Activity_Course 3 Waze project lab

January 21, 2025

1 Waze Project

Course 3 - Go Beyond the Numbers: Translate Data into Insights

Your team is still in the early stages of their user churn project. So far, you've completed a project proposal and used Python to inspect and organize Waze's user data.

You check your inbox and notice a new message from Chidi Ga, your team's Senior Data Analyst. Chidi is pleased with the work you have already completed and requests your assistance with exploratory data analysis (EDA) and further data visualization. Harriet Hadzic, Waze's Director of Data Analysis, will want to review a Python notebook that shows your data exploration and visualization.

A notebook was structured and prepared to help you in this project. Please complete the following questions and prepare an executive summary.

2 Course 3 End-of-course project: Exploratory data analysis

In this activity, you will examine data provided and prepare it for analysis.

The purpose of this project is to conduct exploratory data analysis (EDA) on a provided dataset.

The goal is to continue the examination of the data that you began in the previous Course, adding relevant visualizations that help communicate the story that the data tells.

This activity has 4 parts:

Part 1: Imports, links, and loading

Part 2: Data Exploration * Data cleaning

Part 3: Building visualizations

Part 4: Evaluating and sharing results

Follow the instructions and answer the question below to complete the activity. Then, you will complete an executive summary using the questions listed on the PACE Strategy Document.

Be sure to complete this activity before moving on. The next course item will provide you with a completed exemplar to compare to your own work.

3 Visualize a story in Python

4 PACE stages

Throughout these project notebooks, you'll see references to the problem-solving framework PACE. The following notebook components are labeled with the respective PACE stage: Plan, Analyze, Construct, and Execute.

4.1 PACE: Plan

Consider the questions in your PACE Strategy Document to reflect on the Plan stage.

4.1.1 Task 1. Imports and data loading

For EDA of the data, import the data and packages that will be most helpful, such as pandas, numpy, and matplotlib.

[1]:
```
# Import necessary libraries for data analysis and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read in the data and store it as a dataframe object called df.

Note: As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[2]:
```
# Load the dataset into a dataframe
df = pd.read_csv('waze_dataset.csv')
```

4.2 PACE: Analyze

Consider the questions in your PACE Strategy Document and those below where applicable to complete your code: 1. Does the data need to be restructured or converted into usable formats?

2. Are there any variables that have missing data?

1. Data Restructuring/Conversion: -Some columns may need conversion from object to numeric types -Derive new features like kilometers per drive or drives per day -Ensure consistent data formats across columns -Create categorical encodings for device types

2. Missing Data Analysis: -700 rows (4.67%) have missing values in the 'label' column -Missing labels are distributed proportionally across device types (65% iPhone, 35% Android) -Missing data appears to be random, with no significant deviation in other metrics -Recommended

approach: Remove rows with missing labels due to small percentage and lack of systematic bias

4.2.1 Task 2. Data exploration and cleaning

Consider the following questions:

1. Given the scenario, which data columns are most applicable?
2. Which data columns can you eliminate, knowing they won't solve your problem scenario?
3. How would you check for missing data? And how would you handle missing data (if any)?
4. How would you check for outliers? And how would handle outliers (if any)?

1. Most Applicable Columns: -label (churn status) -sessions -drives -driving_days -device -driven_km_drives -duration_minutes_drives
2. Columns to Eliminate: -ID (unique identifier with no predictive value) -total_navigations_fav1/fav2 (likely not directly related to churn)
3. Missing Data Check: -Use df.isnull().sum() to identify missing values -700 rows missing 'label' -Handle by dropping rows with missing labels
4. Outliers Check and Handling: -Use interquartile range (IQR) method -Identify outliers beyond 1.5 * IQR -Consider log transformation or capping for extreme values -Remove or adjust based on domain knowledge of driving behaviors

Data overview and summary statistics Use the following methods and attributes on the dataframe:

- `head()`
- `size`
- `describe()`
- `info()`

It's always helpful to have this information at the beginning of a project, where you can always refer back to if needed.

[3]:
```
# Data Overview Methods

# View first few rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())
```

```
First 5 rows of the dataset:
   ID     label  sessions  drives  total_sessions  n_days_after_onboarding  \
0   0  retained       283     226      296.748273                     2276
1   1  retained       133     107      326.896596                     1225
2   2  retained       114      95      135.522926                     2651
3   3  retained        49      40       67.589221                       15
4   4  retained        84      68      168.247020                     1562
```

```
   total_navigations_fav1  total_navigations_fav2  driven_km_drives  \
0                     208                       0       2628.845068
1                      19                      64      13715.920550
2                       0                       0       3059.148818
3                     322                       7        913.591123
4                     166                       5       3950.202008

   duration_minutes_drives  activity_days  driving_days   device
0              1985.775061             28            19  Android
1              3160.472914             13            11   iPhone
2              1610.735904             14             8  Android
3               587.196542              7             3   iPhone
4              1219.555924             27            18  Android
```

[4]:
```
# Dataset size
print("\nDataset Size:")
print(f"Total number of elements: {df.size}")
print(f"Number of rows: {len(df)}")
print(f"Number of columns: {len(df.columns)}")
```

```
Dataset Size:
Total number of elements: 194987
Number of rows: 14999
Number of columns: 13
```

Generate summary statistics using the `describe()` method.

[5]:
```
# Descriptive statistics for numeric columns
print("\nSummary Statistics:")
print(df.describe())
```

```
Summary Statistics:
                 ID      sessions        drives  total_sessions  \
count  14999.000000  14999.000000  14999.000000    14999.000000
mean    7499.000000     80.633776     67.281152      189.964447
std     4329.982679     80.699065     65.913872      136.405128
min        0.000000      0.000000      0.000000        0.220211
25%     3749.500000     23.000000     20.000000       90.661156
50%     7499.000000     56.000000     48.000000      159.568115
75%    11248.500000    112.000000     93.000000      254.192341
max    14998.000000    743.000000    596.000000     1216.154633

       n_days_after_onboarding  total_navigations_fav1  \
count             14999.000000            14999.000000
mean               1749.837789              121.605974
std                1008.513876              148.121544
```

```
min                   4.000000               0.000000
25%                 878.000000               9.000000
50%                1741.000000              71.000000
75%                2623.500000             178.000000
max                3500.000000            1236.000000

       total_navigations_fav2  driven_km_drives  duration_minutes_drives  \
count            14999.000000      14999.000000             14999.000000
mean                29.672512       4039.340921              1860.976012
std                 45.394651       2502.149334              1446.702288
min                  0.000000         60.441250                18.282082
25%                  0.000000       2212.600607               835.996260
50%                  9.000000       3493.858085              1478.249859
75%                 43.000000       5289.861262              2464.362632
max                415.000000      21183.401890             15851.727160

       activity_days  driving_days
count   14999.000000  14999.000000
mean       15.537102     12.179879
std         9.004655      7.824036
min         0.000000      0.000000
25%         8.000000      5.000000
50%        16.000000     12.000000
75%        23.000000     19.000000
max        31.000000     30.000000
```

And summary information using the `info()` method.

[6]:
```
# Comprehensive dataset information
print("\nDataset Information:")
df.info()
```

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   ID                       14999 non-null  int64
 1   label                    14299 non-null  object
 2   sessions                 14999 non-null  int64
 3   drives                   14999 non-null  int64
 4   total_sessions           14999 non-null  float64
 5   n_days_after_onboarding  14999 non-null  int64
 6   total_navigations_fav1   14999 non-null  int64
 7   total_navigations_fav2   14999 non-null  int64
 8   driven_km_drives         14999 non-null  float64
```

```
 9   duration_minutes_drives  14999 non-null  float64
 10  activity_days            14999 non-null  int64
 11  driving_days             14999 non-null  int64
 12  device                   14999 non-null  object
dtypes: float64(3), int64(8), object(2)
memory usage: 1.5+ MB
```

4.3 PACE: Construct

Consider the questions in your PACE Strategy Document to reflect on the Construct stage.

Consider the following questions as you prepare to deal with outliers:

1. What are some ways to identify outliers?
2. How do you make the decision to keep or exclude outliers from any future models?

1. Identifying Outliers: -Interquartile Range (IQR): Identify values beyond 1.5 * IQR -Z-score: Detect values more than 3 standard deviations from the mean -Box plots: Visually identify data points beyond whiskers -Scatter plots: Visualize extreme values -Statistical tests like Tukey's method
2. Outlier Handling Decisions: -Understand domain context (e.g., are extreme driving values legitimate?) -Assess impact on model performance

Options: -Remove if clearly erroneous -Transform (log, square root) -Cap at a certain percentile -Use robust statistical methods

-Prioritize preserving real, rare data over arbitrary removal -Consult domain experts for final determination

4.3.1 Task 3a. Visualizations

Select data visualization types that will help you understand and explain the data.

Now that you know which data columns you'll use, it is time to decide which data visualization makes the most sense for EDA of the Waze dataset.

Question: What type of data visualization(s) will be most helpful?

- Line graph
- Bar chart
- Box plot
- Histogram
- Heat map
- Scatter plot
- A geographic map

For the Waze User Churn dataset, the most helpful visualizations would be:

1. Bar Chart: -Compare churn rates by device type -Visualize user distribution (retained vs. churned)

2. Box Plot: -Compare driving behaviors between churned and retained users -Show distribution of key metrics like sessions, drives, kilometers
3. Scatter Plot: -Explore relationship between driving metrics and churn -Visualize kilometers driven vs. number of sessions -Identify potential churn indicators
4. Histogram: -Understand distribution of key numeric variables -Show frequency of driving days, sessions, kilometers
5. Heatmap: -Visualize correlations between numeric variables -Identify potential predictive features for churn

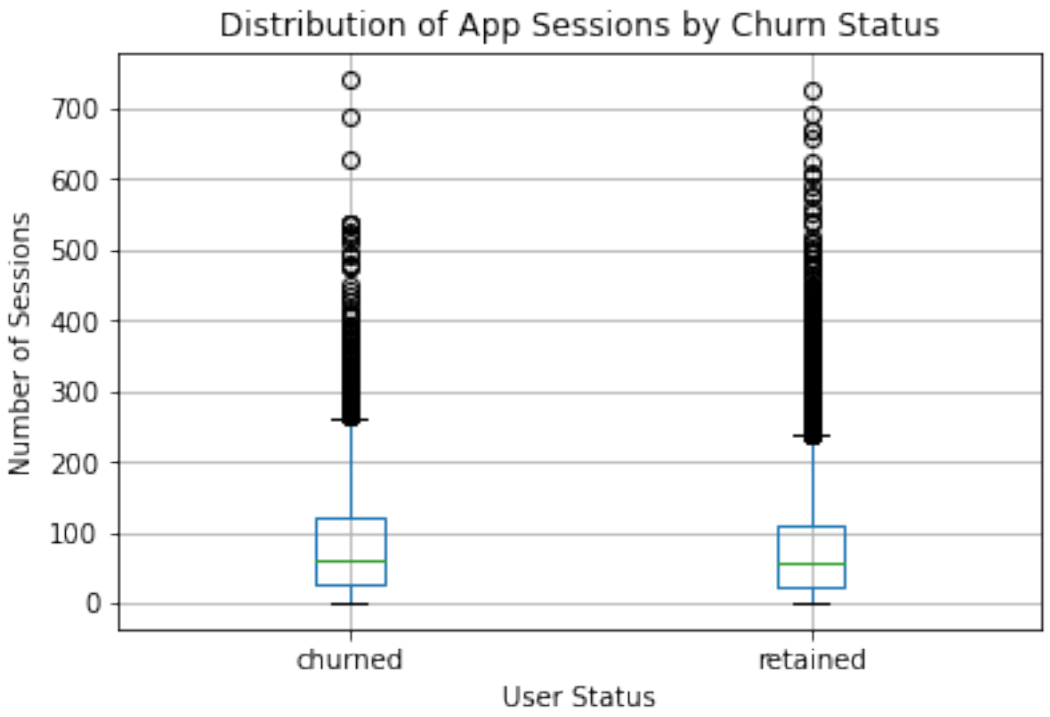
Least Useful: - Geographic map (no location data provided) -Line graph (limited time-series information)

Begin by examining the spread and distribution of important variables using box plots and histograms.

