Activity_Course 3 Waze project lab

April 5, 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]:
```
### YOUR CODE HERE ###
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import datetime as dt
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. No data is okay.

2) Yes Label is missing 700 rows.

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) Label, driving days, device, total distance.
2) ID, one of the total navigation columns.
3) Do df.info to view missing rows. Missing data you would reassign as NA and exclude from your EDA.
4) Doing describe and viewing any variables that are extreme.

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]:
```
df.head()
```

[3]:
```
   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]:
```
df.size
```

[4]:
```
194987
```

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

[5]:
```
df.describe()
```

[5]:
```
                 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]: df.info()
```

```
<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) Count all null numbers. Use info to see non null and data type and get the max and min values.
2) You first have to ask yourself if these outliers make sense with the data you are working with or if they are of error. This will determine on whether to get rid of the outliers or to keep them.

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

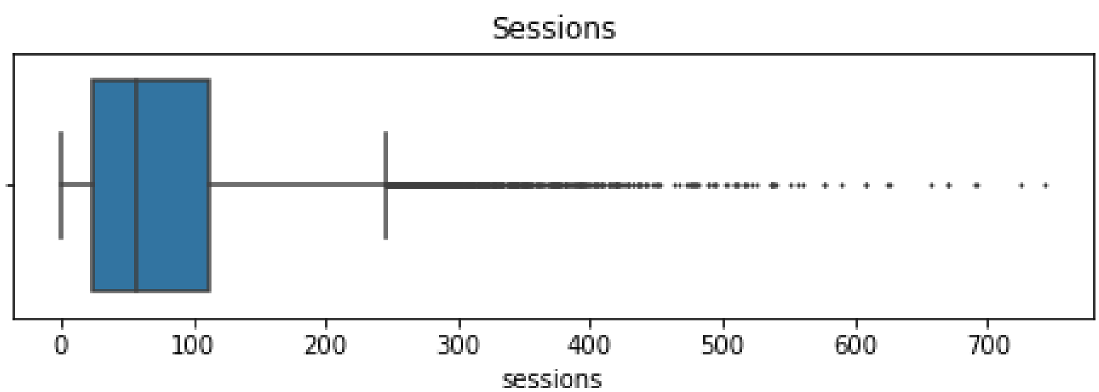
Most important visualizations will be like a Historgram, Box plot, Scatter Plot

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*

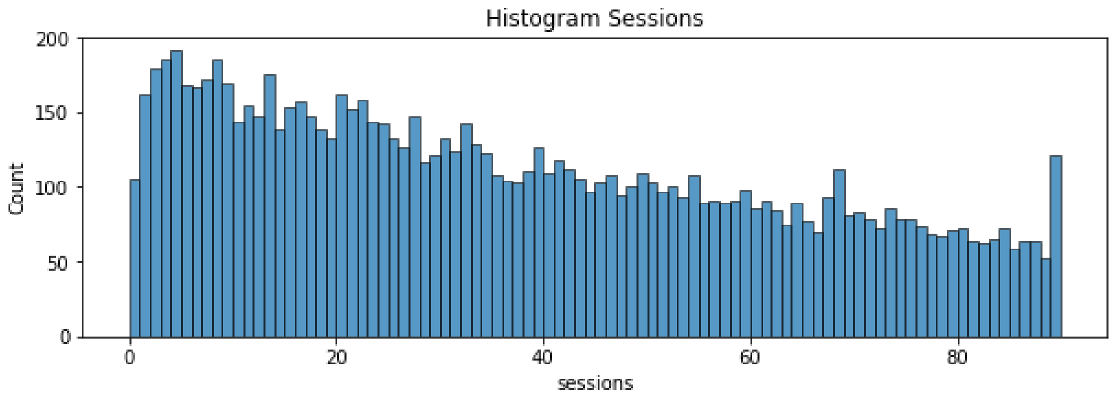
[7]:
```
# Box plot
plt.figure(figsize=(8,2))
plt.title('Sessions')
sns.boxplot(data= None, x=df['sessions'], fliersize = 1)
```

[7]: <matplotlib.axes._subplots.AxesSubplot at 0x704b48635a90>



[8]:
```
# Histogram
plt.figure(figsize=(10,3))
plt.title("Histogram Sessions")
sns.histplot(df['sessions'], bins=range(0,91,1))
```

[8]: <matplotlib.axes._subplots.AxesSubplot at 0x704b4657db50>

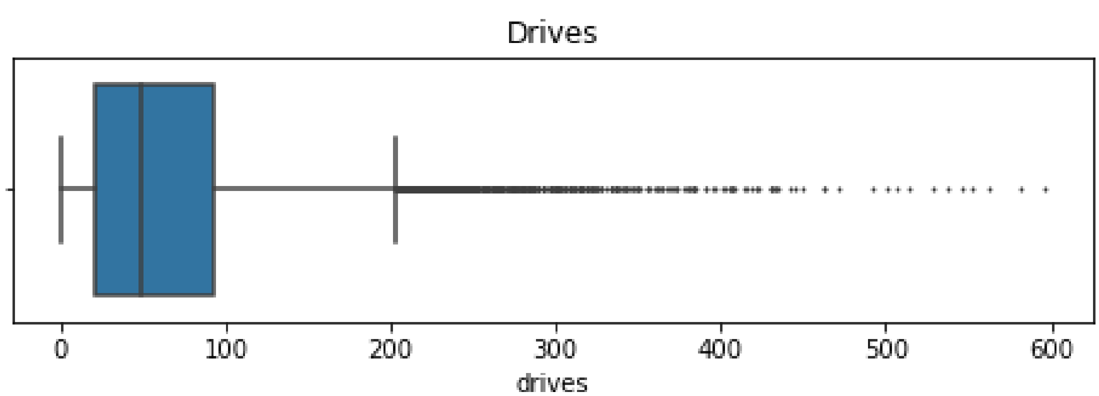


The **sessions** variable is a right-skewed distribution with half of the observations having 56 or fewer sessions. However, as indicated by the boxplot, some users have more than 700.

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

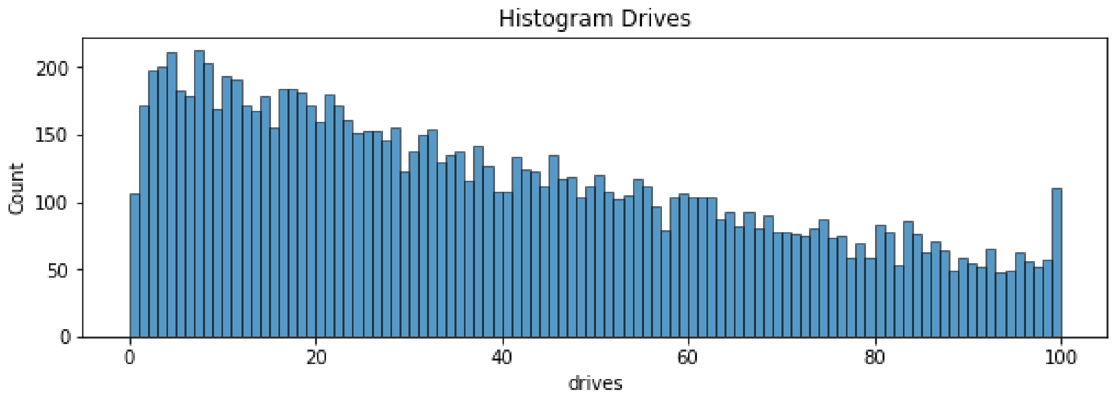
[9]:
```
# Box plot
plt.figure(figsize=(8,2))
plt.title('Drives')
sns.boxplot(data= None, x=df['drives'], fliersize = 1)
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x704b463dfa90>



