Waze Project

Go Beyond the Numbers: Translate Data into Insights

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

I check my inbox and notice a new message from Chidi Ga, my team’s Senior Data Analyst. Chidi is pleased with the work I have already completed and requests my 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 the data exploration and visualization.

Project: Exploratory data analysis

In this activity, we 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 we 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

We follow the instructions and answer the question below to complete the activity. Then, we will complete an executive summary using the questions listed on the [PACE Strategy Document (https://docs.google.com/document/d/1iSHdbfQR6w8RClJNWai8oJXn9tQmYoTKn6QohuaK4-s/template/preview?resourcekey=0-ZIHnbxL1dd2u9A47iEVXvg)](https://docs.google.com/document/d/1iSHdbfQR6w8RClJNWai8oJXn9tQmYoTKn6QohuaK4-s/template/preview?resourcekey=0-ZIHnbxL1dd2u9A47iEVXvg).

Visualize a story in Python

PACE stages

Throughout these project notebooks, we'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.

PACE: Plan

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

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.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

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

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

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?

Answers:

1. The data is already in a structured format. Each row represents a user.
2. Yes, 700 rows have `label` missing. Other variables have no missing values.

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 we eliminate, knowing they won’t solve our problem scenario?
3. How would we check for missing data? And how would we handle missing data (if any)?
4. How would we check for outliers? And how would handle outliers (if any)?

Answers:

1. SInce we are interested in user churn, the `label` column is essential. Besides `label`, variables that tie to user behaviors will be the most applicable. All variables tie to user behavior except `ID`.
2. `ID` can be dropped from the analysis since we are not interested in identifying a particular user. `ID` does not provide meaningful information about the churn (unless `ID` is assigned based on user sign-up time).
3. To check for missing data, we can use `df.info()` and inspect the `Non-Null Count` column. The difference between the number of non-nulls and the number of rows in the data is the number of missing values for the variable.

   If the missing data are missing completely at random (MCAR), meaning that the reason for missingness is independent of the data values themselves, we can proceed with a complete-case analysis by removing the rows with missing values. Otherwise, we need to investigate the root cause of the missingness and make sure it won't interfere with the statistical inference and modeling.
4. See the previous exemplar responses for the outlier question.

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 we can always refer back to if needed.

In [3]:
```
df.head(10)
```

Out[3]:

	ID	label	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1
0	0	retained	283	226	296.748273	2276	208
1	1	retained	133	107	326.896596	1225	19
2	2	retained	114	95	135.522926	2651	0
3	3	retained	49	40	67.589221	15	322
4	4	retained	84	68	168.247020	1562	166
5	5	retained	113	103	279.544437	2637	0
6	6	retained	3	2	236.725314	360	185
7	7	retained	39	35	176.072845	2999	0
8	8	retained	57	46	183.532018	424	0
9	9	churned	84	68	244.802115	2997	72

In [4]:
```
df.size
```

Out[4]: 194987

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

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

Out[5]:

	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	
mean	7499.000000	80.633776	67.281152	189.964447	1749.837789	
std	4329.982679	80.699065	65.913872	136.405128	1008.513876	
min	0.000000	0.000000	0.000000	0.220211	4.000000	
25%	3749.500000	23.000000	20.000000	90.661156	878.000000	
50%	7499.000000	56.000000	48.000000	159.568115	1741.000000	
75%	11248.500000	112.000000	93.000000	254.192341	2623.500000	
max	14998.000000	743.000000	596.000000	1216.154633	3500.000000	

