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

In [2]:
```
df = pd.read_csv('My Uber Drives - 2016.csv') #Reading Dataset
df
```

Out[2]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
0	1/1/2016 21:11	1/1/2016 21:17	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	1/2/2016 1:25	1/2/2016 1:37	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	1/2/2016 20:25	1/2/2016 20:38	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	1/5/2016 17:31	1/5/2016 17:45	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	1/6/2016 14:42	1/6/2016 15:49	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit
...	...	...	...	...	...	...	...
1151	12/31/2016 13:24	12/31/2016 13:42	Business	Kar?chi	Unknown Location	3.9	Temporary Site
1152	12/31/2016 15:03	12/31/2016 15:38	Business	Unknown Location	Unknown Location	16.2	Meeting
1153	12/31/2016 21:32	12/31/2016 21:50	Business	Katunayake	Gampaha	6.4	Temporary Site
1154	12/31/2016 22:08	12/31/2016 23:51	Business	Gampaha	Ilukwatta	48.2	Temporary Site
1155	Totals	NaN	NaN	NaN	NaN	12204.7	NaN

1156 rows × 7 columns

In [3]:
```
df.columns #Columns in dataset
```

Out[3]:
```
Index(['START_DATE*', 'END_DATE*', 'CATEGORY*', 'START*', 'STOP*', 'MILES*',
       'PURPOSE*'],
      dtype='object')
```

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

Out[4]:
```
(1156, 7)
```

In [5]:
```
df.isnull().sum() #Checking Null values in Dataset
```

Out[5]:
```
START_DATE*      0
END_DATE*        1
CATEGORY*        1
START*           1
STOP*            1
MILES*           0
PURPOSE*       503
dtype: int64
```

In [6]:
```
df[df.duplicated()] #Finding Duplicate rows
```

Out[6]:

	START_DATE*	END_DATE*	CATEGORY*	START*	STOP*	MILES*	PURPOSE*
492	6/28/2016 23:34	6/28/2016 23:59	Business	Durham	Cary	9.9	Meeting

In [7]:
```
df.drop_duplicates(inplace = True) #Removing duplicated row
```

In [8]:
```
#Removing rows, which have same start and end time i.e., Zero Trip distance and zero mile time
df.drop(df.index[[751, 761, 798, 807]], inplace = True)
```

In [9]:
```
#Renaming columns name
df.columns = ['START_DATE', 'END_DATE', 'CATEGORY', 'START', 'STOP', 'MILES', 'PURPOSE']
```

In [10]:
```
#Converting START_DATE and END_DATE into datetime
df['START_DATE'] = pd.to_datetime(df['START_DATE'], errors = 'coerce')
df['END_DATE'] = pd.to_datetime(df['END_DATE'], errors = 'coerce')
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1151 entries, 0 to 1155
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   START_DATE  1150 non-null   datetime64[ns]
 1   END_DATE    1150 non-null   datetime64[ns]
 2   CATEGORY    1150 non-null   object        
 3   START       1150 non-null   object        
 4   STOP        1150 non-null   object        
 5   MILES       1151 non-null   float64       
 6   PURPOSE     652 non-null    object        
dtypes: datetime64[ns](2), float64(1), object(4)
memory usage: 71.9+ KB
```