[10]:
```
# Histogram
plt.figure(figsize=(10,3))
plt.title("Histogram Drives")
sns.histplot(df['drives'], bins=range(0,101,1))
```

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x704b46331e50>

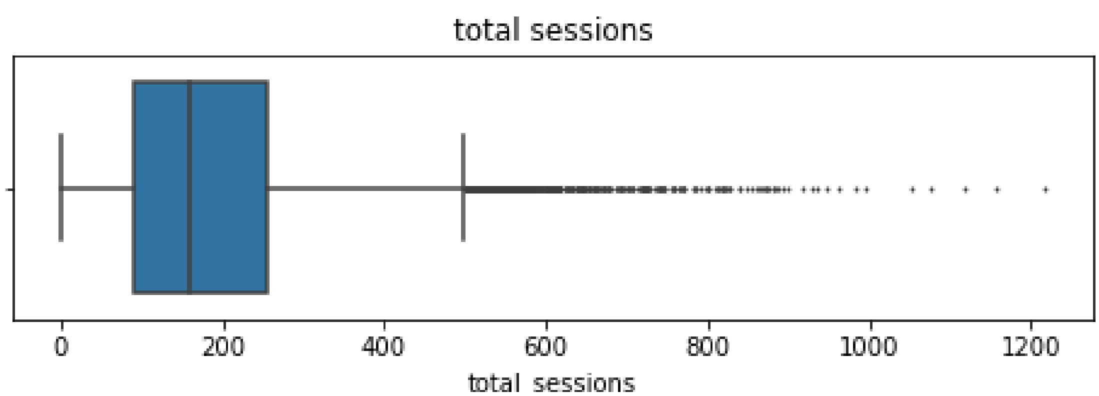


The **drives** information follows a distribution similar to the **sessions** variable. It is right-skewed, approximately log-normal, with a median of 48. However, some drivers had over 400 drives in the last month.

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

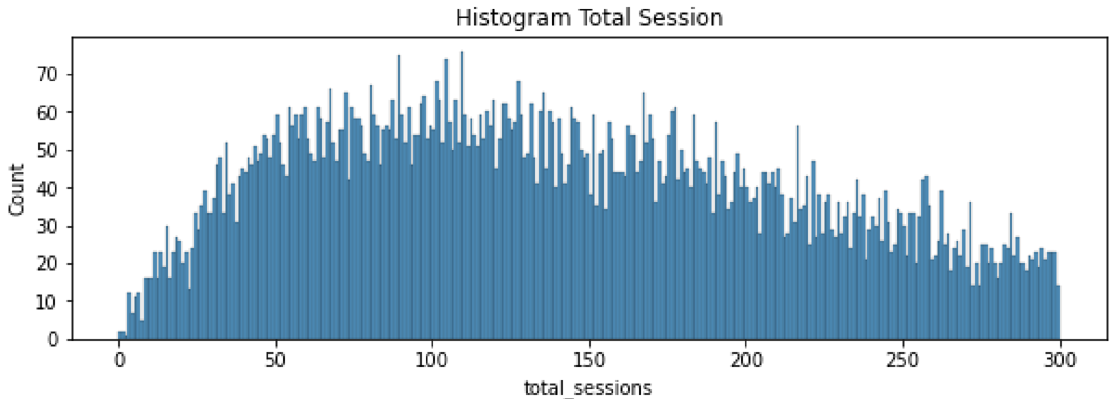
[11]:
```
# Box plot
plt.figure(figsize=(8,2))
plt.title('total sessions')
sns.boxplot(data= None, x=df['total_sessions'], fliersize = 1)
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x704b46365710>
```



[12]:
```
# Histogram
plt.figure(figsize=(10,3))
plt.title("Histogram Total Session")
sns.histplot(df['total_sessions'], bins=range(0,301,1))
```

[12]: <matplotlib.axes._subplots.AxesSubplot at 0x704b4613dd10>

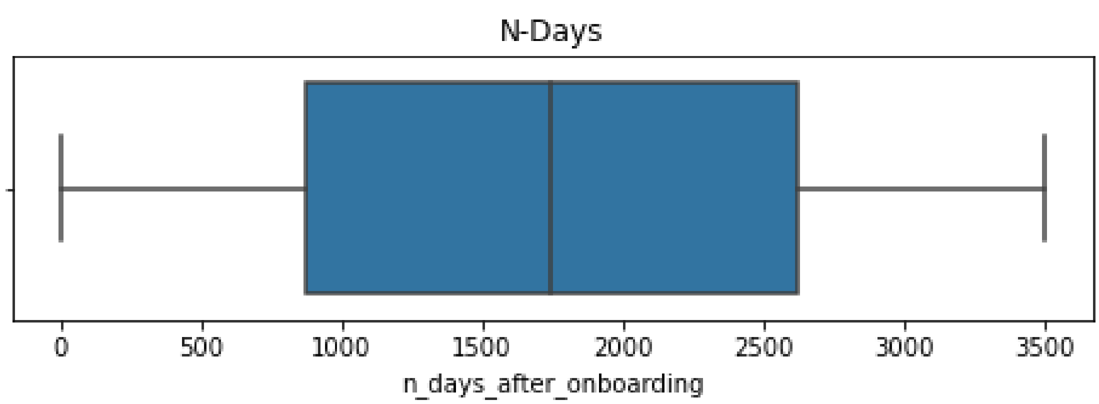


The `total_sessions` is a right-skewed distribution. The median total number of sessions is 159.6. This is interesting information because, if the median number of sessions in the last month was 48 and the median total sessions was ~160, then it seems that a large proportion of a user's total drives 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*

