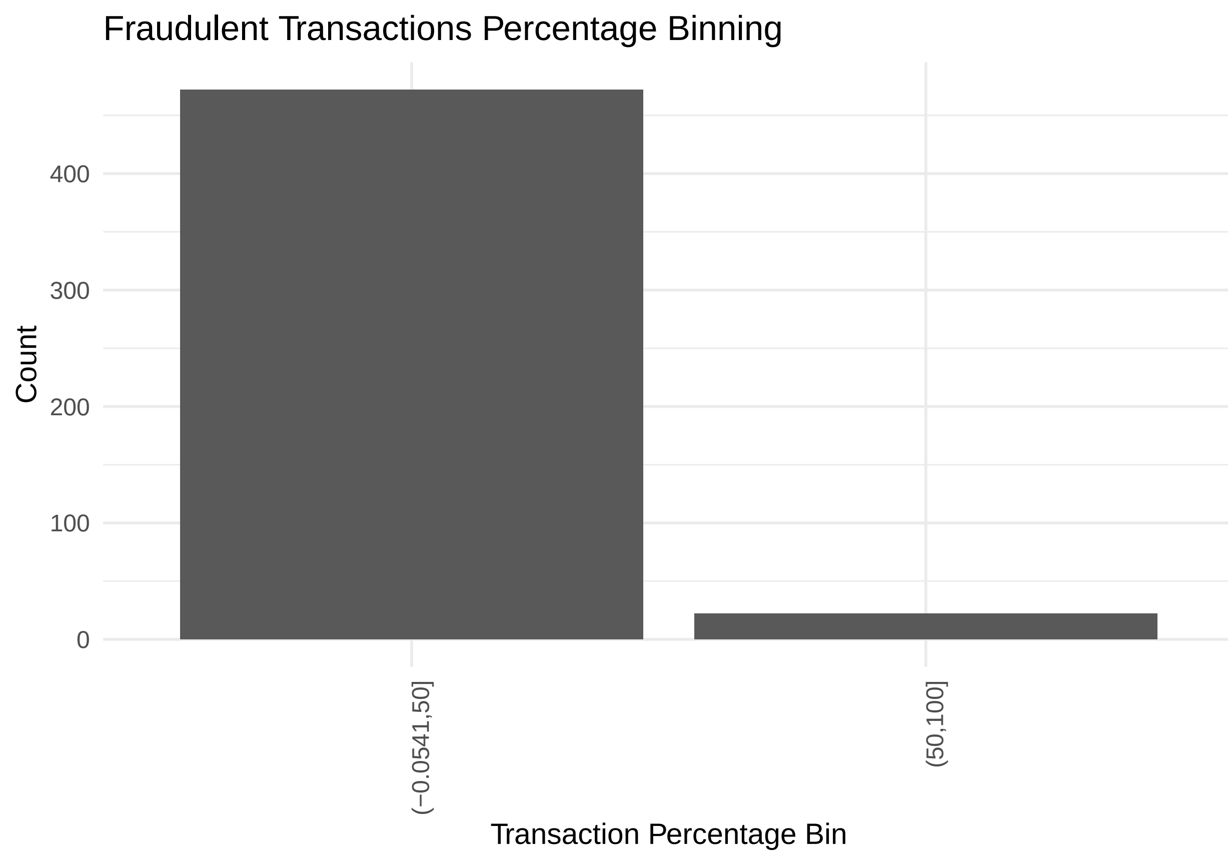



```
# Constructing the job-transaction count distribution
df_job = aggregate(trans_num ~ job, data = df, FUN = length)
names(df_job) <- c('job', 'job_count')

# Creating the job-fraud distribution
df_fraud_job = aggregate(trans_num ~ job + is_fraud, data = df, FUN = length)
names(df_fraud_job) = c('job', 'is_fraud', 'Transaction_count')

# Merging with job counts
df_fraud_job = merge(df_fraud_job, df_job, by = 'job')

# Calculating Transaction percentage
df_fraud_job$Transaction_percentage <- (df_fraud_job$Transaction_count / df_fraud_job$job_count) * 100

# Viewing the top 20 jobs with high fraudulent transaction volumes
top_fraud_jobs = subset(df_fraud_job, is_fraud == 1)
top_fraud_jobs = top_fraud_jobs[order(-top_fraud_jobs$Transaction_percentage), ]
head(top_fraud_jobs, 20)
```

```
##                                  job is_fraud Transaction_count job_count
## 3                Accountant, chartered        1                11        11
## 40              Air traffic controller        1                17        17
## 69       Armed forces technical officer        1                 8         8
## 100               Broadcast journalist        1                 9         9
## 119                    Careers adviser        1                15        15
```

```
## 208        Contracting civil engineer        1                 7          7
## 229                            Dancer        1                19         19
## 336                    Engineer, site        1                12         12
## 341                   Engineer, water        1                 8          8
## 394           Forest/woodland manager        1                 9          9
## 445                         Homeopath        1                11         11
## 472                  Industrial buyer        1                10         10
## 475               Information officer        1                 8          8
## 522                   Legal secretary        1                12         12
## 625     Operational investment banker        1                11         11
## 652                 Personnel officer        1                12         12
## 797 Sales promotion account executive        1                14         14
## 828                       Ship broker        1                 7          7
## 835                 Software engineer        1                11         11
## 838                         Solicitor        1                11         11
##     Transaction_percentage
## 3                      100
## 40                     100
## 69                     100
## 100                    100
## 119                    100
## 208                    100
## 229                    100
## 336                    100
## 341                    100
## 394                    100
## 445                    100
## 472                    100
## 475                    100
## 522                    100
## 625                    100
## 652                    100
## 797                    100
## 828                    100
## 835                    100
## 838                    100
```

```r
# Filter the data for only fraudulent transactions
df_fraud_job = subset(df_fraud_job, is_fraud == 1)

# Create the plot using ggplot
ggplot(df_fraud_job, aes(x = cut(`Transaction_percentage`, breaks = 2), fill = `Transaction_percentage`
  geom_bar(stat = "count") +
  labs(title = "Fraudulent Transactions Percentage Binning",
       x = "Transaction Percentage Bin",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Filter and print jobs with more than 50% fraudulent transactions
fraudulent_jobs = df_fraud_job %>%
  filter(is_fraud == 1, `Transaction_percentage` >= 50) %>%
  select(job)

# Print the list of jobs
cat("Jobs with more than 50% fraudulent transactions:\n")
```

```
## Jobs with more than 50% fraudulent transactions:
```

```r
print(fraudulent_jobs$job)
```