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

In [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
```

PACE: Construct

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

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

1. What are some ways to identify outliers?
    - Use numpy functions to investigate the `mean()` and `median()` of the data and understand range of data values
    - Use a boxplot to visualize the distribution of the data
2. How do you make the decision to keep or exclude outliers from any future models?
    - There are three main options for dealing with outliers: keeping them as they are, deleting them, or reassigning them. Whether we keep outliers as they are, delete them, or reassign values is a decision that we make on a dataset-by-dataset basis, according to what your goals are for the model we are planning to construct. To help us make the decision, we can start with these general guidelines:
        - Delete them: If we are sure the outliers are mistakes, typos, or errors and the dataset will be used for modeling or machine learning, then we are more likely to decide to delete outliers. Of the three choices, we'll use this one the least.
        - Reassign them: If the dataset is small and/or the data will be used for modeling or machine learning, we are more likely to choose a path of deriving new values to replace the outlier values.
        - Leave them: For a dataset that we plan to do EDA/analysis on and nothing else, or for a dataset we are preparing for a model that is resistant to outliers, it is most likely that we are going to leave them in.

Task 3a. Visualizations

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

Now that we know which data columns we'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

Answer:

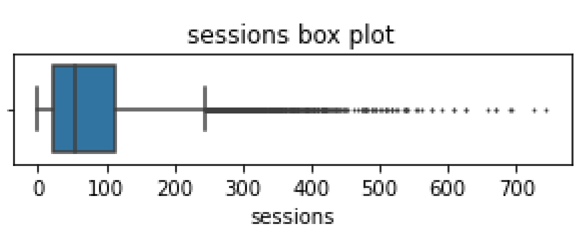
- Box plots will be helpful to determine outliers and where the bulk of the data points reside in terms of `drives` , `sessions` and all other continuous numeric variables
- Histograms are essential to understand the distribution of variables
- Scatter plots will be helpful to visualize relationships between variables
- Bar charts are useful for communicating levels and quantities, especially for categorical information

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

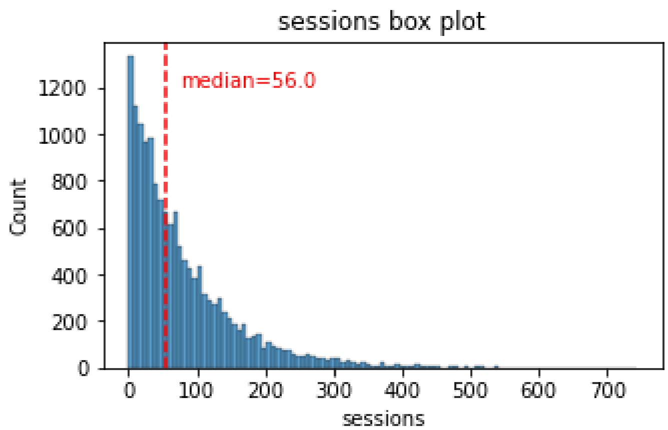
`sessions`

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

In [7]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['sessions'], fliersize=1)
plt.title('sessions box plot');
```



In [8]:
```
# Histogram
plt.figure(figsize=(5,3))
sns.histplot(x=df['sessions'])
median = df['sessions'].median()
plt.axvline(median, color='red', linestyle='--')
plt.text(75,1200, 'median=56.0', color='red')
plt.title('sessions box plot');
```

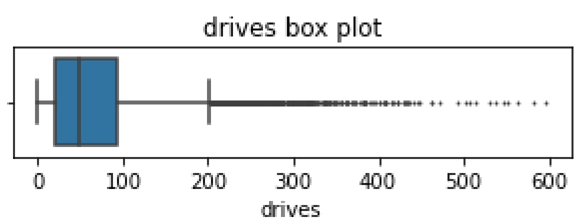


The `sessions` variable is a right-skewed distribution with half of the observations having 56 or fewer sessions. However, as indicated by the boxplot, some users have more than 700.

drives

An occurrence of driving at least 1 km during the month

In [9]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['drives'], fliersize=1)
plt.title('drives box plot');
```

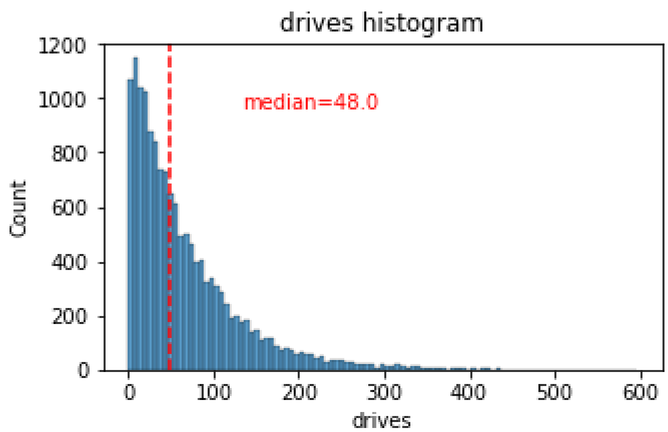


As we perform EDA, we'll find that many tasks get repeated, such as plotting histograms of features. Remember that whenever we find ourselves copy/pasting code, it's worth considering whether a function would help make the work more efficient. Sometimes it's not worth it. Other times, defining a function will help a lot.

The following code block defines a function that helps plot histograms with a particular style/format using this particular dataset. We don't have to do this, but in this case it's helpful.

In [10]:
```
# Helper function to plot histograms based on the
# format of the `sessions` histogram
def histogrammer(column_str, median_text=True, **kwargs):    # **kwargs = any k
                                                             # from the sns.his
    median=round(df[column_str].median(), 1)
    plt.figure(figsize=(5,3))
    ax = sns.histplot(x=df[column_str], **kwargs)            # Plot the histogr
    plt.axvline(median, color='red', linestyle='--')         # Plot the median
    if median_text==True:                                    # Add median text
        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');
```

In [11]:
```
# Histogram
histogrammer('drives')
```

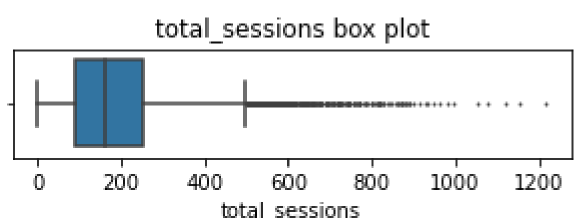


The `drives` information follows a distribution similar to the `sessions` variable. It is right-skewed, approximately log-normal, with a median of 48. However, some drivers had over 400 drives in the last month.

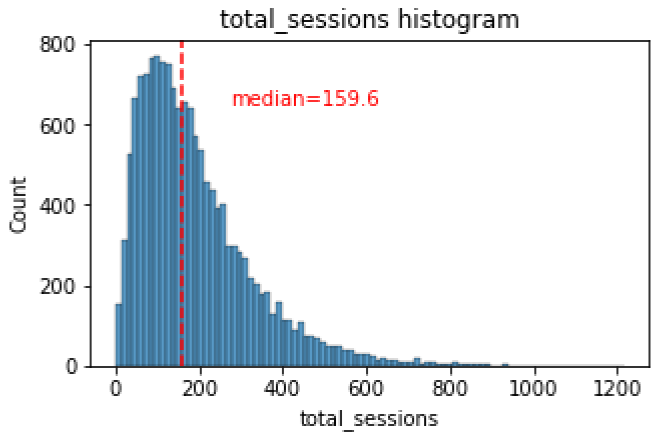
`total_sessions`

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

In [12]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['total_sessions'], fliersize=1)
plt.title('total_sessions box plot');
```