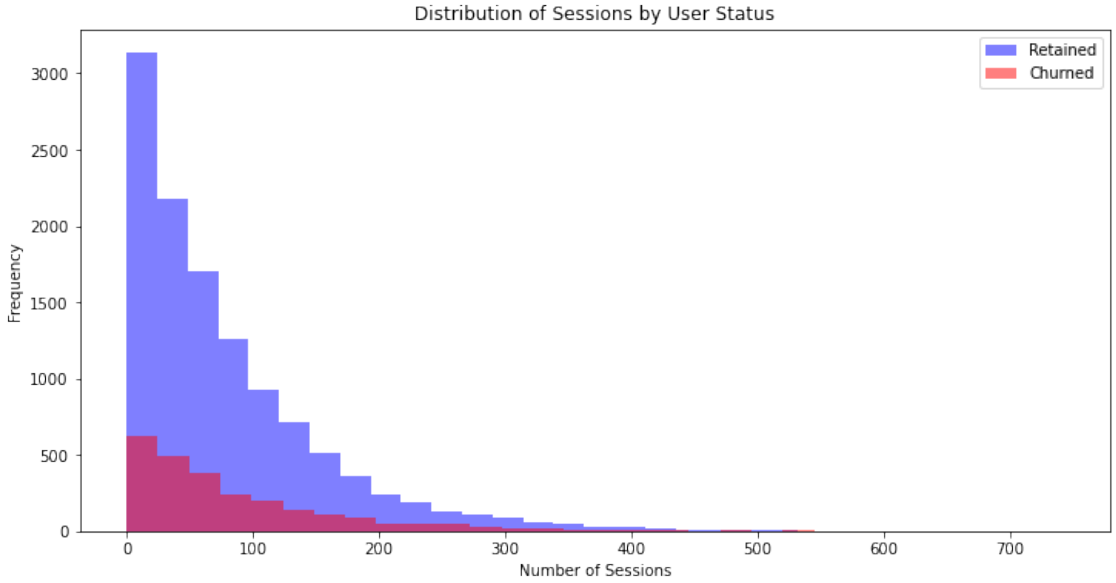
sessions *The number of occurrence of a user opening the app during the month*

[13]:
```
# Box plot
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('waze_dataset.csv')

# Clean the data by removing rows with missing labels
df_cleaned = df.dropna(subset=['label'])

# Basic box plot for sessions by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='sessions', by='label')
plt.title('Distribution of App Sessions by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Number of Sessions')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[14]:
```
# Histogram
# Histogram for sessions
plt.figure(figsize=(12, 6))

# Histogram for retained users
plt.hist(df_cleaned[df_cleaned['label'] == 'retained']['sessions'],
        bins=30, alpha=0.5, label='Retained', color='blue')

# Histogram for churned users
plt.hist(df_cleaned[df_cleaned['label'] == 'churned']['sessions'],
        bins=30, alpha=0.5, label='Churned', color='red')

plt.title('Distribution of Sessions by User Status')
plt.xlabel('Number of Sessions')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

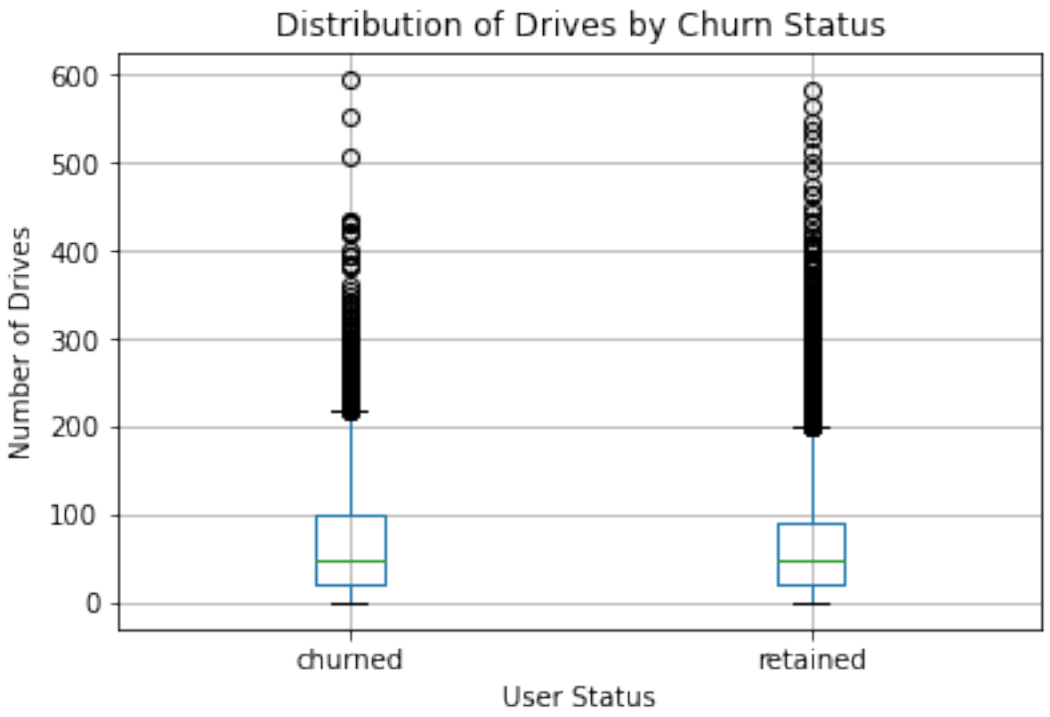


The `sessions` variable is a right-skewed distribution with half of the observations having 56 or fewer sessions. However, as indicated by the boxplot, some users have more than 700.

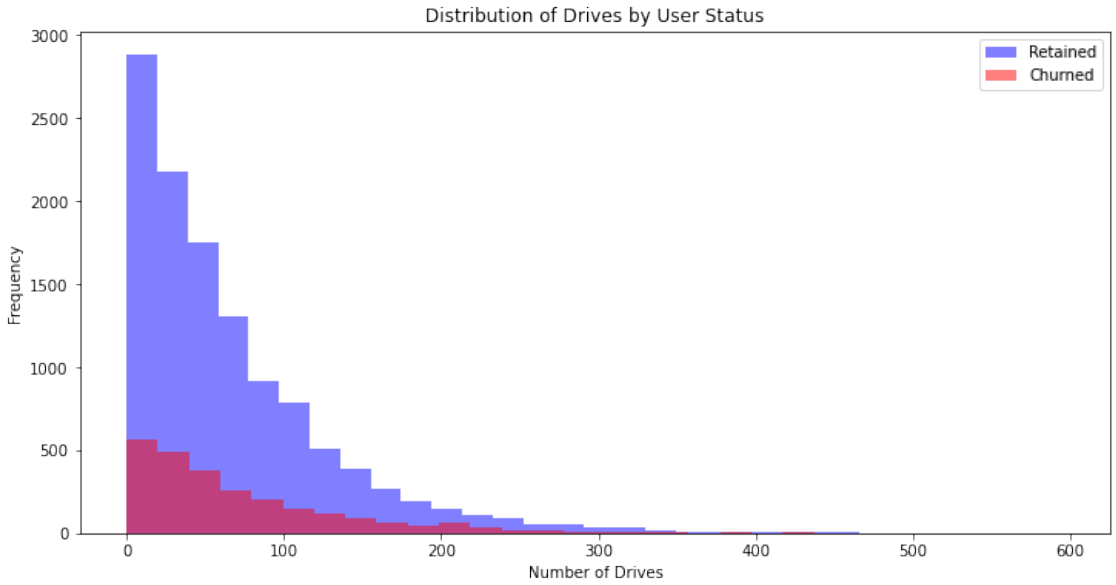
`drives` *An occurrence of driving at least 1 km during the month*

[15]:
```
# Box plot
# Box plot for drives by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='drives', by='label')
plt.title('Distribution of Drives by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Number of Drives')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[17]:
```
# Histogram
# Histogram for drives
plt.figure(figsize=(12, 6))

# Histogram for retained users
plt.hist(df_cleaned[df_cleaned['label'] == 'retained']['drives'],
        bins=30, alpha=0.5, label='Retained', color='blue')

# Histogram for churned users
plt.hist(df_cleaned[df_cleaned['label'] == 'churned']['drives'],
        bins=30, alpha=0.5, label='Churned', color='red')

plt.title('Distribution of Drives by User Status')
plt.xlabel('Number of Drives')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

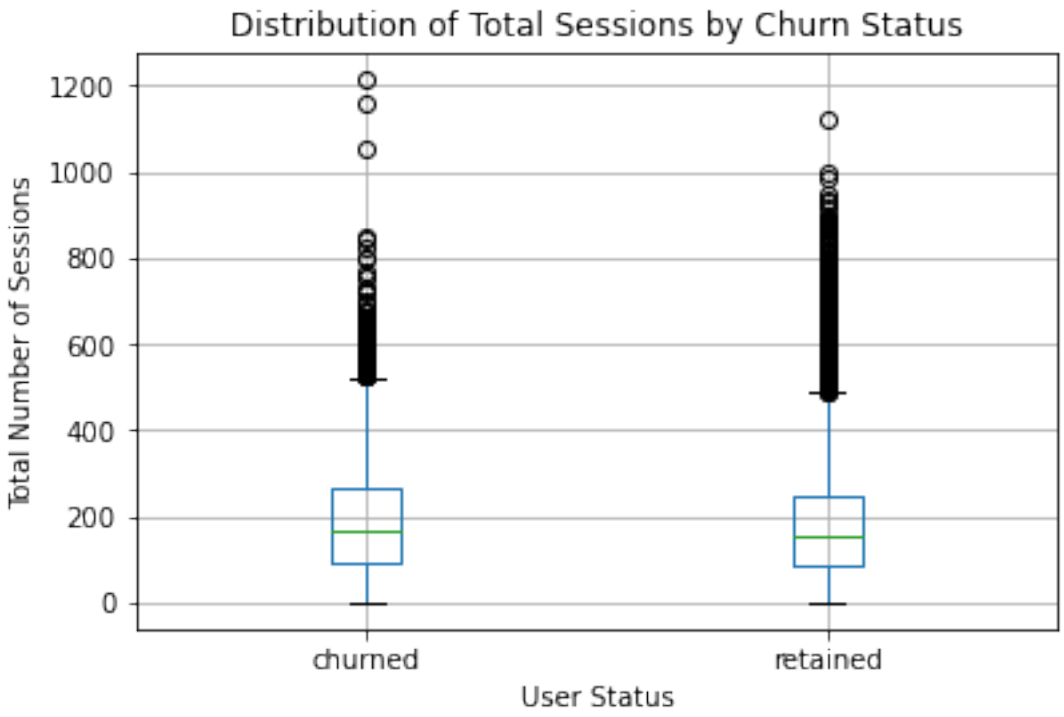


The `drives` information follows a distribution similar to the `sessions` variable. It is right-skewed, approximately log-normal, with a median of 48. However, some drivers had over 400 drives in the last month.

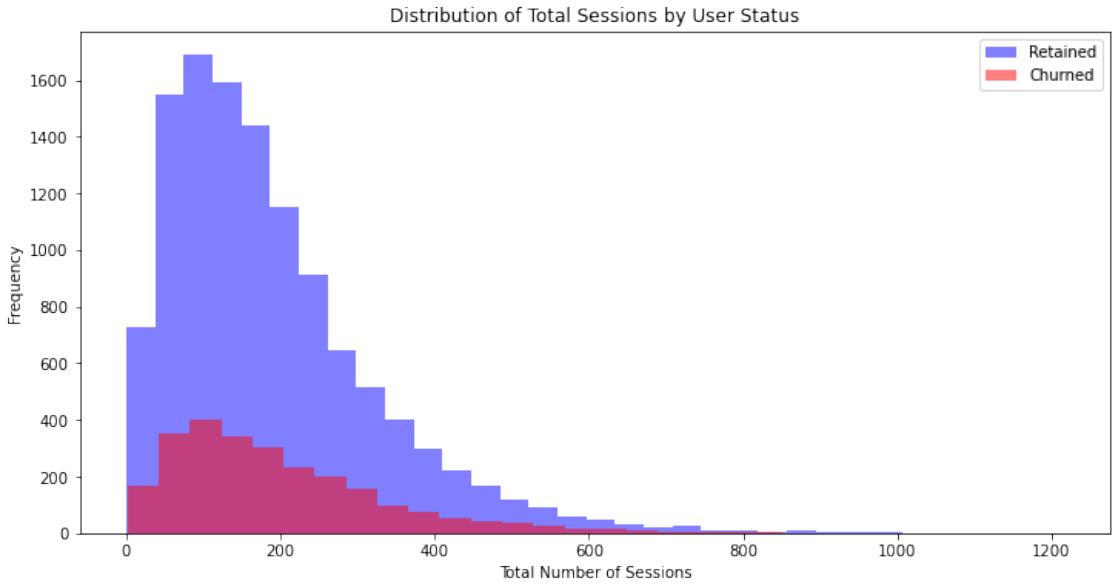
`total_sessions` *A model estimate of the total number of sessions since a user has onboarded*

[18]:
```
# Box plot
# Box plot for total_sessions by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='total_sessions', by='label')
plt.title('Distribution of Total Sessions by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Total Number of Sessions')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[19]:
```
# Histogram
# Histogram for total_sessions
plt.figure(figsize=(12, 6))

