# Activity_Course 3 Waze project lab

July 14, 2023

## 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 pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[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. There are 700 missing value in the label column
2. We can round all the numerical columns to 2 decimals
3. Can we reformat the days columns and add years columns?

### 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, drives, total sessions, n_days_after_onboarding, activity_days, driving_days, device, sessions
2. ID, total_navigations_fav1, total_navigations_fav2
3. Rows and Columns dropping, use isnull() function to check all the missing values
4. Building Box plot help to visualize the distribution

[3]:
```python
#Take a look at the dataset:
df.head(5)
```

[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]:
```python
#What is the shape of the dataset?
print("There are ",df.shape[0]," Rows in the data and ",df.shape[1]," Columns.")
```

```
There are  14999  Rows in the data and  13  Columns.
```

[5]:
```python
#List of the columns 'Features':
features=list(df.columns)
features
```

[5]:
```
['ID',
 'label',
 'sessions',
 'drives',
 'total_sessions',
 'n_days_after_onboarding',
 'total_navigations_fav1',
 'total_navigations_fav2',
 'driven_km_drives',
```

```
 'duration_minutes_drives',
 'activity_days',
 'driving_days',
 'device']
```

[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
```

[7]:
```
df.info('Dtype')
```

```
<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
```

[8]:
```
df.drop('ID',axis=1,inplace=True)
```

[9]:
```
#Statistics
df.describe()
```

[9]:
```
              sessions        drives  total_sessions  n_days_after_onboarding  \
count  14999.000000  14999.000000    14999.000000             14999.000000
mean      80.633776     67.281152      189.964447              1749.837789
std       80.699065     65.913872      136.405128              1008.513876
min        0.000000      0.000000        0.220211                 4.000000
25%       23.000000     20.000000       90.661156               878.000000
50%       56.000000     48.000000      159.568115              1741.000000
75%      112.000000     93.000000      254.192341              2623.500000
max      743.000000    596.000000     1216.154633              3500.000000

       total_navigations_fav1  total_navigations_fav2  driven_km_drives  \
count            14999.000000            14999.000000      14999.000000
mean               121.605974               29.672512       4039.340921
std                148.121544               45.394651       2502.149334
min                  0.000000                0.000000         60.441250
25%                  9.000000                0.000000       2212.600607
50%                 71.000000                9.000000       3493.858085
75%                178.000000               43.000000       5289.861262
max               1236.000000              415.000000      21183.401890

       duration_minutes_drives  activity_days  driving_days
count             14999.000000   14999.000000  14999.000000
mean               1860.976012      15.537102     12.179879
std                1446.702288       9.004655      7.824036
min                  18.282082       0.000000      0.000000
25%                 835.996260       8.000000      5.000000
50%                1478.249859      16.000000     12.000000
75%                2464.362632      23.000000     19.000000
max               15851.727160      31.000000     30.000000
```

[10]:
```
df.isnull().sum()
```

[10]:
```
label                      700
sessions                     0
drives                       0
total_sessions               0
n_days_after_onboarding      0
total_navigations_fav1       0
total_navigations_fav2       0
```

```
driven_km_drives           0
duration_minutes_drives    0
activity_days              0
driving_days               0
device                     0
dtype: int64
```

## 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. Plotting box plot

2. Interquarel range: Interquartile range method

   Sort your data from low to high Identify the first quartile (Q1), the median, and the third quartile (Q3). Calculate your IQR = Q3 – Q1 Calculate your upper fence = Q3 + (1.5 * IQR) Calculate your lower fence = Q1 – (1.5 * IQR) Use your fences to highlight any outliers, all values that fall outside your fences.

3. Using Statistical outlier detection (z-score)

### 4.3.1 Task 3a. Visualizations

**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

1. Box plot, Bar chart, Histogram, Heat map

### 4.3.2 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*

[11]:
```
import seaborn as sp
import matplotlib.pyplot as pt
import plotly.express as px
```

[12]:
```
#calculating the mean
mean=np.mean(df['sessions'])
median=np.median(df['sessions'])
print("The mean: ",mean)
#calculating the median
print("The median: ",median)
```

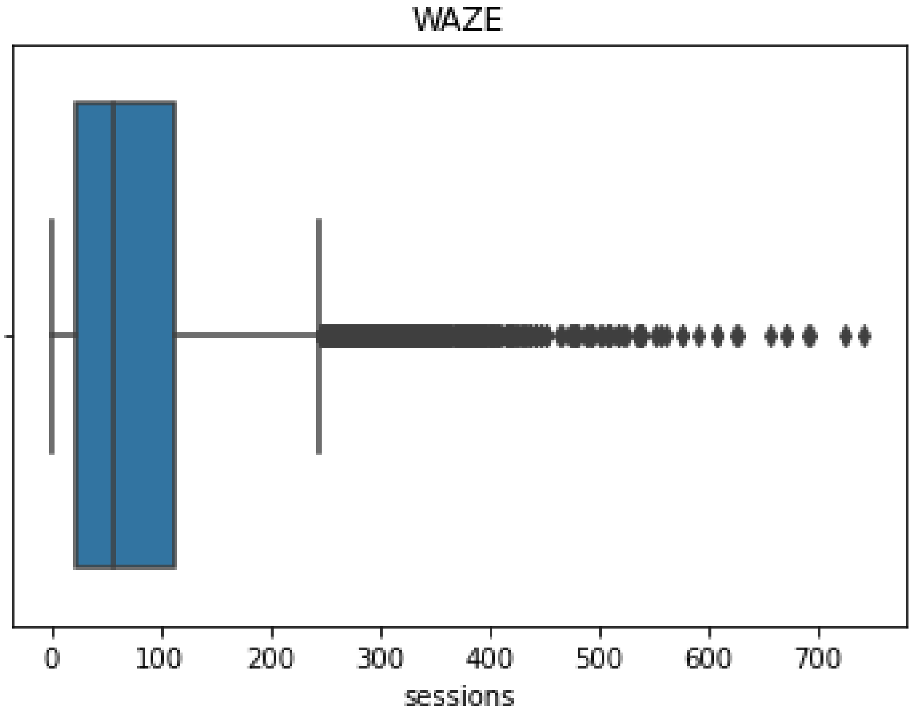
```
The mean:  80.633775585039
The median:  56.0
```

[13]:
```
if mean>median:
    print('The data more likely to be skewed to the RIGHT!')
else:
    print('The data more likely to be skewed to the LEFT!')
```

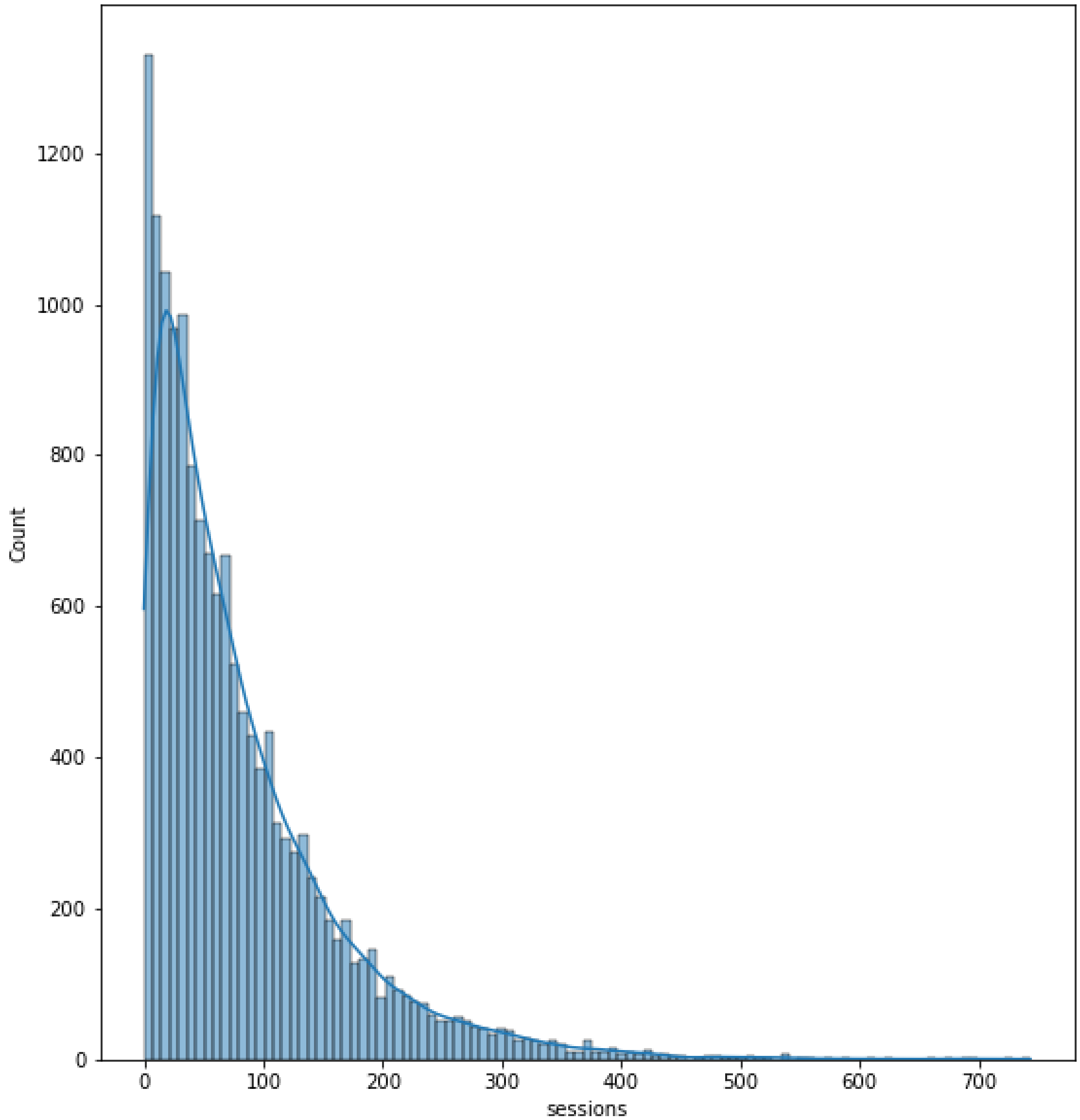
```
The data more likely to be skewed to the RIGHT!
```

[14]:
```
def skeweness(column):
    mean=np.mean(df['sessions'])
    median=np.median(df['sessions'])
    print("The mean: ",mean)
    #calculating the median
    print("The median: ",median)
    if mean>median:
        print('The data more likely to be skewed to the RIGHT!')
    else:
        print('The data more likely to be skewed to the LEFT!')
```

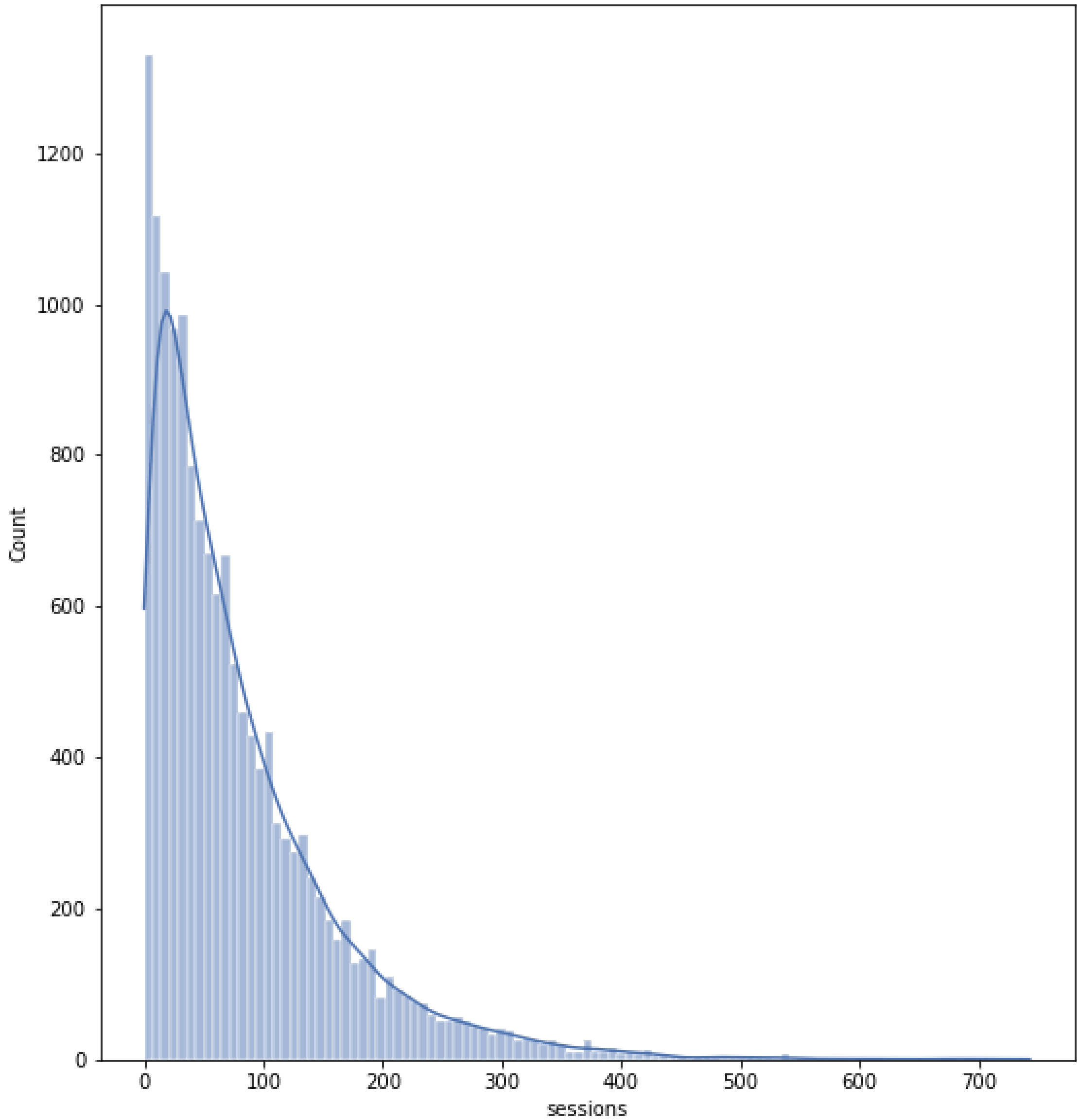
[15]:
```
# Box plot
box = sp.boxplot(x=df['sessions'])
g = plt.gca()
#box.set_xticklabels(np.array([readable_numbers(x) for x in g.get_xticks()]))
plt.xlabel('sessions')
plt.title('WAZE');
```




[16]:
```
fig, ax = plt.subplots(figsize=(9, 10))
sp.histplot(data=df, x="sessions", kde=True)
plt.show()
```