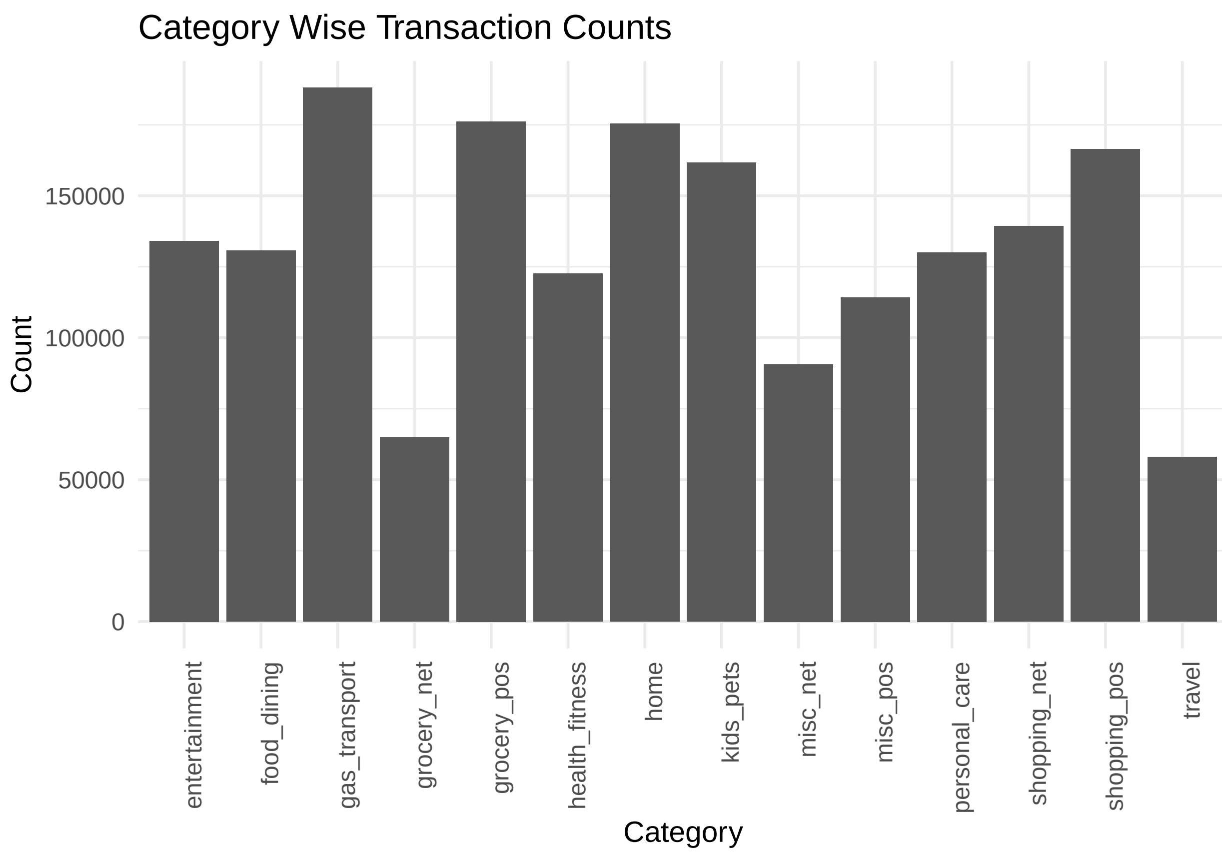
```
##  [1] "Accountant, chartered"             "Air traffic controller"
##  [3] "Armed forces technical officer"    "Broadcast journalist"
##  [5] "Careers adviser"                   "Contracting civil engineer"
##  [7] "Dancer"                            "Engineer, site"
##  [9] "Engineer, water"                   "Forest/woodland manager"
## [11] "Homeopath"                         "Industrial buyer"
## [13] "Information officer"               "Legal secretary"
## [15] "Operational investment banker"     "Personnel officer"
## [17] "Sales promotion account executive" "Ship broker"
## [19] "Software engineer"                 "Solicitor"
## [21] "Veterinary surgeon"                "Warehouse manager"
```