# Histogram for retained users
plt.hist(df_cleaned[df_cleaned['label'] == 'retained']['total_sessions'],
        bins=30, alpha=0.5, label='Retained', color='blue')

# Histogram for churned users
plt.hist(df_cleaned[df_cleaned['label'] == 'churned']['total_sessions'],
        bins=30, alpha=0.5, label='Churned', color='red')

plt.title('Distribution of Total Sessions by User Status')
plt.xlabel('Total Number of Sessions')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

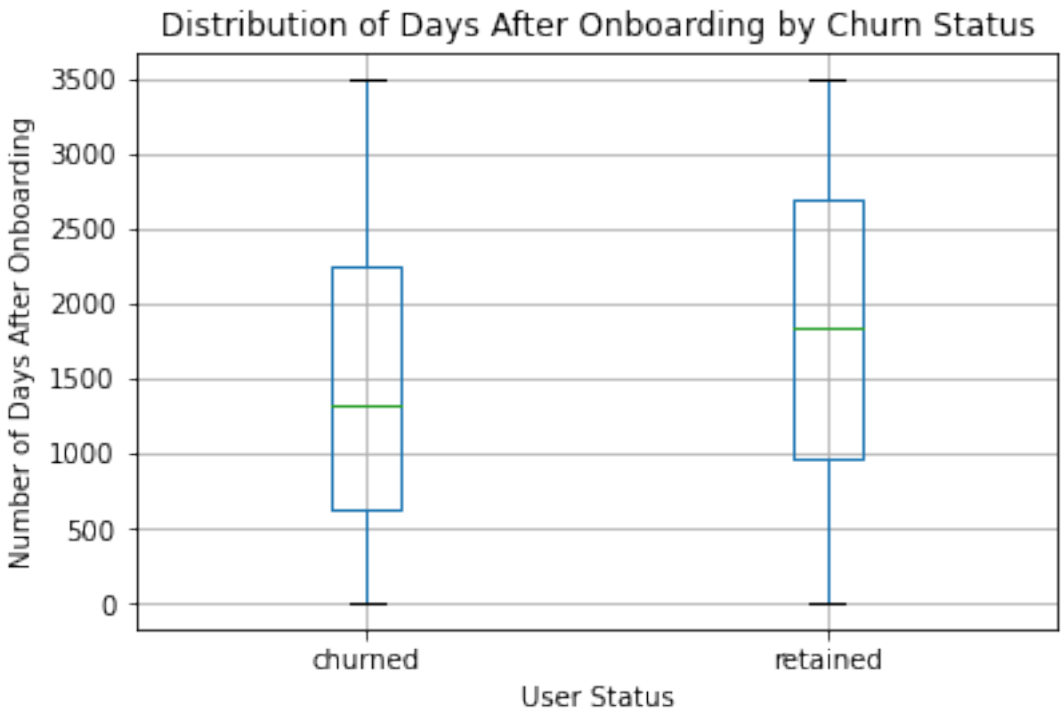


The `total_sessions` is a right-skewed distribution. The median total number of sessions is 159.6. This is interesting information because, if the median number of sessions in the last month was 48 and the median total sessions was ~160, then it seems that a large proportion of a user's total drives might have taken place in the last month. This is something you can examine more closely later.

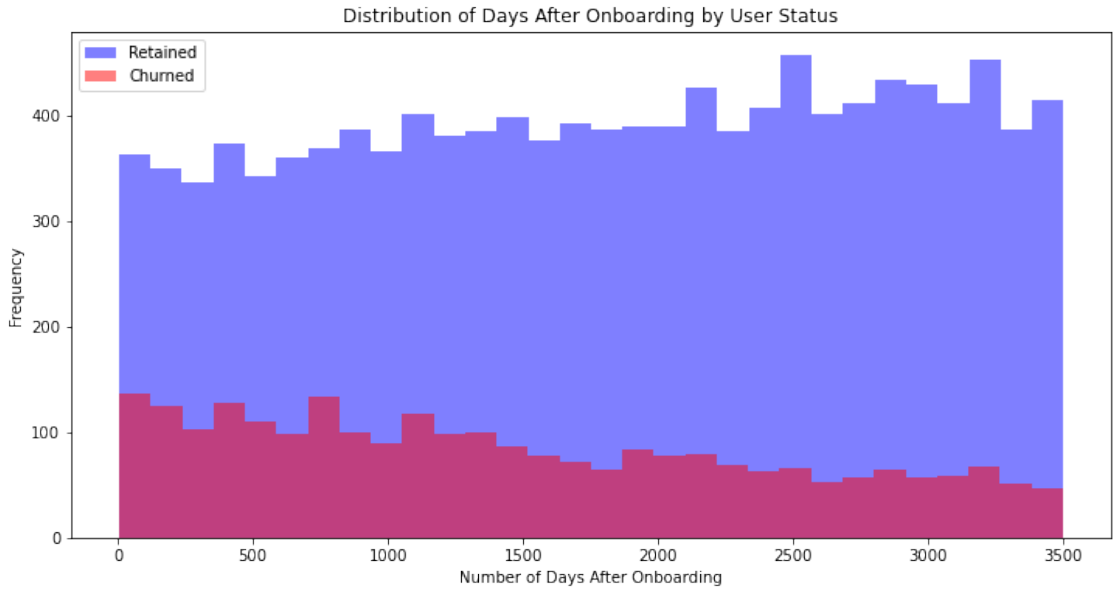
`n_days_after_onboarding` *The number of days since a user signed up for the app*

[20]:
```
# Box plot
# Box plot for n_days_after_onboarding by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='n_days_after_onboarding', by='label')
plt.title('Distribution of Days After Onboarding by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Number of Days After Onboarding')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[21]:
```
# Histogram
# Histogram for n_days_after_onboarding
plt.figure(figsize=(12, 6))

# Histogram for retained users
plt.hist(df_cleaned[df_cleaned['label'] ==␣
 ↪'retained']['n_days_after_onboarding'],
         bins=30, alpha=0.5, label='Retained', color='blue')

# Histogram for churned users
plt.hist(df_cleaned[df_cleaned['label'] ==␣
 ↪'churned']['n_days_after_onboarding'],
         bins=30, alpha=0.5, label='Churned', color='red')

plt.title('Distribution of Days After Onboarding by User Status')
plt.xlabel('Number of Days After Onboarding')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

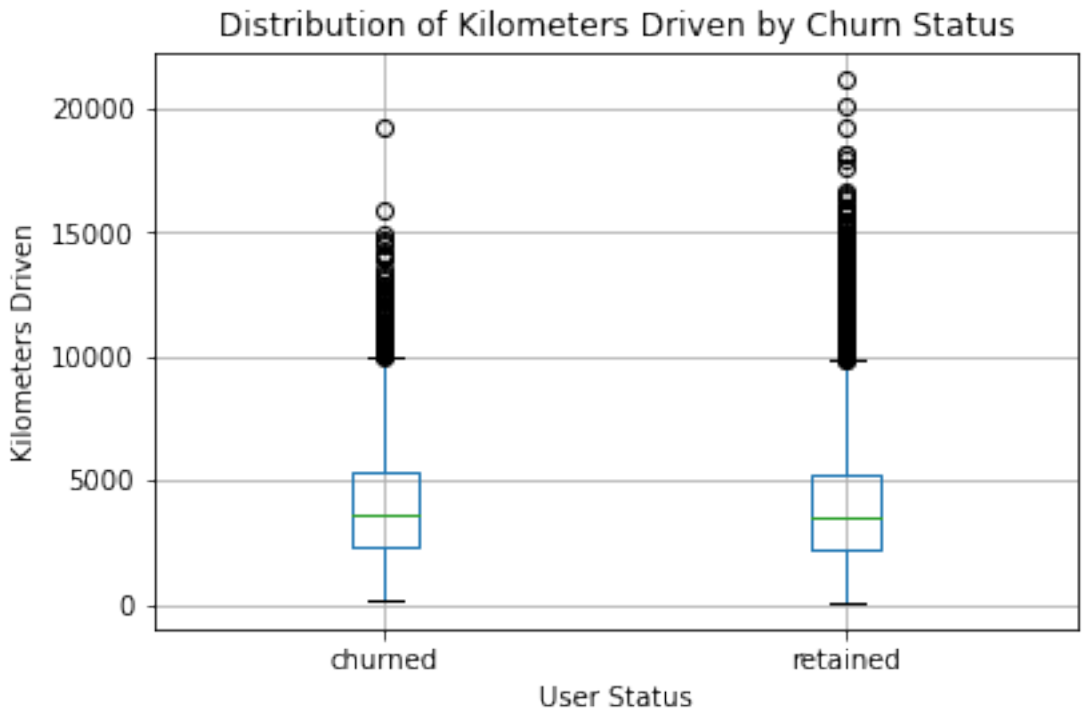


The total user tenure (i.e., number of days since onboarding) is a uniform distribution with values ranging from near-zero to ~3,500 (~9.5 years).

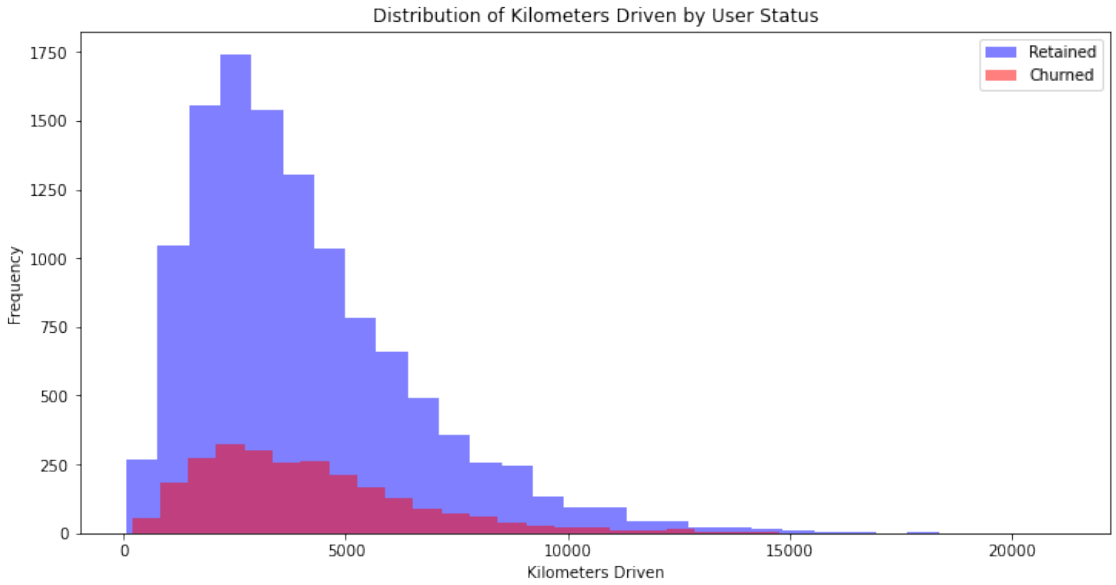
`driven_km_drives` *Total kilometers driven during the month*

[22]:
```
# Box plot for driven_km_drives by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='driven_km_drives', by='label')
plt.title('Distribution of Kilometers Driven by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Kilometers Driven')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[23]:
```
# Histogram
# Histogram for driven_km_drives
plt.figure(figsize=(12, 6))

# Histogram for retained users
plt.hist(df_cleaned[df_cleaned['label'] == 'retained']['driven_km_drives'],
        bins=30, alpha=0.5, label='Retained', color='blue')

# Histogram for churned users
plt.hist(df_cleaned[df_cleaned['label'] == 'churned']['driven_km_drives'],
        bins=30, alpha=0.5, label='Churned', color='red')

plt.title('Distribution of Kilometers Driven by User Status')
plt.xlabel('Kilometers Driven')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

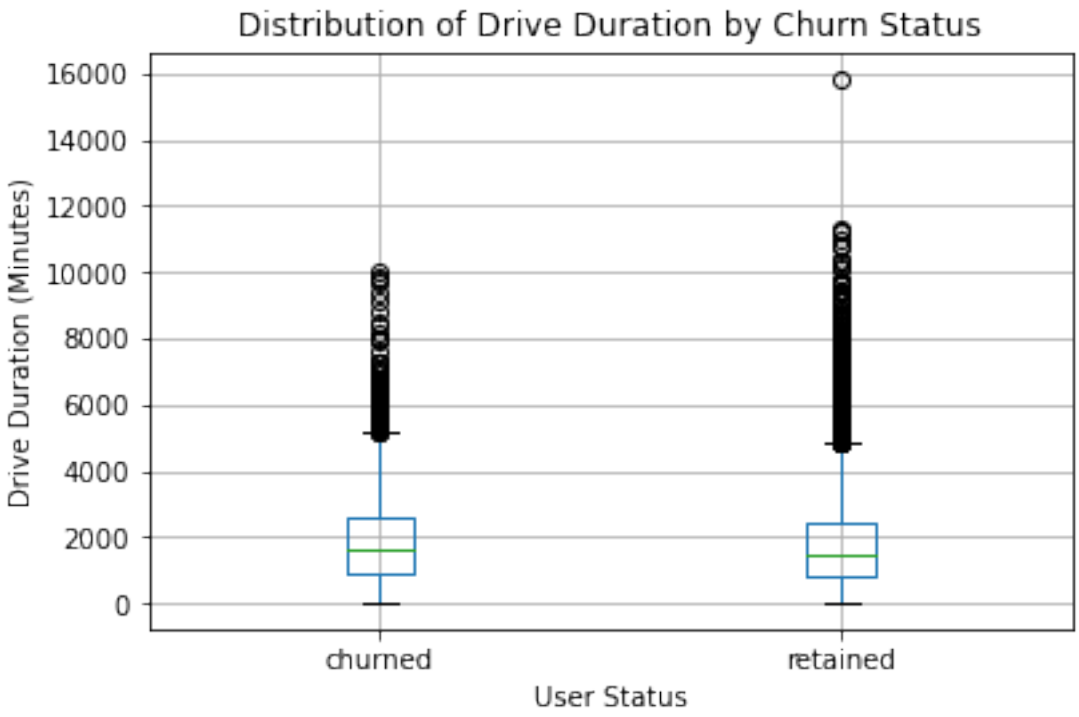


The number of drives driven in the last month per user is a right-skewed distribution with half the users driving under 3,495 kilometers. As you discovered in the analysis from the previous course, the users in this dataset drive *a lot*. The longest distance driven in the month was over half the circumferene of the earth.