```
[17]: # Histogram
      # setting the dimensions of the plot
      fig, ax = plt.subplots(figsize=(9, 10))
      sp.set(style="darkgrid")
      sp.histplot(data=df, x="sessions", kde=True)
      plt.show()
```



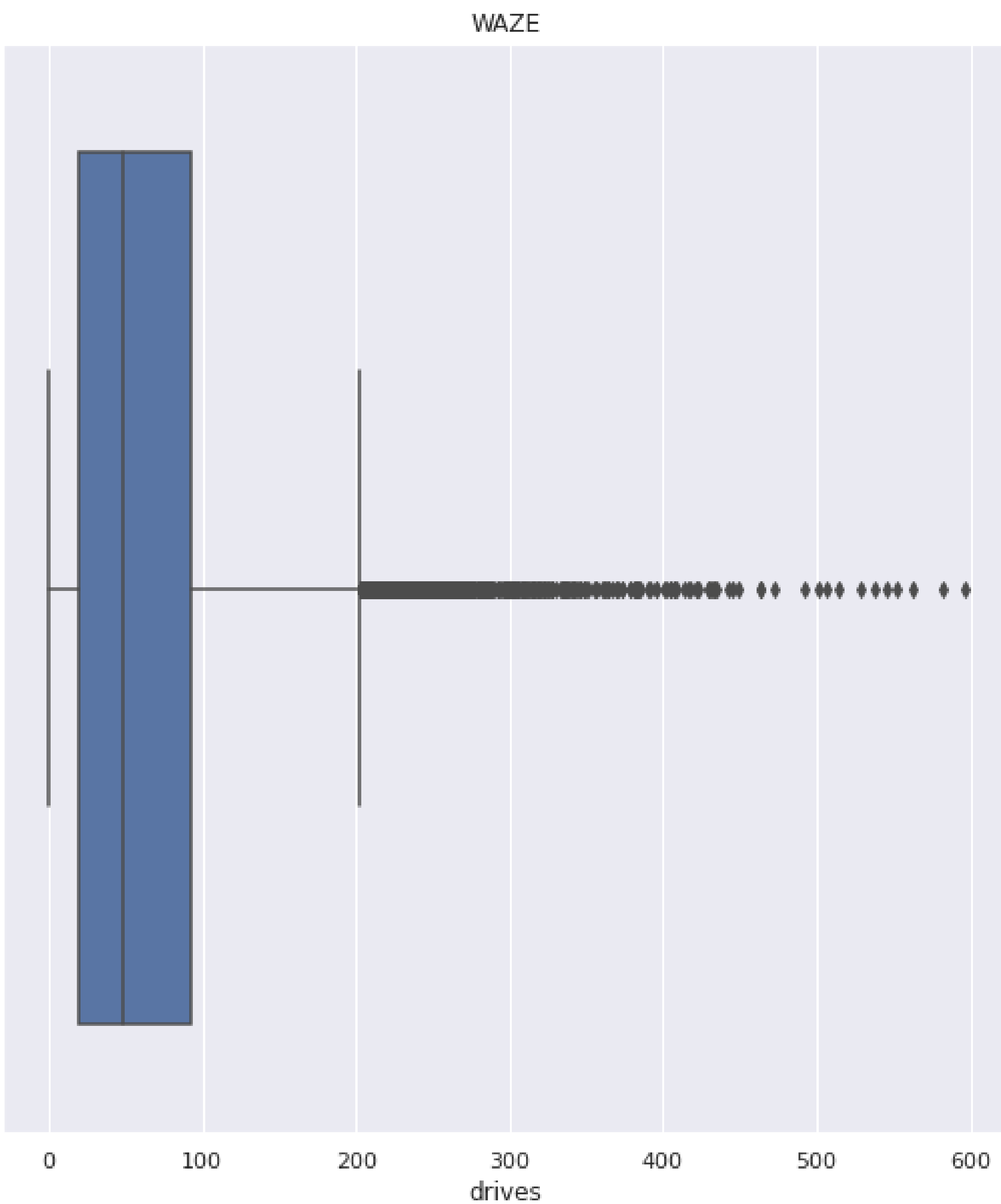

The **sessions** variable is a right-skewed distribution with half of the observations having 56 or fewer sessions. However, as indicated by the boxplot, some users have more than 700.

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

[18]:
```
skeweness(df['drives'])
```

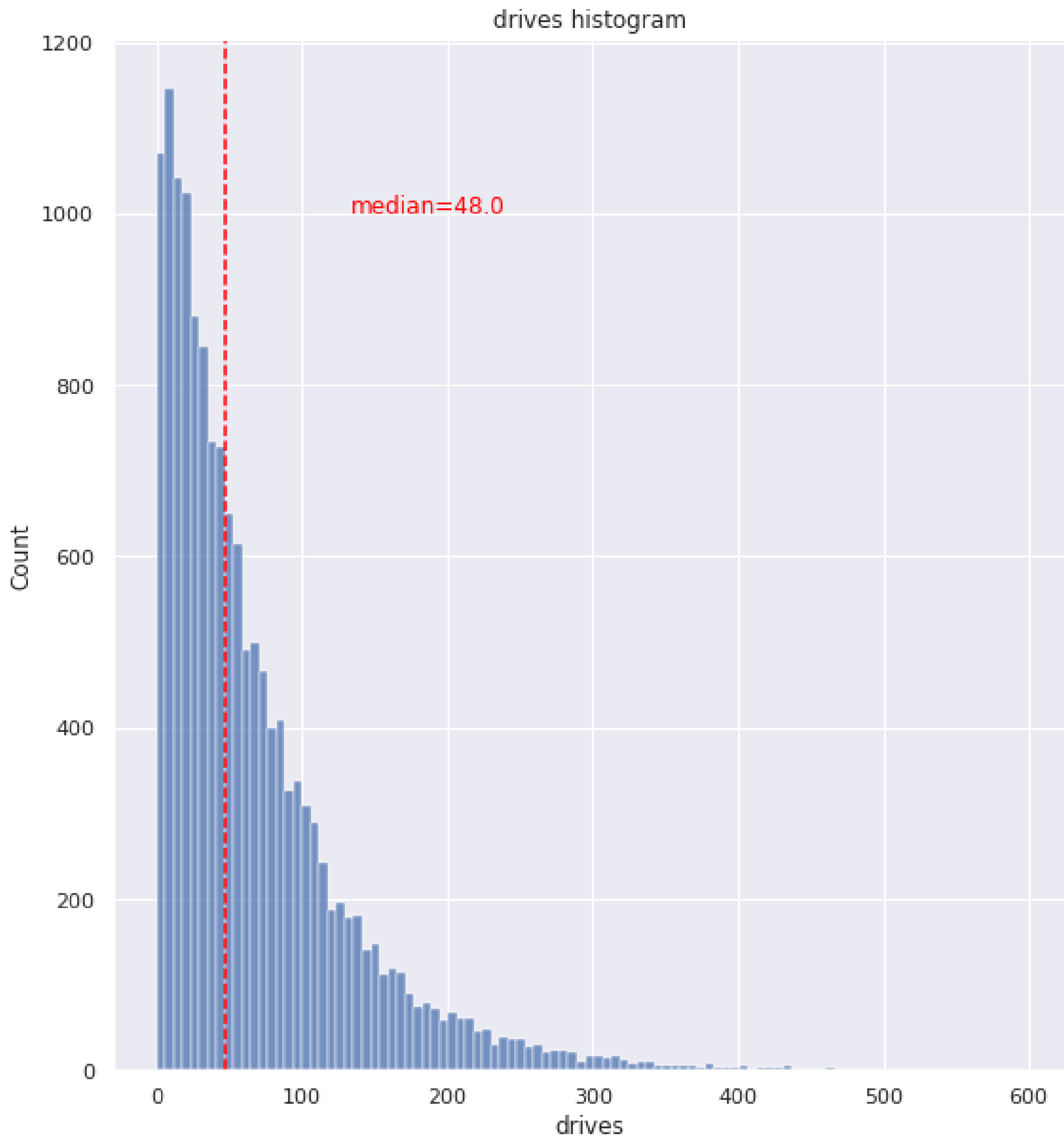
```
The mean:  80.633775585039
The median:  56.0
The data more likely to be skewed to the RIGHT!
```

[19]:
```python
# Box plot
fig, ax = plt.subplots(figsize=(9, 10))
box = sp.boxplot(x=df['drives'])
g = plt.gca()
#box.set_xticklabels(np.array([readable_numbers(x) for x in g.get_xticks()]))
plt.xlabel('drives')
plt.title('WAZE');
```



[20]:
```
# 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)
    fig, ax = plt.subplots(figsize=(9, 10))
    ax = sp.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');
```

[21]:
```
# Histogram
histogrammer('drives',True)
```




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. It is best to use the median when the distribution is either skewed or there are outliers present.

[22]:
```
def boxplotter(column,xlabel,title):
    # Box plot
    fig, ax = plt.subplots(figsize=(9, 10))
    box = sp.boxplot(x=df[column])
    g = plt.gca()
```

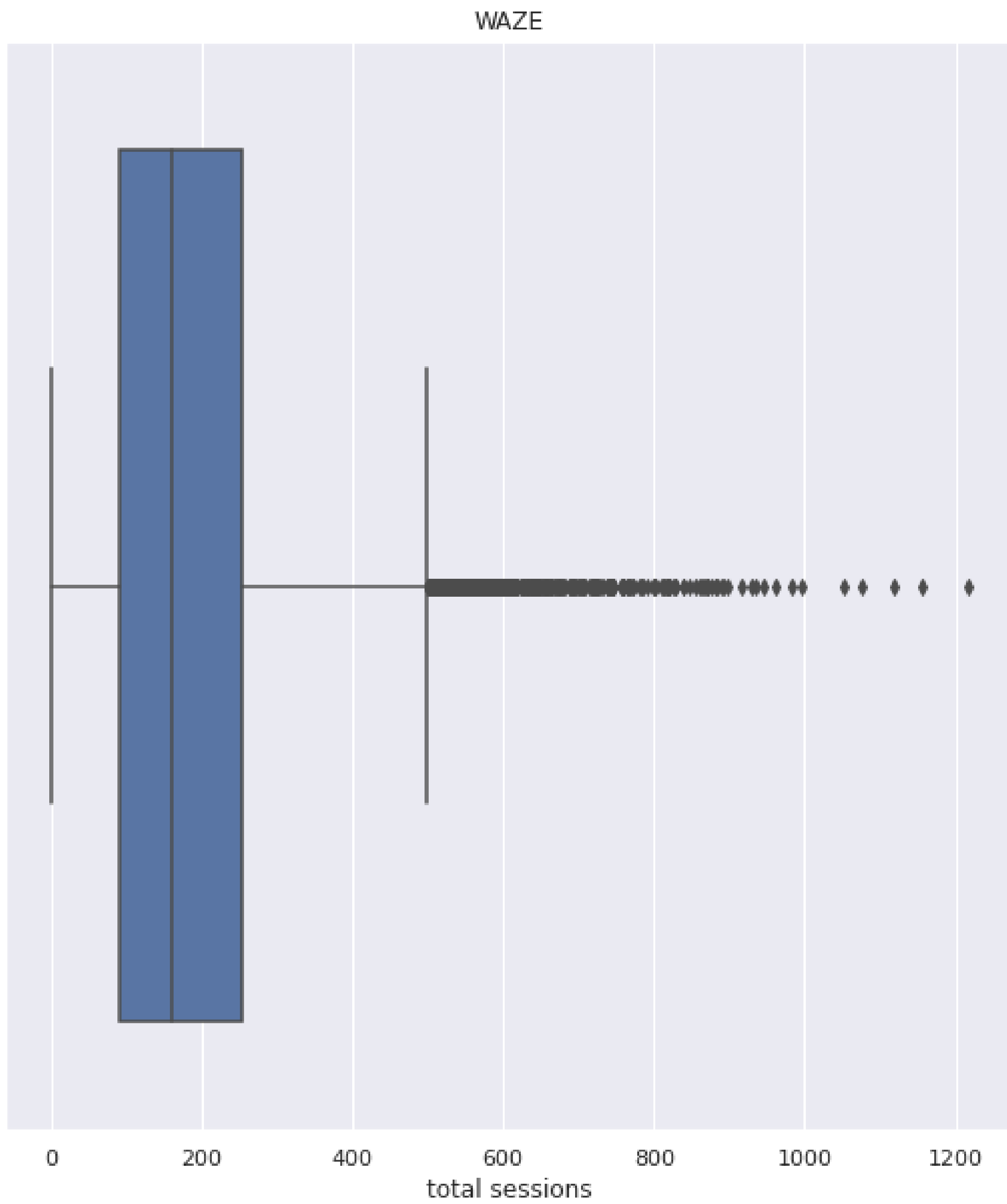
```
    #box.set_xticklabels(np.array([readable_numbers(x) for x in g.
↪get_xticks()]))
    plt.xlabel(xlabel)
    plt.title(title)
    plt.show()
```

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

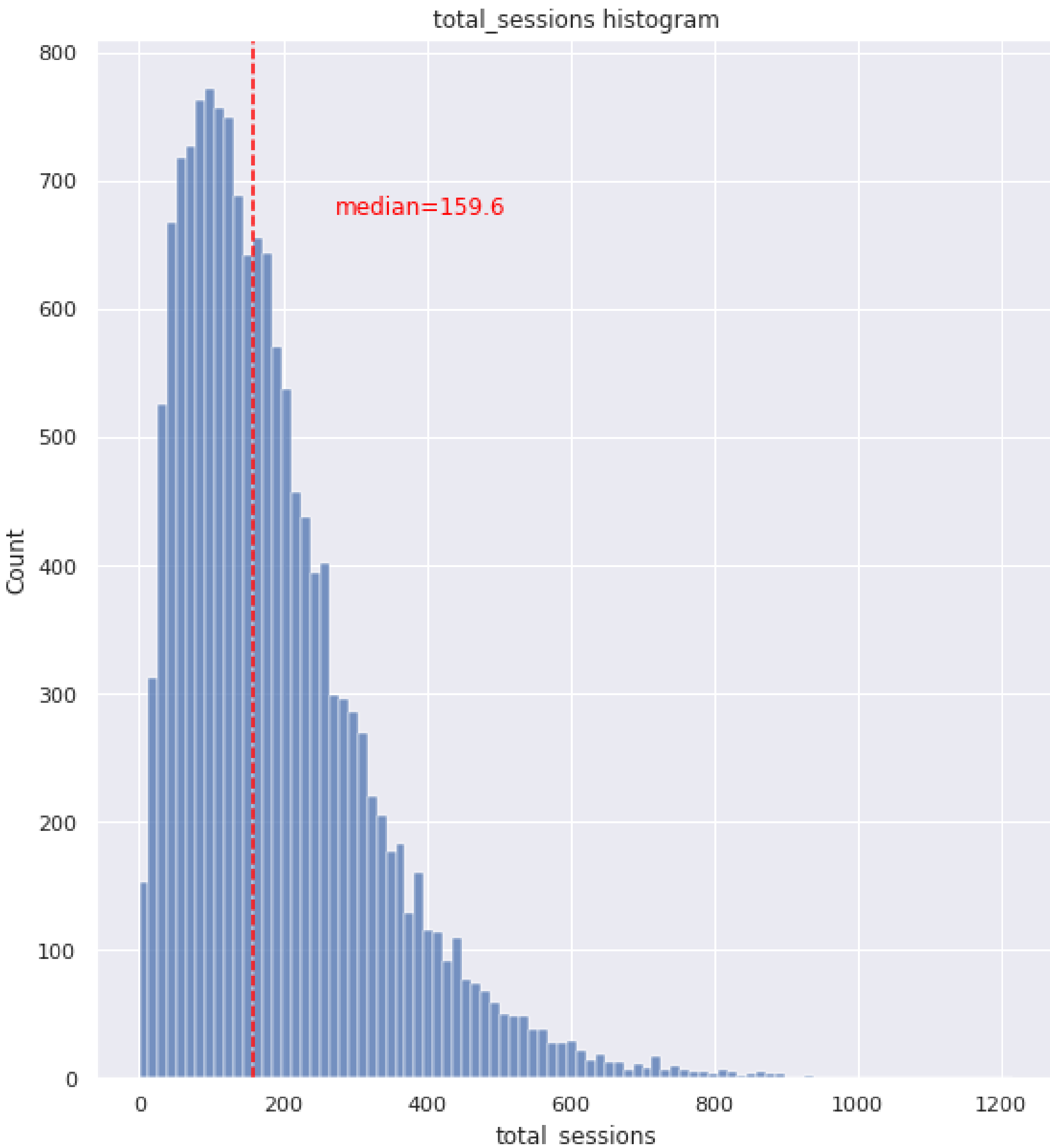
[23]:
```
skeweness(df['total_sessions'])
```