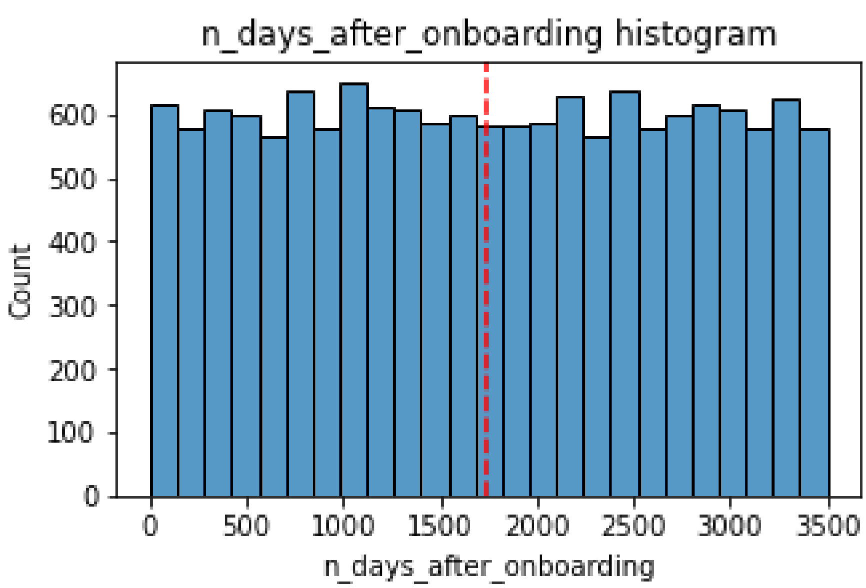
[13]:
```
# Box plot
plt.figure(figsize=(8,2))
plt.title('N-Days')
sns.boxplot(data= None, x=df['n_days_after_onboarding'], fliersize = 1)
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x704b4613d7d0>



[14]:
```
# Helper function to plot histograms based on the
# format of the `sessions` histogram
def histogrammer(column_str, median_text=True, **kwargs):     # **kwargs = any␣
 ↪keyword arguments
                                                              # from the sns.
 ↪histplot() function
    median=round(df[column_str].median(), 1)
    plt.figure(figsize=(5,3))
    ax = sns.histplot(x=df[column_str], **kwargs)             # Plot the␣
 ↪histogram
    plt.axvline(median, color='red', linestyle='--')          # Plot the median␣
 ↪line
    if median_text==True:                                     # Add median text␣
 ↪unless set to False
        ax.text(0.25, 0.85, f'median={median}', color='red',
            ha='left', va='top', transform=ax.transAxes)
    else:
        print('Median:', median)
    plt.title(f'{column_str} histogram');