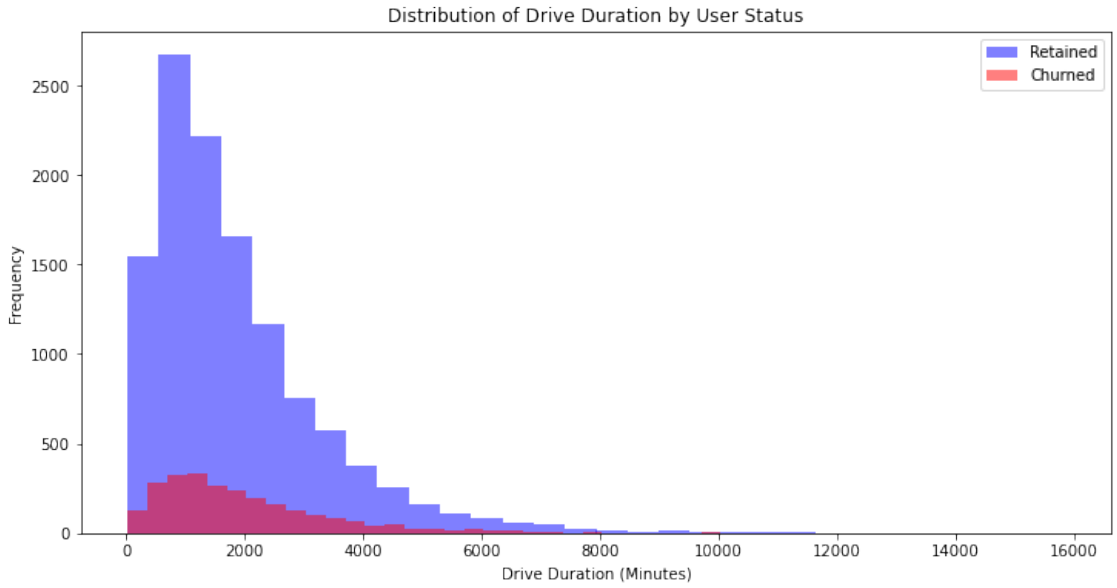
duration_minutes_drives *Total duration driven in minutes during the month*

[24]:
```
# Box plot
# Box plot for duration_minutes_drives by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='duration_minutes_drives', by='label')
plt.title('Distribution of Drive Duration by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Drive Duration (Minutes)')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[25]:
```
# Histogram

# Histogram for duration_minutes_drives
plt.figure(figsize=(12, 6))

# Histogram for retained users
plt.hist(df_cleaned[df_cleaned['label'] ==␣
 ↪'retained']['duration_minutes_drives'],
         bins=30, alpha=0.5, label='Retained', color='blue')

# Histogram for churned users
plt.hist(df_cleaned[df_cleaned['label'] ==␣
 ↪'churned']['duration_minutes_drives'],
         bins=30, alpha=0.5, label='Churned', color='red')

plt.title('Distribution of Drive Duration by User Status')
plt.xlabel('Drive Duration (Minutes)')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

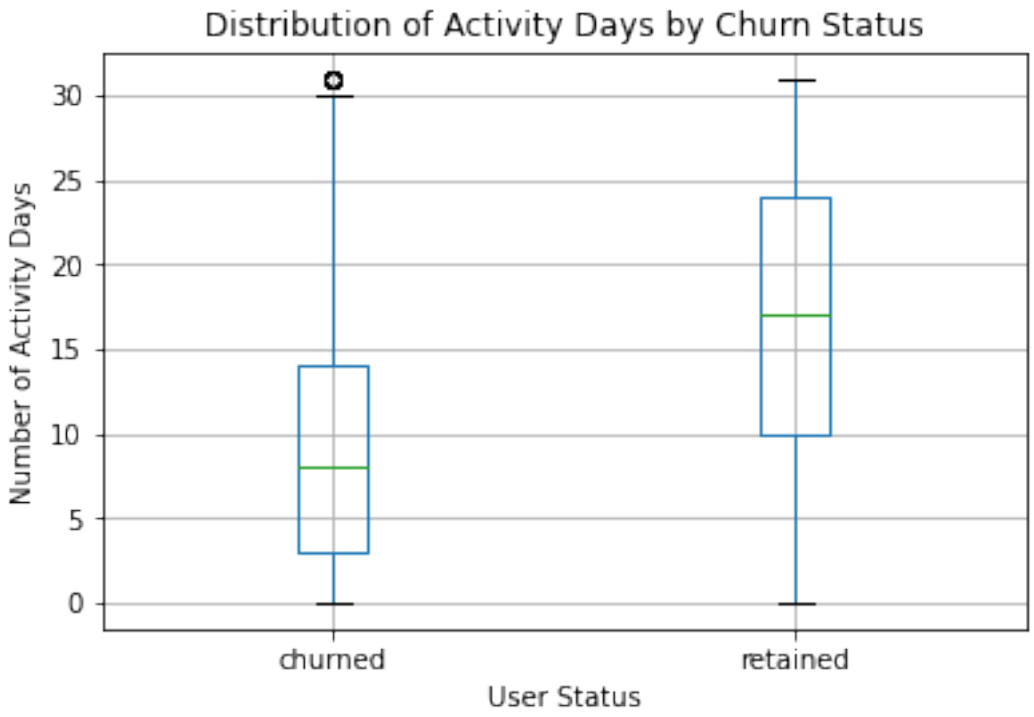


The `duration_minutes_drives` variable has a heavily skewed right tail. Half of the users drove less than ~1,478 minutes (~25 hours), but some users clocked over 250 hours over the month.

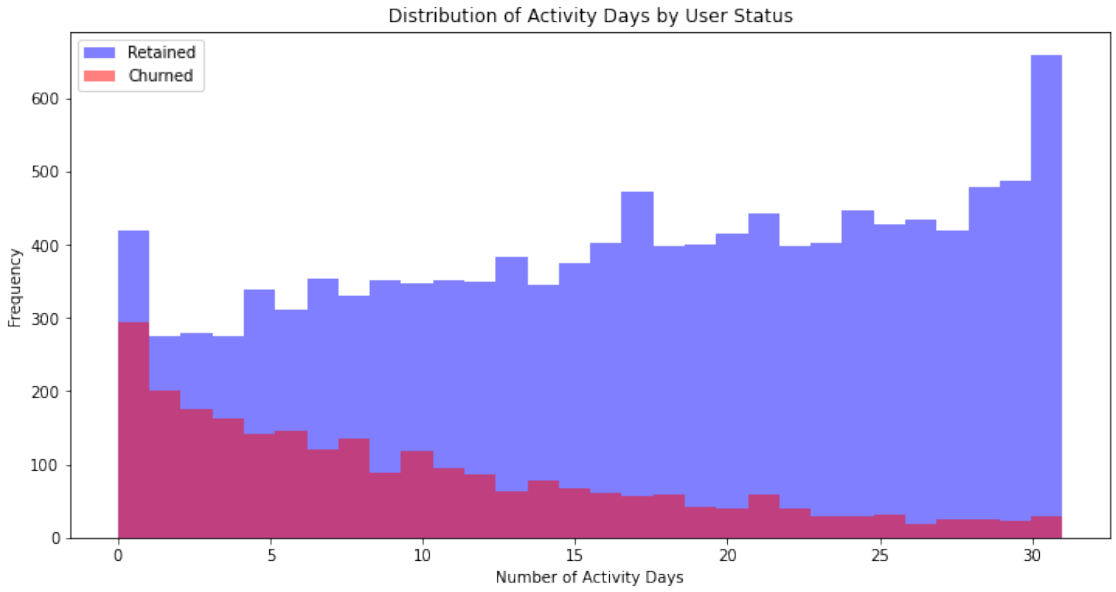
`activity_days` *Number of days the user opens the app during the month*

[27]:
```
# Box plot
# Box plot for activity_days by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='activity_days', by='label')
plt.title('Distribution of Activity Days by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Number of Activity Days')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[28]:
```
# Histogram
# Histogram for activity_days
plt.figure(figsize=(12, 6))

# Histogram for retained users
plt.hist(df_cleaned[df_cleaned['label'] == 'retained']['activity_days'],
        bins=30, alpha=0.5, label='Retained', color='blue')

# Histogram for churned users
plt.hist(df_cleaned[df_cleaned['label'] == 'churned']['activity_days'],
        bins=30, alpha=0.5, label='Churned', color='red')

plt.title('Distribution of Activity Days by User Status')
plt.xlabel('Number of Activity Days')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



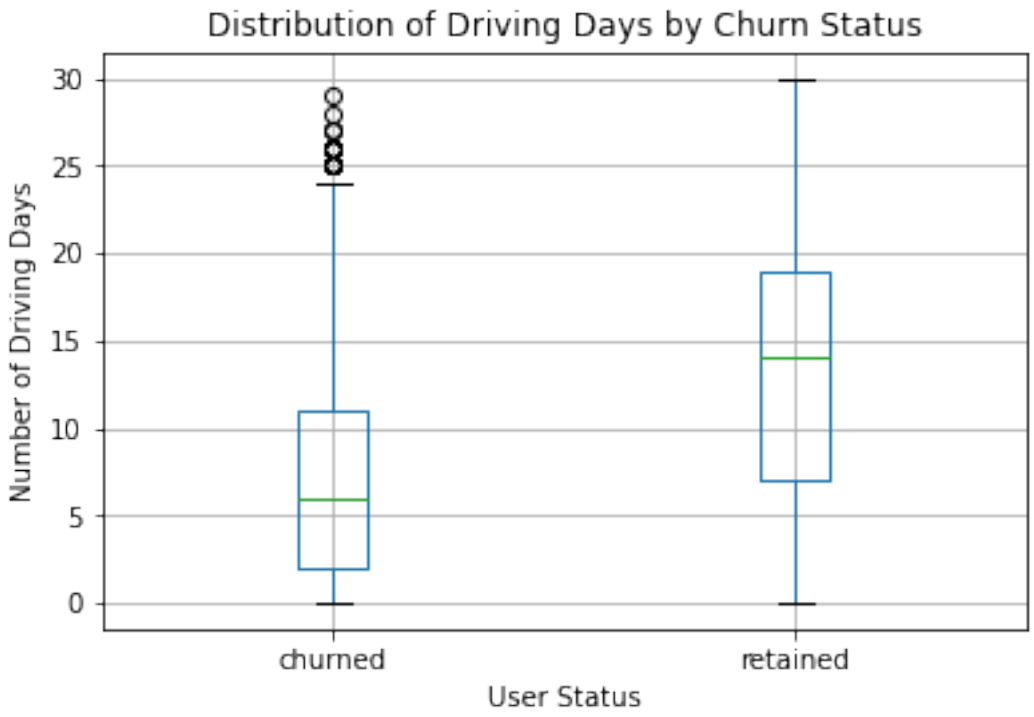
Within the last month, users opened the app a median of 16 times. The box plot reveals a centered distribution. The histogram shows a nearly uniform distribution of ~500 people opening the app on each count of days. However, there are ~250 people who didn't open the app at all and ~250 people who opened the app every day of the month.