In [13]:
```
# Histogram
histogrammer('total_sessions')
```

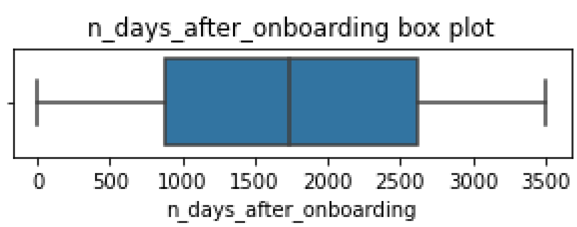


The `total_sessions` is a right-skewed distribution. The median total number of sessions is 159.6. This is interesting information because, if the median number of sessions in the last month was 48 and the median total sessions was ~160, then it seems that a large proportion of a user's total drives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

`n_days_after_onboarding`

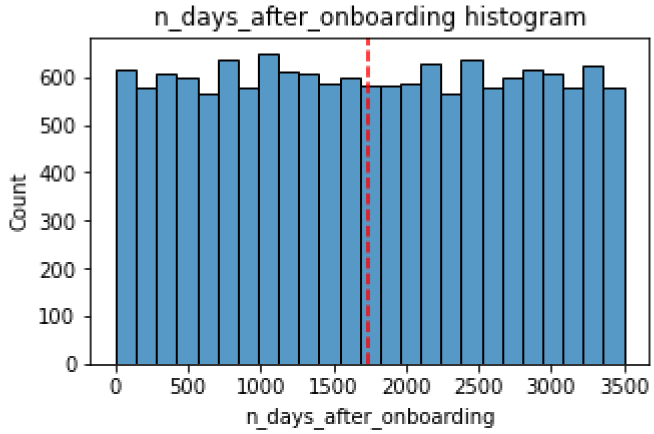
The number of days since a user signed up for the app

In [14]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['n_days_after_onboarding'], fliersize=1)
plt.title('n_days_after_onboarding box plot');
```



In [15]:
```
# Histogram
histogrammer('n_days_after_onboarding', median_text=False)
```

```
Median: 1741.0
```

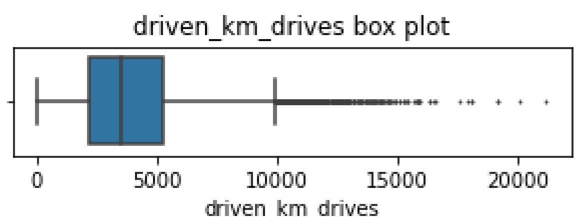


The total user tenure (i.e., number of days since onboarding) is a uniform distribution with values ranging from near-zero to ~3,500 (~9.5 years).

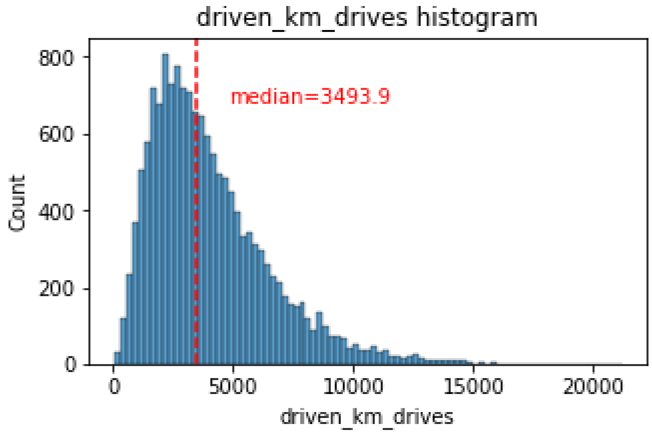
driven_km_drives

Total kilometers driven during the month

In [16]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['driven_km_drives'], fliersize=1)
plt.title('driven_km_drives box plot');
```