Exploring Category feature

```r
prop.table(table(df$category))
```

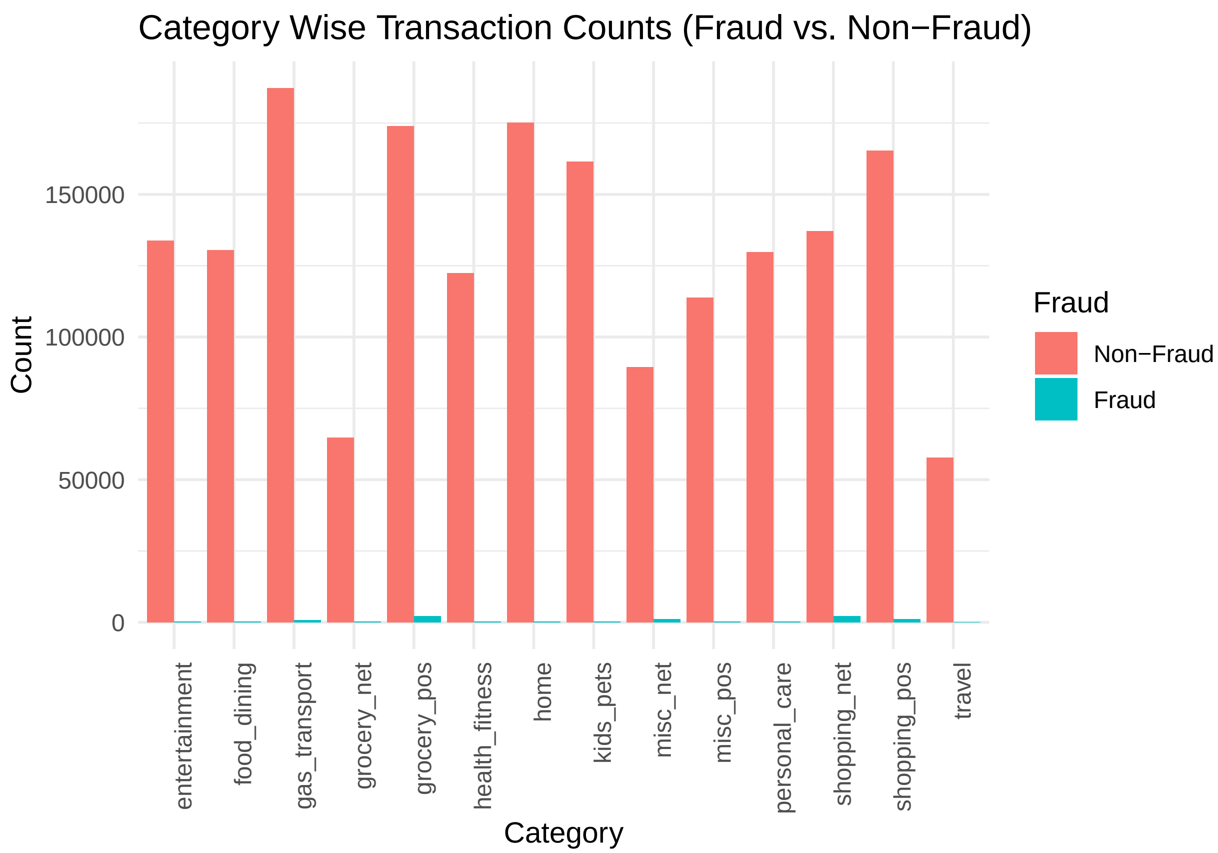
```
##
##   entertainment     food_dining   gas_transport     grocery_net     grocery_pos
##      0.07240252      0.07057300      0.10150594      0.03502387      0.09511529
## health_fitness            home       kids_pets        misc_net        misc_pos
##      0.06615925      0.09472067      0.08730702      0.04893883      0.06166561
##   personal_care    shopping_net    shopping_pos          travel
##      0.07022534      0.07521186      0.08986371      0.03128708
```

```r
# Create the plot using ggplot
ggplot(df, aes(x = category)) +
  geom_bar() +
  labs(title = "Category Wise Transaction Counts",
       x = "Category",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```r
# Create the plot using ggplot
ggplot(df, aes(x = category, fill = factor(is_fraud))) +
```

```
  geom_bar(position = "dodge") +
  labs(title = "Category Wise Transaction Counts (Fraud vs. Non-Fraud)",
       x = "Category",
       y = "Count") +
  scale_fill_discrete(name = "Fraud",
                      labels = c("Non-Fraud", "Fraud")) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



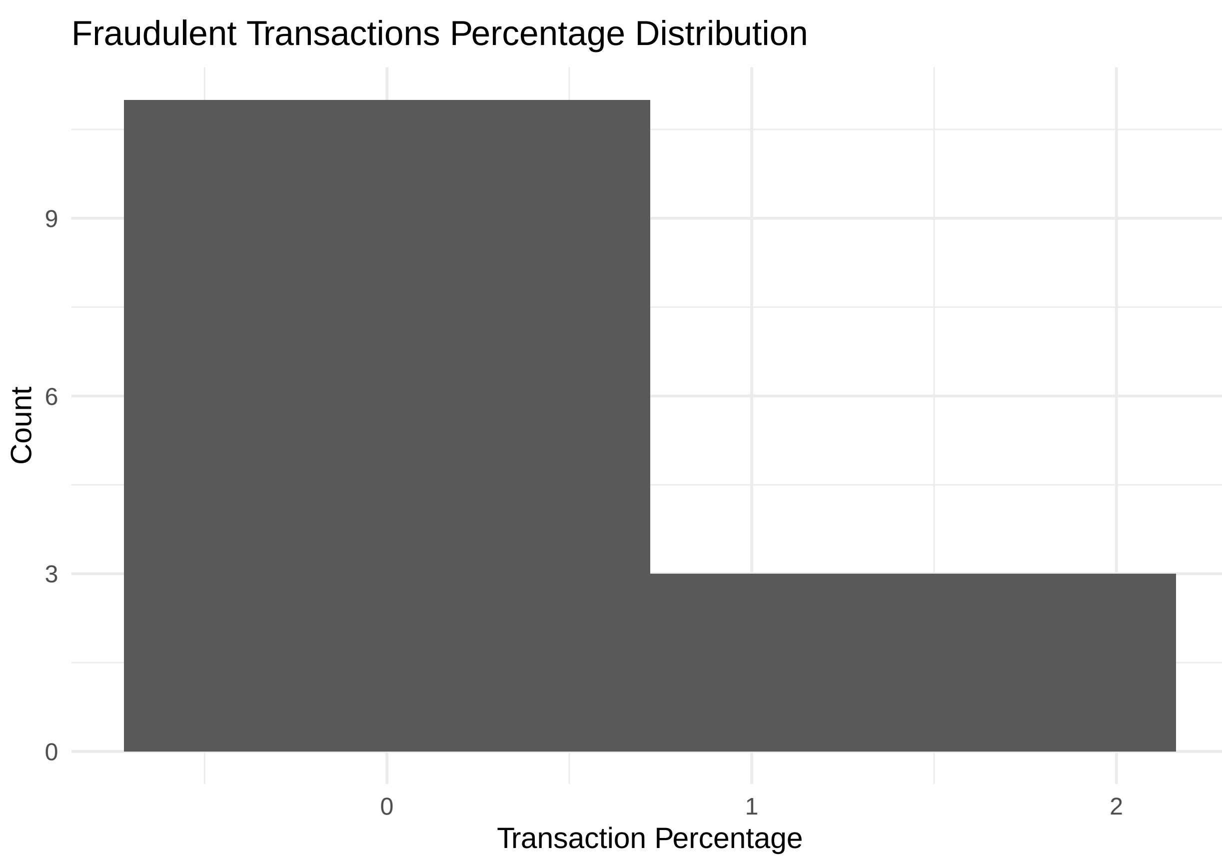