In [12]:
```
df.head()
```

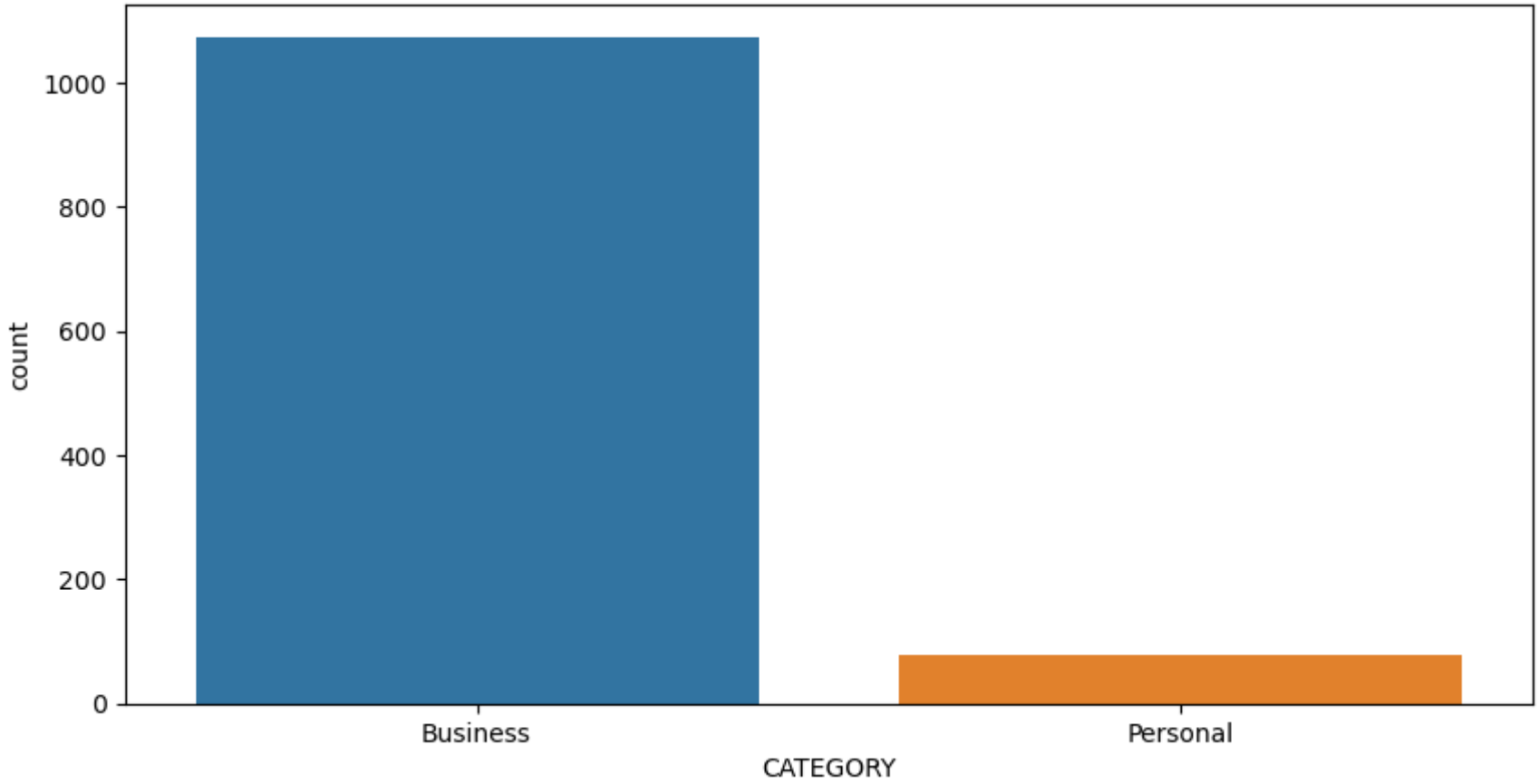
Out[12]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NaN
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit

In [13]:
```
#Count Plot
plt.figure(figsize = (10, 5))
sns.countplot(df['CATEGORY'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]:
```
<AxesSubplot:xlabel='CATEGORY', ylabel='count'>
```



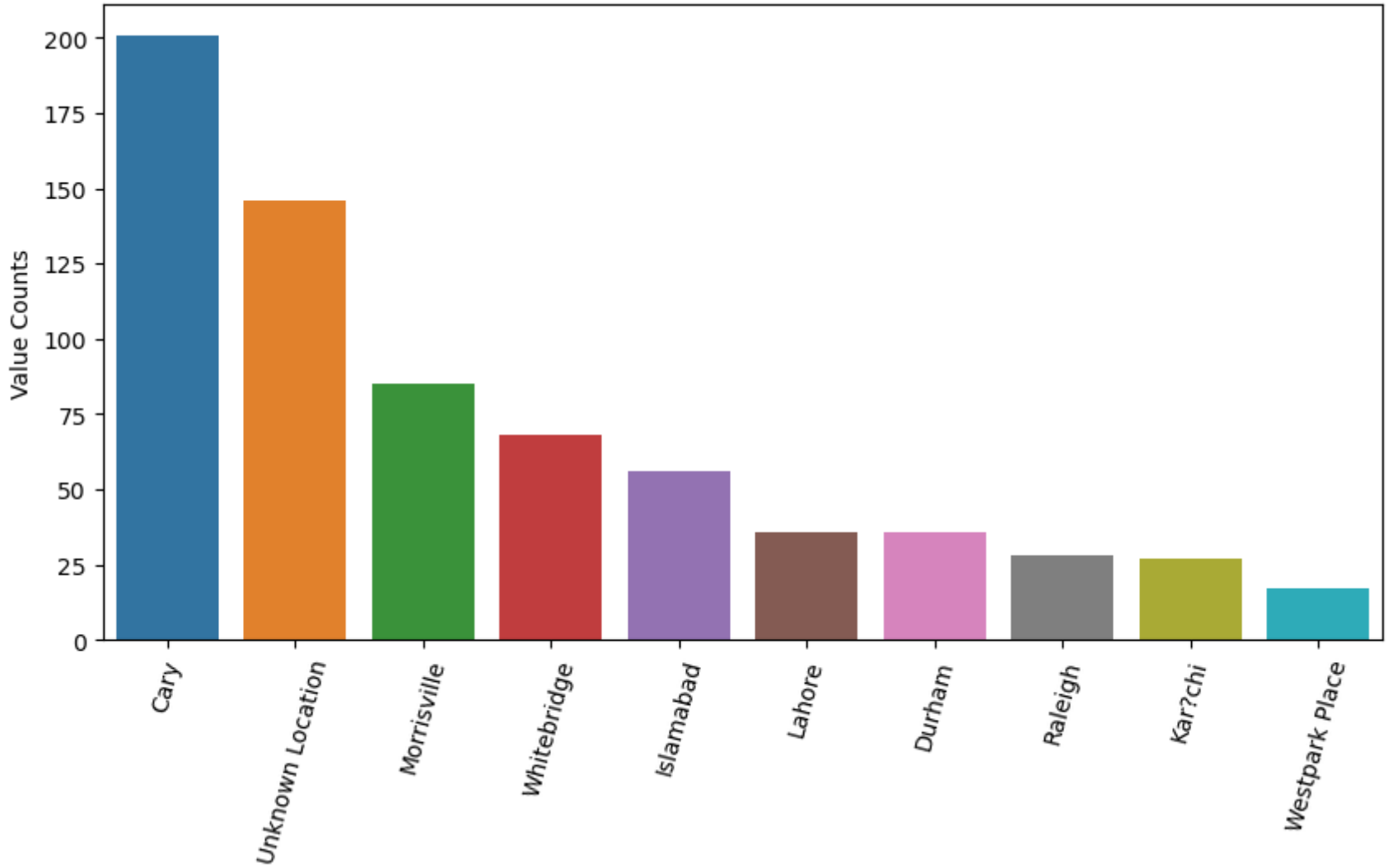
In [14]:
```
start_labels = df.START.value_counts().nlargest(10)
start_labels
```

Out[14]:
```
Cary               201
Unknown Location   146
Morrisville         85
Whitebridge         68
Islamabad           56
Lahore              36
Durham              36
Raleigh             28
Kar?chi             27
Westpark Place      17
Name: START, dtype: int64
```

In [15]:
```
#Bar Plot
plt.figure(figsize = (10, 5))
plt.xticks(rotation = 75)
sns.barplot(start_labels.index, start_labels)
plt.ylabel('Value Counts')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[15]:
```
Text(0, 0.5, 'Value Counts')
```