In [17]:
```
# Histogram
histogrammer('driven_km_drives')
```

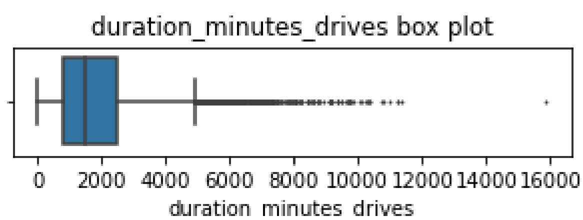


The number of drives driven in the last month per user is a right-skewed distribution with half the users driving under 3,495 kilometers. As we discovered in the analysis from the previous course, the users in this dataset drive *a lot*. The longest distance driven in the month was over half the circumferene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

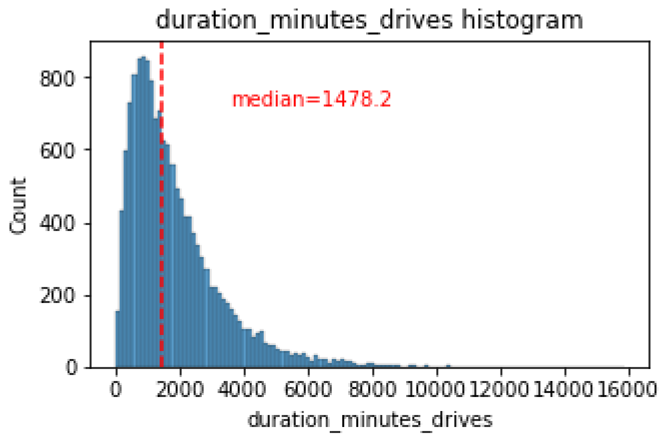
duration_minutes_drives

Total duration driven in minutes during the month

In [18]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['duration_minutes_drives'], fliersize=1)
plt.title('duration_minutes_drives box plot');
```



In [19]:
```
# Histogram
histogrammer('duration_minutes_drives')
```

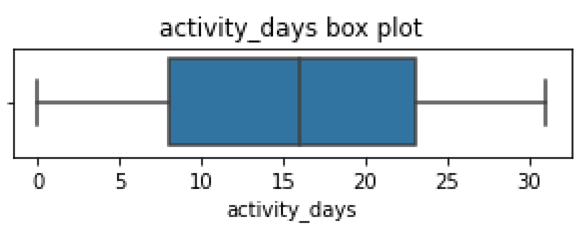


The `duration_minutes_drives` variable has a heavily skewed right tail. Half of the users drove less than ~1,478 minutes (~25 hours), but some users clocked over 250 hours over the month.

`activity_days`

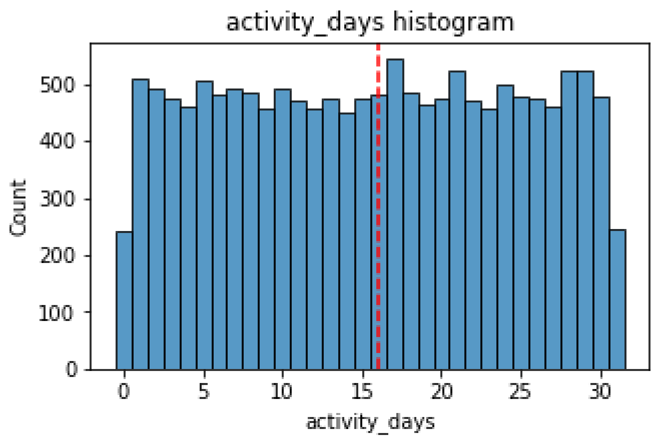
Number of days the user opens the app during the month

In [20]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['activity_days'], fliersize=1)
plt.title('activity_days box plot');
```



In [21]:
```
# Histogram
histogrammer('activity_days', median_text=False, discrete=True)
```

```
Median: 16.0
```



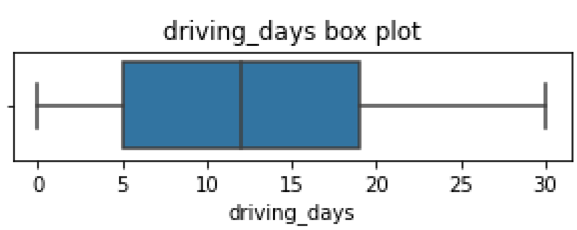
Within the last month, users opened the app a median of 16 times. The box plot reveals a centered distribution. The histogram shows a nearly uniform distribution of ~500 people opening the app on each count of days. However, there are ~250 people who didn't open the app at all and ~250 people who opened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

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

`driving_days`

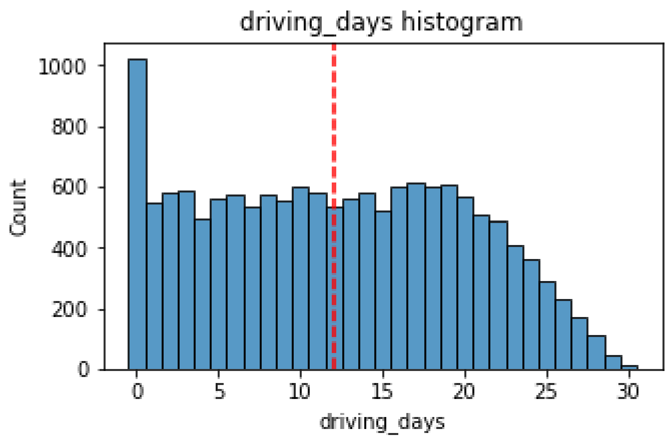
Number of days the user drives (at least 1 km) during the month

