In [3]:
```
import pandas as pd

file_path = r'C:\Users\chuck\Documents\Jupyter Notebooks\Waze_dataset.csv'

df = pd.read_csv(file_path)

import numpy as np
```

In [4]:
```
df.head(10)
```

Out[4]:

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

Out[5]:
```
<bound method DataFrame.info of           ID     label  sessions  drives  tot
al_sessions  \
0          0  retained       283     226      296.748273
1          1  retained       133     107      326.896596
2          2  retained       114      95      135.522926
3          3  retained        49      40       67.589221
4          4  retained        84      68      168.247020
...      ...       ...       ...     ...             ...
14994  14994  retained        60      55      207.875622
14995  14995  retained        42      35      187.670313
14996  14996  retained       273     219      422.017241
14997  14997   churned       149     120      180.524184
14998  14998  retained        73      58      353.419797

       n_days_after_onboarding  total_navigations_fav1  \
0                         2276                     208
1                         1225                      19
2                         2651                       0
3                           15                     322
4                         1562                     166
...                        ...                     ...
14994                      140                     317
14995                     2505                      15
14996                     1873                      17
14997                     3150                      45
14998                     3383                      13

       total_navigations_fav2  driven_km_drives  duration_minutes_drives  \
0                           0       2628.845068              1985.775061
1                          64      13715.920550              3160.472914
2                           0       3059.148818              1610.735904
3                           7        913.591123               587.196542
4                           5       3950.202008              1219.555924
...                       ...               ...                      ...
14994                       0       2890.496901              2186.155708
14995                      10       4062.575194              1208.583193
14996                       0       3097.825028              1031.278706
14997                       0       4051.758549               254.187763
14998                      51       6030.498773              3042.436423

       activity_days  driving_days   device
0                 28            19  Android
1                 13            11   iPhone
2                 14             8  Android
3                  7             3   iPhone
4                 27            18  Android
...              ...           ...      ...
14994             25            17   iPhone
14995             25            20  Android
14996             18            17   iPhone
14997              6             6   iPhone
14998             14            13   iPhone

[14999 rows x 13 columns]>
```

In [7]:
```
# Isolate rows with null values
null_df = df[df['label'].isnull()]
null_df.describe()
```

Out[7]:

	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_navig
count	700.000000	700.000000	700.000000	700.000000	700.000000	
mean	7405.584286	80.837143	67.798571	198.483348	1709.295714	
std	4306.900234	79.987440	65.271926	140.561715	1005.306562	
min	77.000000	0.000000	0.000000	5.582648	16.000000	
25%	3744.500000	23.000000	20.000000	94.056340	869.000000	
50%	7443.000000	56.000000	47.500000	177.255925	1650.500000	
75%	11007.000000	112.250000	94.000000	266.058022	2508.750000	
max	14993.000000	556.000000	445.000000	1076.879741	3498.000000	

In [9]:
```
# Isolate rows without null values
not_null_df = df[~df['label'].isnull()]
not_null_df.describe()
```

Out[9]:

	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_
count	14299.000000	14299.000000	14299.000000	14299.000000	14299.000000	
mean	7503.573117	80.623820	67.255822	189.547409	1751.822505	
std	4331.207621	80.736502	65.947295	136.189764	1008.663834	
min	0.000000	0.000000	0.000000	0.220211	4.000000	
25%	3749.500000	23.000000	20.000000	90.457733	878.500000	
50%	7504.000000	56.000000	48.000000	158.718571	1749.000000	
75%	11257.500000	111.000000	93.000000	253.540450	2627.500000	
max	14998.000000	743.000000	596.000000	1216.154633	3500.000000	

In [10]:
```
# Get count of null values by device
null_df['device'].value_counts()
```