```
The mean:  80.633775585039
The median:  56.0
The data more likely to be skewed to the RIGHT!
```

[24]:
```
# Box plot
boxplotter('total_sessions','total sessions','WAZE')
```




[25]:
```
# Histogram
histogrammer('total_sessions',True)
```



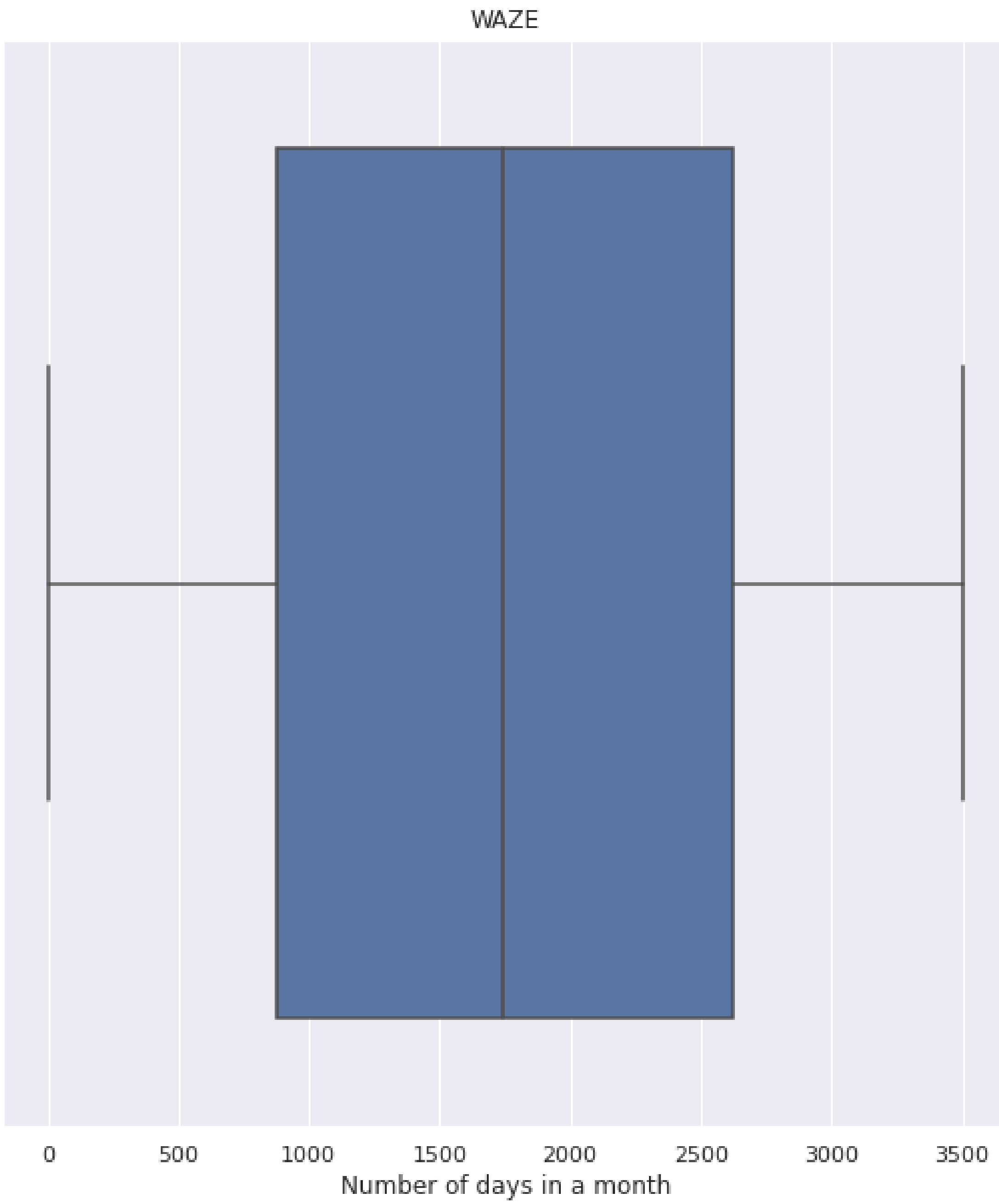

The `total_sessions` is a right-skewed distribution that looks more normal than the previous variables. The median total number of sessions is 159.6. This is interesting information because, if the median number of sessions in the last month was 48 and the median total sessions was ~160, then it seems 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*

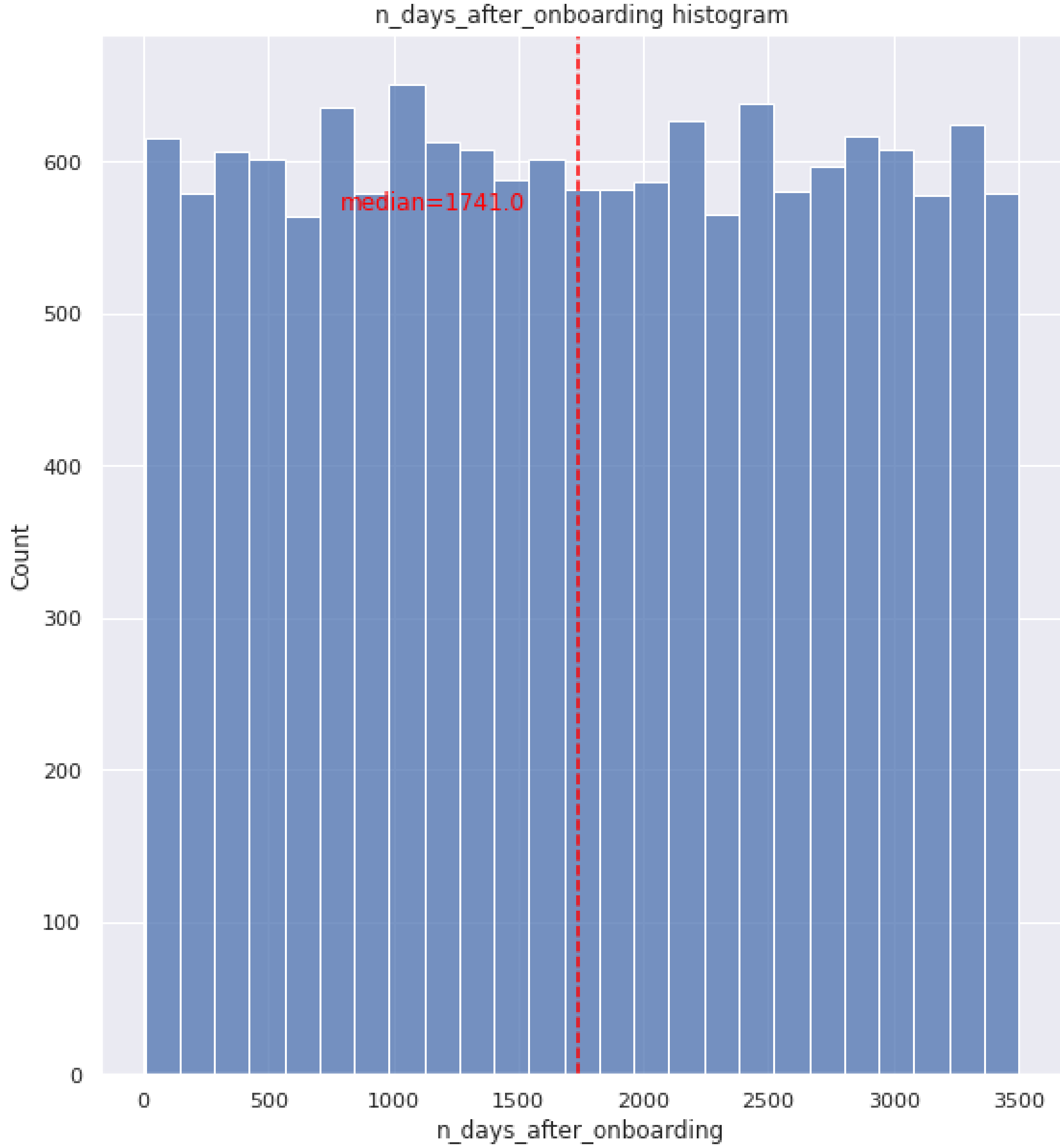
[26]:
```
skeweness(df['n_days_after_onboarding'])
```