In [22]:
```
# Box plot
plt.figure(figsize=(5,1))
sns.boxplot(x=df['driving_days'], fliersize=1)
plt.title('driving_days box plot');
```



In [23]:
```
# Histogram
histogrammer('driving_days', median_text=False, discrete=True)
```

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.

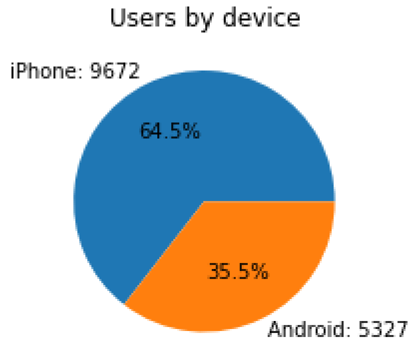
However, there were almost twice as many users (~1,000 vs. ~550) who did not drive at all during the month. This might seem counterintuitive when considered together with the information from `activity_days` . That variable had ~500 users opening the app on each of most of the day counts, but there were 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 we do not plot a box plot for it. A good plot for a binary categorical variable is a pie chart.

In [24]:
```
# Pie chart
fig = plt.figure(figsize=(3,3))
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');
```



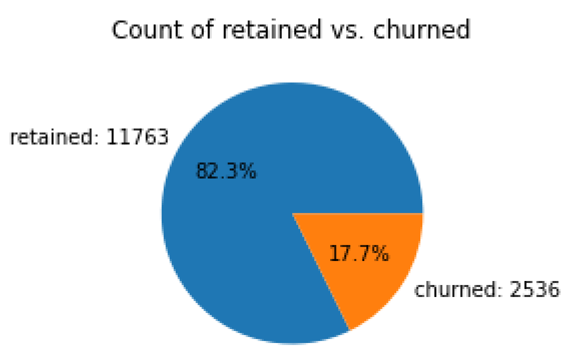
There are nearly twice as many iPhone users as Android users represented in this data.

`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.

In [25]:
```
# Pie chart
fig = plt.figure(figsize=(3,3))
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('Count of retained vs. churned');
```



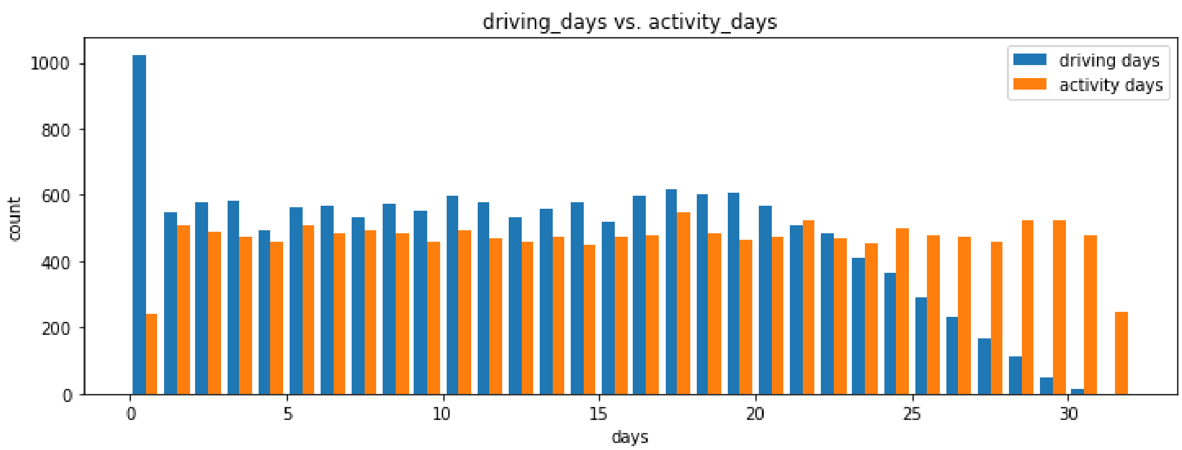
Less than 18% of the users churned.

`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, we 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`.