histogrammer('n_days_after_onboarding', median_text=False)
```

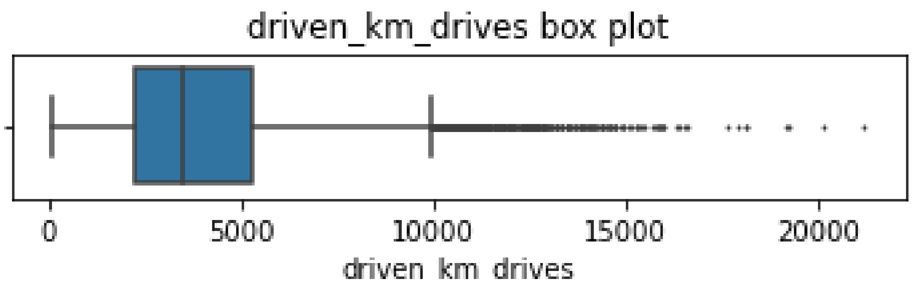
```
Median: 1741.0
```



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).

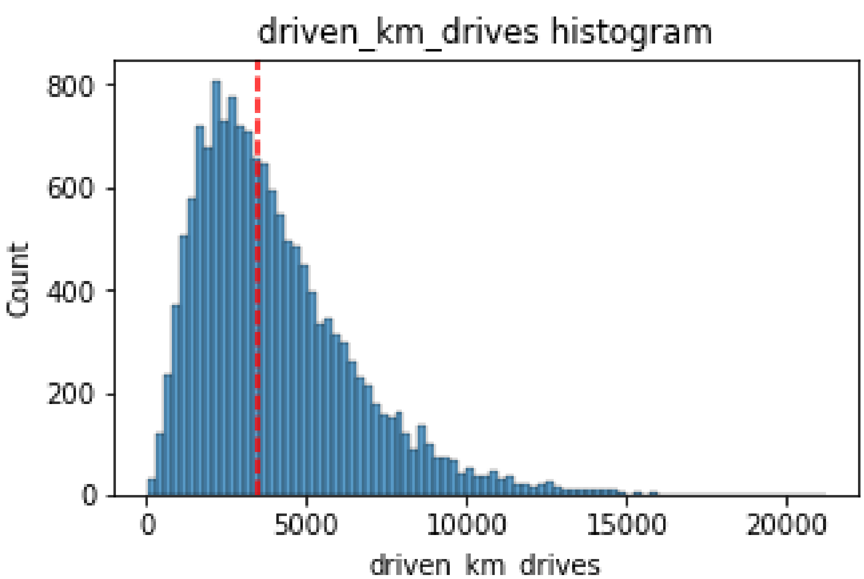
driven_km_drives *Total kilometers driven during the month*

[15]:
```
# Box plot
plt.figure(figsize=(6,1))
sns.boxplot(x=df['driven_km_drives'], fliersize=1)
plt.title('driven_km_drives box plot');
```



[16]:
```
# Histogram
histogrammer('driven_km_drives', median_text=False)
```

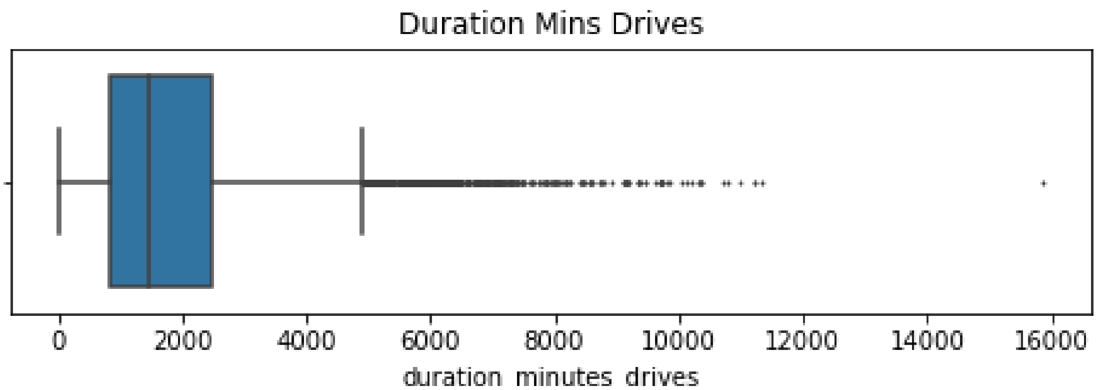
```
Median: 3493.9
```



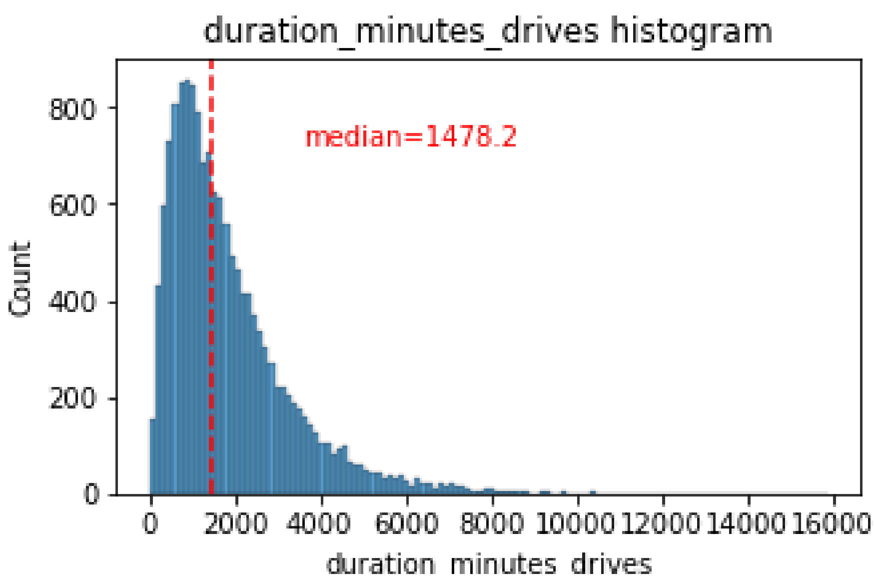
The number of drives driven in the last month per user is a right-skewed distribution with half the users driving under 3,495 kilometers. As you discovered in the analysis from the previous course, the users in this dataset drive *a lot*. The longest distance driven in the month was over half the circumferene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

duration_minutes_drives *Total duration driven in minutes during the month*

[17]:
```
# Box plot
plt.figure(figsize=(8,2))
sns.boxplot(x=df['duration_minutes_drives'], fliersize=1)
plt.title('Duration Mins Drives');
```



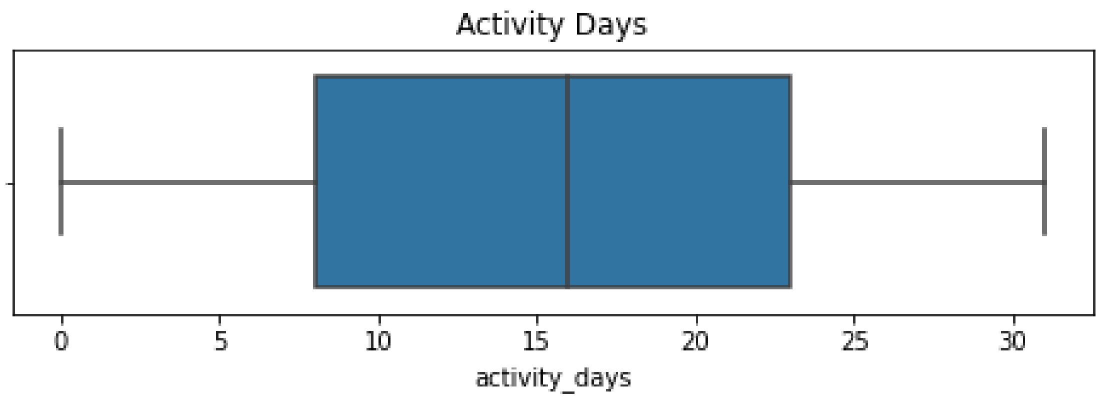
[18]:
```
# Histogram
histogrammer('duration_minutes_drives')
```



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.

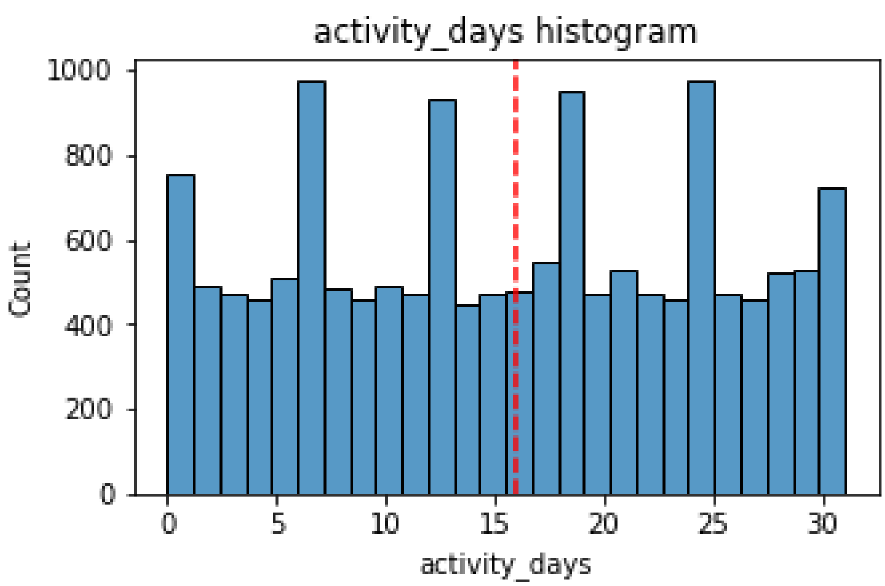
`activity_days` *Number of days the user opens the app during the month*

[19]:
```
# Box plot
plt.figure(figsize=(8,2))
sns.boxplot(x=df['activity_days'], fliersize=1)
plt.title('Activity Days');
```



[20]:
```
# Histogram
histogrammer('activity_days', median_text=False)
```