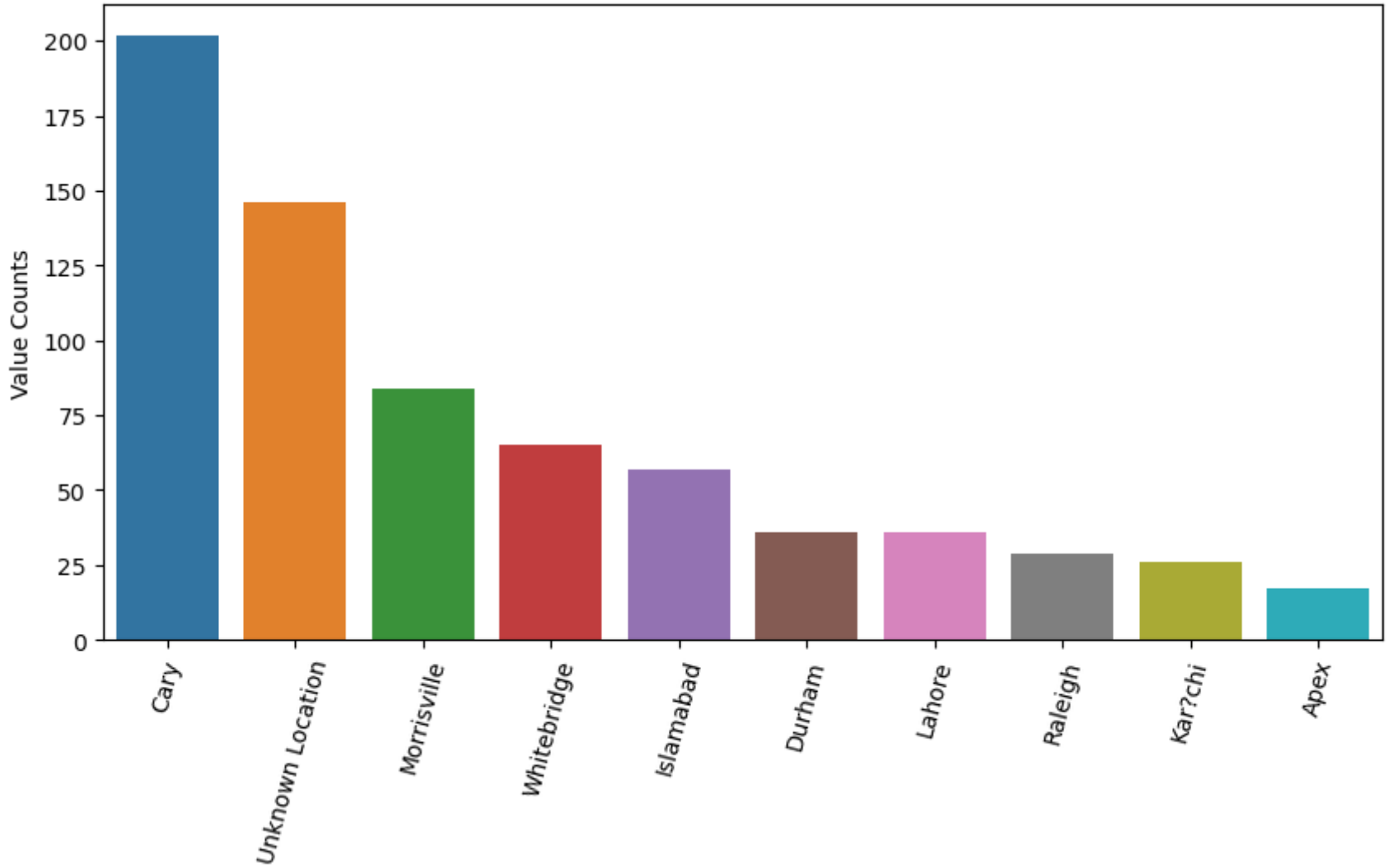
In [16]:
```
stop_labels = df.STOP.value_counts().nlargest(10)
stop_labels
```

Out[16]:
```
Cary                202
Unknown Location    146
Morrisville          84
Whitebridge          65
Islamabad            57
Durham               36
Lahore               36
Raleigh              29
Kar?chi              26
Apex                 17
Name: STOP, dtype: int64
```