This distribution is noteworthy because it does not mirror the `sessions` distribution, which you might think would be closely correlated with `activity_days`.

`driving_days` *Number of days the user drives (at least 1 km) during the month*

[29]:
```
# Box plot
# Box plot for driving_days by churn status
plt.figure(figsize=(10, 6))
df_cleaned.boxplot(column='driving_days', by='label')
plt.title('Distribution of Driving Days by Churn Status')
plt.suptitle('')  # Remove automatic suptitle
plt.xlabel('User Status')
plt.ylabel('Number of Driving Days')
plt.show()
```

```
<Figure size 720x432 with 0 Axes>
```



[ ]:
```
# Histogram
### YOUR CODE HERE ###
```

The number of days users drove each month is almost uniform, and it largely correlates with the number of days they opened the app that month, except the `driving_days` distribution tails off on the right.

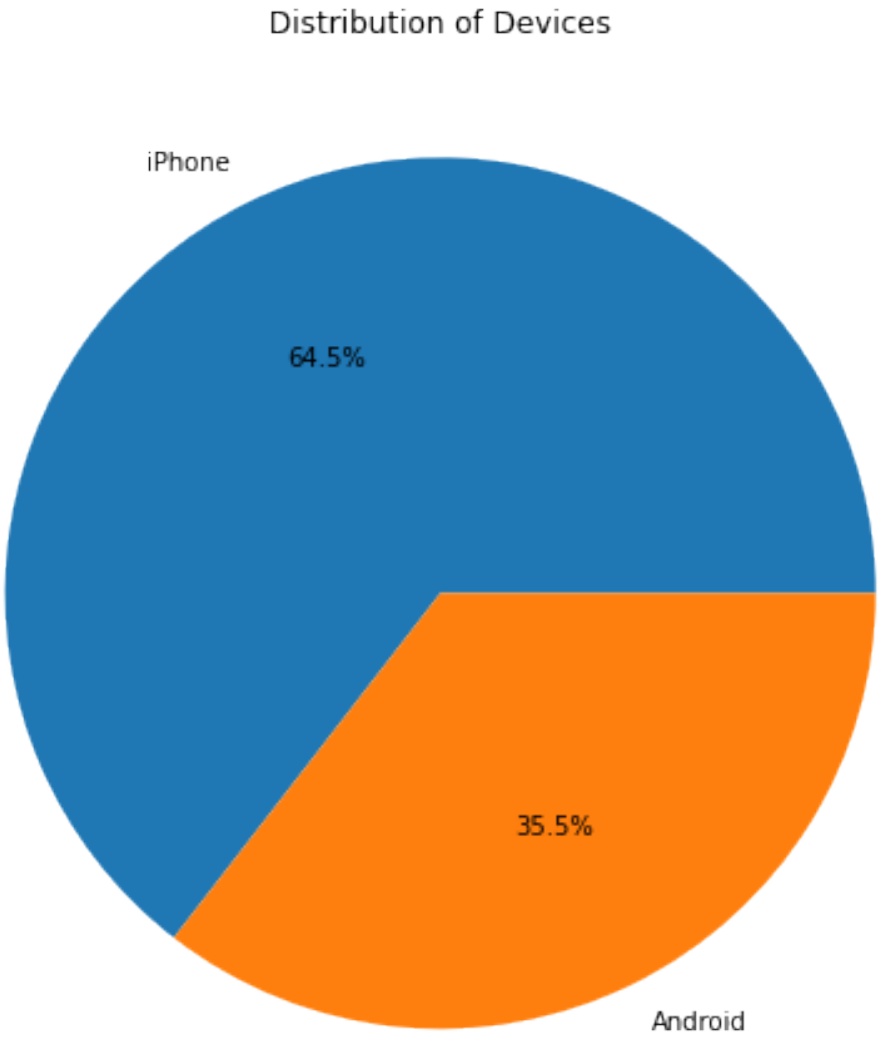
However, there were almost twice as many users (~1,000 vs. ~550) who did not drive at all during the month. This might seem counterintuitive when considered together with the information from `activity_days`. That variable had ~500 users opening the app on each of most of the day counts, but there were only ~250 users who did not open the app at all during the month and ~250 users who opened the app every day. Flag this for further investigation later.

`device` *The type of device a user starts a session with*

This is a categorical variable, so you do not plot a box plot for it. A good plot for a binary categorical variable is a pie chart.

[32]:
```
# Pie chart
# Pie chart for device distribution
plt.figure(figsize=(8, 8))
device_counts = df_cleaned['device'].value_counts()
plt.pie(device_counts, labels=device_counts.index, autopct='%1.1f%%')
plt.title('Distribution of Devices')
```

```
plt.show()

# Detailed device distribution by churn status
print("\nDevice Distribution:")
print(df_cleaned.groupby(['label', 'device']).size())
print("\nDevice Distribution Percentages:")
print(df_cleaned.groupby(['label', 'device']).size().groupby(level=0).
 ↪apply(lambda x: 100 * x / x.sum()).round(2))
```




```
Device Distribution:
label     device
```

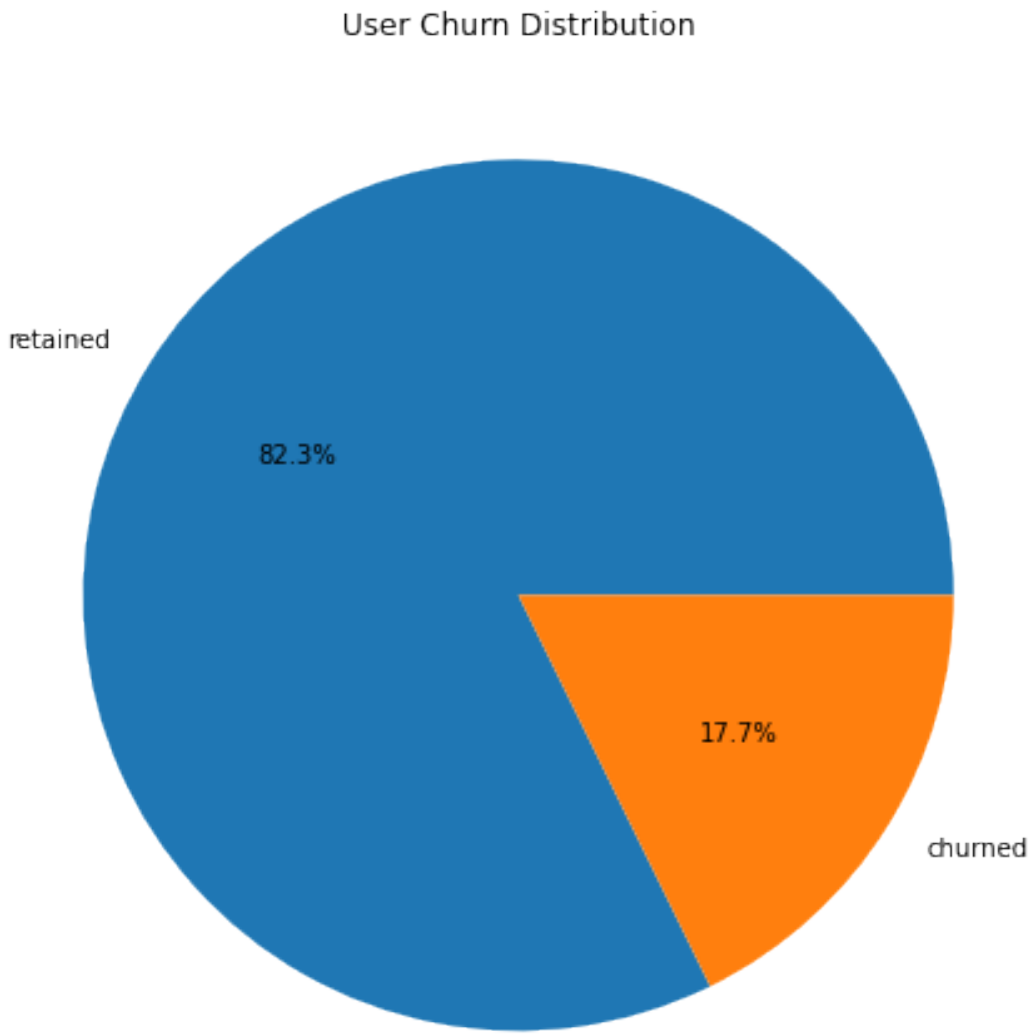
```
churned   Android     891
          iPhone     1645
retained  Android    4183
          iPhone     7580
dtype: int64

Device Distribution Percentages:
label     device
churned   Android    35.13
          iPhone     64.87
retained  Android    35.56
          iPhone     64.44
dtype: float64
```

There are nearly twice as many iPhone users as Android users represented in this data.

label *Binary target variable ("retained" vs "churned") for if a user has churned anytime during the course of the month*

This is also a categorical variable, and as such would not be plotted as a box plot. Plot a pie chart instead.

[33]:
```
# Pie chart
# Pie chart for churn distribution
plt.figure(figsize=(8, 8))
label_counts = df_cleaned['label'].value_counts()
plt.pie(label_counts, labels=label_counts.index, autopct='%1.1f%%')
plt.title('User Churn Distribution')
plt.show()

# Detailed churn percentages
print("\nChurn Distribution:")
print(label_counts)
print("\nChurn Percentages:")
print(label_counts / len(df_cleaned) * 100)
```



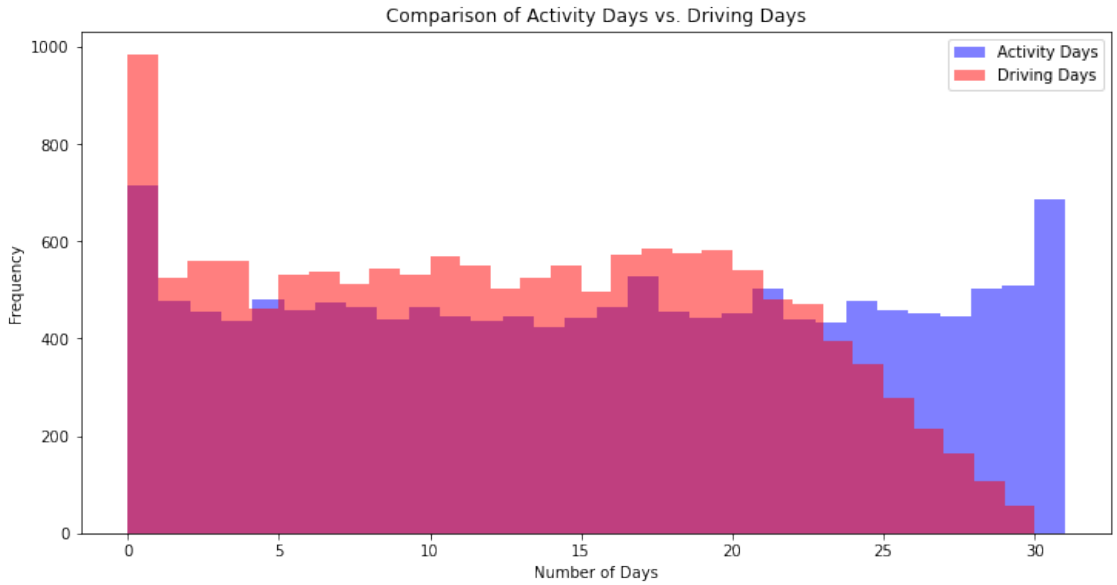

```
Churn Distribution:
retained    11763
churned      2536
Name: label, dtype: int64

Churn Percentages:
retained    82.264494
churned     17.735506
Name: label, dtype: float64
```

Less than 18% of the users churned.

`driving_days` vs. `activity_days` Because both `driving_days` and `activity_days` represent counts of days over a month and they're also closely related, you can plot them together on a single histogram. This will help to better understand how they relate to each other without having to scroll back and forth comparing histograms in two different places.

Plot a histogram that, for each day, has a bar representing the counts of `driving_days` and `activity_days`.