In [26]:
```
# 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

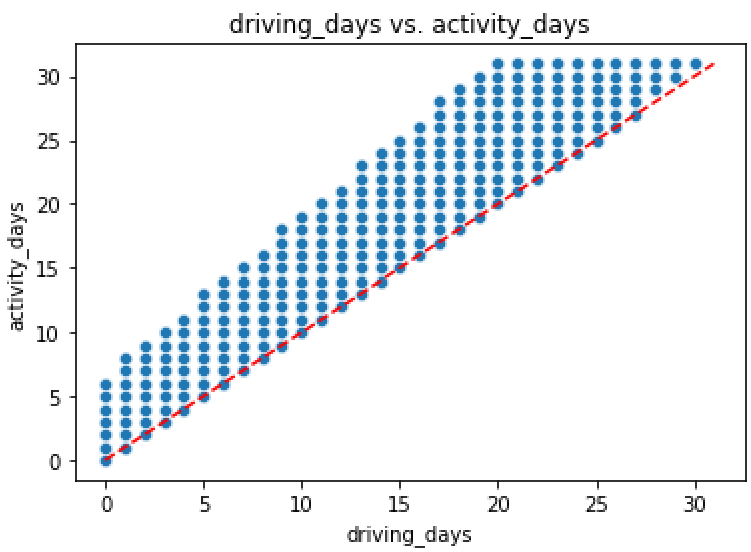
In [27]:
```
print(df['driving_days'].max())
print(df['activity_days'].max())
```

```
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.

In [28]:
```
# 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='red', linestyle='--');
```

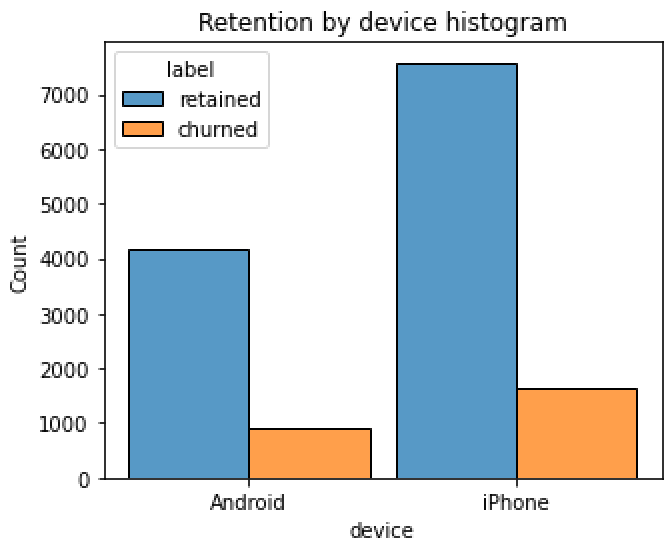


Notice that there is a theoretical limit. If we use the app to drive, then by definition it must count as a day-use as well. In other words, we cannot have more drive-days than activity-days. None of the samples in this data violate this rule, which is good.

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.

In [29]:
```
# 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, we 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.

In [30]:
```
# 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()
```

Out[30]:
```
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 we 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.

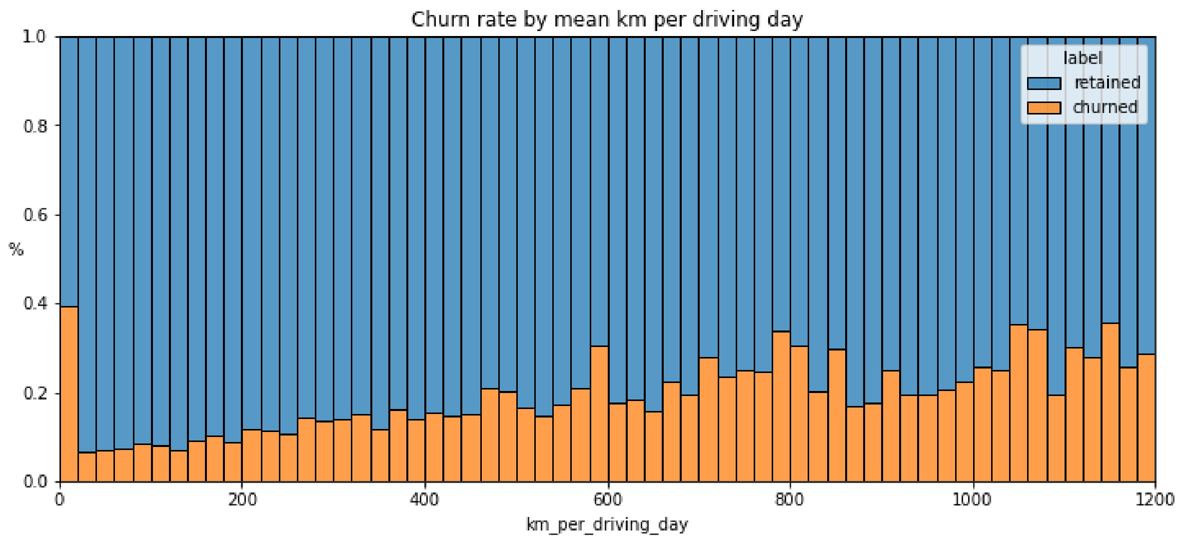
In [31]:
```
# 1. Convert infinite values to zero
df.loc[df['km_per_driving_day']==np.inf, 'km_per_driving_day'] = 0

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

Out[31]:
```
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()` (https://seaborn.pydata.org/generated/seaborn.histplot.html) function