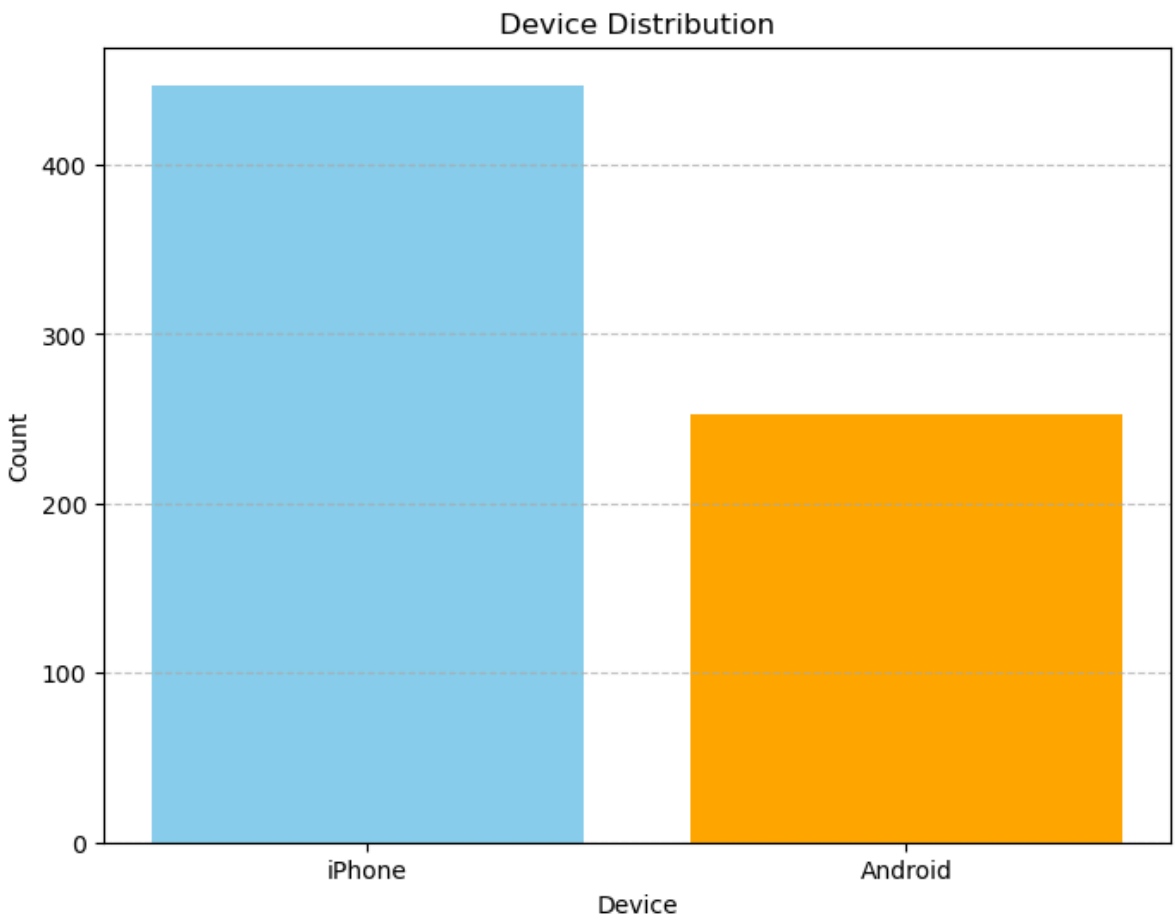
Out[10]:
```
device
iPhone     447
Android    253
Name: count, dtype: int64
```

In [31]:
```
# Create bar chart
import matplotlib.pyplot as plt

# Create a DataFrame with the device counts
device_counts = pd.DataFrame({
    'Device': ['iPhone', 'Android'],
    'Count': [447, 253]
})

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(device_counts['Device'], device_counts['Count'], color=['skyblue', 'ora
plt.xlabel('Device')
plt.ylabel('Count')
plt.title('Device Distribution')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



In [12]:
```
# Caculate % of null iPhones and Andriod
null_df['device'].value_counts(normalize=True)
```

Out[12]:
```
device
iPhone     0.638571
Android    0.361429
Name: proportion, dtype: float64
```

In [13]:
```
# Calculate % of iPhone and Android users in complete dataset
df['device'].value_counts(normalize=True)
```

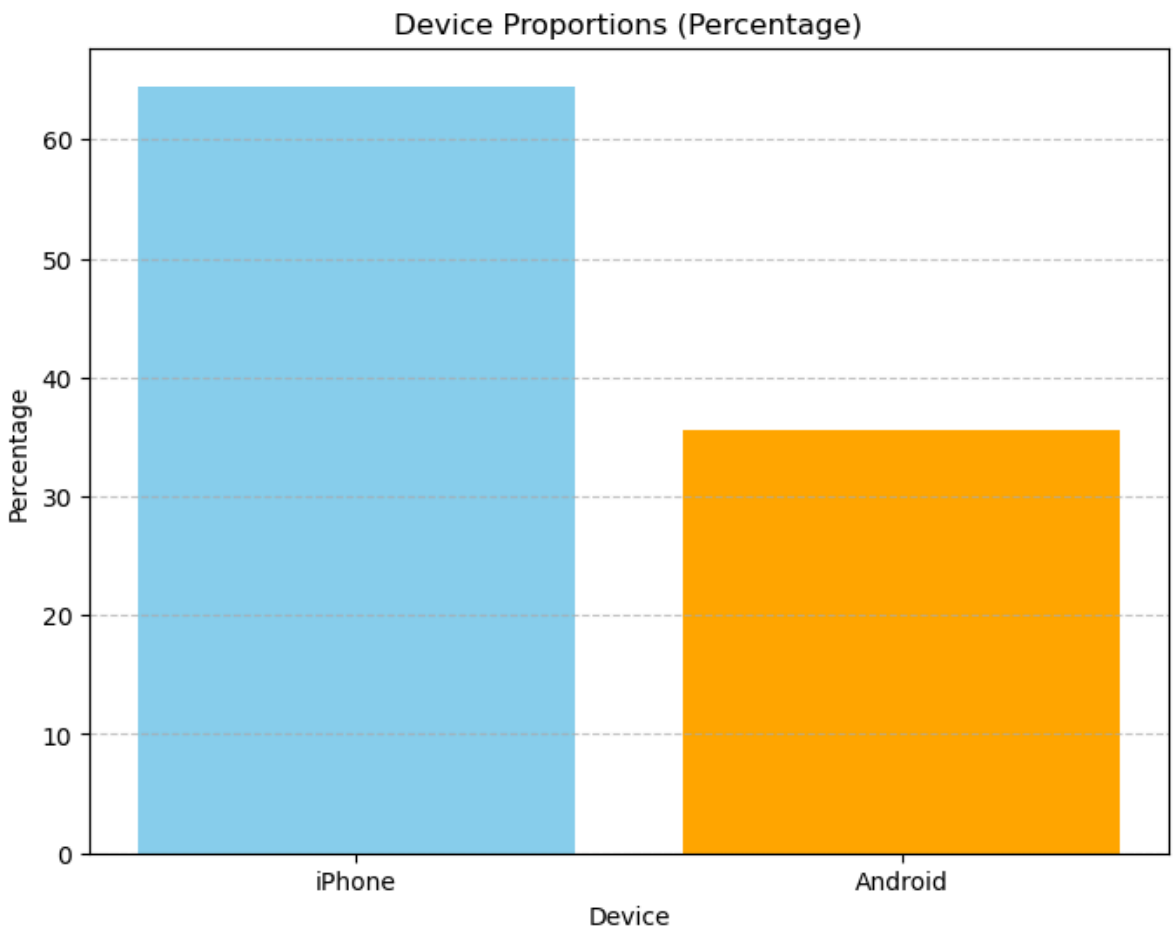
Out[13]:
```
device
iPhone     0.644843
Android    0.355157
Name: proportion, dtype: float64
```

In [34]:
```
# Create a DataFrame with the device proportions
device_proportions = pd.DataFrame({
    'Device': ['iPhone', 'Android'],
    'Proportion': [0.644843, 0.355157]
})