[34]:
```
# Histogram
# Histogram comparing driving_days and activity_days
plt.figure(figsize=(12, 6))

# Histogram for activity_days
plt.hist(df_cleaned['activity_days'], bins=30, alpha=0.5, label='Activity␣
↪Days', color='blue')

# Histogram for driving_days
plt.hist(df_cleaned['driving_days'], bins=30, alpha=0.5, label='Driving Days',␣
↪color='red')

plt.title('Comparison of Activity Days vs. Driving Days')
plt.xlabel('Number of Days')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# Descriptive statistics for comparison
print("\nComparison of Activity and Driving Days:")
print("Activity Days Summary:")
print(df_cleaned['activity_days'].describe())
print("\nDriving Days Summary:")
print(df_cleaned['driving_days'].describe())

# Calculate correlation
correlation = df_cleaned['activity_days'].corr(df_cleaned['driving_days'])
print(f"\nCorrelation between Activity Days and Driving Days: {correlation:.
↪4f}")
```




```
Comparison of Activity and Driving Days:
Activity Days Summary:
count    14299.000000
mean        15.544653
std          9.016088
min          0.000000
25%          8.000000
50%         16.000000
75%         23.000000
max         31.000000
Name: activity_days, dtype: float64

Driving Days Summary:
count    14299.000000
mean        12.182530
std          7.833835
min          0.000000
25%          5.000000
50%         12.000000
75%         19.000000
max         30.000000
Name: driving_days, dtype: float64

Correlation between Activity Days and Driving Days: 0.9477
```

As observed previously, this might seem counterintuitive. After all, why are there *fewer* people who didn't use the app at all during the month and *more* people who didn't drive at all during the

month?

On the other hand, it could just be illustrative of the fact that, while these variables are related to each other, they're not the same. People probably just open the app more than they use the app to drive—perhaps to check drive times or route information, to update settings, or even just by mistake.

Nonetheless, it might be worthwile to contact the data team at Waze to get more information about this, especially because it seems that the number of days in the month is not the same between variables.

Confirm the maximum number of days for each variable—`driving_days` and `activity_days`.

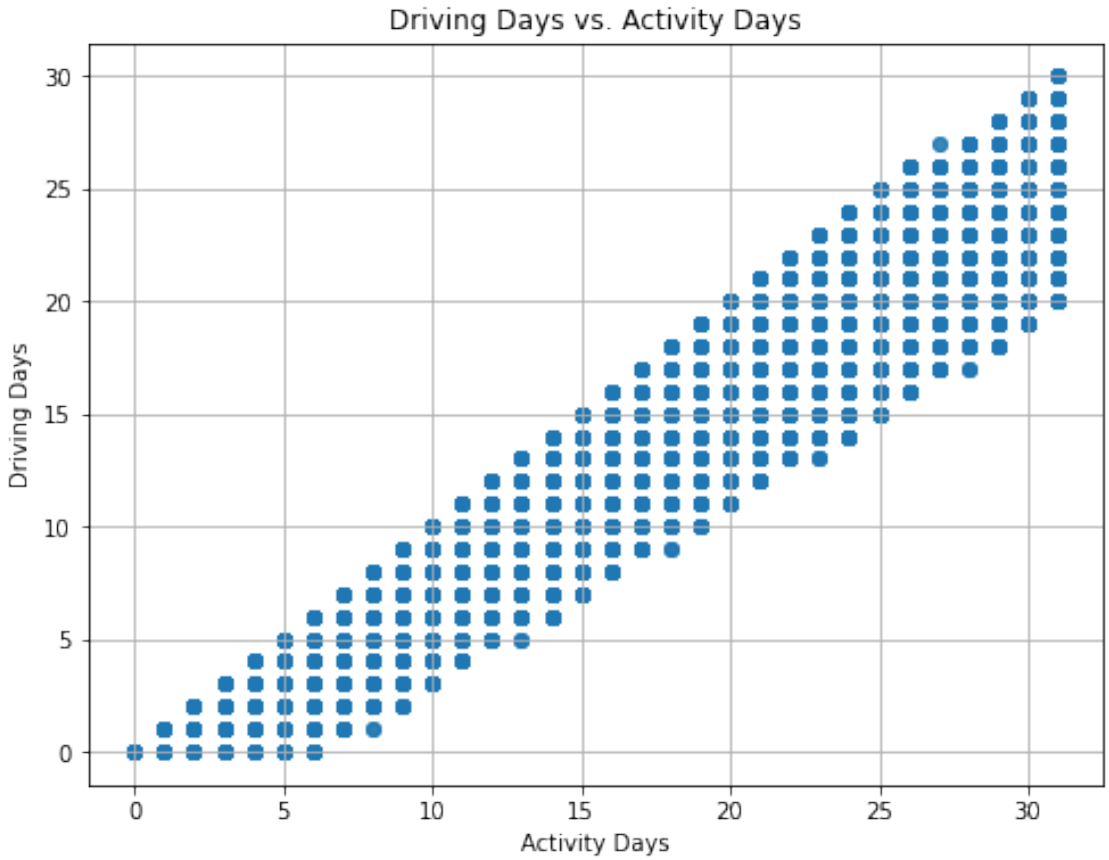
[35]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming `df` is the loaded dataset

# Check the maximum number of days for `driving_days` and `activity_days`
max_driving_days = df['driving_days'].max()
max_activity_days = df['activity_days'].max()

print(f"Maximum driving days: {max_driving_days}")
print(f"Maximum activity days: {max_activity_days}")
```

```
Maximum driving days: 30
Maximum activity days: 31
```

It's true. Although it's possible that not a single user drove all 31 days of the month, it's highly unlikely, considering there are 15,000 people represented in the dataset.

One other way to check the validity of these variables is to plot a simple scatter plot with the x-axis representing one variable and the y-axis representing the other.

[36]:
```
# Scatter plot
# Plotting a scatter plot to visualize the relationship
plt.figure(figsize=(8, 6))
plt.scatter(df['activity_days'], df['driving_days'], alpha=0.5)
plt.title('Driving Days vs. Activity Days')
plt.xlabel('Activity Days')
plt.ylabel('Driving Days')
plt.grid(True)
plt.show()
```

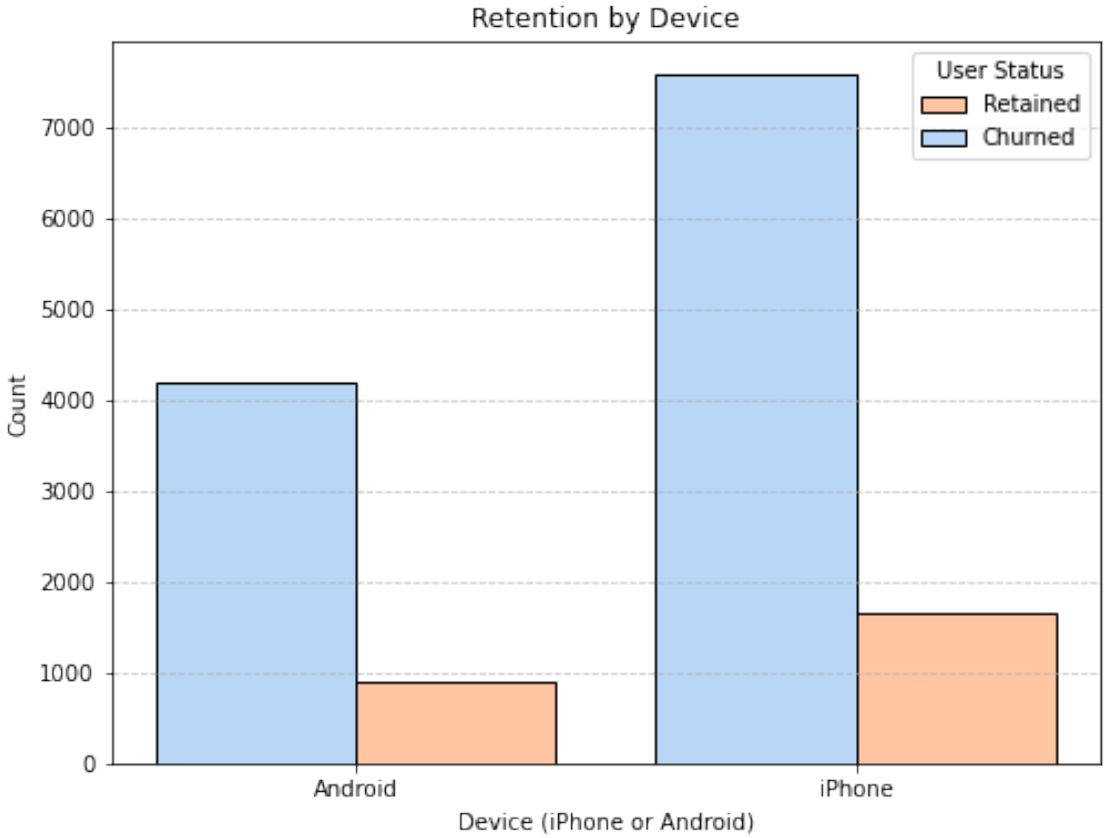


Notice that there is a theoretical limit. If you use the app to drive, then by definition it must count as a day-use as well. In other words, you cannot have more drive-days than activity-days. None of the samples in this data violate this rule, which is good.

Retention by device Plot a histogram that has four bars—one for each device-label combination—to show how many iPhone users were retained/churned and how many Android users were retained/churned.

[37]:
```
# Histogram
import seaborn as sns

# Creating a histogram for retention by device
plt.figure(figsize=(8, 6))
sns.histplot(
    data=df,
    x="device",
    hue="label",
    multiple="dodge",
    shrink=0.8,
```

```
    palette="pastel",
    stat="count"
)
plt.title("Retention by Device")
plt.xlabel("Device (iPhone or Android)")
plt.ylabel("Count")
plt.legend(title="User Status", labels=["Retained", "Churned"])
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



The proportion of churned users to retained users is consistent between device types.

Retention by kilometers driven per driving day In the previous course, you discovered that the median distance driven per driving day last month for users who churned was 697.54 km, versus 289.55 km for people who did not churn. Examine this further.

1. Create a new column in `df` called `km_per_driving_day`, which represents the mean distance driven per driving day for each user.
2. Call the `describe()` method on the new column.

[38]:
```
# 1. Create `km_per_driving_day` column
# Create a new column `km_per_driving_day`
df['km_per_driving_day'] = df['driven_km_drives'] / df['driving_days']

# Handle division by zero or NaN values
df['km_per_driving_day'].replace([float('inf'), -float('inf')], 0, inplace=True)
df['km_per_driving_day'].fillna(0, inplace=True)


# 2. Call `describe()` on the new column
# Generate summary statistics for `km_per_driving_day`
km_per_driving_day_stats = df['km_per_driving_day'].describe()

print(km_per_driving_day_stats)
```

```
count    14999.000000
mean       578.963113
std       1030.094384
min          0.000000
25%        136.238895
50%        272.889272
75%        558.686918
max      15420.234110
Name: km_per_driving_day, dtype: float64
```

What do you notice? The mean value is infinity, the standard deviation is NaN, and the max value is infinity. Why do you think this is?

This is the result of there being values of zero in the `driving_days` column. Pandas imputes a value of infinity in the corresponding rows of the new column because division by zero is undefined.

1. Convert these values from infinity to zero. You can use `np.inf` to refer to a value of infinity.
2. Call `describe()` on the `km_per_driving_day` column to verify that it worked.

[41]:
```
# 1. Convert infinite values to zero
import numpy as np

# Replace infinity values with zero
df['km_per_driving_day'].replace([np.inf, -np.inf], 0, inplace=True)


# 2. Confirm that it worked
# Call `describe()` on the corrected `km_per_driving_day` column
corrected_stats = df['km_per_driving_day'].describe()

print(corrected_stats)
```

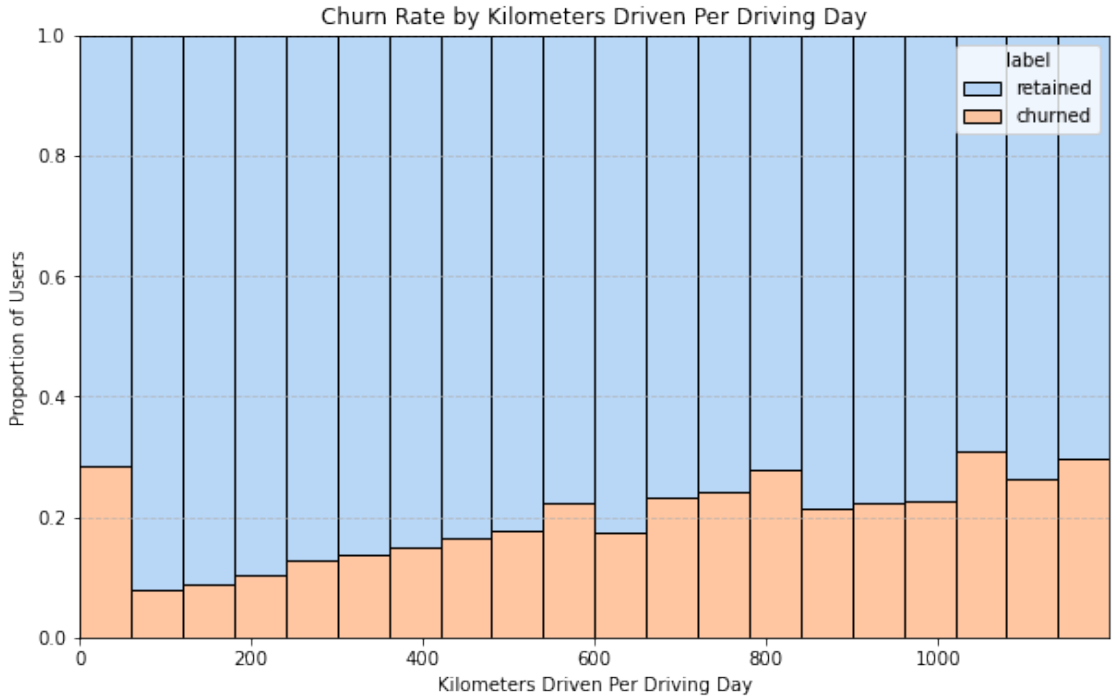
```
count    14999.000000
mean       578.963113
```

```
std       1030.094384
min          0.000000
25%        136.238895
50%        272.889272
75%        558.686918
max      15420.234110
Name: km_per_driving_day, dtype: float64
```

The maximum value is 15,420 kilometers *per drive day*. This is physically impossible. Driving 100 km/hour for 12 hours is 1,200 km. It's unlikely many people averaged more than this each day they drove, so, for now, disregard rows where the distance in this column is greater than 1,200 km.