```
Median: 16.0
```

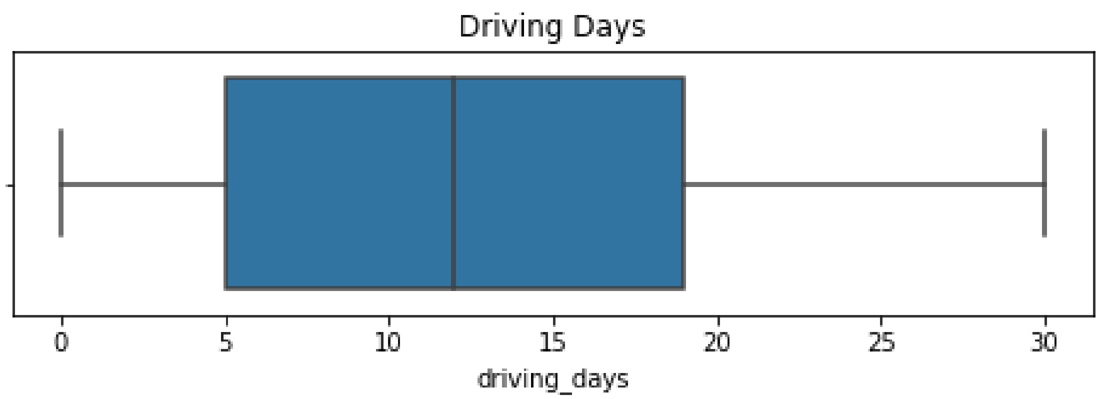


Within the last month, users opened the app a median of 16 times. The box plot reveals a centered distribution. The histogram shows a nearly uniform distribution of ~500 people opening the app on each count of days. However, there are ~250 people who didn't open the app at all and ~250 people who opened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

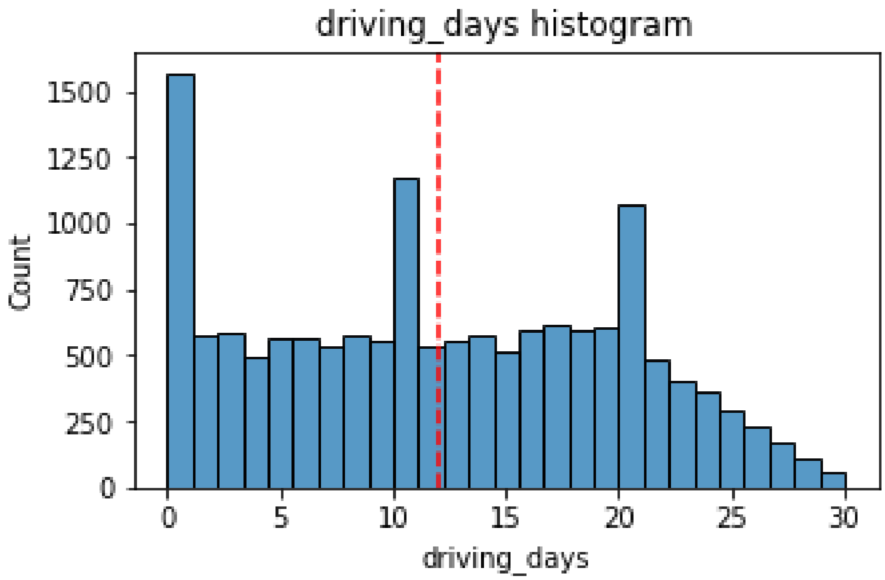
`driving_days` *Number of days the user drives (at least 1 km) during the month*

[21]:
```
# Box plot
plt.figure(figsize=(8,2))
sns.boxplot(x=df['driving_days'], fliersize=1)
plt.title('Driving Days');
```



[22]:
```
# Histogram
histogrammer('driving_days', median_text=False)
```