# Convert proportions to percentages
device_proportions['Percentage'] = device_proportions['Proportion'] * 100

# Create a bar chart with percentages
plt.figure(figsize=(8, 6))
plt.bar(device_proportions['Device'], device_proportions['Percentage'], color=[
plt.xlabel('Device')
plt.ylabel('Percentage')
plt.title('Device Proportions (Percentage)')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



In [17]:
```python
# Calculatte counts of churned vs. retained
print(df['label'].value_counts())
print()
print(df['label'].value_counts(normalize=True))
```

```
label
retained    11763
churned      2536
Name: count, dtype: int64

label
retained    0.822645
churned     0.177355
Name: proportion, dtype: float64
```

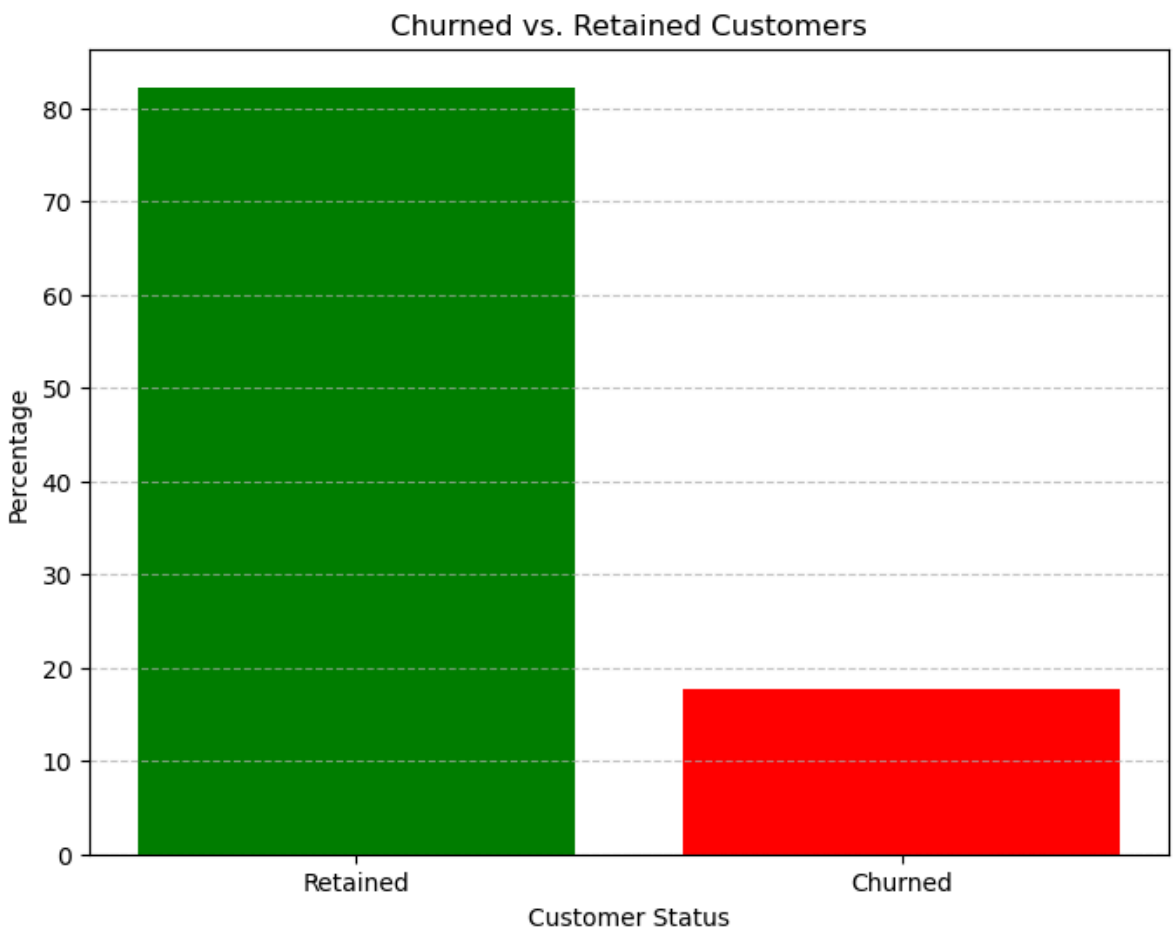
In [35]:
```
# Create a DataFrame with the churned and retained counts
churn_retained_counts = pd.DataFrame({
    'Label': ['Retained', 'Churned'],
    'Count': [11763, 2536]
})

# Calculate the proportions
churn_retained_counts['Proportion'] = churn_retained_counts['Count'] / churn_re

# Convert proportions to percentages
churn_retained_counts['Percentage'] = churn_retained_counts['Proportion'] * 100