```
# Constructing the category-transaction count distribution
df_category = df %>%
  group_by(category) %>%
  summarize(category_count = n()) %>%
  ungroup()

# Creating the category-fraud distribution
df_fraud_category = df %>%
  group_by(category, is_fraud) %>%
  summarize(`Transaction count` = n()) %>%
  ungroup() %>%
  left_join(df_category, by = "category") %>%
  mutate(`Transaction percentage` = (`Transaction count` / category_count) * 100)
```

```
## `summarise()` has grouped output by 'category'. You can override using the
## `.groups` argument.
```

```r
# Viewing the top categories with high fraudulent transaction volumes
df_fraud_category %>%
  filter(is_fraud == 1) %>%
  arrange(desc(`Transaction percentage`)) %>%
  head()
```

```
## # A tibble: 6 x 5
##   category      is_fraud `Transaction count` category_count Transaction percen~1
##   <chr>            <int>               <int>          <int>                <dbl>
## 1 shopping_net         1                2219         139322                1.59
## 2 misc_net             1                1182          90654                1.30
## 3 grocery_pos          1                2228         176191                1.26
## 4 shopping_pos         1                1056         166463                0.634
## 5 gas_transport        1                 772         188029                0.411
## 6 misc_pos             1                 322         114229                0.282
## # ... with abbreviated variable name 1: `Transaction percentage`
```

```r
# Create the plot using ggplot
ggplot(df_fraud_category[df_fraud_category$is_fraud == 1, ], aes(x = `Transaction percentage`)) +
  geom_histogram(bins = 2) +
  labs(title = "Fraudulent Transactions Percentage Distribution",
       x = "Transaction Percentage",
       y = "Count") +
  theme_minimal()
```



```r
# Filter and print categories with more than one percent fraudulent transactions
fraudulent_categories = df_fraud_category %>%
  filter(is_fraud == 1, `Transaction percentage` >= 1) %>%
  select(category)

# Print the list of categories
cat("Categories with more than 1% fraudulent transactions:\n")
```

```
## Categories with more than 1% fraudulent transactions:
```

```r
print(fraudulent_categories$category)
```

```
## [1] "grocery_pos"  "misc_net"     "shopping_net"
```

Exploring Merchant feature

```r
length(unique(df$merchant))
```

```
## [1] 693
```

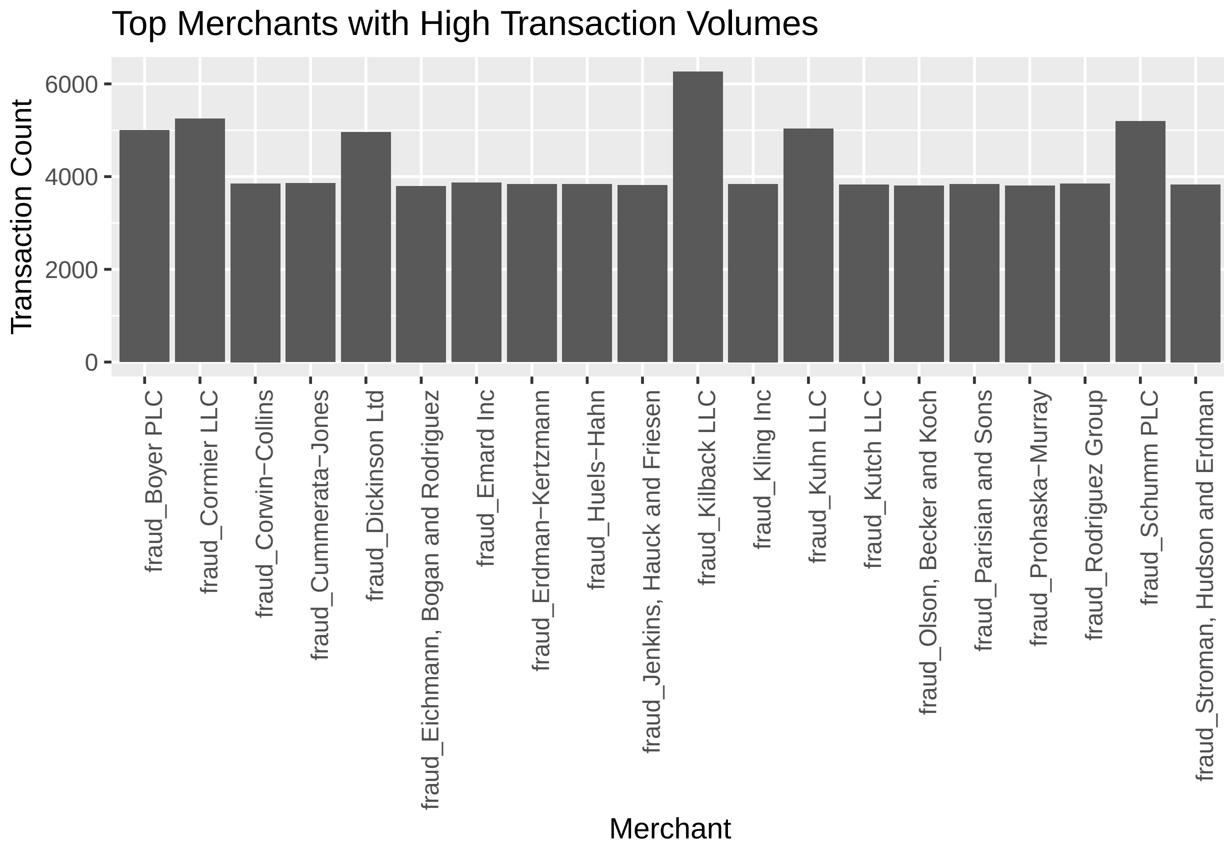
```r
# Get the top 20 high transaction merchants
high_trans_merchants = names(head(sort(table(df$merchant), decreasing = TRUE), 20))