Plot a histogram of the new `km_per_driving_day` column, disregarding those users with values greater than 1,200 km. Each bar should be the same length and have two colors, one color representing the percent of the users in that bar that churned and the other representing the percent that were retained. This can be done by setting the `multiple` parameter of seaborn's `histplot()` function to `fill`.

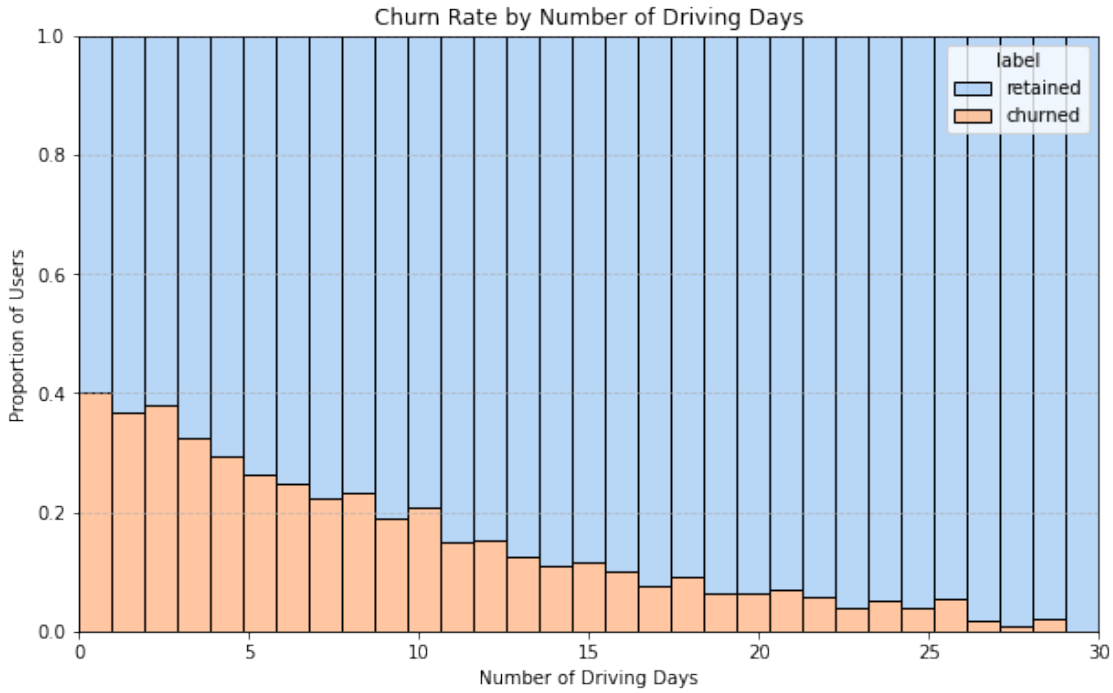
[42]:
```
# Histogram
import seaborn as sns
import matplotlib.pyplot as plt

# Filter the dataset to exclude rows where `km_per_driving_day` > 1200
filtered_df = df[df['km_per_driving_day'] <= 1200]

# Plot the histogram with churn/retention breakdown
plt.figure(figsize=(10, 6))
sns.histplot(
    data=filtered_df,
    x='km_per_driving_day',
    hue='label',  # Differentiates churned vs retained
    multiple='fill',  # Ensures bars are filled proportionally
    bins=20,  # Adjust the number of bins for clarity
    palette='pastel'
)
plt.title('Churn Rate by Kilometers Driven Per Driving Day')
plt.xlabel('Kilometers Driven Per Driving Day')
plt.ylabel('Proportion of Users')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



The churn rate tends to increase as the mean daily distance driven increases, confirming what was found in the previous course. It would be worth investigating further the reasons for long-distance users to discontinue using the app.

Churn rate per number of driving days Create another histogram just like the previous one, only this time it should represent the churn rate for each number of driving days.

[43]:
```
# Histogram
# Plot the histogram with churn/retention breakdown for driving days
plt.figure(figsize=(10, 6))
sns.histplot(
    data=df,
    x='driving_days',
    hue='label',  # Differentiates churned vs retained
    multiple='fill',  # Ensures bars are filled proportionally
    bins=31,  # One bin per possible number of driving days in a month
    palette='pastel'
)
plt.title('Churn Rate by Number of Driving Days')
plt.xlabel('Number of Driving Days')
plt.ylabel('Proportion of Users')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



The churn rate is highest for people who didn't use Waze much during the last month. The more times they used the app, the less likely they were to churn. While 40% of the users who didn't use the app at all last month churned, nobody who used the app 30 days churned.

This isn't surprising. If people who used the app a lot churned, it would likely indicate dissatisfaction. When people who don't use the app churn, it might be the result of dissatisfaction in the past, or it might be indicative of a lesser need for a navigational app. Maybe they moved to a city with good public transportation and don't need to drive anymore.

Proportion of sessions that occurred in the last month Create a new column `percent_sessions_in_last_month` that represents the percentage of each user's total sessions that were logged in their last month of use.

[44]:
```
# Create the new column for percentage of sessions in the last month
df['percent_sessions_in_last_month'] = (df['sessions'] / df['total_sessions'])␣
↪* 100
```

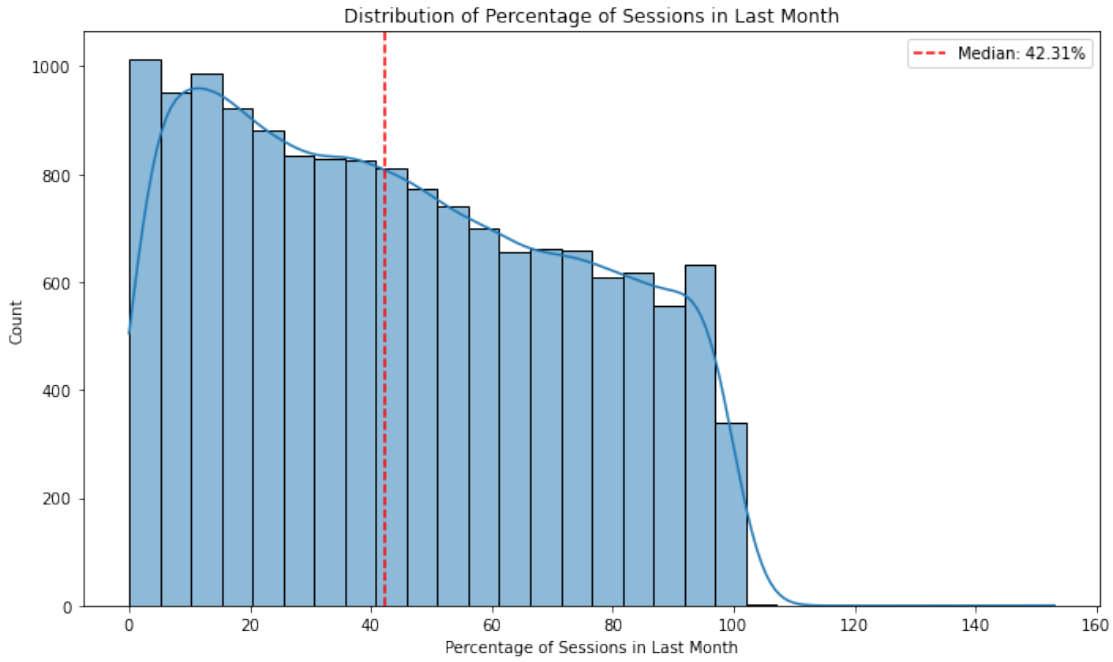
What is the median value of the new column?

[45]:
```
# Calculate the median value of the new column
median_percent_sessions = df['percent_sessions_in_last_month'].median()

print(f"The median percentage of sessions in the last month is␣
↪{median_percent_sessions:.2f}%")
```

```
The median percentage of sessions in the last month is 42.31%
```

Now, create a histogram depicting the distribution of values in this new column.

[46]:
```
# Histogram
import matplotlib.pyplot as plt
import seaborn as sns

# Create the histogram
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='percent_sessions_in_last_month', bins=30, kde=True)

# Customize the plot
plt.title('Distribution of Percentage of Sessions in Last Month')
plt.xlabel('Percentage of Sessions in Last Month')
plt.ylabel('Count')

# Add a vertical line at the median
median_value = df['percent_sessions_in_last_month'].median()
plt.axvline(x=median_value, color='red', linestyle='--', label=f'Median:␣
↪{median_value:.2f}%')

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



Check the median value of the `n_days_after_onboarding` variable.

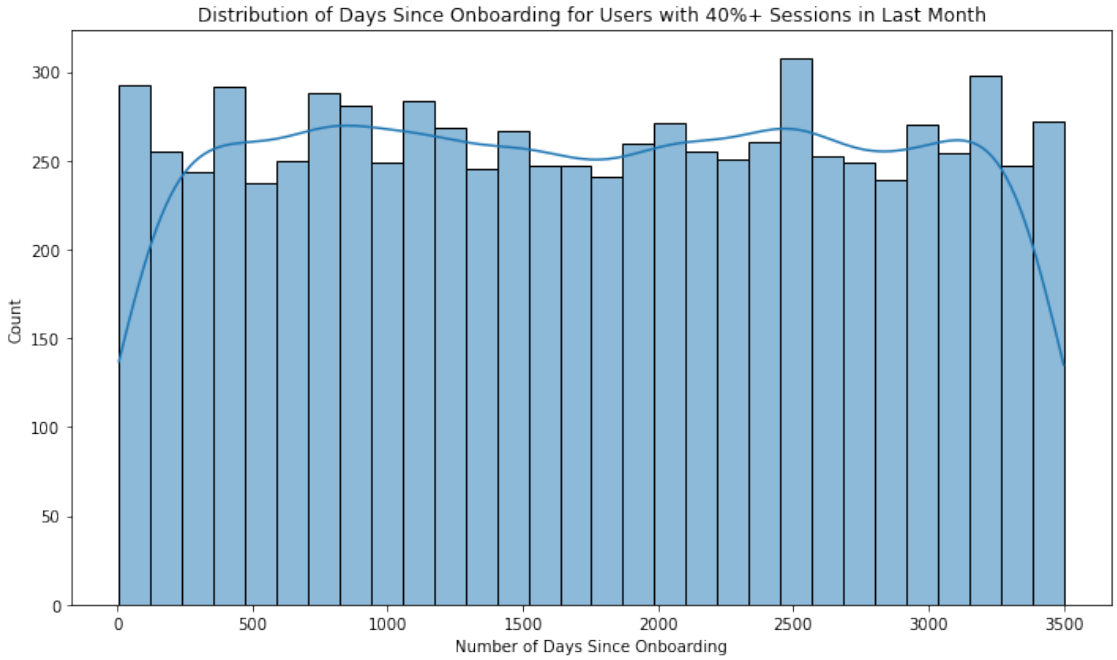
[47]:
```
median_days = df['n_days_after_onboarding'].median()
print(f"The median value of n_days_after_onboarding is: {median_days:.2f} days")
```

```
The median value of n_days_after_onboarding is: 1741.00 days
```

Half of the people in the dataset had 40% or more of their sessions in just the last month, yet the overall median time since onboarding is almost five years.

Make a histogram of `n_days_after_onboarding` for just the people who had 40% or more of their total sessions in the last month.

[48]:
```
# Histogram
import matplotlib.pyplot as plt
import seaborn as sns

# Filter the DataFrame for users with 40% or more sessions in the last month
filtered_df = df[df['percent_sessions_in_last_month'] >= 40]

# Create the histogram
plt.figure(figsize=(10, 6))
sns.histplot(data=filtered_df, x='n_days_after_onboarding', bins=30, kde=True)

# Customize the plot
plt.title('Distribution of Days Since Onboarding for Users with 40%+ Sessions␣
 ↪in Last Month')
plt.xlabel('Number of Days Since Onboarding')
plt.ylabel('Count')

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



The number of days since onboarding for users with 40% or more of their total sessions occurring in just the last month is a uniform distribution. This is very strange. It's worth asking Waze why so many long-time users suddenly used the app so much in the last month.