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(churn_retained_counts['Label'], churn_retained_counts['Percentage'], co
plt.xlabel('Customer Status')
plt.ylabel('Percentage')
plt.title('Churned vs. Retained Customers')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



In [18]:
```
# Calculate median values of all columns for churned and retained users
df.groupby('label').median(numeric_only=True)
```

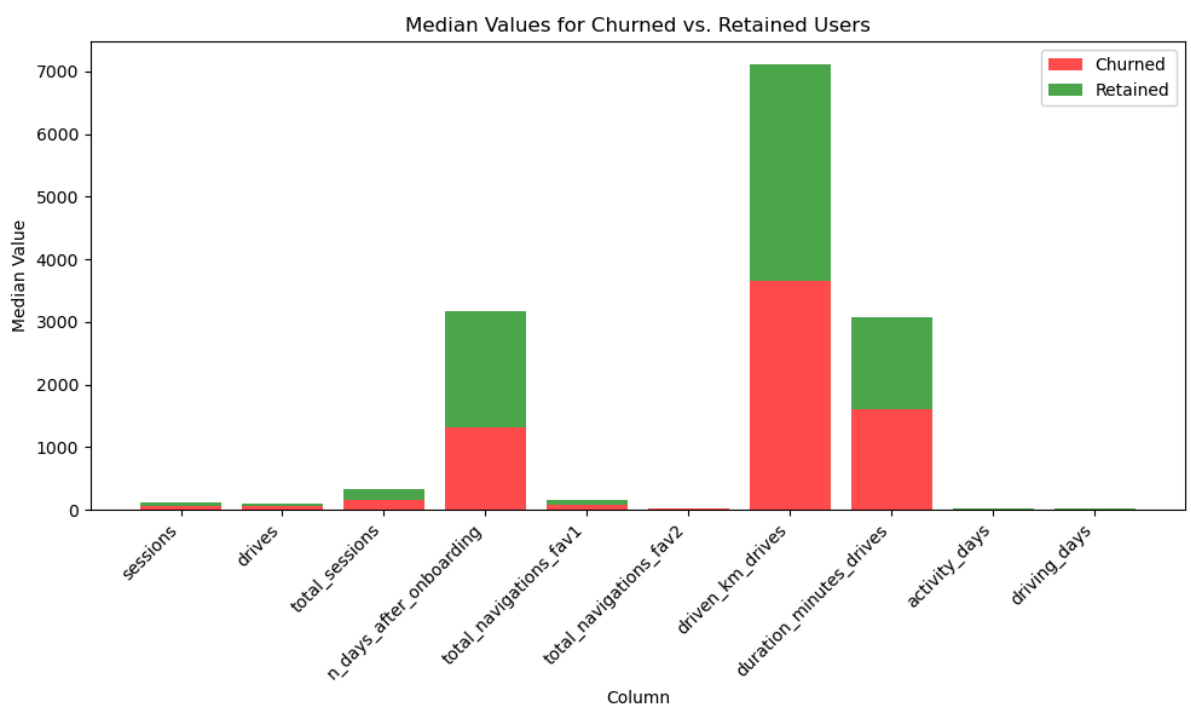
Out[18]:

label	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1
churned	7477.5	59.0	50.0	164.339042	1321.0	84.5
retained	7509.0	56.0	47.0	157.586756	1843.0	68.0

In [36]:
```
# Create a DataFrame with the provided median values
median_values = pd.DataFrame({
    'Column': [
        'sessions', 'drives', 'total_sessions', 'n_days_after_onboarding',
        'total_navigations_fav1', 'total_navigations_fav2', 'driven_km_drives',
        'duration_minutes_drives', 'activity_days', 'driving_days'
    ],
    'Churned': [59.0, 50.0, 164.339042, 1321.0, 84.5, 11.0, 3652.655666, 1607.1
    'Retained': [56.0, 47.0, 157.586756, 1843.0, 68.0, 9.0, 3464.684614, 1458.6
})