# Print the list of high transaction merchants
cat("High transaction merchants:\n")
```

```
## High transaction merchants:
```

```r
print(high_trans_merchants)
```

```
##  [1] "fraud_Kilback LLC"                   "fraud_Cormier LLC"
##  [3] "fraud_Schumm PLC"                    "fraud_Kuhn LLC"
##  [5] "fraud_Boyer PLC"                     "fraud_Dickinson Ltd"
##  [7] "fraud_Emard Inc"                     "fraud_Cummerata-Jones"
##  [9] "fraud_Corwin-Collins"                "fraud_Rodriguez Group"
## [11] "fraud_Kling Inc"                     "fraud_Erdman-Kertzmann"
## [13] "fraud_Parisian and Sons"             "fraud_Huels-Hahn"
## [15] "fraud_Stroman, Hudson and Erdman"    "fraud_Kutch LLC"
## [17] "fraud_Jenkins, Hauck and Friesen"    "fraud_Prohaska-Murray"
## [19] "fraud_Olson, Becker and Koch"        "fraud_Eichmann, Bogan and Rodriguez"
```

```r
ggplot(df[df$merchant %in% high_trans_merchants, ], aes(x = merchant)) +
  geom_bar() +
  labs(title = "Top Merchants with High Transaction Volumes",
       x = "Merchant",
       y = "Transaction Count") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Now, as we done with the EDA, we will move to the feature encoding

Feature Encoding

- One hot encoding

```
# One-hot encode the category variable
category_onehot = model.matrix(~0 + category, data = df)

# Rename the columns
colnames(category_onehot) = gsub("category", "category_", colnames(category_onehot))

# Remove the intercept column
category_onehot = category_onehot[, -1]

# One-hot encode the gender variable
gender_onehot = model.matrix(~0 + gender, data = df)
colnames(gender_onehot) <- gsub("gender", "gender_", colnames(gender_onehot))
gender_onehot = gender_onehot[, -1]

# One-hot encode the day_of_week variable
day_of_week_onehot = model.matrix(~0 + trans_day_of_week, data = df)
colnames(day_of_week_onehot) = gsub("trans_day_of_week", "day_", colnames(day_of_week_onehot))
day_of_week_onehot = day_of_week_onehot[, -1]
```

```r
# One-hot encode the age variable
age_onehot = model.matrix(~0 + age, data = df)
colnames(age_onehot) = gsub("age", "age_", colnames(age_onehot))
age_onehot = age_onehot[, -1]
```

```r
# Combine the one-hot encoded matrices with the original data frame
df1 = cbind(df, category_onehot, gender_onehot, day_of_week_onehot, age_onehot)
head(df1)
```

```
##   X       cc_num                            merchant      category    amt gender
## 1 0 2.703186e+15          fraud_Rippin, Kub and Mann      misc_net   4.97      F
## 2 1 6.304233e+11     fraud_Heller, Gutmann and Zieme   grocery_pos 107.23      F
## 3 2 3.885949e+13                fraud_Lind-Buckridge entertainment 220.11      M
## 4 3 3.534094e+15 fraud_Kutch, Hermiston and Farrell gas_transport  45.00      M
## 5 4 3.755342e+14                 fraud_Keeling-Crist      misc_pos  41.96      M
## 6 5 4.767265e+15   fraud_Stroman, Hudson and Erdman gas_transport  94.63      F
##                            street           city state   zip     lat      long
## 1                561 Perry Cove Moravian Falls    NC 28654 36.0788  -81.1781
## 2 43039 Riley Greens Suite 393         Orient    WA 99160 48.8878 -118.2105
## 3     594 White Dale Suite 530     Malad City    ID 83252 42.1808 -112.2620
## 4  9443 Cynthia Court Apt. 038        Boulder    MT 59632 46.2306 -112.1138
## 5              408 Bradley Rest       Doe Hill    VA 24433 38.4207  -79.4629
## 6             4655 David Island         Dublin    PA 18917 40.3750  -75.2045
##   city_pop                               job                        trans_num
## 1     3495         Psychologist, counselling 0b242abb623afc578575680df30655b9
## 2      149 Special educational needs teacher 1f76529f8574734946361c461b024d99
## 3     4154       Nature conservation officer a1a22d70485983eac12b5b88dad1cf95
## 4     1939                   Patent attorney 6b849c168bdad6f867558c3793159a81
## 5       99    Dance movement psychotherapist a41d7549acf90789359a9aa5346dcb46
## 6     2158                 Transport planner 189a841a0a8ba03058526bcfe566aab5
##    unix_time merch_lat merch_long is_fraud trans_hour trans_day_of_week
## 1 1325376018  36.01129  -82.04832        0          0           Tuesday
## 2 1325376044  49.15905 -118.18646        0          0           Tuesday
## 3 1325376051  43.15070 -112.15448        0          0           Tuesday
## 4 1325376076  47.03433 -112.56107        0          0           Tuesday
## 5 1325376186  38.67500  -78.63246        0          0           Tuesday
## 6 1325376248  40.65338  -76.15267        0          0           Tuesday
##   trans_year_month age age_bin category_food_dining category_gas_transport
## 1            19-01  31   30-45                    0                      0
## 2            19-01  41   30-45                    0                      0
## 3            19-01  57   46-60                    0                      0
## 4            19-01  52   46-60                    0                      1
## 5            19-01  33   30-45                    0                      0
## 6            19-01  58   46-60                    0                      1
##   category_grocery_net category_grocery_pos category_health_fitness
## 1                    0                    0                       0
## 2                    0                    1                       0
## 3                    0                    0                       0
## 4                    0                    0                       0
## 5                    0                    0                       0
## 6                    0                    0                       0
##   category_home category_kids_pets category_misc_net category_misc_pos
## 1             0                  0                 1                 0
```

```
## 2              0                   0                  0                  0
## 3              0                   0                  0                  0
## 4              0                   0                  0                  0
## 5              0                   0                  0                  1
## 6              0                   0                  0                  0
##   category_personal_care category_shopping_net category_shopping_pos
## 1                      0                     0                     0
## 2                      0                     0                     0
## 3                      0                     0                     0
## 4                      0                     0                     0
## 5                      0                     0                     0
## 6                      0                     0                     0
##   category_travel gender_onehot day_Monday day_Saturday day_Sunday day_Thursday
## 1               0             0          0            0          0            0
## 2               0             0          0            0          0            0
## 3               0             1          0            0          0            0
## 4               0             1          0            0          0            0
## 5               0             1          0            0          0            0
## 6               0             0          0            0          0            0
##   day_Tuesday day_Wednesday
## 1           1             0
## 2           1             0
## 3           1             0
## 4           1             0
## 5           1             0
## 6           1             0
```

```r
# Drop specified columns
df1 = df1 %>%
  select(-cc_num, -trans_num)

# Print the dimensions of the data frame
print(dim(df1))
```

```
## [1] 1852394      42
```

```r
# Print the column names
print(names(df1))
```

```
##  [1] "X"                      "merchant"
##  [3] "category"               "amt"
##  [5] "gender"                 "street"
##  [7] "city"                   "state"
##  [9] "zip"                    "lat"
## [11] "long"                   "city_pop"
## [13] "job"                    "unix_time"
## [15] "merch_lat"              "merch_long"
## [17] "is_fraud"               "trans_hour"
## [19] "trans_day_of_week"      "trans_year_month"
## [21] "age"                    "age_bin"
## [23] "category_food_dining"   "category_gas_transport"
## [25] "category_grocery_net"   "category_grocery_pos"
## [27] "category_health_fitness" "category_home"
```

```
## [29] "category_kids_pets"      "category_misc_net"
## [31] "category_misc_pos"       "category_personal_care"
## [33] "category_shopping_net"   "category_shopping_pos"
## [35] "category_travel"         "gender_onehot"
## [37] "day_Monday"              "day_Saturday"
## [39] "day_Sunday"              "day_Thursday"
## [41] "day_Tuesday"             "day_Wednesday"
```