```
The mean:  80.633775585039
The median:  56.0
The data more likely to be skewed to the RIGHT!
```

[27]:
```python
# Box plot
boxplotter('n_days_after_onboarding','Number of days in a month','WAZE')
```



[28]:
```
# Histogram
histogrammer('n_days_after_onboarding',True)
```




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).

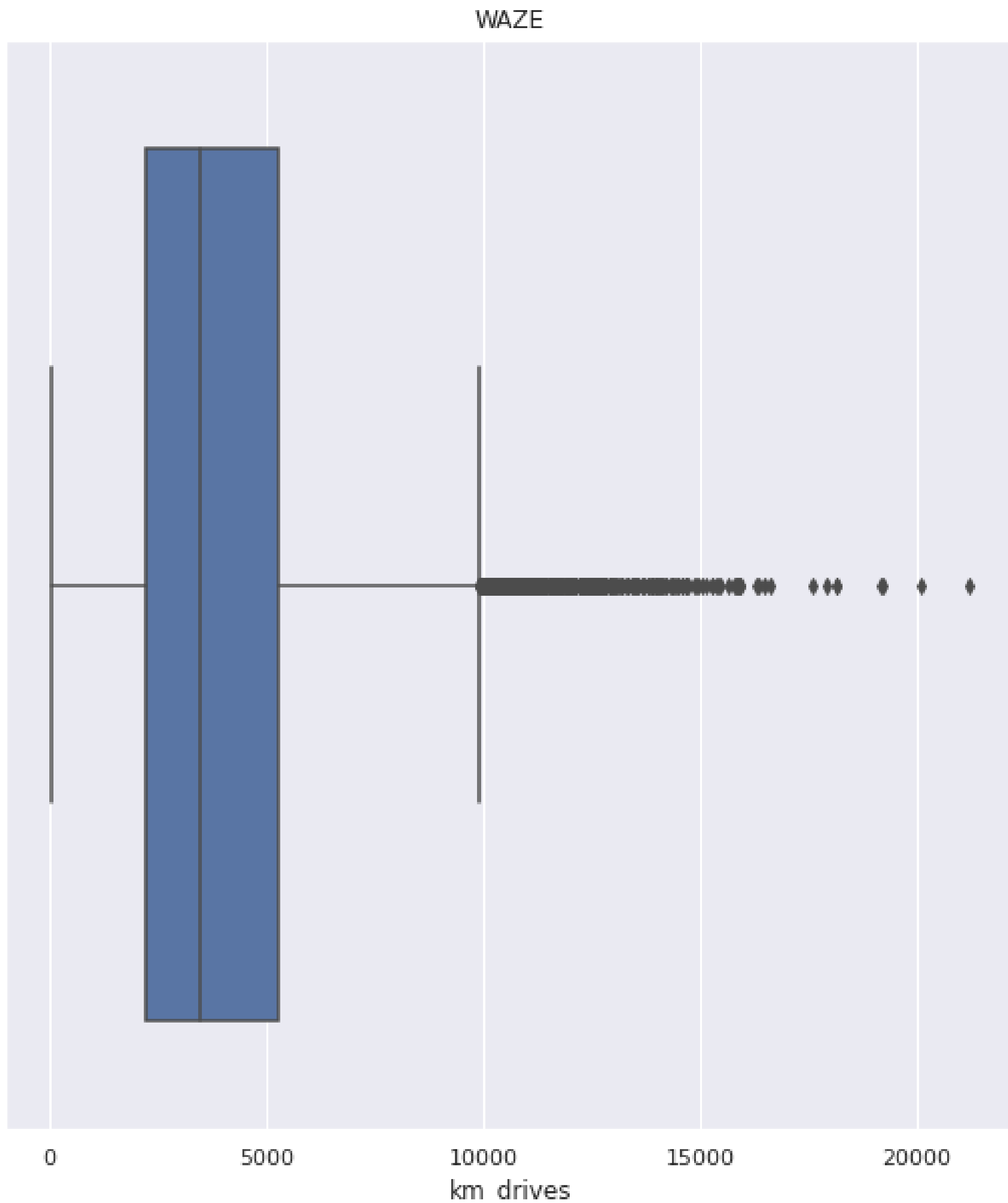
**driven_km_drives** *Total kilometers driven during the month*

[29]:
```
skeweness(df['driven_km_drives'])
```

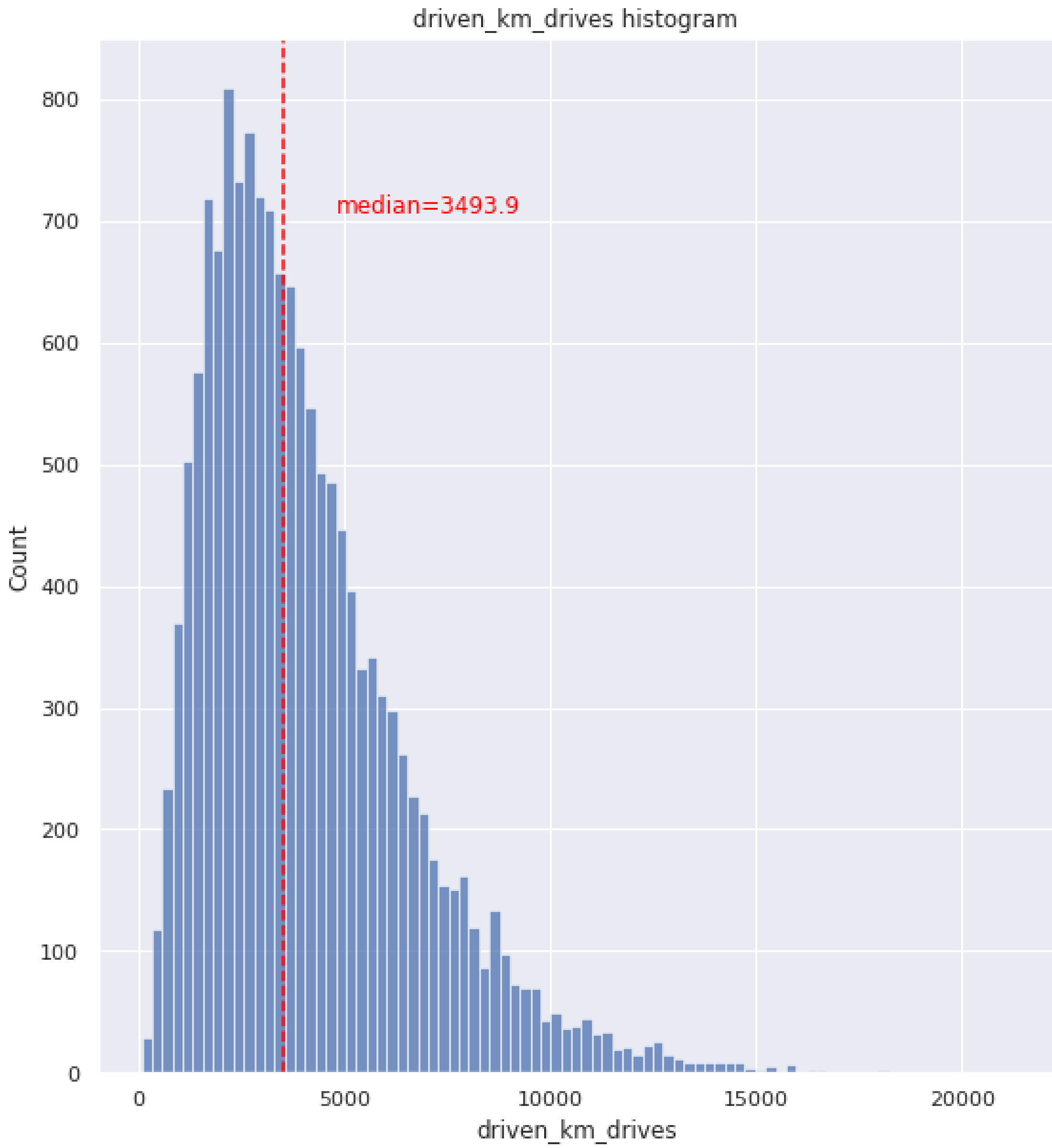
```
The mean:  80.633775585039
The median:  56.0
The data more likely to be skewed to the RIGHT!
```

[30]:

```
# Box plot
boxplotter('driven_km_drives','km_drives','WAZE')
```



[31]:
```python
# Histogram
histogrammer('driven_km_drives',True)
```



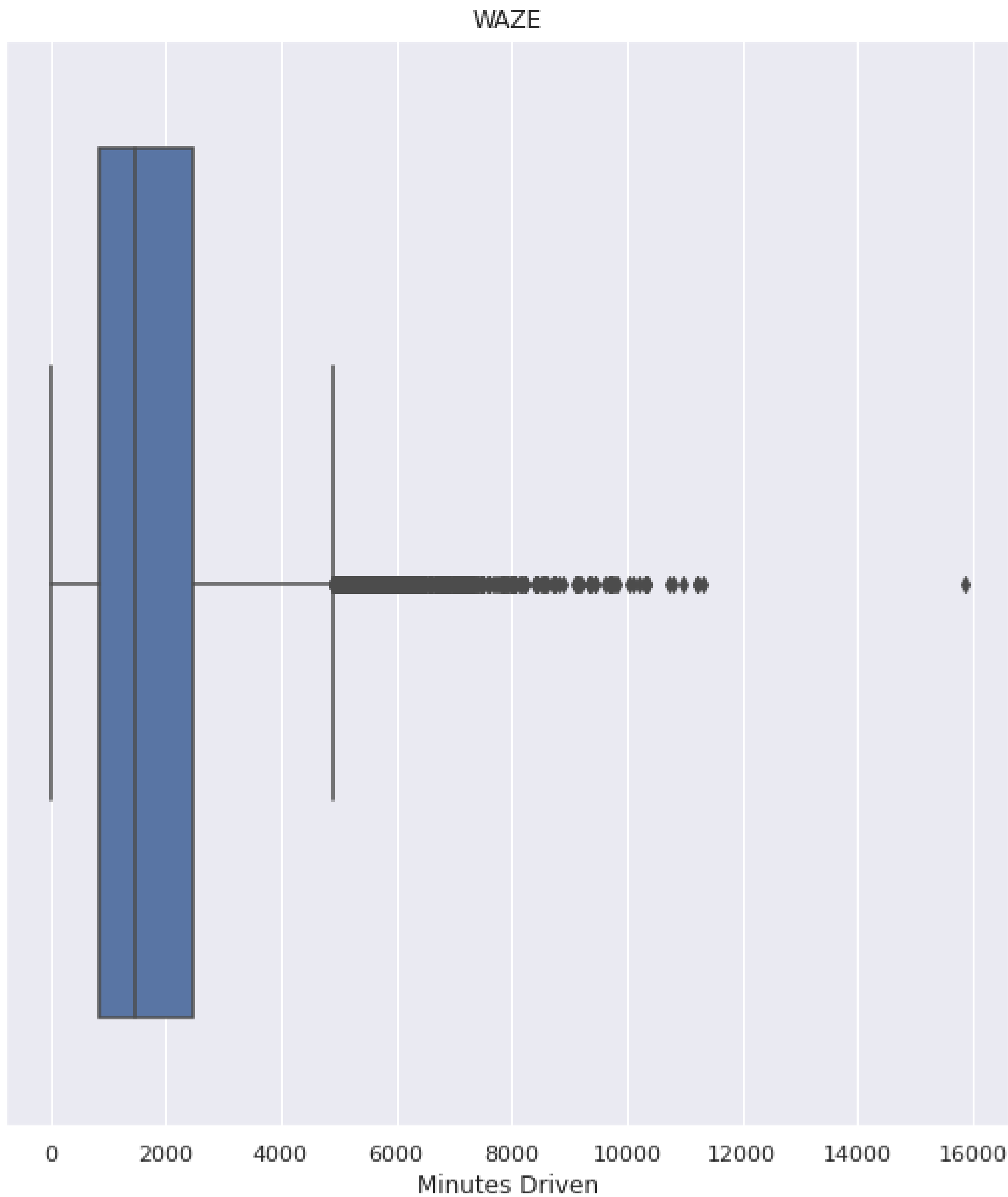

The number of drives driven in the last month per user resembles a right-skewed normal distribution with half the users driving under 3,495 kilometers. As you discovered in the analysis from the previous course, the users in this dataset drive *a lot*. The longest distance driven in the month was over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

**duration_minutes_drives** *Total duration driven in minutes during the month*

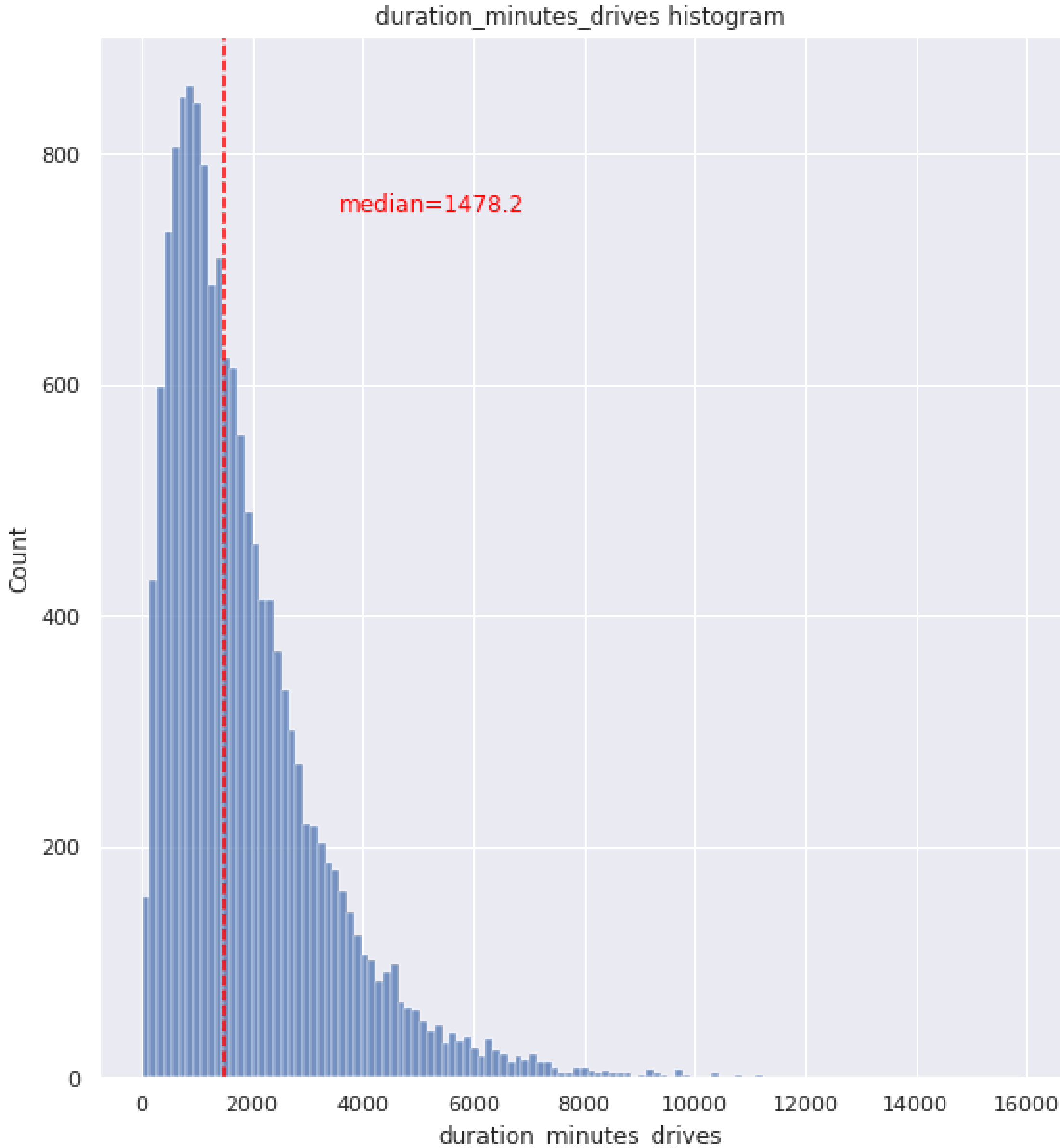
[32]:
```
skeweness(df['duration_minutes_drives'])
```

```
The mean:  80.633775585039
The median:  56.0
The data more likely to be skewed to the RIGHT!
```

[33]:
```
# Box plot
boxplotter('duration_minutes_drives','Minutes Driven','WAZE')
```



[34]:
```python
# Histogram
histogrammer('duration_minutes_drives',True)
```



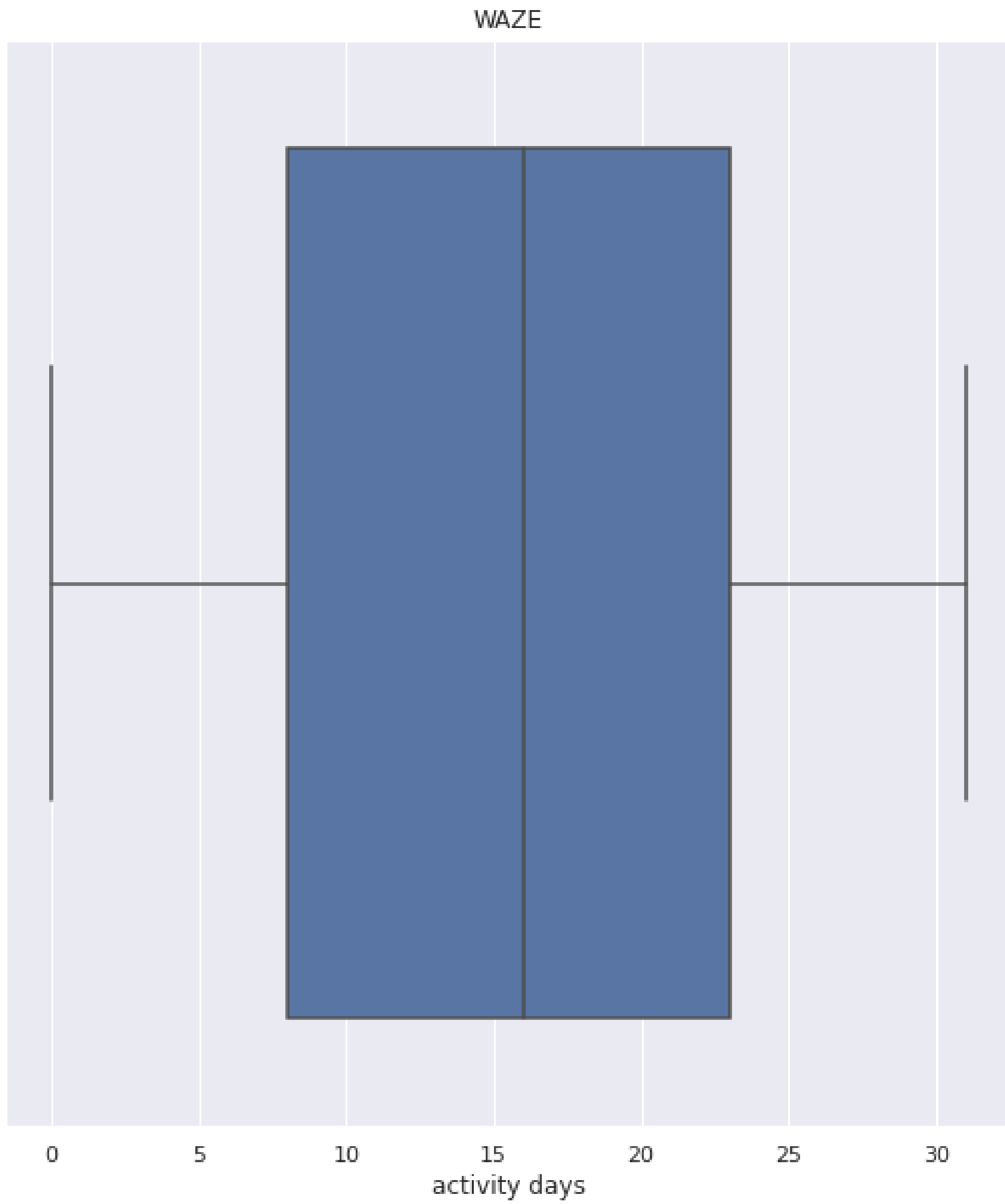

The `duration_minutes_drives` variable has a normalish distribution with a heavily skewed right tail. Half of the users drove less than ~1,478 minutes (~25 hours), but some users clocked over 250 hours over the month.

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

[35]:
```
skeweness(df['activity_days'])
```

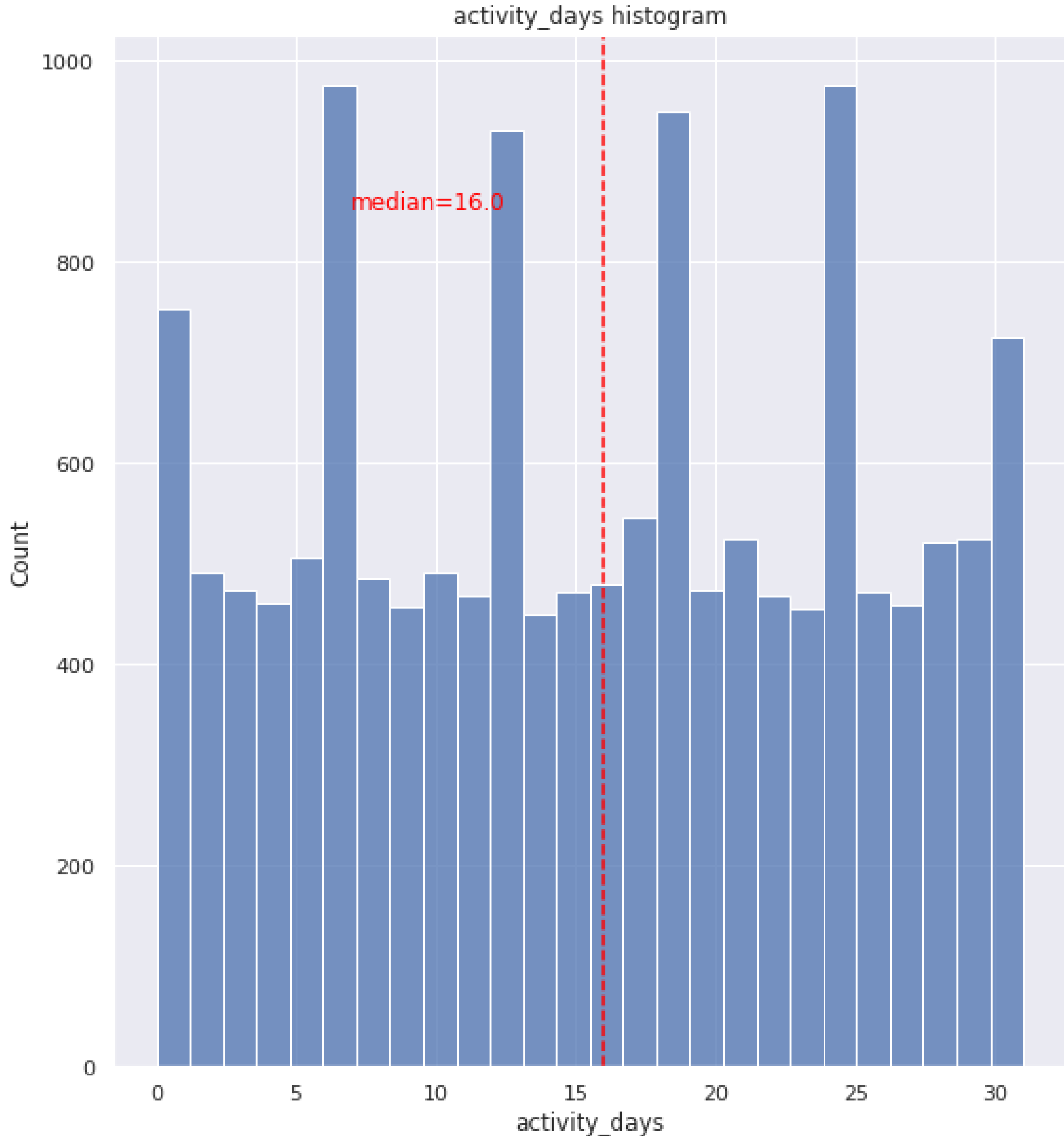
```
The mean:  80.633775585039
The median:  56.0
The data more likely to be skewed to the RIGHT!
```

[36]:
```
# Box plot
boxplotter('activity_days','activity days','WAZE')
```



[37]:

```
# Histogram
histogrammer('activity_days',True)
```




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.

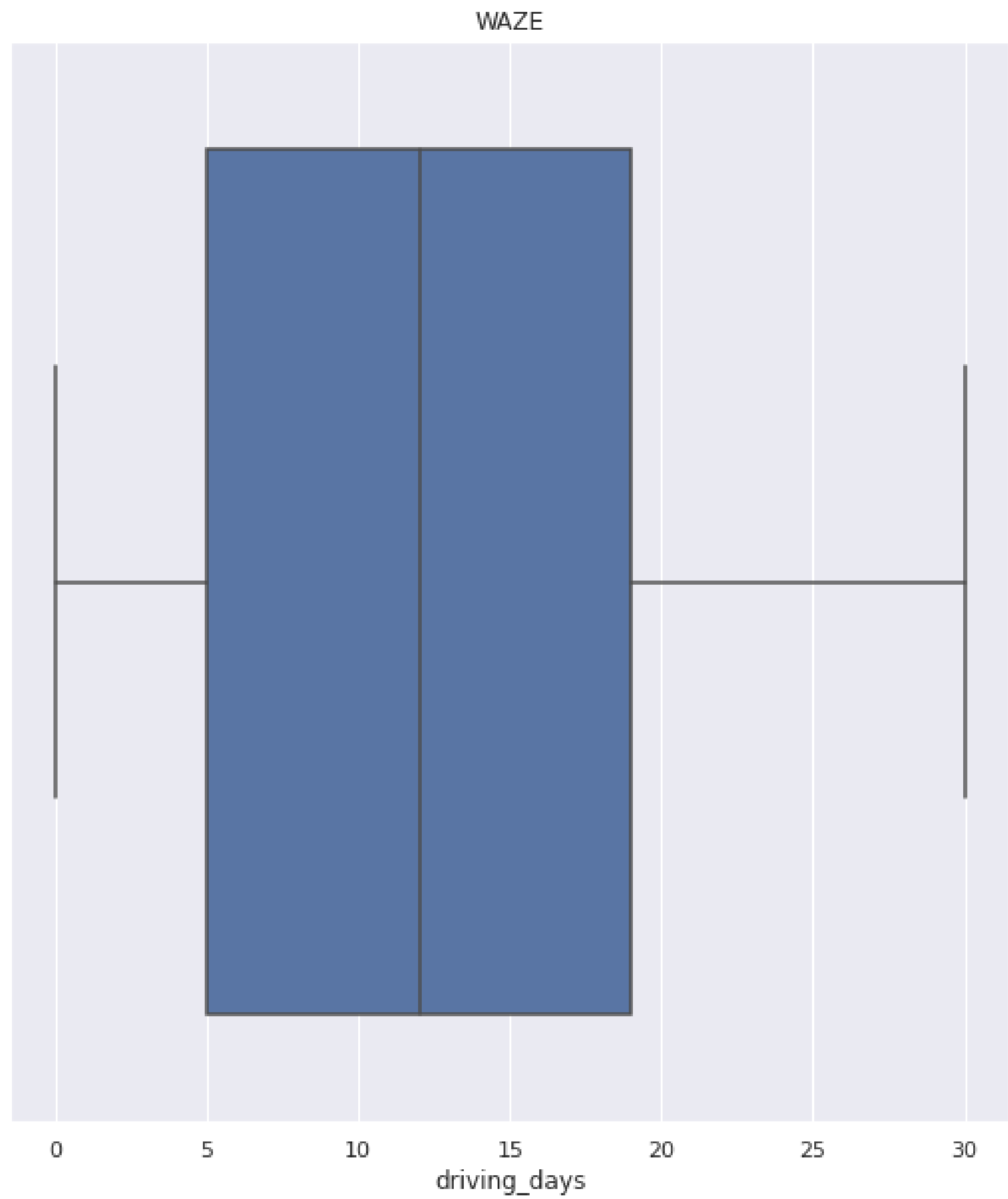
This distribution is noteworthy because it does not mirror the `sessions` distribution, which you might think would be closely correlated with `activity_days`.

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

[38]:
```
skeweness(df['driving_days'])
```

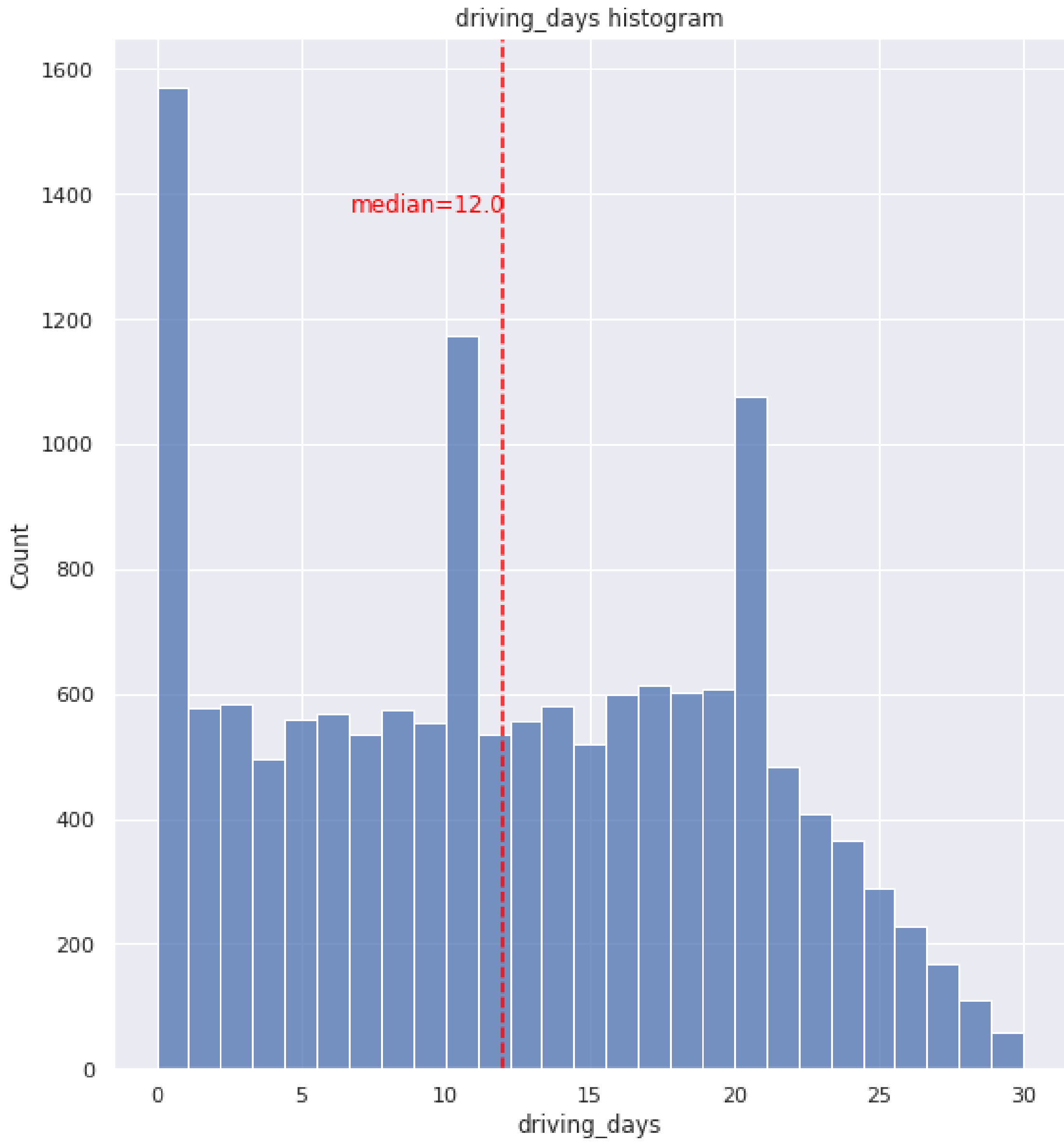
```
The mean:  80.633775585039
The median:  56.0
The data more likely to be skewed to the RIGHT!
```

[39]:
```
# Box plot
boxplotter('driving_days','driving_days','WAZE')
```



[40]:

```python
# Histogram
histogrammer('driving_days',True)
```




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.

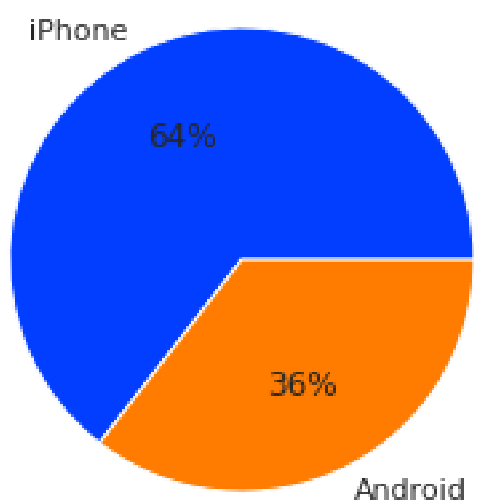
However, there were almost twice as many users (~1,000 vs. ~550) who did not drive at all during the month. This might seem counterintuitive when considered together with the information from `activity_days`. That variable had ~500 users opening the app on each of most of the day counts, but there were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

[41]:
```
def pirplotter(data,keys):
    # define Seaborn color palette to use
    palette_color = sp.color_palette('bright')