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(median_values['Column'], median_values['Churned'], label='Churned', col
plt.bar(median_values['Column'], median_values['Retained'], label='Retained', c
plt.xlabel('Column')
plt.ylabel('Median Value')
plt.title('Median Values for Churned vs. Retained Users')
plt.xticks(rotation=45, ha='right')
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```



In [21]:
```
# Add a column to df 'km_per_drive'
df['km_per_drive'] = df['driven_km_drives'] / df['drives']

# Group by 'label', calculate the median, and isolate for km per drive
median_km_per_drive = df.groupby('label').median(numeric_only=True)[['km_per_dr
median_km_per_drive
```

Out[21]:

label	km_per_drive
churned	74.109416
retained	75.014702

In [22]:
```
# Add a column to df 'km_per_driving_day'
df['km_per_driving_day'] = df['driven_km_drives'] / df['driving_days']

# Group by 'label', calculate the median, and isolate for km per driving day
median_km_per_driving_day = df.groupby('label').median(numeric_only=True)[['km_
median_km_per_driving_day
```

Out[22]:

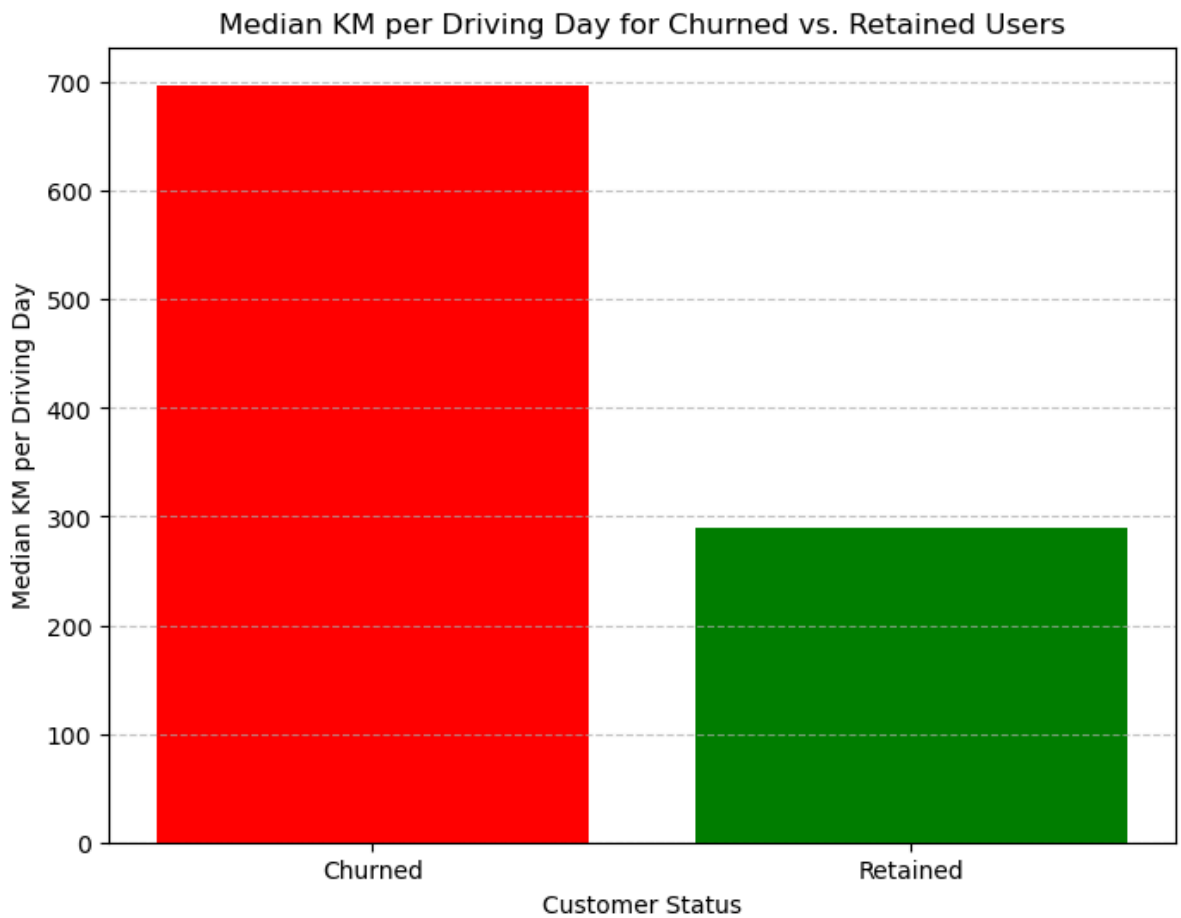
label	km_per_driving_day
churned	697.541999
retained	289.549333