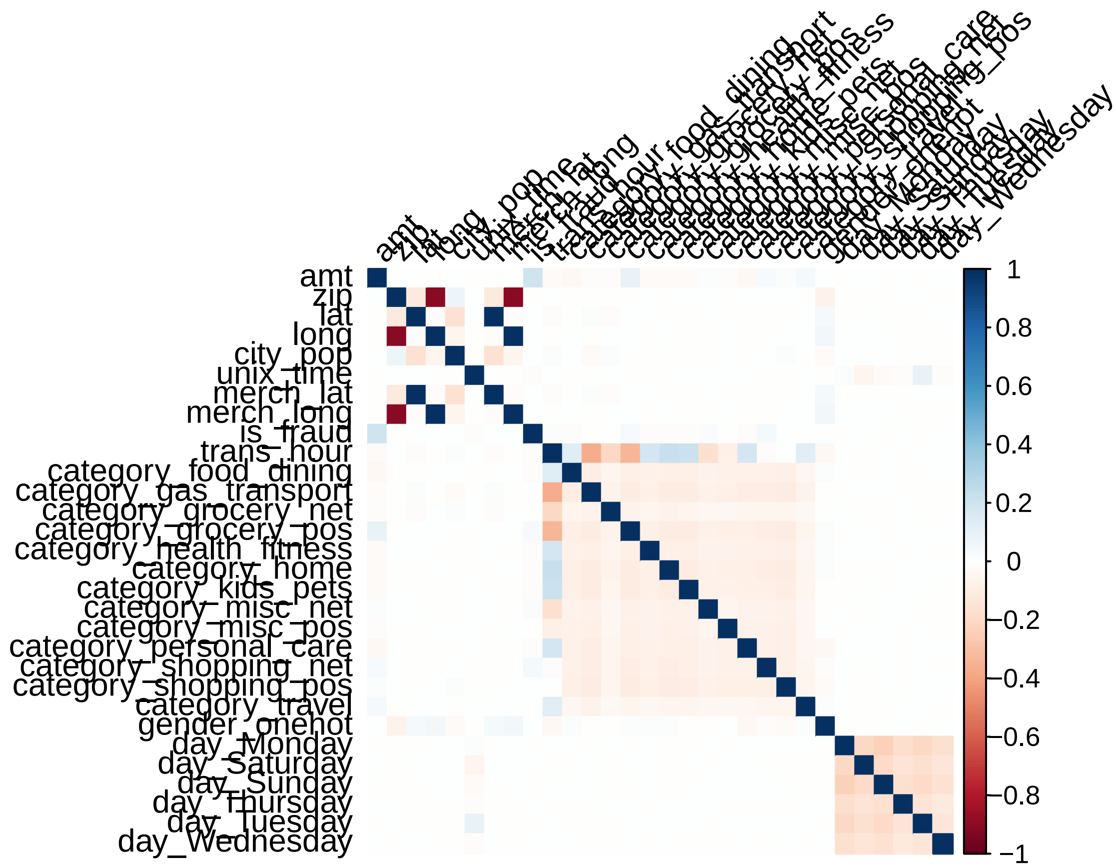
- In the above df1 Data frame, the feature 'merchant' can be dropped since it has lot of unique values and it is hard to encode all of them. And the same applies to the variables - 'street', 'city', 'state' and 'job'
- Similarly, the variables - 'age', 'category', 'gender', 'trans_day_of_week' can also be dropped since they have already been encoded.

```
# Drop specified columns
df1 = df1 %>%
  select(-merchant, -street, -city, -state, -job,
         -category, -gender, -trans_day_of_week, -age)

# Print the column names
print(names(df1))
```

```
##  [1] "X"                       "amt"
##  [3] "zip"                     "lat"
##  [5] "long"                    "city_pop"
##  [7] "unix_time"               "merch_lat"
##  [9] "merch_long"              "is_fraud"
## [11] "trans_hour"              "trans_year_month"
## [13] "age_bin"                 "category_food_dining"
## [15] "category_gas_transport"  "category_grocery_net"
## [17] "category_grocery_pos"    "category_health_fitness"
## [19] "category_home"           "category_kids_pets"
## [21] "category_misc_net"       "category_misc_pos"
## [23] "category_personal_care"  "category_shopping_net"
## [25] "category_shopping_pos"   "category_travel"
## [27] "gender_onehot"           "day_Monday"
## [29] "day_Saturday"            "day_Sunday"
## [31] "day_Thursday"            "day_Tuesday"
## [33] "day_Wednesday"
```

```
# Drop specified columns
df1 = df1 %>%
  select(-X)
```

```
# Select only the numeric columns for correlation calculation
numeric_cols = sapply(df1, is.numeric)
df_numeric = df1[, numeric_cols]

# Calculate the correlations
df_random_under_corr = cor(df_numeric)

# Plotting the correlation heatmap
corrplot(df_random_under_corr, method="color", type=c("full", "lower", "upper"), tl.col="black", tl.srt=
```