In [32]:
```
# 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');
```

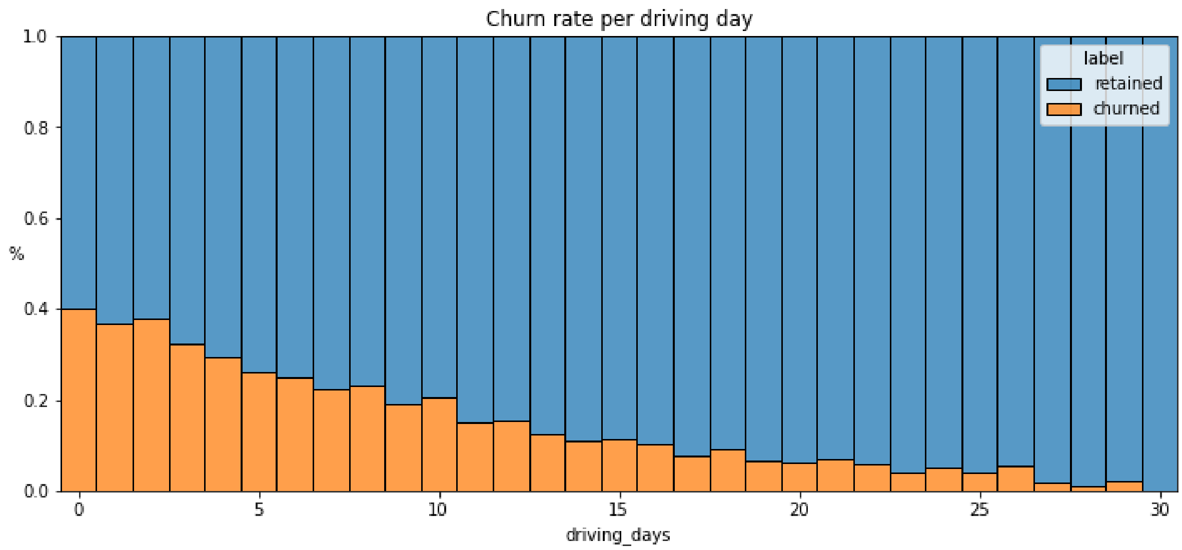


The churn rate tends to increase as the mean daily distance driven increases, confirming what was found in the previous course. It would be worth investigating further the reasons for long-distance users to discontinue using the app.

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.

In [33]:
```
# 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 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.

In [34]:
```
df['percent_sessions_in_last_month'] = df['sessions'] / df['total_sessions']
```

What is the median value of the new column?

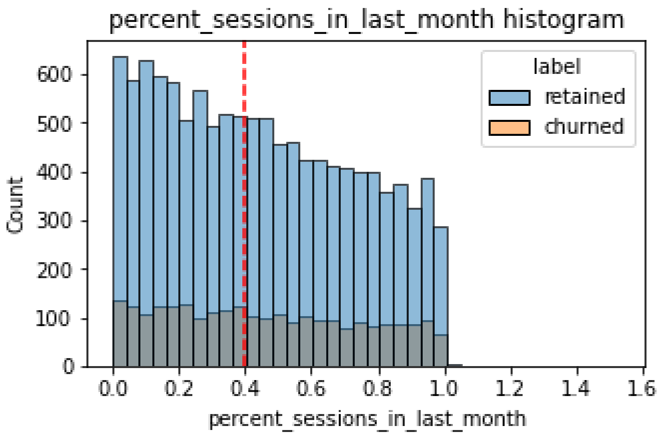
In [35]:
```
df['percent_sessions_in_last_month'].median()
```

Out[35]:
```
0.42309702992763176
```

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

In [36]:
```
# 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.

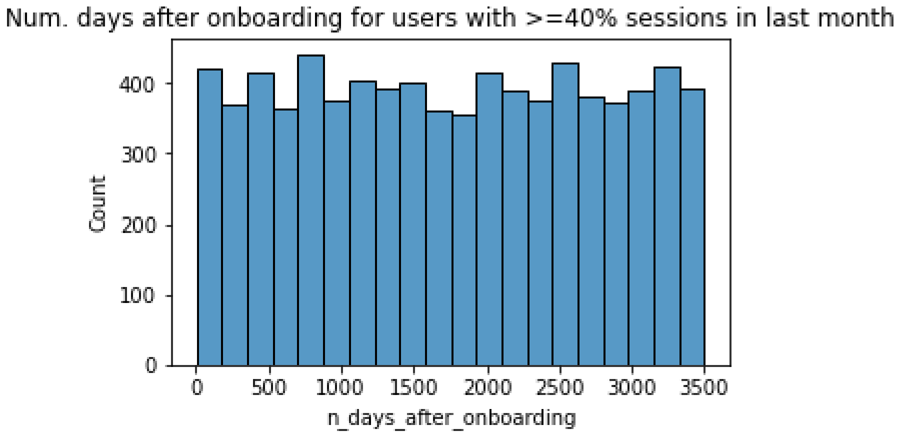
In [37]:
```
df['n_days_after_onboarding'].median()
```

Out[37]:
```
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.

In [38]:
```
# 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 mo
```



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.

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 we'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.

In [39]:
```
def outlier_imputer(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, percen
```

Next, apply that function to the following columns:

- `sessions`
- `drives`
- `total_sessions`

- driven_km_drives
- duration_minutes_drives

In [40]:
```
for column in ['sessions', 'drives', 'total_sessions',
               'driven_km_drives', 'duration_minutes_drives']:
               outlier_imputer(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.