In [38]:
```
import matplotlib.pyplot as plt

# Create a DataFrame with the median km_per_driving_day values
median_km_per_driving_day = pd.DataFrame({
    'Label': ['Churned', 'Retained'],
    'Median km_per_driving_day': [697.541999, 289.549333]
})

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(median_km_per_driving_day['Label'], median_km_per_driving_day['Median k
plt.xlabel('Customer Status')
plt.ylabel('Median KM per Driving Day')
plt.title('Median KM per Driving Day for Churned vs. Retained Users')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



In [23]:
```
# Add a column to df 'drives_per_driving_day'
df['drives_per_driving_day'] = df['drives'] / df['driving_days']

# Group by 'label', calculate the median, and isolate for drives per driving da
median_drives_per_driving_day = df.groupby('label').median(numeric_only=True)[|
median_drives_per_driving_day
```

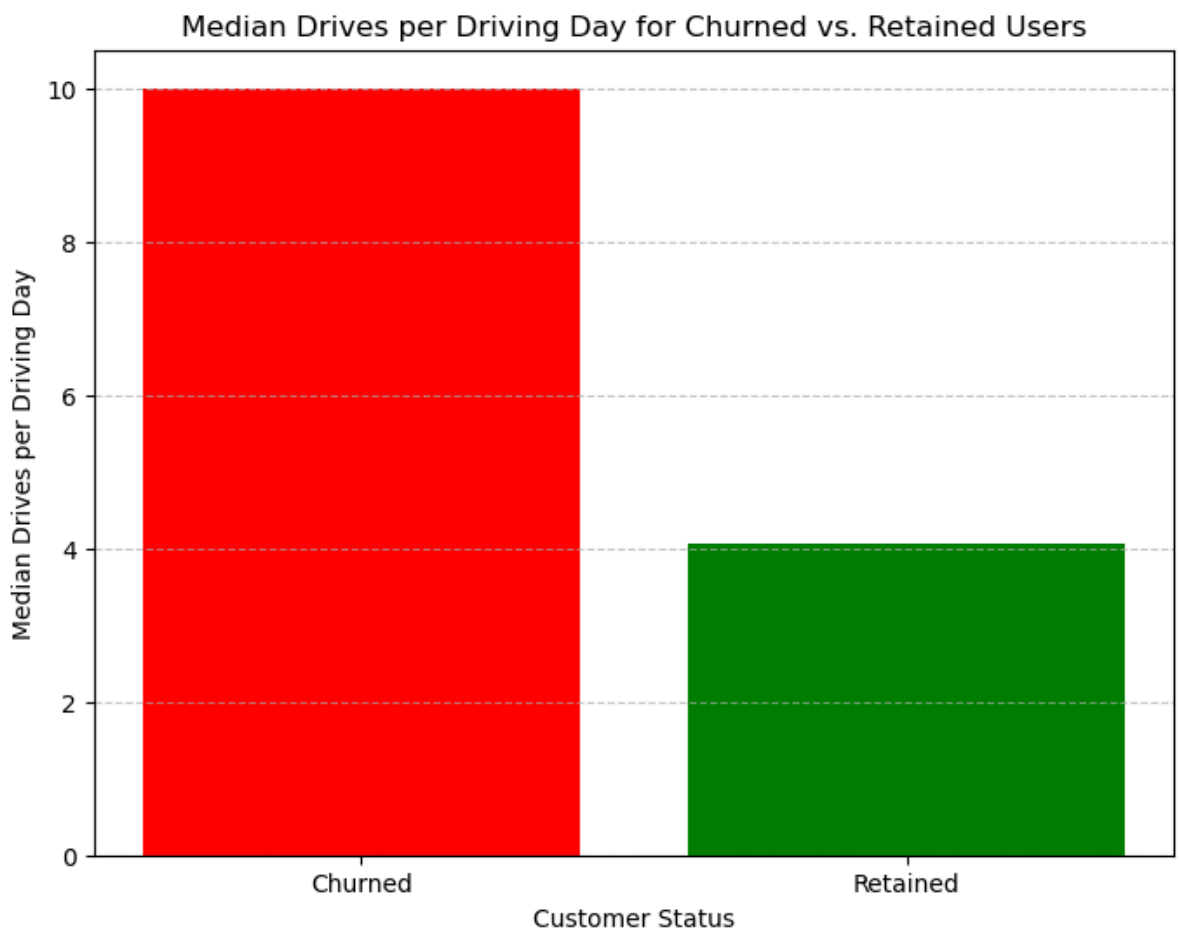
Out[23]:

label	drives_per_driving_day
churned	10.0000
retained	4.0625

In [44]:
```
# Create a DataFrame with the median drives_per_driving_day values
median_drives_per_driving_day = pd.DataFrame({
    'Label': ['Churned', 'Retained'],
    'Median drives_per_driving_day': [10.0, 4.0625]
})

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(median_drives_per_driving_day['Label'], median_drives_per_driving_day['
plt.xlabel('Customer Status')
plt.ylabel('Median Drives per Driving Day')
plt.title('Median Drives per Driving Day for Churned vs. Retained Users')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