- Now, since there are a lot of variables let us get the variables that have high correlation using a function that outputs the variables with correlation between them above a certain threshold.

```
# Function to return highly correlated columns above a threshold
correlation = function(dataset, threshold) {
  numeric_cols = sapply(dataset, is.numeric)
  numeric_dataset = dataset[, numeric_cols]

  col_corr = c()  # This vector stores the highly correlated columns
  corr_matrix = cor(numeric_dataset, use = "pairwise.complete.obs")  # Correlation matrix

  # Traversing the correlation matrix
  for (i in 1:(ncol(corr_matrix) - 1)) {
    for (j in (i + 1):ncol(corr_matrix)) {
      if (!is.na(corr_matrix[i, j]) && abs(corr_matrix[i, j]) > threshold) {
        colname <- colnames(corr_matrix)[i]  # Selecting columns above threshold
        col_corr <- c(col_corr, colname)  # Adding columns to vector
      }
    }
  }
  return(col_corr)
}

# Example usage
highly_correlated_cols = correlation(df1, threshold = 0.7)
print(highly_correlated_cols)
```

```
## [1] "zip"  "zip"  "lat"  "long"
```

```
highly_correlated_cols = correlation(df1, threshold = 0.95)
print(highly_correlated_cols)
```

```
## [1] "lat"  "long"
```

Implementing Algorithm

```
# Storing the number of values in each class
non_fraud_count = sum(df1$is_fraud == 0)
fraud_count = sum(df1$is_fraud == 1)
```

```
# Storing the numerical columns of the data and removing unnecessary variables
df_num = df1 %>%
  select_if(is.numeric) %>%
  select(-c(zip, lat, long, city_pop, unix_time, merch_lat, merch_long))

# To see the column names
colnames(df_num)
```

```
##  [1] "amt"                       "is_fraud"
##  [3] "trans_hour"                "category_food_dining"
##  [5] "category_gas_transport"    "category_grocery_net"
##  [7] "category_grocery_pos"      "category_health_fitness"
##  [9] "category_home"             "category_kids_pets"
## [11] "category_misc_net"         "category_misc_pos"
## [13] "category_personal_care"    "category_shopping_net"
## [15] "category_shopping_pos"     "category_travel"
## [17] "gender_onehot"             "day_Monday"
## [19] "day_Saturday"              "day_Sunday"
## [21] "day_Thursday"              "day_Tuesday"
## [23] "day_Wednesday"
```

```
summary(df_num)
```

```
##       amt               is_fraud          trans_hour    category_food_dining
##  Min.   :    1.00   Min.   :0.00000   Min.   : 0.00   Min.   :0.00000
##  1st Qu.:    9.64   1st Qu.:0.00000   1st Qu.: 7.00   1st Qu.:0.00000
##  Median :   47.45   Median :0.00000   Median :14.00   Median :0.00000
##  Mean   :   70.06   Mean   :0.00521   Mean   :12.81   Mean   :0.07057
##  3rd Qu.:   83.10   3rd Qu.:0.00000   3rd Qu.:19.00   3rd Qu.:0.00000
##  Max.   :28948.90   Max.   :1.00000   Max.   :23.00   Max.   :1.00000
##  category_gas_transport category_grocery_net category_grocery_pos
##  Min.   :0.0000         Min.   :0.00000      Min.   :0.00000
##  1st Qu.:0.0000         1st Qu.:0.00000      1st Qu.:0.00000
##  Median :0.0000         Median :0.00000      Median :0.00000
```

```
##  Mean   :0.1015         Mean   :0.03502      Mean   :0.09512
##  3rd Qu.:0.0000         3rd Qu.:0.00000      3rd Qu.:0.00000
##  Max.   :1.0000         Max.   :1.00000      Max.   :1.00000
##  category_health_fitness category_home     category_kids_pets category_misc_net
##  Min.   :0.00000         Min.   :0.00000   Min.   :0.00000    Min.   :0.00000
##  1st Qu.:0.00000         1st Qu.:0.00000   1st Qu.:0.00000    1st Qu.:0.00000
##  Median :0.00000         Median :0.00000   Median :0.00000    Median :0.00000
##  Mean   :0.06616         Mean   :0.09472   Mean   :0.08731    Mean   :0.04894
##  3rd Qu.:0.00000         3rd Qu.:0.00000   3rd Qu.:0.00000    3rd Qu.:0.00000
##  Max.   :1.00000         Max.   :1.00000   Max.   :1.00000    Max.   :1.00000
##  category_misc_pos category_personal_care category_shopping_net
##  Min.   :0.00000   Min.   :0.00000        Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000        1st Qu.:0.00000
##  Median :0.00000   Median :0.00000        Median :0.00000
##  Mean   :0.06167   Mean   :0.07023        Mean   :0.07521
##  3rd Qu.:0.00000   3rd Qu.:0.00000        3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000        Max.   :1.00000
##  category_shopping_pos category_travel   gender_onehot      day_Monday
##  Min.   :0.00000       Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000       1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000       Median :0.00000   Median :0.0000   Median :0.0000
##  Mean   :0.08986       Mean   :0.03129   Mean   :0.4522   Mean   :0.1994
##  3rd Qu.:0.00000       3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :1.00000       Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##   day_Saturday      day_Sunday       day_Thursday      day_Tuesday
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.1421   Mean   :0.1855   Mean   :0.1116   Mean   :0.1459
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  day_Wednesday
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.09928
##  3rd Qu.:0.00000
##  Max.   :1.00000
```

```r
# Save the df_num DataFrame to a CSV file named 'processed.csv'
write.csv(df_num, file = 'processed.csv', row.names = FALSE)
```

```r
dataset = read.csv("processed.csv")
```