    # plotting data on chart
    plt.pie(data, labels=keys, colors=palette_color, autopct='%.0f%%')

    # displaying chart
    plt.show()
```

**`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.

[42]:
```
# Pie chart
keys=list(df['device'].value_counts().index)
data=list(df['device'].value_counts().values)
pirplotter(data,keys)
```

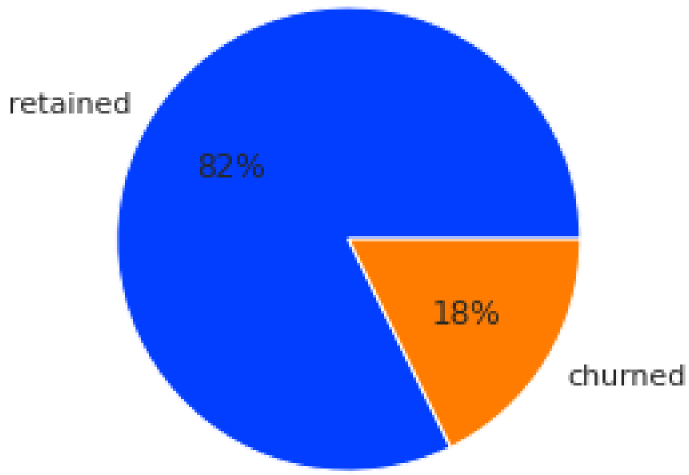



There are nearly twice as many iPhone users as Android users represented in this data.

**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.

[43]:
```
# Pie chart
keys=list(df['label'].value_counts().index)
data=list(df['label'].value_counts().values)
pirplotter(data,keys)
```

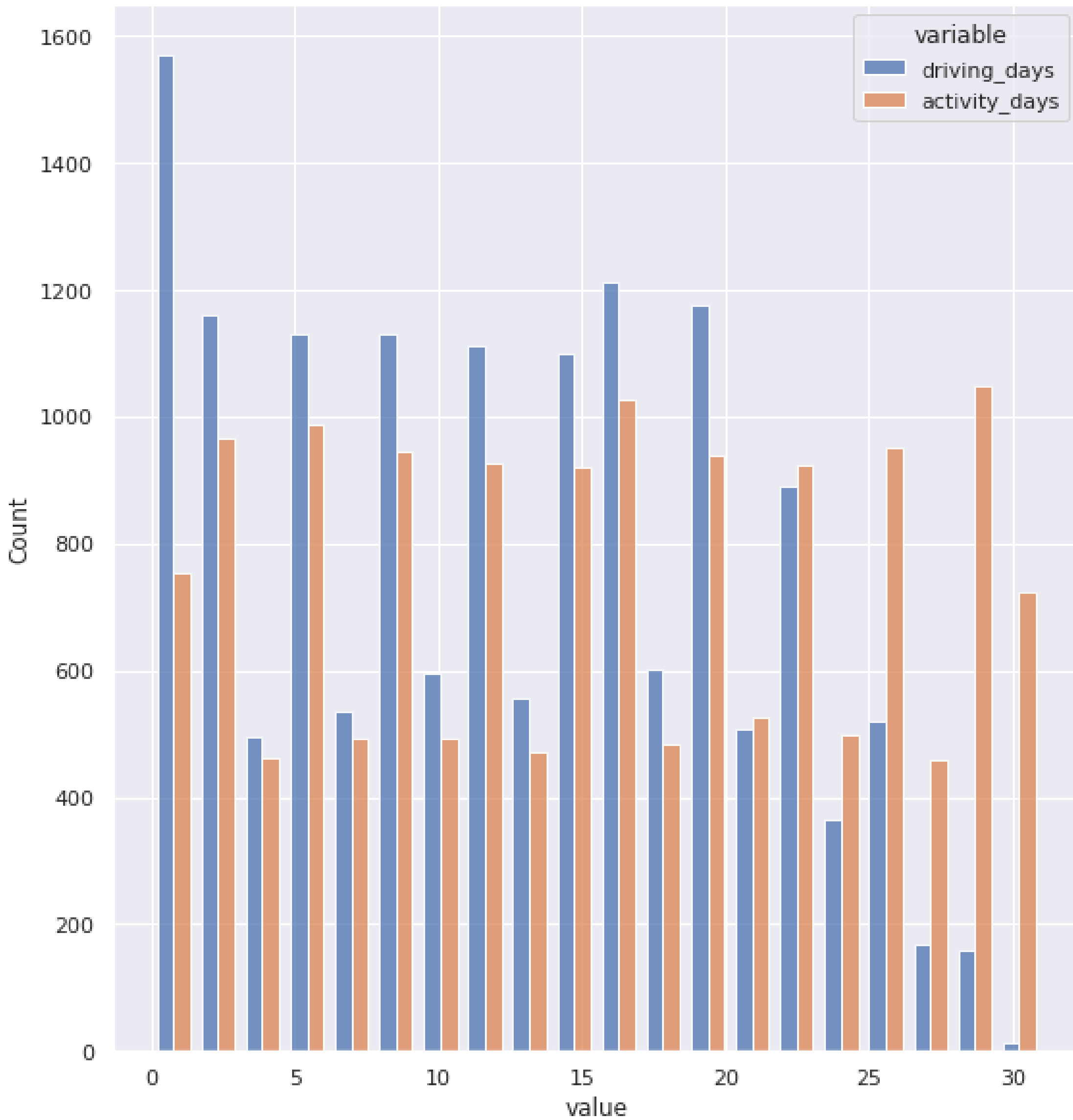



Less than 18% of the users churned.

**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 `user_days`.

[44]:
```
# Histogram
# plot melted dataframe in a single command
fig, ax = plt.subplots(figsize=(9, 10))
df_copy=df[['driving_days', 'activity_days']]
sp.histplot(df_copy.melt(), x='value', hue='variable',
             multiple='dodge', shrink=.75, bins=20);
```




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`.

[45]:
```
max_driving=df['driving_days'].max()
max_active=df['activity_days'].max()
```

[46]:
```
print('The maximum days drove in a month: ',max_driving)
print('The maximum days been active in a month: ',max_active)
```

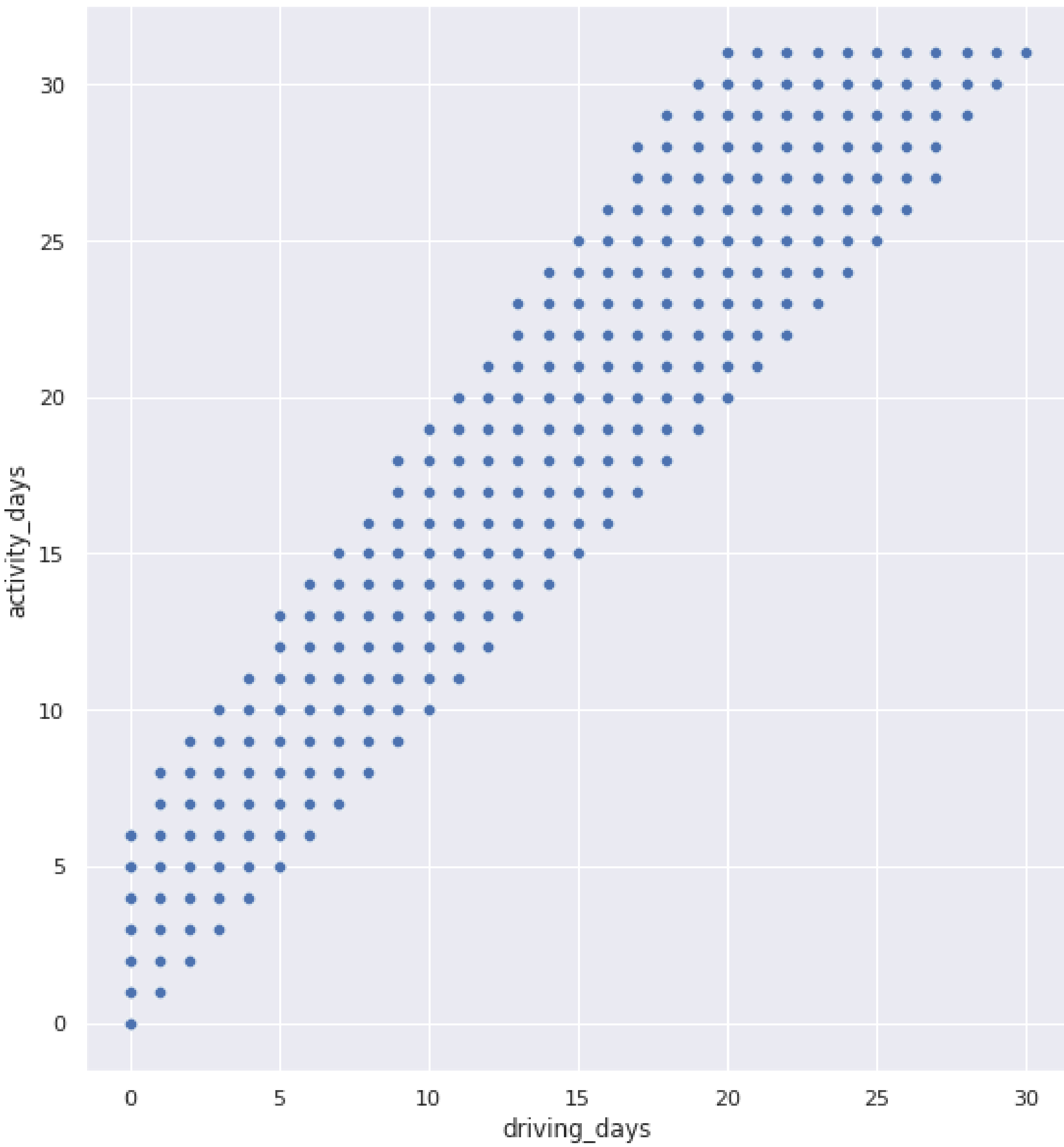
```
The maximum days drove in a month:  30
The maximum days been active in a month:  31
```

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.

[47]:
```
# Scatter plot
fig, ax = plt.subplots(figsize=(9, 10))
sp.scatterplot(x="driving_days",
                    y="activity_days",
                    data=df)
```

[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc28d62cb10>
```

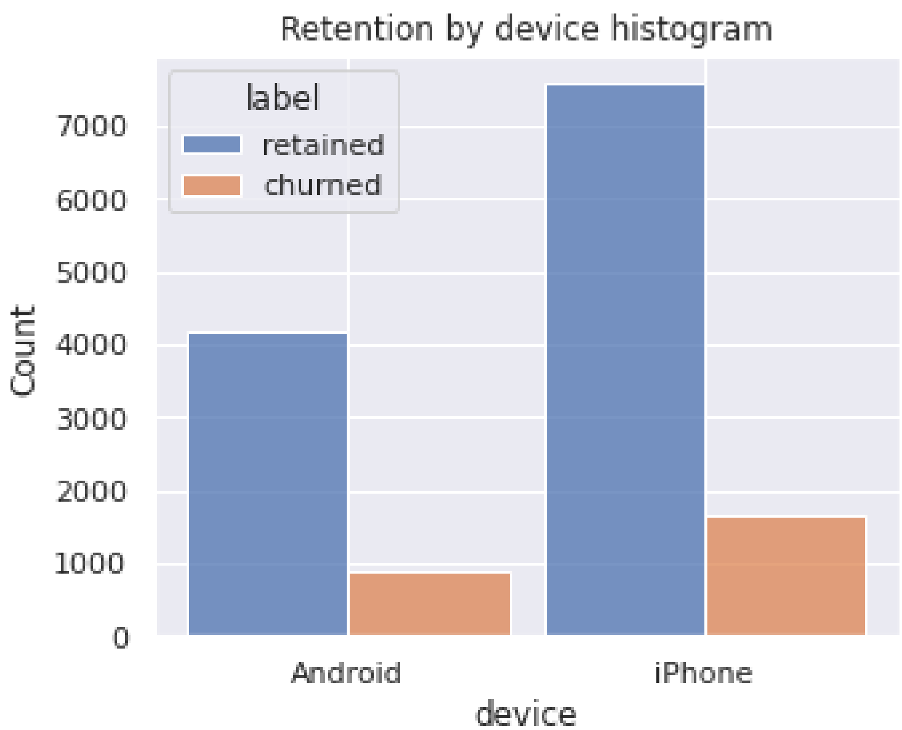



Notice that there is a theoretical limit. If you use the app to drive, then by definition it must count as a day-use as well. In other words, you cannot have more drive-days than activity-days. None of the samples in this data violate this rule, which is good.

**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.

[48]:
```
# Histogram
plt.figure(figsize=(5,4))
```

```
sp.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 last month for users who churned was 8.33 km, versus 3.36 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.

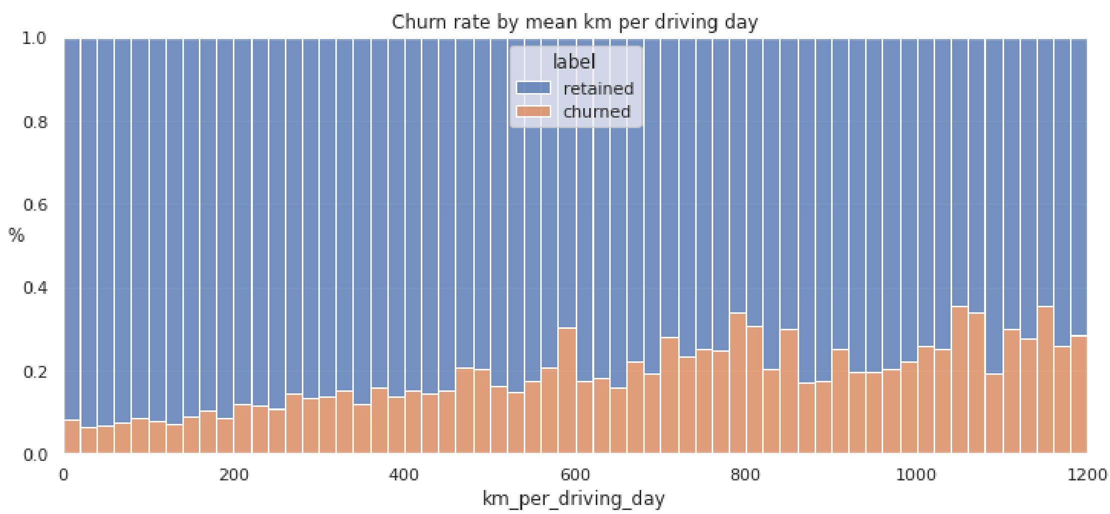
[49]:
```
# 1. Create `km_per_driving_day` column
df['km_per_driving_day']= df['driven_km_drives'] / df['driving_days']

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

[49]:
```
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
```

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`.

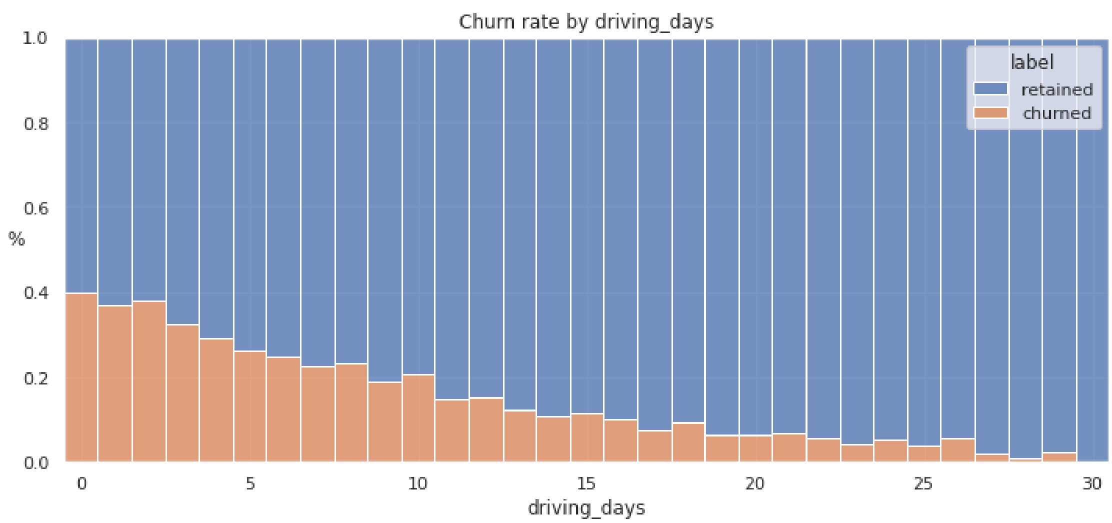
[50]:
```
# Histogram
plt.figure(figsize=(12,5))
sp.histplot(data=df,
             x='km_per_driving_day',
             bins=range(0,1201,20), #this is to specify the range to get only␣
 ↪users drove at most 1200 km
             hue='label',
             multiple='fill')
plt.ylabel('%', rotation=0)
plt.title('Churn 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.

[51]:
```
# Histogram

plt.figure(figsize=(12,5))
sp.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 driving_days');
```




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.