In [17]:
```
#Bar Plot
plt.figure(figsize = (10, 5))
plt.xticks(rotation = 75)
sns.barplot(stop_labels.index, stop_labels)
plt.ylabel('Value Counts')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[17]:
```
Text(0, 0.5, 'Value Counts')
```



In [18]:
```
df['MONTH'] = pd.DatetimeIndex(df['START_DATE']).month #Extracting months from column START_DATE
```

In [19]:
```
month_label = {1.0:'Jan', 2.0:'Feb', 3.0:'Mar', 4.0:'Apr', 5.0:'May', 6.0:'Jun', 7.0:'July', 8.0:'Aug', 9.0:'Sept', 10.0:'Oct',
               11.0:'Nov', 12.0:'Dec'}
df['MONTH'] =  df.MONTH.map(month_label)
df.MONTH.unique()
```

Out[19]:
```
array(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'July', 'Aug', 'Sept',
       'Oct', 'Nov', 'Dec', nan], dtype=object)
```

In [20]:
```
df.head()
```

Out[20]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	MONTH
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	Jan
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NaN	Jan
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	Jan
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	Jan
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	Jan

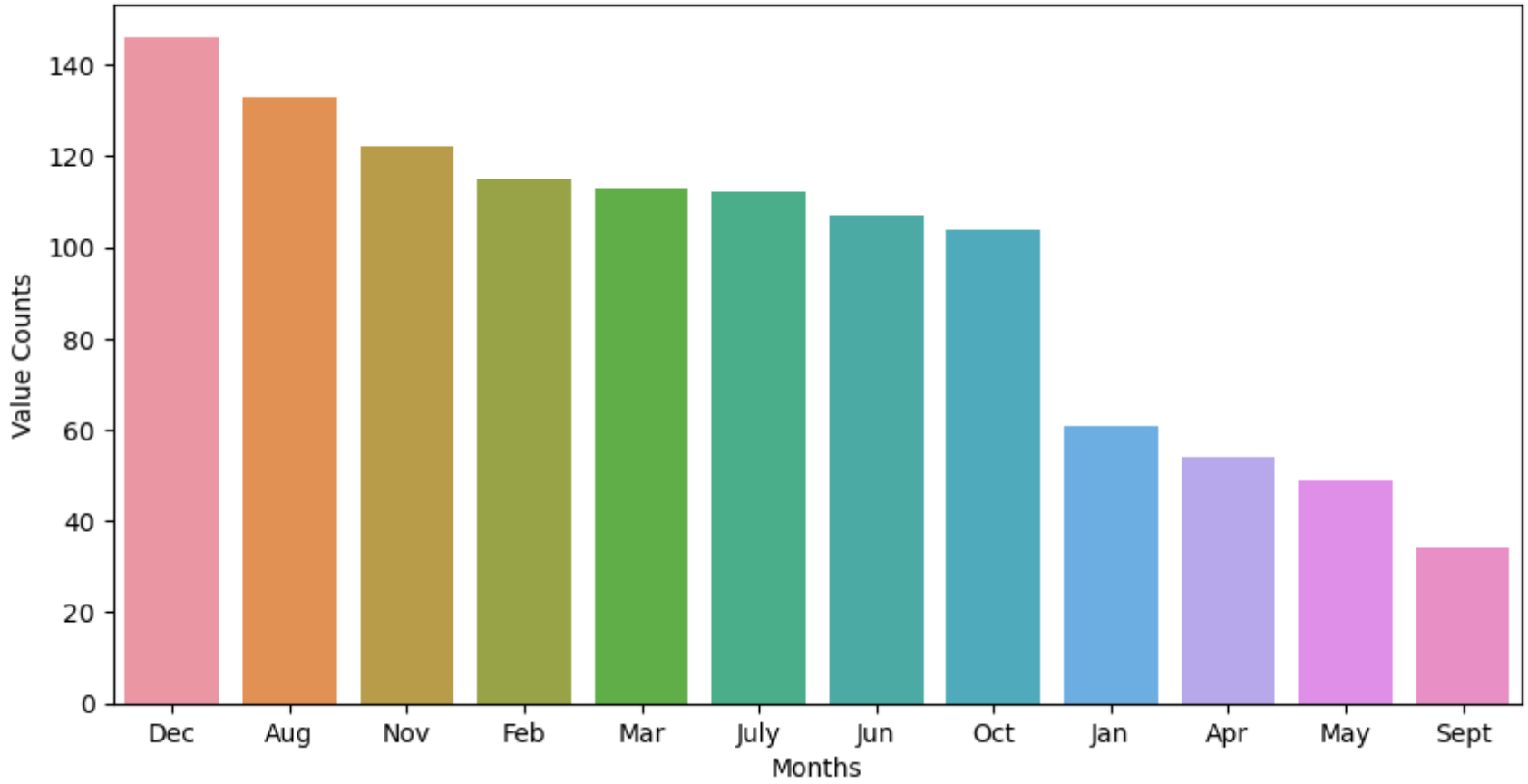
In [21]:
```
month_count = df.MONTH.value_counts()
month_count
```

Out[21]:
```
Dec     146
Aug     133
Nov     122
Feb     115
Mar     113
July    112
Jun     107
Oct     104
Jan      61
Apr      54
May      49
Sept     34
Name: MONTH, dtype: int64
```

In [22]:
```
#Bar Plot
plt.figure(figsize = (10, 5))
sns.barplot(month_count.index, month_count)
plt.xlabel('Months')
plt.ylabel('Value Counts')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[22]:
```
Text(0, 0.5, 'Value Counts')
```