- splitting the dataset

```r
set.seed(123)
split = sample.split(dataset$is_fraud, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

```r
# Feature Scaling
training_set[-2] = scale(training_set[-2])
test_set[-2] = scale(test_set[-2])
```

Implementing Logistic regression algorithm

```r
# Fitting Logistic Regression to the Training set
classifier = glm(formula = is_fraud ~ .,
                 family = binomial,
                 data = training_set)
```

```r
# Predicting the Test set results
prob_pred = predict(classifier, type = 'response', newdata = test_set[-2])
y_pred = ifelse(prob_pred > 0.5, 1, 0)
```

```r
# Making the Confusion Matrix
cm = table(test_set[, 2], y_pred > 0.5)
print(cm)
```

```
##
##      FALSE   TRUE
##   0 460513    173
##   1   2413      0
```

```r
# Calculate precision
precision <- cm[2, 2] / sum(cm[, 2])

# Calculate recall
recall <- cm[2, 2] / sum(cm[2, ])

# Calculate F1 score
f1_score <- 2 * (precision * recall) / (precision + recall)

# Calculate accuracy score
accuracy <- sum(diag(cm)) / sum(cm)

# Create a data frame for the metrics
metrics_df <- data.frame(
  Metric = c("Precision", "Recall", "F1 Score", "Accuracy"),
  Value = c(precision, recall, f1_score, accuracy)
)

# Print the metrics table
kable(metrics_df, format = "html", caption = "Evaluation Metrics For model_1")
```

Evaluation Metrics For model_1

Metric

Value

Precision

0.0000000

Recall

0.0000000

F1 Score

NaN

Accuracy

0.9944159

- our accuracy is high but f1 score is Nan. this is because of the data imbalace proble, we have to deal with it.

Resampling technique (Over sampling)

```
# Perform oversampling using ROSE
oversampled_data = ovun.sample(is_fraud ~ ., data = dataset, method = "over", N = 2500000)$data

# Check the class distribution after oversampling
table(oversampled_data$is_fraud)
```

```
##
##       0       1
## 1842743  657257
```

- splitting the dataset

```
set.seed(123)
split = sample.split(oversampled_data$is_fraud, SplitRatio = 0.75)
training_set1 = subset(oversampled_data, split == TRUE)
test_set1 = subset(oversampled_data, split == FALSE)
```

```
# Feature Scaling
training_set1[-2] = scale(training_set1[-2])
test_set1[-2] = scale(test_set1[-2])
```

- Fitting the model for Oversampled data

```
# Fitting Logistic Regression to the Training set
classifier1 = glm(formula = is_fraud ~ .,
                  family = binomial,
                  data = training_set1)

# Predicting the Test set results
prob_pred = predict(classifier1, type = 'response', newdata = test_set1[-2])
y_pred1 = ifelse(prob_pred > 0.5, 1, 0)
```

```
# Making the Confusion Matrix
cm = table(test_set1[, 2], y_pred1 > 0.5)
print(cm)
```

```
##
##        FALSE   TRUE
##    0 452826   7860
##    1  45305 119009
```

```r
# Calculate precision
precision <- cm[2, 2] / sum(cm[, 2])

# Calculate recall
recall <- cm[2, 2] / sum(cm[2, ])

# Calculate F1 score
f1_score <- 2 * (precision * recall) / (precision + recall)

# Calculate accuracy score
accuracy <- sum(diag(cm)) / sum(cm)

# Create a data frame for the metrics
metrics_df <- data.frame(
  Metric = c("Precision", "Recall", "F1 Score", "Accuracy"),
  Value = c(precision, recall, f1_score, accuracy)
)

# Print the metrics table
kable(metrics_df, format = "html", caption = "Evaluation Metrics after using Over Sampling")
```

Evaluation Metrics after using Over Sampling

Metric

Value

Precision

0.9380463

Recall

0.7242779

F1 Score

0.8174172

Accuracy

0.9149360

Resampling technique (Under sampling)

```r
# Perform oversampling using ROSE
undersampled_data = ovun.sample(is_fraud ~ ., data = dataset, method = "under", N = 35000)$data

# Check the class distribution after oversampling
table(undersampled_data$is_fraud)
```

```
##
##     0     1
## 25349  9651
```

- splitting the dataset

```
set.seed(123)
split = sample.split(undersampled_data$is_fraud, SplitRatio = 0.75)
training_set2 = subset(undersampled_data, split == TRUE)
test_set2 = subset(undersampled_data, split == FALSE)
```

```
# Feature Scaling
training_set2[-2] = scale(training_set2[-2])
test_set2[-2] = scale(test_set2[-2])
```

- Fitting the model for Undersampled data

```
# Fitting Logistic Regression to the Training set
classifier2 = glm(formula = is_fraud ~ .,
                 family = binomial,
                 data = training_set2)

# Predicting the Test set results
prob_pred = predict(classifier2, type = 'response', newdata = test_set2[-2])
y_pred2 = ifelse(prob_pred > 0.5, 1, 0)
```

```
# Making the Confusion Matrix
cm = table(test_set2[, 2], y_pred2 > 0.5)
print(cm)
```

```
##
##     FALSE TRUE
##   0  6215  122
##   1   654 1759
```

```
# Calculate precision
precision <- cm[2, 2] / sum(cm[, 2])

# Calculate recall
recall <- cm[2, 2] / sum(cm[2, ])

# Calculate F1 score
f1_score <- 2 * (precision * recall) / (precision + recall)

# Calculate accuracy score
accuracy <- sum(diag(cm)) / sum(cm)

# Create a data frame for the metrics
metrics_df <- data.frame(
  Metric = c("Precision", "Recall", "F1 Score", "Accuracy"),
  Value = c(precision, recall, f1_score, accuracy)
```

```r
)

# Print the metrics table
kable(metrics_df, format = "html", caption = "Evaluation Metrics after using Under Sampling")
```

Evaluation Metrics after using Under Sampling

Metric

Value

Precision

0.9351409

Recall

0.7289681

F1 Score

0.8192827

Accuracy

0.9113143

Conclusion

Out of three model, Logistic regression(with under sampling) is the best model.