```
Median: 12.0
```



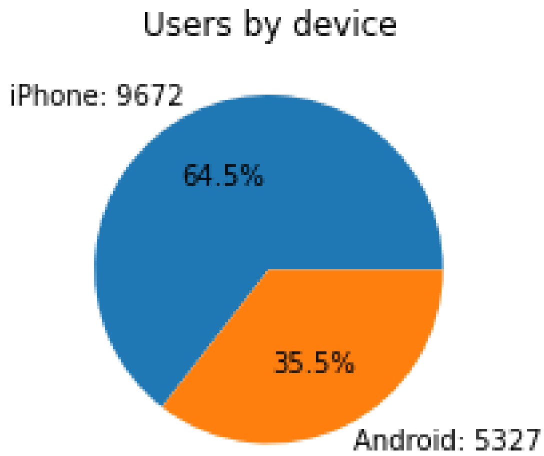
The number of days users drove each month is almost uniform, and it largely correlates with the number of days they opened the app that month, except the `driving_days` distribution tails off on the right.

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.

[24]:
```
# Pie chart
fig = plt.figure(figsize=(4,3))
data=df['device'].value_counts()
plt.pie(data,
        labels=[f'{data.index[0]}: {data.values[0]}',
                f'{data.index[1]}: {data.values[1]}'],
        autopct='%1.1f%%'
        )
plt.title('Users by device');
```

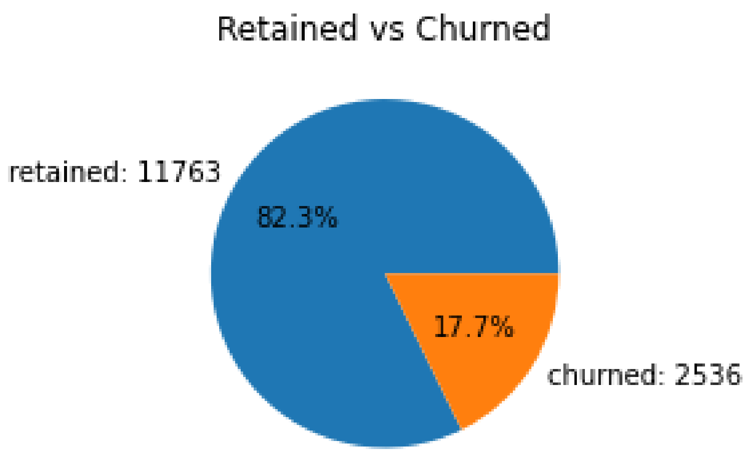


There are nearly twice as many iPhone users as Android users represented in this data.

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.

[25]:
```
# Pie chart
fig = plt.figure(figsize=(4,3))
data=df['label'].value_counts()
plt.pie(data,
        labels=[f'{data.index[0]}: {data.values[0]}',
                f'{data.index[1]}: {data.values[1]}'],
        autopct='%1.1f%%'
        )
plt.title('Retained vs Churned');
```

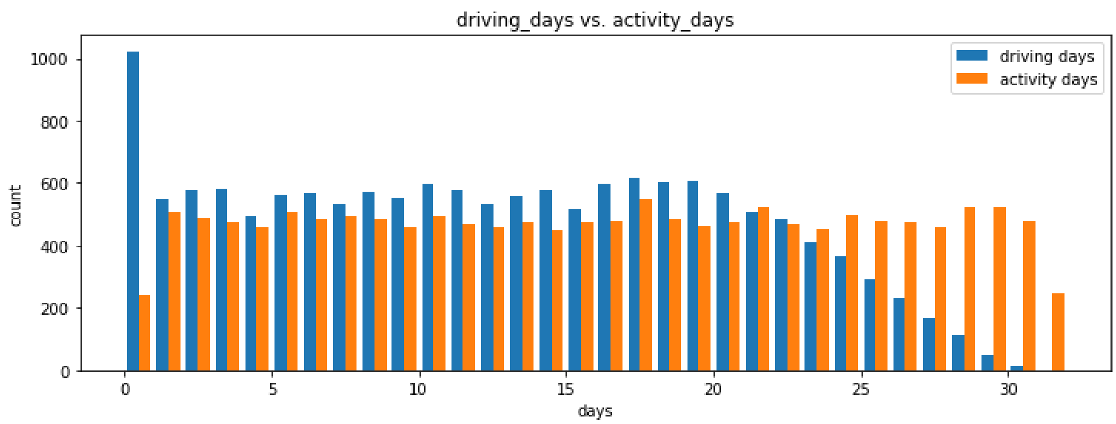


Less than 18% of the users churned.

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`.

[27]:
```
# Histogram
plt.figure(figsize=(12,4))
label=['driving days', 'activity days']
plt.hist([df['driving_days'], df['activity_days']],
         bins=range(0,33),
         label=label)
plt.xlabel('days')
plt.ylabel('count')
plt.legend()
plt.title('driving_days vs. activity_days');
```



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`.

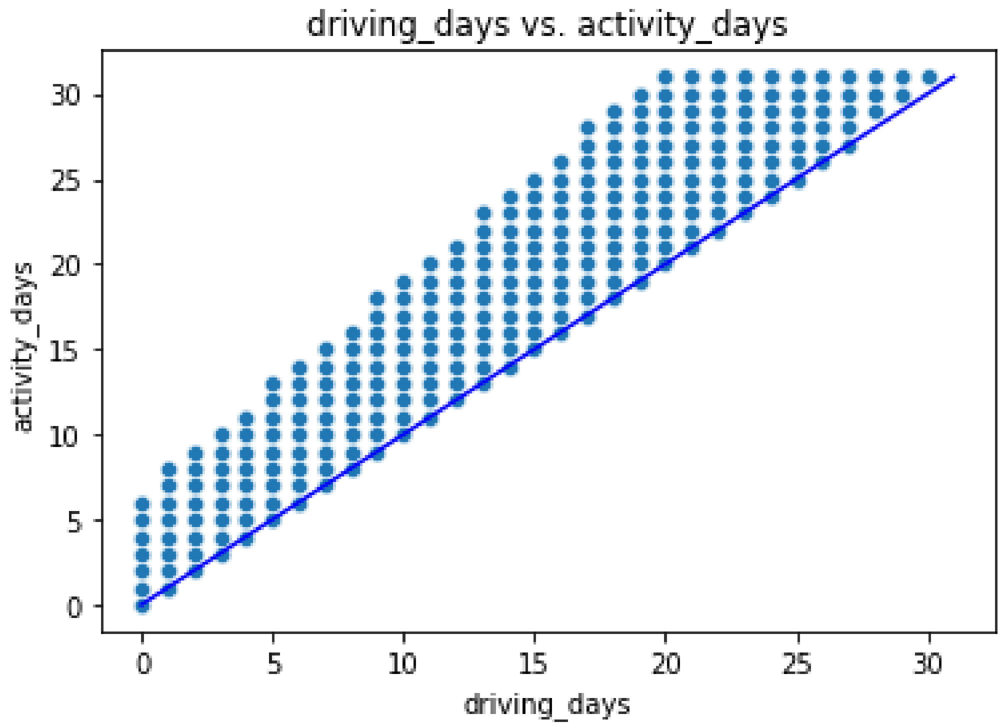
[32]:
```
#Count max number of days
print(df['driving_days'].max())
print(df['activity_days'].max())
```

```
30
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.

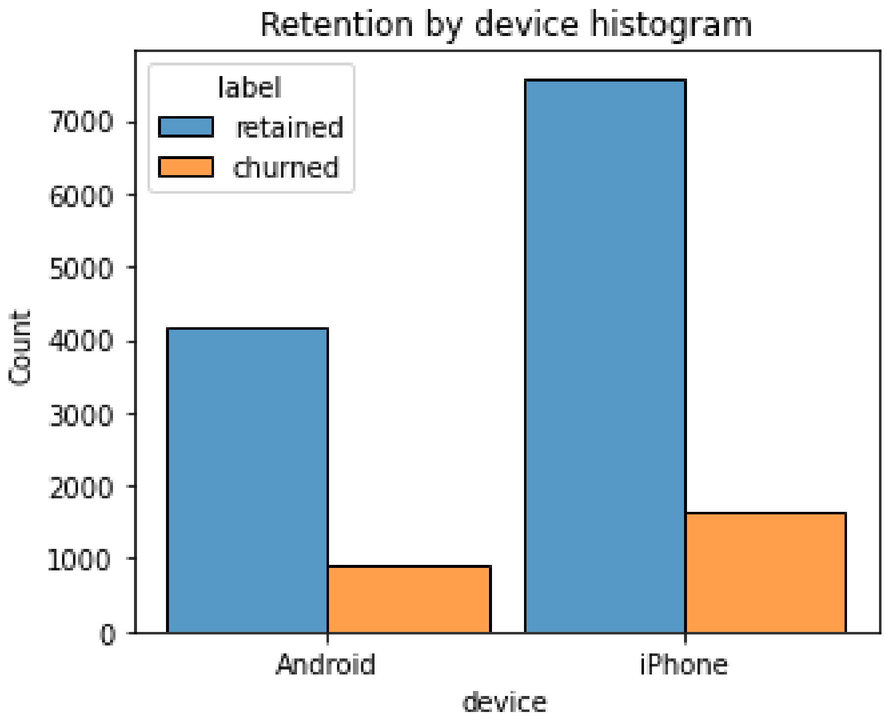
[34]:
```
# Scatter plot
sns.scatterplot(data=df, x='driving_days', y='activity_days')
plt.title('driving_days vs. activity_days')
plt.plot([0,31], [0,31], color='blue');
```



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.