In [23]:
```
#Creating dictionaries, that would contain info about the miles column
miles_dic = {}

for i in df.MILES:
    if i < 10:
        if '0-10 miles' not in miles_dic:
            miles_dic['0-10 miles'] = [i]
        else:
            miles_dic['0-10 miles'].append(i)

    elif i >= 10 and i < 20:
        if '10-20 miles' not in miles_dic:
            miles_dic['10-20 miles'] = [i]
        else:
            miles_dic['10-20 miles'].append(i)
```

```
    elif i >= 20 and i < 30:
        if '20-30 miles' not in miles_dic:
            miles_dic['20-30 miles'] = [i]
        else:
            miles_dic['20-30 miles'].append(i)

    elif i >= 30 and i < 40:
        if '30-40 miles' not in miles_dic:
            miles_dic['30-40 miles'] = [i]
        else:
            miles_dic['30-40 miles'].append(i)

    elif i >= 40 and i < 50:
        if '40-50 miles' not in miles_dic:
            miles_dic['40-50 miles'] = [i]
        else:
            miles_dic['40-50 miles'].append(i)

    else:
        if 'Above 50 miles' not in miles_dic:
            miles_dic['Above 50 miles'] = [i]
        else:
            miles_dic['Above 50 miles'].append(i)
```

In [24]:
```
len_miles = []
for key in miles_dic:
    len_miles.append((key, len(miles_dic[key])))
```