In [ ]:
```
# NOTE ATTENTION NEEDED The data shows that WAZE users drive a lot.  Perhaps th
```

In [27]:
```
df.groupby(['label', 'device']).size()
```

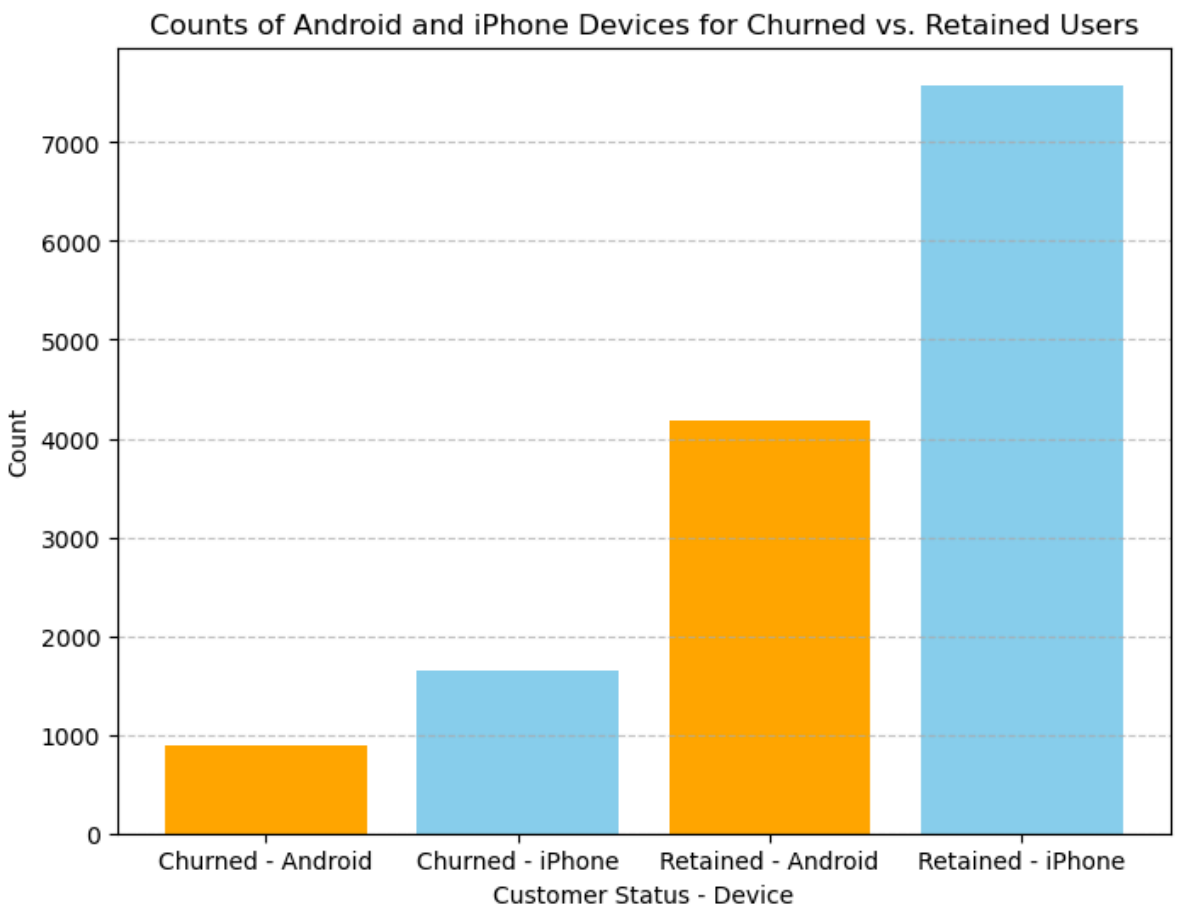
Out[27]:
```
label     device
churned   Android     891
          iPhone     1645
retained  Android    4183
          iPhone     7580
dtype: int64
```