[35]:
```
# Histogram
plt.figure(figsize=(5,4))
sns.histplot(data=df,
             x='device',
             hue='label',
             multiple='dodge',
             shrink=0.9
             )
plt.title('Retention by device histogram');
```



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
df['km_per_driving_day'] = df['driven_km_drives'] / df['driving_days']

# 2. Call `describe()` on the new column
df['km_per_driving_day'].describe()
```

[38]:
```
count    1.499900e+04
mean              inf
std               NaN
min      3.022063e+00
25%      1.672804e+02
50%      3.231459e+02
75%      7.579257e+02
```

```
max                 inf
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.

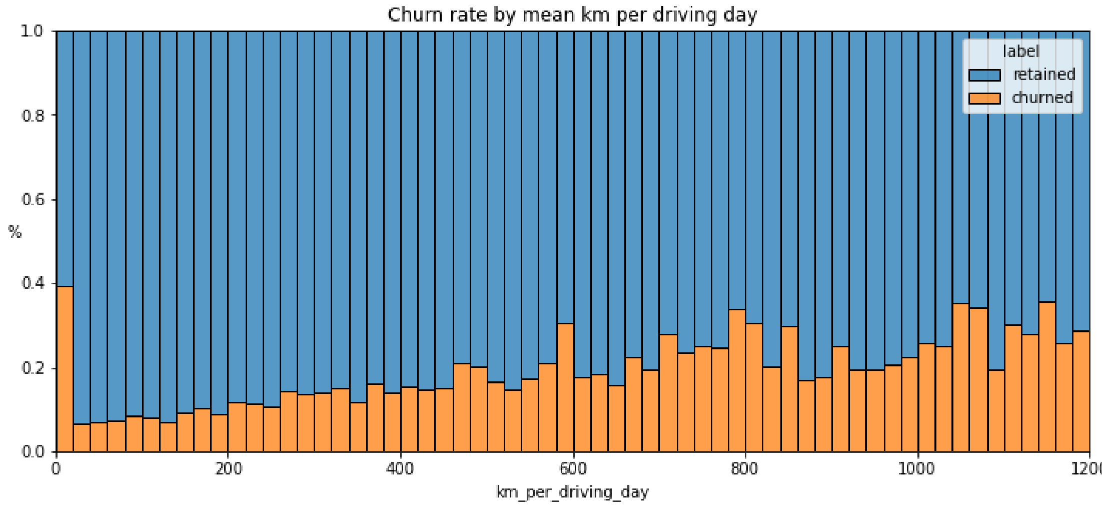
[39]:
```
# 1. Convert infinite values to zero
df.replace ([np. inf, -np. inf], 0, inplace= True)

# 2. Confirm that it worked
df['km_per_driving_day'].describe()
```

[39]:
```
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`.

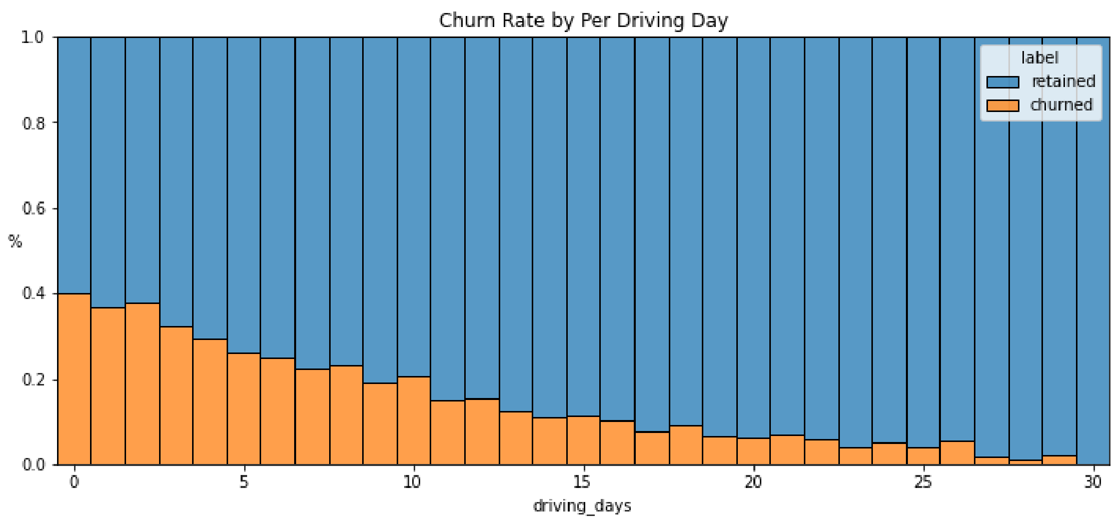
[40]:
```
# Histogram
plt.figure(figsize=(12,5))
sns.histplot(data=df,
             x='km_per_driving_day',
             bins=range(0,1201,20),
             hue='label',
             multiple='fill')
plt.ylabel('%', rotation=0)
plt.title('Churn rate by mean km per driving day');
```



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
plt.figure(figsize=(12,5))
sns.histplot(data=df,
             x='driving_days',
             bins=range(1,32),
             hue='label',
             multiple='fill',
            discrete=True)
plt.ylabel('%', rotation=0)
plt.title('Churn Rate by Per Driving Day');
```



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]:
```
#HD
df['percent_sessions_in_last_month'] = df['sessions'] / df['total_sessions']
```

What is the median value of the new column?