4.3.2 Task 3b. Handling outliers

The box plots from the previous section indicated that many of these variables have outliers. These outliers do not seem to be data entry errors; they are present because of the right-skewed distributions.

Depending on what you'll be doing with this data, it may be useful to impute outlying data with more reasonable values. One way of performing this imputation is to set a threshold based on a percentile of the distribution.

To practice this technique, write a function that calculates the 95th percentile of a given column, then imputes values $>$ the 95th percentile with the value at the 95th percentile. such as the 95th percentile of the distribution.

[49]:
```
import pandas as pd
import numpy as np

def impute_outliers(df, column):
    percentile_95 = np.percentile(df[column], 95)
    df[column] = df[column].apply(lambda x: min(x, percentile_95))
    return df
```

Next, apply that function to the following columns: * `sessions` * `drives` * `total_sessions` * `driven_km_drives` * `duration_minutes_drives`

[50]:
```
columns_to_impute = ['sessions', 'drives', 'total_sessions',␣
↪'driven_km_drives', 'duration_minutes_drives']

for column in columns_to_impute:
    df = impute_outliers(df, column)
```

Call `describe()` to see if your change worked.

[51]:
```
columns_to_check = ['sessions', 'drives', 'total_sessions', 'driven_km_drives',␣
↪'duration_minutes_drives']

for column in columns_to_check:
    print(f"\nDescriptive statistics for {column}:")
    print(df[column].describe())
```

```
Descriptive statistics for sessions:
count    14999.000000
mean        76.568705
std         67.297958
min          0.000000
25%         23.000000
50%         56.000000
75%        112.000000
max        243.000000
Name: sessions, dtype: float64

Descriptive statistics for drives:
count    14999.000000
mean        64.058204
std         55.306924
min          0.000000
25%         20.000000
50%         48.000000
75%         93.000000
max        201.000000
Name: drives, dtype: float64

Descriptive statistics for total_sessions:
count    14999.000000
mean       184.031320
std        118.600463
min          0.220211
25%         90.661156
50%        159.568115
```

```
75%        254.192341
max        454.363204
Name: total_sessions, dtype: float64

Descriptive statistics for driven_km_drives:
count    14999.000000
mean      3939.632764
std       2216.041510
min         60.441250
25%       2212.600607
50%       3493.858085
75%       5289.861262
max       8889.794236
Name: driven_km_drives, dtype: float64

Descriptive statistics for duration_minutes_drives:
count    14999.000000
mean      1789.647426
std       1222.705167
min         18.282082
25%        835.996260
50%       1478.249859
75%       2464.362632
max       4668.899349
Name: duration_minutes_drives, dtype: float64
```

Conclusion Analysis revealed that the overall churn rate is ~17%, and that this rate is consistent between iPhone users and Android users.

Perhaps you feel that the more deeply you explore the data, the more questions arise. This is not uncommon! In this case, it's worth asking the Waze data team why so many users used the app so much in just the last month.

Also, EDA has revealed that users who drive very long distances on their driving days are *more* likely to churn, but users who drive more often are *less* likely to churn. The reason for this discrepancy is an opportunity for further investigation, and it would be something else to ask the Waze data team about.

4.4 PACE: Execute

Consider the questions in your PACE Strategy Document to reflect on the Execute stage.

4.4.1 Task 4a. Results and evaluation

Having built visualizations in Python, what have you learned about the dataset? What other questions have your visualizations uncovered that you should pursue?

Pro tip: Put yourself in your client's perspective. What would they want to know?

Use the following code fields to pursue any additional EDA based on the visualizations you've already plotted. Also use the space to make sure your visualizations are clean, easily understandable, and accessible.

Ask yourself: Did you consider color, contrast, emphasis, and labeling?

I have learned: -The overall churn rate is approximately 17%, consistent across both iPhone and Android users. -User behavior exhibits right-skewed distributions for variables such as sessions, drives, and total_sessions, indicating that a small number of users are highly active. -There's a surprising pattern where half of the users had 40% or more of their total sessions in just the last month, despite a median user tenure of almost five years. -Users who drive very long distances on their driving days are more likely to churn, while users who drive more frequently are less likely to churn. -The churn rate is highest for users who rarely used the app in the last month, with 40% of non-users churning compared to 0% for daily users.

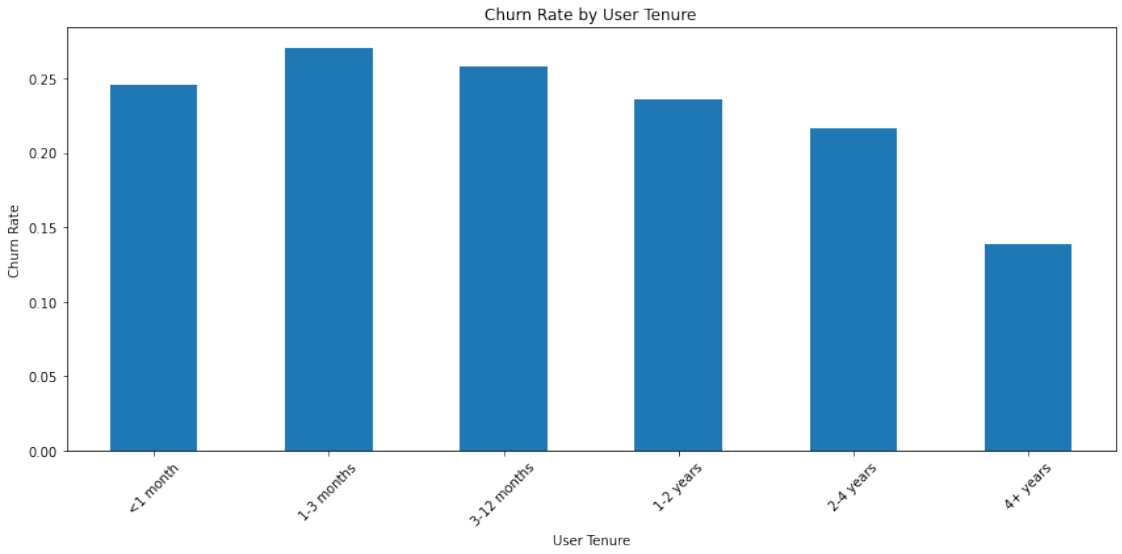
My other questions are: -Why did so many long-time users suddenly increase their app usage in the last month? -What factors contribute to the higher churn rate among users who drive long distances? -Is there a specific threshold of usage frequency that significantly reduces churn risk? -Are there any seasonal patterns in user behavior or churn rates? -How do external factors (e.g., gas prices, traffic patterns) correlate with app usage and churn?

My client would likely want to know: -What specific actions can be taken to reduce churn based on the identified patterns? -Are there any user segments that are particularly at risk of churning? -How can we re-engage users who are showing signs of decreased app usage? -What features or improvements could be implemented to better retain long-distance drivers? -Is there an optimal engagement level that balances user retention with app resources? -How does the churn rate compare to industry standards or competitors? -What is the estimated impact on revenue if we could reduce churn by a certain percentage?

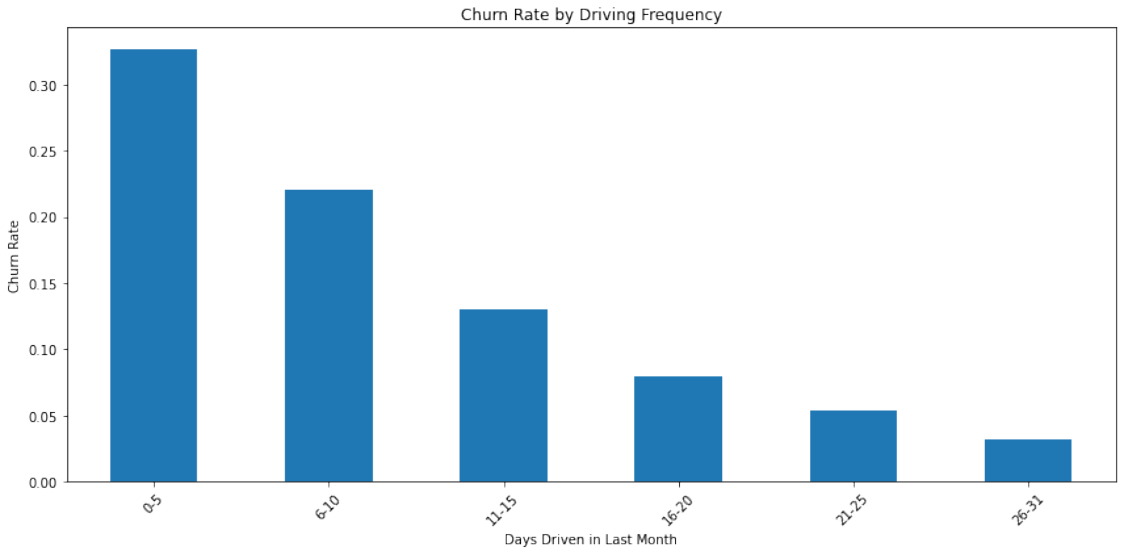
Use the following two code blocks (add more blocks if you like) to do additional EDA you feel is important based on the given scenario.

[53]:
```
# Tenure vs. Churn
import matplotlib.pyplot as plt
import seaborn as sns

# Create tenure bins
df['tenure_bins'] = pd.cut(df['n_days_after_onboarding'],
                           bins=[0, 30, 90, 365, 730, 1460, float('inf')],
                           labels=['<1 month', '1-3 months', '3-12 months',␣
 ↪'1-2 years', '2-4 years', '4+ years'])

# Calculate churn rate for each tenure bin
churn_by_tenure = df.groupby('tenure_bins')['label'].
 ↪value_counts(normalize=True).unstack()

# Plot
plt.figure(figsize=(12, 6))
```

```
churn_by_tenure['churned'].plot(kind='bar')
plt.title('Churn Rate by User Tenure')
plt.xlabel('User Tenure')
plt.ylabel('Churn Rate')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[54]:
```
# Drive Frequency vs. Churn
# Create driving frequency bins
df['driving_frequency'] = pd.cut(df['driving_days'],
                                 bins=[0, 5, 10, 15, 20, 25, 31],
                                 labels=['0-5', '6-10', '11-15', '16-20',␣
 ↪'21-25', '26-31'])

# Calculate churn rate for each driving frequency bin
churn_by_frequency = df.groupby('driving_frequency')['label'].
 ↪value_counts(normalize=True).unstack()

# Plot
plt.figure(figsize=(12, 6))
churn_by_frequency['churned'].plot(kind='bar')
plt.title('Churn Rate by Driving Frequency')
plt.xlabel('Days Driven in Last Month')
plt.ylabel('Churn Rate')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



4.4.2 Task 4b. Conclusion

Now that you've explored and visualized your data, the next step is to share your findings with Harriet Hadzic, Waze's Director of Data Analysis. Consider the following questions as you prepare to write your executive summary. Think about key points you may want to share with the team, and what information is most relevant to the user churn project.

Questions:

1. What types of distributions did you notice in the variables? What did this tell you about the data?
2. Was there anything that led you to believe the data was erroneous or problematic in any way?
3. Did your investigation give rise to further questions that you would like to explore or ask the Waze team about?
4. What percentage of users churned and what percentage were retained?
5. What factors correlated with user churn? How?
6. Did newer uses have greater representation in this dataset than users with longer tenure? How do you know?

1. Most variables showed right-skewed distributions, indicating a small number of highly active users. The n_days_after_onboarding variable had a uniform distribution.
2. The driving_days variable had a maximum of 30 days, while activity_days had 31, which seems inconsistent. The km_per_driving_day column initially had infinity values due to division by zero.

3. Why did many long-time users suddenly increase app usage in the last month? What caused the discrepancy between driving_days and activity_days?

4. Approximately 18% of users churned, while 82% were retained.

5. Factors correlated with churn: • Higher churn for users who drove longer distances per day • Lower churn for users who used the app more frequently • Highest churn (40%) for users who didn't use the app at all in the last month

6. The dataset had equal representation of users across different tenure lengths, as evidenced by the uniform distribution of the n_days_after_onboarding variable.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.