In [46]:
```
import matplotlib.pyplot as plt

# Create a DataFrame with the provided device counts
device_counts = pd.DataFrame({
    'Label': ['Churned', 'Churned', 'Retained', 'Retained'],
    'Device': ['Android', 'iPhone', 'Android', 'iPhone'],
    'Count': [891, 1645, 4183, 7580]
})

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(device_counts['Label'] + ' - ' + device_counts['Device'], device_counts
plt.xlabel('Customer Status - Device')
plt.ylabel('Count')
plt.title('Counts of Android and iPhone Devices for Churned vs. Retained Users'
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



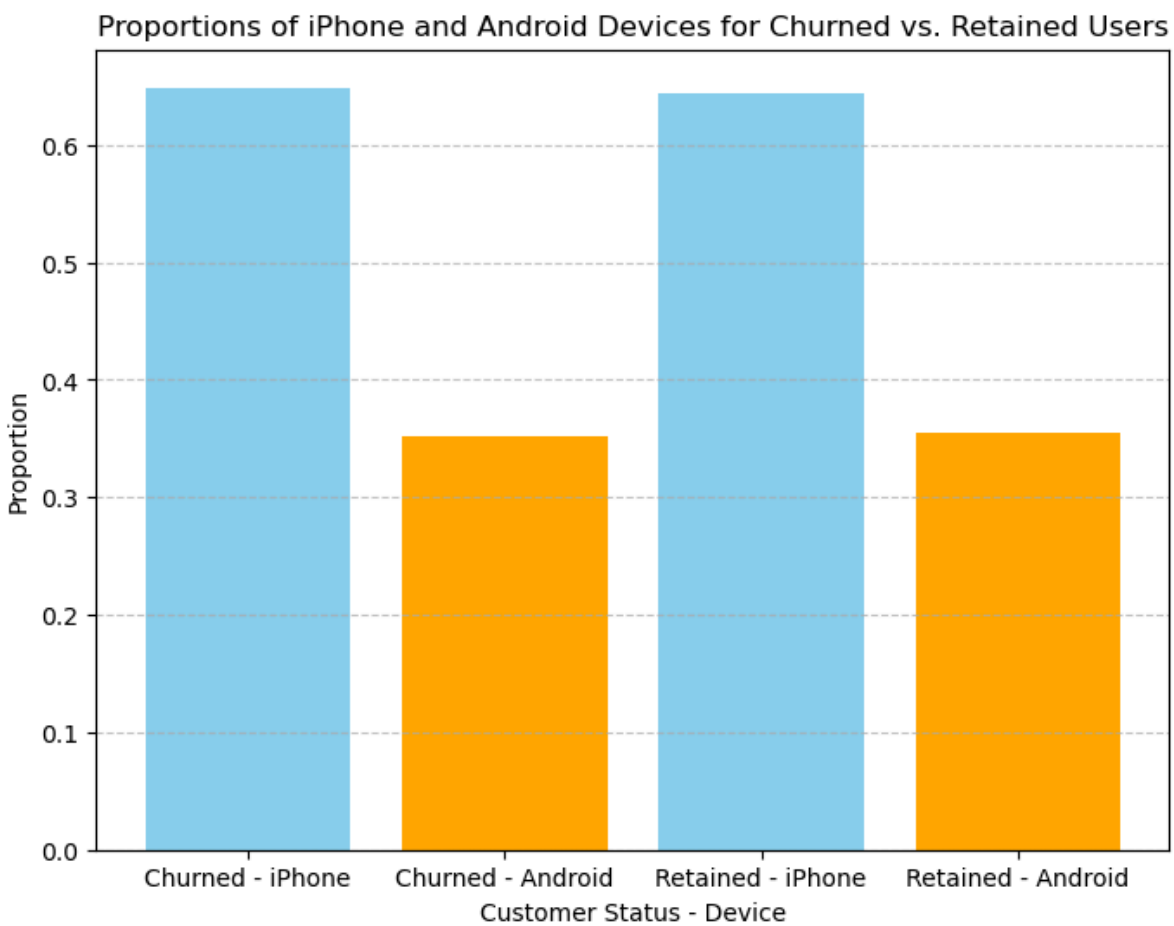
In [30]:
```python
# For each label, calculate the % of Android and iPhone users
df.groupby('label')['device'].value_counts(normalize=True)
```

Out[30]:
```
label     device
churned   iPhone     0.648659
          Android    0.351341
retained  iPhone     0.644393
          Android    0.355607
Name: proportion, dtype: float64
```

In [48]:
```
# Create a DataFrame with the provided proportions
device_proportions = pd.DataFrame({
    'Label': ['Churned', 'Churned', 'Retained', 'Retained'],
    'Device': ['iPhone', 'Android', 'iPhone', 'Android'],
    'Proportion': [0.648659, 0.351341, 0.644393, 0.355607]
})

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(device_proportions['Label'] + ' - ' + device_proportions['Device'], dev
plt.xlabel('Customer Status - Device')
plt.ylabel('Proportion')
plt.title('Proportions of iPhone and Android Devices for Churned vs. Retained U
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```



In [ ]:

Analysis of Retained vs Churned Users (2023) by iPhone/Android

Analysis and Presentation by Chuck Smit

Total count of the original dataset = 14999

Items removed due to incomplete data = 700

iPhone = 447

Android = 253


64%
APPLE
36%
ANDROID


2023 Users Retained = 11763 (82%)


2023 Users Churned = 2536 (18%)

How Many KM’s are Retained Users Driving?

How Many KM’s are Churned Users Driving?

Churned users drive 141% more than Retained users

How can this be explained?


Typically, Over-The-Road truckers drive 805-965 km’s per da
and Class A/5th Wheel RV’ers drive 480-560 km’s per day
Why are the drivers travelling the most churning?

More market analysis needs to be undertaken to make a solid assessment

Until then, a few talks with OTR drivers. as well as an internet search, some of the most sought out features.....

The app should consider truck-specific issues,
low bridges, weight limits, hazmat routes,
rest stop amenities, weather alerts and fuel price tracking.

Most of these would also benefit RV'ers