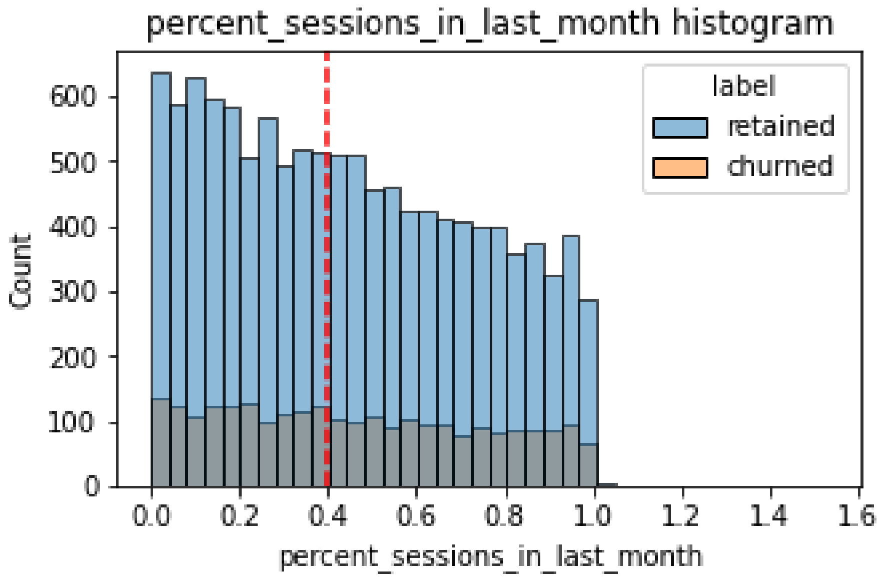
[45]:
```
df['percent_sessions_in_last_month'].median()
```

[45]: 0.42309702992763176

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

[46]:
```
# Histogram
# Histogram
histogrammer('percent_sessions_in_last_month',
             hue=df['label'],
             multiple='layer',
             median_text=False)
```

```
Median: 0.4
```



Check the median value of the `n_days_after_onboarding` variable.

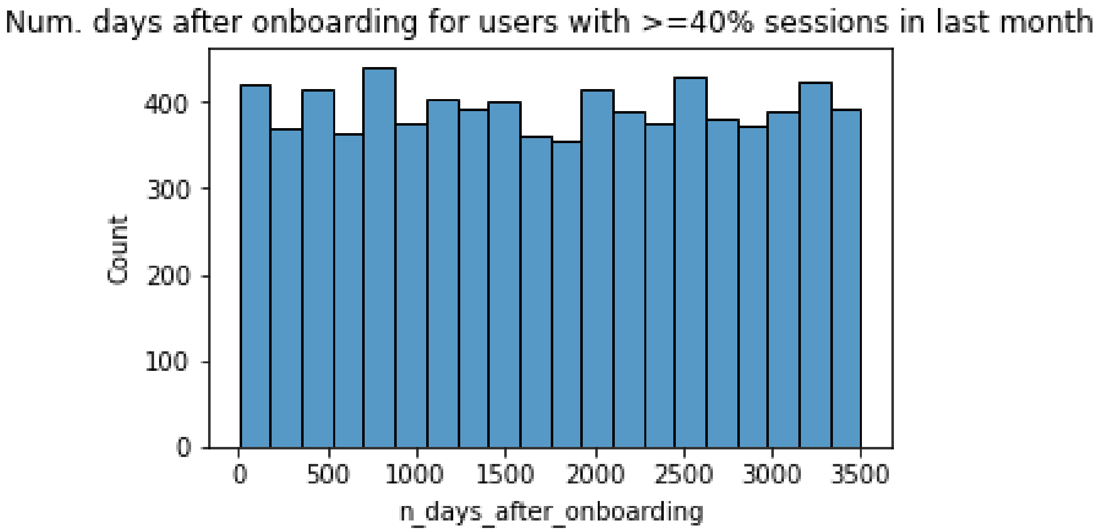
[47]:
```
df['n_days_after_onboarding'].median()
```

[47]:
```
1741.0
```

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.

[49]:
```
# Histogram
data = df.loc[df['percent_sessions_in_last_month']>=0.4]
plt.figure(figsize=(5,3))
sns.histplot(x=data['n_days_after_onboarding'])
plt.title('Num. days after onboarding for users with >=40% sessions in last␣
↪month');
```



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.

[50]:
```
### YOUR CODE HERE ###
def outlier(column_name, percentile):
    # Calculate threshold
    threshold = df[column_name].quantile(percentile)
    # Impute threshold for values > than threshold
    df.loc[df[column_name] > threshold, column_name] = threshold

    print('{:>25} | percentile: {} | threshold: {}'.format(column_name,
 ↪percentile, threshold))
```

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

[53]:

```
### YOUR CODE HERE ###
for column in ['sessions', 'drives', 'total_sessions',
               'driven_km_drives', 'duration_minutes_drives']:
               outlier(column, 0.95)
```

```
               sessions | percentile: 0.95 | threshold: 243.0
                 drives | percentile: 0.95 | threshold: 201.0
         total_sessions | percentile: 0.95 | threshold: 454.3632037399997
       driven_km_drives | percentile: 0.95 | threshold: 8889.7942356
duration_minutes_drives | percentile: 0.95 | threshold: 4668.899348999999
```

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

[54]:

```
df.describe()
```

[54]:

```
                 ID      sessions        drives  total_sessions  \
count  14999.000000  14999.000000  14999.000000    14999.000000
mean    7499.000000     76.568705     64.058204      184.031320
std     4329.982679     67.297958     55.306924      118.600463
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    243.000000    201.000000      454.363204

       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       3939.632764              1789.647426
std                 45.394651       2216.041510              1222.705167
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       8889.794236              4668.899349

       activity_days  driving_days  km_per_driving_day  \
count   14999.000000  14999.000000        14999.000000
mean       15.537102     12.179879          578.963113
```

```
std         9.004655      7.824036           1030.094384
min         0.000000      0.000000              0.000000
25%         8.000000      5.000000            136.238895
50%        16.000000     12.000000            272.889272
75%        23.000000     19.000000            558.686918
max        31.000000     30.000000          15420.234110

       percent_sessions_in_last_month
count                    14999.000000
mean                         0.449255
std                          0.286919
min                          0.000000
25%                          0.196221
50%                          0.423097
75%                          0.687216
max                          1.530637
```

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?

==> ENTER YOUR RESPONSE HERE

I have learned ….

My other questions are ….

My client would likely want to know …

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.

[ ]:
```
### YOUR CODE HERE ###
I have learned that there is some inconsistency with the data in the driving␣
↪columns,
the number of sessions and drives are strongly proportional.
More iphone users than android but the churned consistency is the same.
The number of churned users had more drives than number of retained users.
```

[ ]:
```
### YOUR CODE HERE ###
My other questions are...
what are the demographics & locations of these users.
Where is the missing data coming from?
The users with extremly long drives what there occupation is.
Seperate data for android and iphone users

My client would likely want to know
What is the correlation between longer drives and churn?
How can we further see into the users by there location or demographic.
```

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?

==> ENTER YOUR RESPONSES TO QUESTIONS 1-6 HERE

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.