```
[52]: df['percent_sessions_in_last_month']=df['sessions']/df['total_sessions']
```

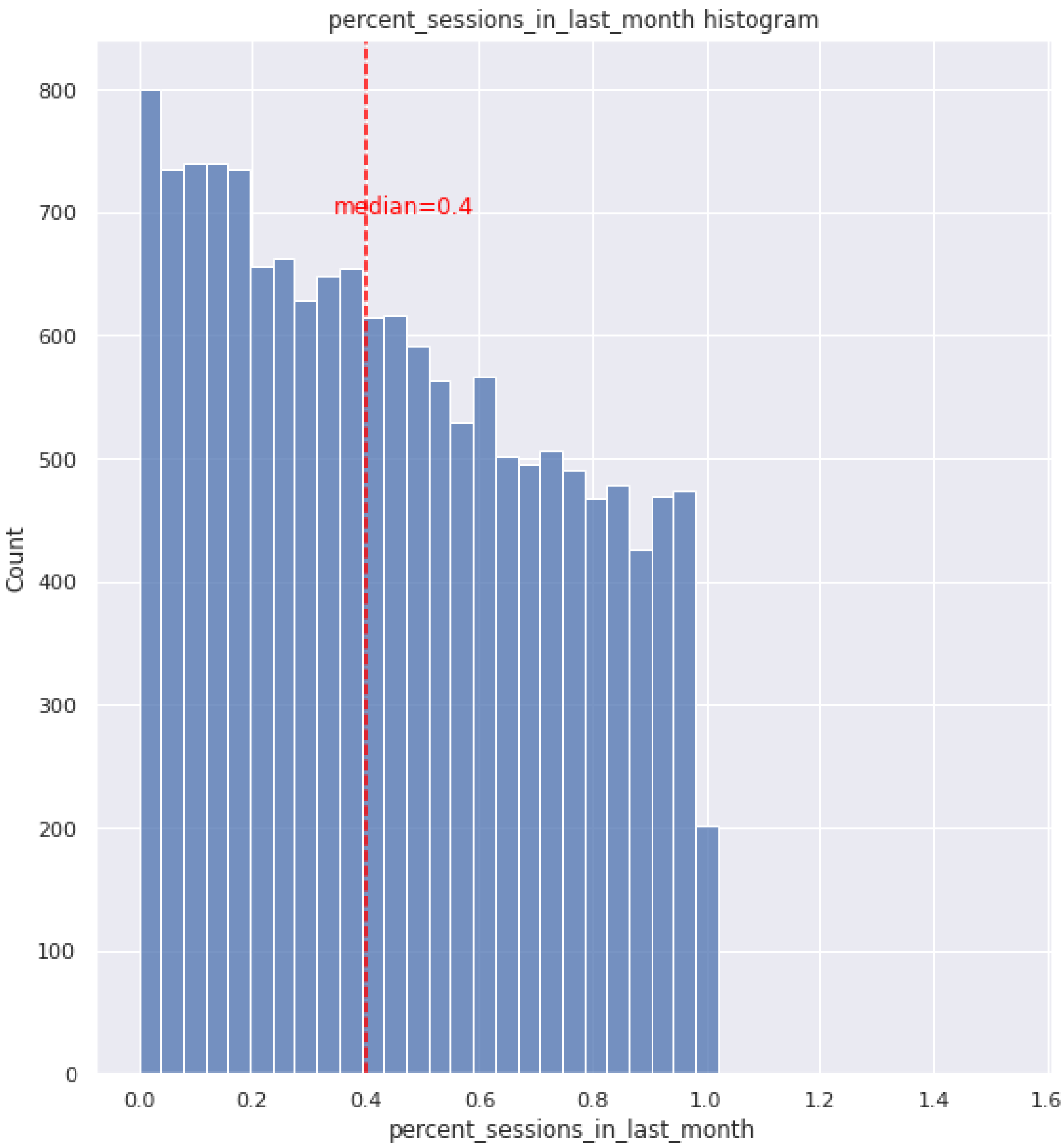
What is the median value of the new column?

```
[53]: print("The median session percenatage for the last month over the total seesion␣
       ↪for the user: ",df['percent_sessions_in_last_month'].median())
```

```
The median session percenatage for the last month over the total seesion for the
user:  0.42309702992763176
```

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

```
[54]: # Histogram
      histogrammer('percent_sessions_in_last_month',True)
```




Check the median value of the `n_days_after_onboarding` variable.

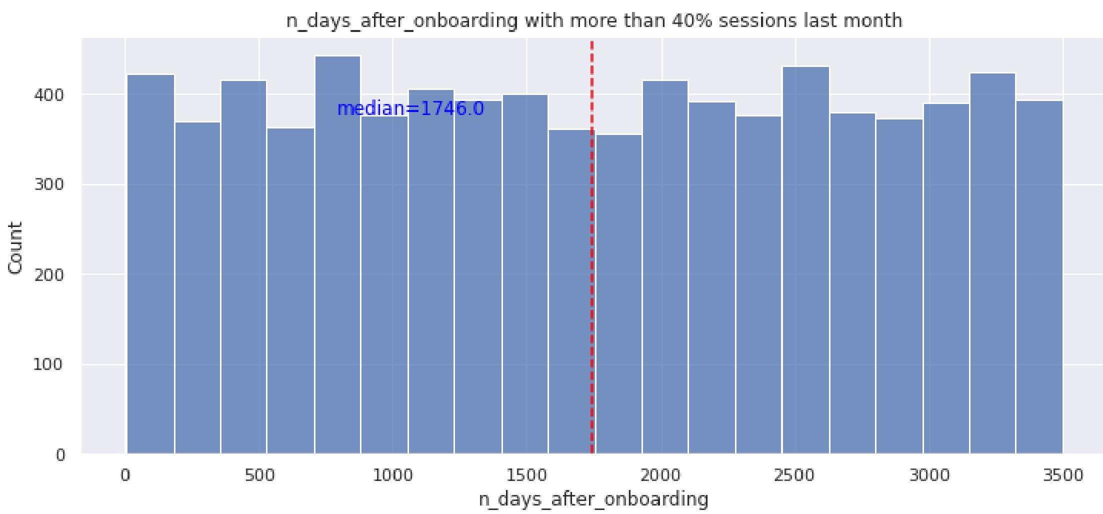
[55]:
```
print("The median number of days for users after onboarding␣
↪",df['n_days_after_onboarding'].median())
```

```
The median number of days for users after onboarding  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.

[56]:
```
# Histogram
plt.figure(figsize=(12,5))
#first filter the data to have only those with more then 40% sessions last month
data=df[df['percent_sessions_in_last_month']>=0.40]
median=round(data['n_days_after_onboarding'].median(), 1)
ax = sp.histplot(x=data['n_days_after_onboarding'])          # Plot the␣
↪histogram
plt.axvline(median, color='red', linestyle='--')         # Plot the median line
ax.text(0.25, 0.85, f'median={median}', color='blue',
            ha="left", va="top", transform=ax.transAxes)
plt.title('n_days_after_onboarding with more than 40% sessions 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.

### 4.3.3 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.

[57]:
```python
def prec95(column):
    # Calculate 75th percentile of annual strikes
    percentile95 = df[column].quantile(0.95)
    df.loc[ df[column] >percentile95, column] = percentile95
```

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

[58]:
```python
prec95('sessions')
prec95('drives')
prec95('total_sessions')
prec95('driven_km_drives')
prec95('duration_minutes_drives')
```

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

[59]:
```python
df.describe()
```

[59]:

| | sessions | drives | total_sessions | n_days_after_onboarding | \ |
|---|---|---|---|---|---|
| count | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | |
| mean | 76.568705 | 64.058204 | 184.031320 | 1749.837789 | |
| std | 67.297958 | 55.306924 | 118.600463 | 1008.513876 | |
| min | 0.000000 | 0.000000 | 0.220211 | 4.000000 | |
| 25% | 23.000000 | 20.000000 | 90.661156 | 878.000000 | |
| 50% | 56.000000 | 48.000000 | 159.568115 | 1741.000000 | |
| 75% | 112.000000 | 93.000000 | 254.192341 | 2623.500000 | |
| max | 243.000000 | 201.000000 | 454.363204 | 3500.000000 | |

| | total_navigations_fav1 | total_navigations_fav2 | driven_km_drives | \ |
|---|---|---|---|---|
| count | 14999.000000 | 14999.000000 | 14999.000000 | |
| mean | 121.605974 | 29.672512 | 3939.632764 | |
| std | 148.121544 | 45.394651 | 2216.041510 | |
| min | 0.000000 | 0.000000 | 60.441250 | |
| 25% | 9.000000 | 0.000000 | 2212.600607 | |
| 50% | 71.000000 | 9.000000 | 3493.858085 | |
| 75% | 178.000000 | 43.000000 | 5289.861262 | |
| max | 1236.000000 | 415.000000 | 8889.794236 | |

| | duration_minutes_drives | activity_days | driving_days | \ |
|---|---|---|---|---|
| count | 14999.000000 | 14999.000000 | 14999.000000 | |
| mean | 1789.647426 | 15.537102 | 12.179879 | |
| std | 1222.705167 | 9.004655 | 7.824036 | |
| min | 18.282082 | 0.000000 | 0.000000 | |
| 25% | 835.996260 | 8.000000 | 5.000000 | |
| 50% | 1478.249859 | 16.000000 | 12.000000 | |
| 75% | 2464.362632 | 23.000000 | 19.000000 | |
| max | 4668.899349 | 31.000000 | 30.000000 | |

```
       km_per_driving_day  percent_sessions_in_last_month
count        1.499900e+04                    14999.000000
mean                  inf                        0.449255
std                   NaN                        0.286919
min          3.022063e+00                        0.000000
25%          1.672804e+02                        0.196221
50%          3.231459e+02                        0.423097
75%          7.579257e+02                        0.687216
max                   inf                        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

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.

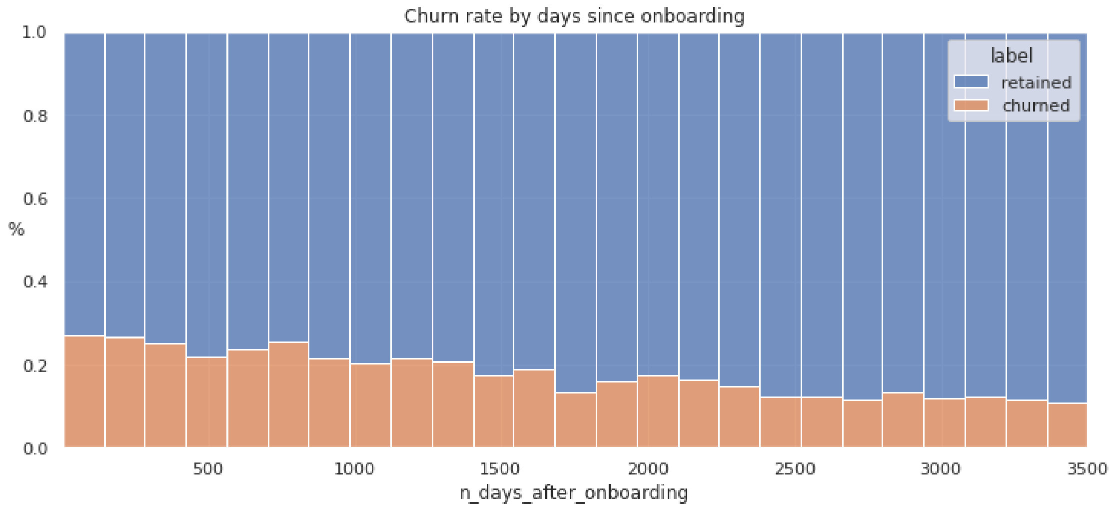
Use the following two code blocks (add more blocks if you like) to do additional EDA you feel is important based on the given scenario.

[67]:
```
#Analysis of newer users using "n_days_after_onboarding" column
df['n_months_after_onboarding']=(df['n_days_after_onboarding']/30)
```

[70]:
```
df.drop('n_months_after_onboarding',axis=1,inplace=True)
```

[79]:
```
plt.figure(figsize=(12,5))
sp.histplot(data=df,
             x='n_days_after_onboarding',
             #bins=range(0,1500,20),
             hue='label',
             multiple='fill'
```

```
              #discrete=True
             )
plt.ylabel('%', rotation=0)
plt.title('Churn rate by days since onboarding');
```




#### 4.4.2 Task 4b. Conclusion

**Questions:**

1. most of the variables are rigth skewed
2. Why does the activity and driving days have different max value?
3. ~18% chruned and 82% users retained
4. What factors correlated with user churn? How?
5. Did newer uses have greater representation in this dataset than users with longer tenure? No, the histogram plot shows that the data is normally ditributed and churn has no correlation with the days since onboarding column

[ ]:

```

```