In [41]:
```
df.describe()
```

Out[41]:

	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	
mean	7499.000000	76.568705	64.058204	184.031320	1749.837789	
std	4329.982679	67.297958	55.306924	118.600463	1008.513876	
min	0.000000	0.000000	0.000000	0.220211	4.000000	
25%	3749.500000	23.000000	20.000000	90.661156	878.000000	
50%	7499.000000	56.000000	48.000000	159.568115	1741.000000	
75%	11248.500000	112.000000	93.000000	254.192341	2623.500000	
max	14998.000000	243.000000	201.000000	454.363204	3500.000000	

Conclusion

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

Perhaps we feel that the more deeply we 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.

PACE: Execute

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

Task 4a. Results and evaluation

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

Pro tip: Put yourself in the 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 we consider color, contrast, emphasis, and labeling?

We have learned ....

- There is missing data in the user churn label, so we might need further data processing before further analysis.
- There are many outlying observations for drives, so we might consider a variable transformation to stabilize the variation.
- The number of drives and the number of sessions are both strongly correlated, so they might provide redundant information when we incorporate both in a model.
- On average, retained users have fewer drives than churned users.

My other questions are ....

- How does the missingness in the user churn label arise?
- Who are the users with an extremely large number of drives? Are they ridesharing drivers or commercial drivers?
- Why do retained users have fewer drives than churned users? Is it because churned users have a longer history of using the Waze app?
- What is the user demographic for retained users and churned users?

My client would likely want to know ...

- What are the key variables associated with user churn?
- Can we implement policies to reduce user churn?

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.

In [42]:
```
df['monthly_drives_per_session_ratio'] = (df['drives']/df['sessions'])
```

In [43]:
```
df.head(10)
```

Out[43]:

ı_km_drives	duration_minutes_drives	activity_days	driving_days	device	km_per_driving_day	peı
2628.845068	1985.775061	28	19	Android	138.360267	
8889.794236	3160.472914	13	11	iPhone	1246.901868	
3059.148818	1610.735904	14	8	Android	382.393602	
913.591123	587.196542	7	3	iPhone	304.530374	
3950.202008	1219.555924	27	18	Android	219.455667	
901.238699	439.101397	15	11	iPhone	81.930791	
5249.172828	726.577205	28	23	iPhone	228.224906	
7892.052468	2466.981741	22	20	iPhone	394.602623	
2651.709764	1594.342984	25	20	Android	132.585488	
6043.460295	2341.838528	7	3	iPhone	2014.486765	

Task 4b. Conclusion

Now that we've explored and visualized the data, the next step is to share the findings with Harriet Hadzic, Waze's Director of Data Analysis. Consider the following questions as we prepare to write the executive summary. Think about key points we 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 we notice in the variables? What did this tell us about the data?

> *Nearly all the variables were either very right-skewed or uniformly distributed. For the right-skewed distributions, this means that most users had values in the lower end of the range for that variable. For the uniform distributions, this means that users were generally equally likely to have values anywhere within the range for that variable.*

2. Was there anything that led us to believe the data was erroneous or problematic in any way?

> *Most of the data was not problematic, and there was no indication that any single variable was completely wrong. However, several variables had highly improbable or perhaps even impossible outlying values, such as `driven_km_drives` . Some of the monthly variables also might be problematic, such as `activity_days` and `driving_days` , because one has a max value of 31 while the other has a max value of 30, indicating that data collection might not have occurred in the same month for both of these variables.*

3. Did our investigation give rise to further questions that we would like to explore or ask the Waze team about?

> *Yes. we'd want to ask the Waze data team to confirm that the monthly variables were collected during the same month, given the fact that some have max values of 30 days while others have 31 days. we'd also want to learn why so many long-time users suddenly started using the app so much in just the last month. Was there anything that changed in the last month that might prompt this kind of behavior?*

4. What percentage of users churned and what percentage were retained?

> *Less than 18% of users churned, and ~82% were retained.*

5. What factors correlated with user churn? How?

> *Distance driven per driving day had a positive correlation with user churn. The farther a user drove on each driving day, the more likely they were to churn. On the other hand, number of driving days had a negative correlation with churn. Users who drove more days of the last month were less likely to churn.*

6. Did newer uses have greater representation in this dataset than users with longer tenure? How do you know?

> *No. Users of all tenures from brand new to ~10 years were relatively evenly represented in the data. This is borne out by the histogram for `n_days_after_onboarding` , which reveals a uniform distribution for this variable.*