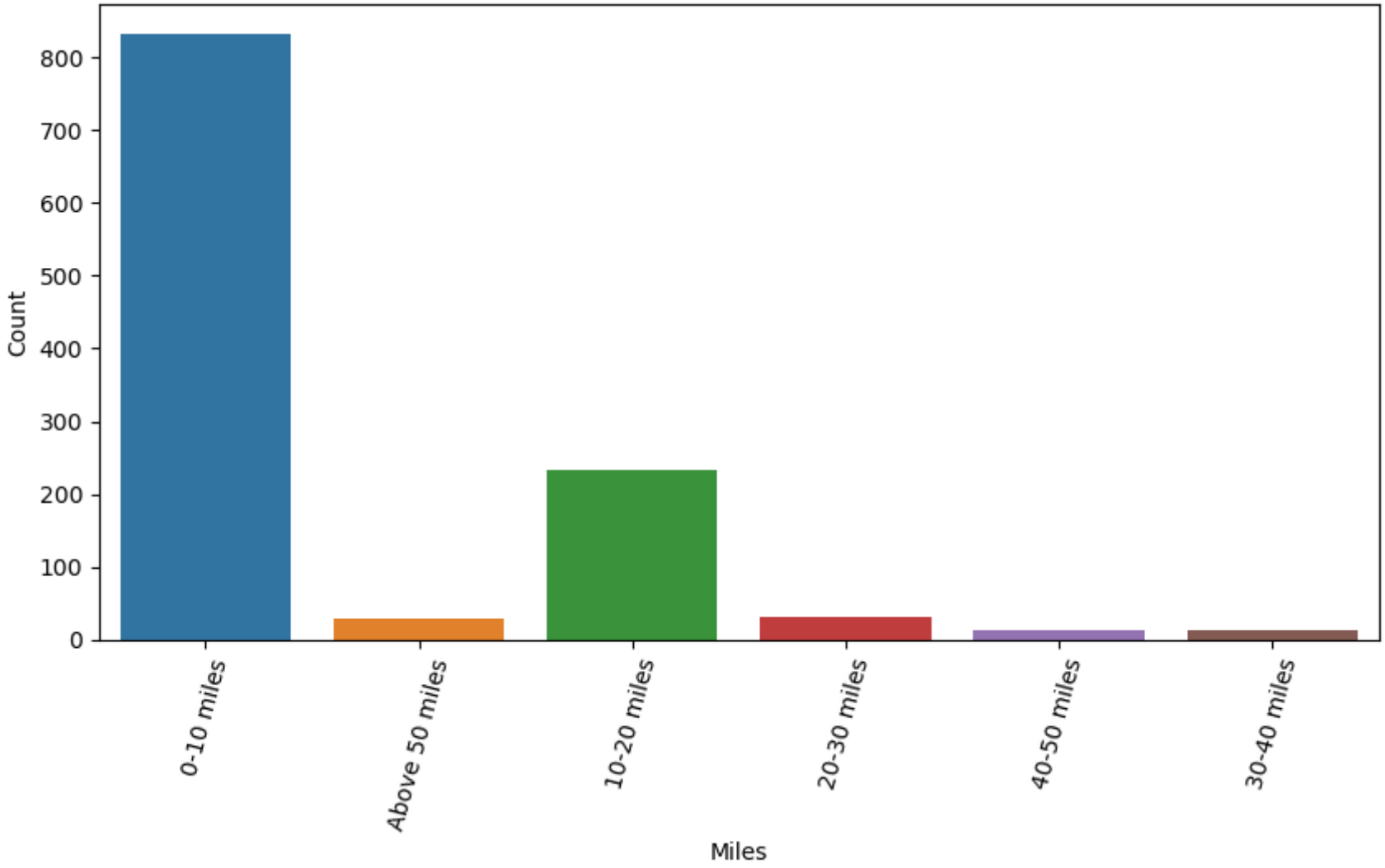
In [25]:
```
a, b = [], []
for i, j in len_miles:
    a.append(i)
    b.append(j)

plt.figure(figsize = (10, 5))
plt.xticks(rotation = 75)
sns.barplot(a, b)
plt.xlabel('Miles')
plt.ylabel('Count')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[25]:

```
Text(0, 0.5, 'Count')
```



In [26]:

```
df.head()
```

Out[26]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	MONTH
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	Jan
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NaN	Jan
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	Jan
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	Jan
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	Jan

In [27]:
```
#Distinguising Day and Night trips as per the time
t = pd.to_datetime(['18:00:00']).time
```

In [28]:
```
def check_time(tim):
    if t > tim:
        tim = 'DAY RIDE'
    else:
        tim = 'NIGHT RIDE'
```

In [29]:
```
df['DAY/NIGHT'] = df.apply(lambda x: 'NIGHT RIDE' if pd.notna(x['START_DATE']) and pd.Timestamp(x['START_DATE']).time() > t
                           else 'DAY RIDE', axis=1)
```

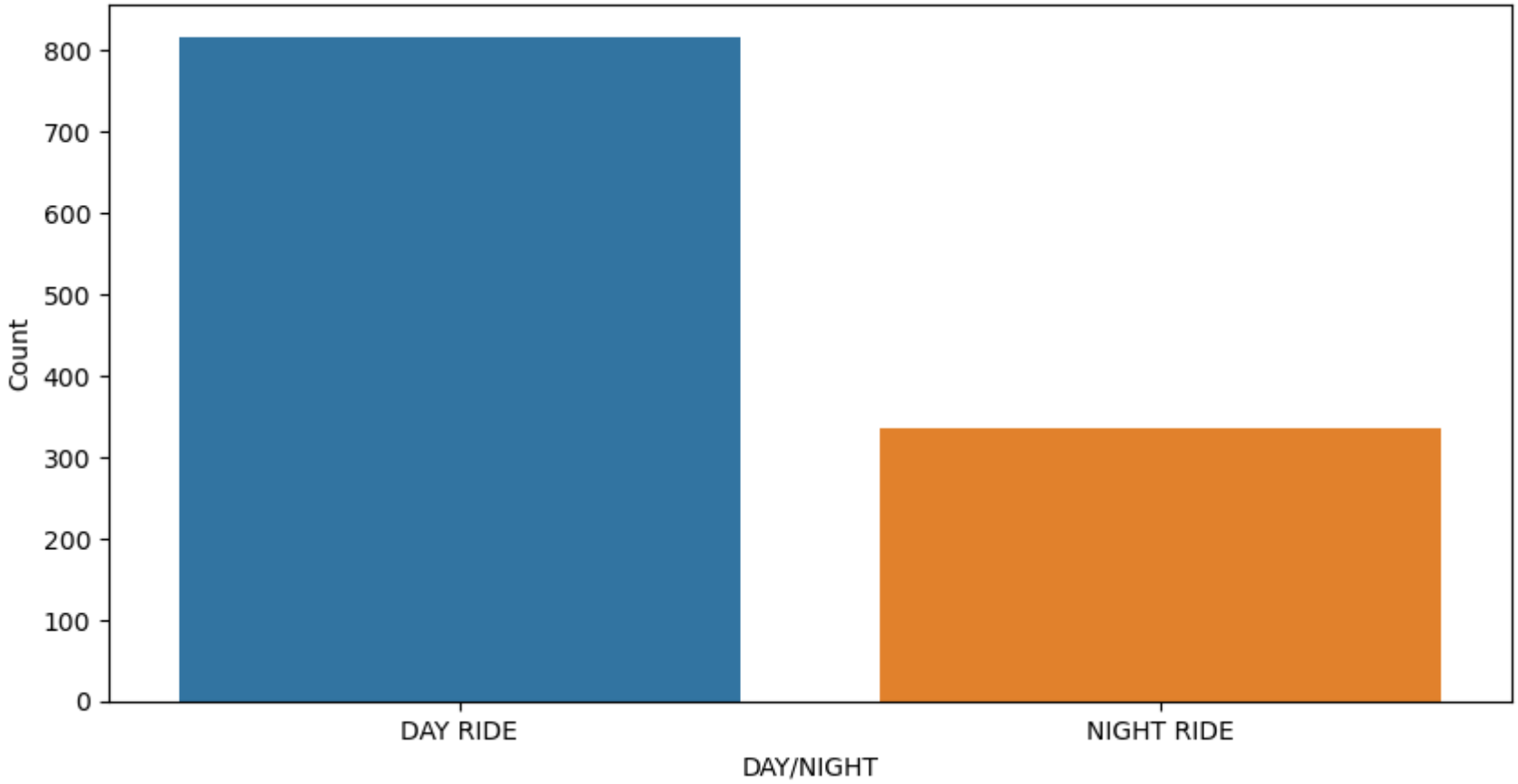
In [30]:
```
day_night_label = df['DAY/NIGHT'].value_counts()
day_night_label
```

Out[30]:
```
DAY RIDE      815
NIGHT RIDE    336
Name: DAY/NIGHT, dtype: int64
```

In [31]:
```
plt.figure(figsize = (10, 5))
sns.barplot(day_night_label.index, day_night_label)
plt.ylabel('Count')
plt.xlabel('DAY/NIGHT')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[31]:
```
Text(0.5, 0, 'DAY/NIGHT')
```



In [32]:
```
df['DAY'] = df.START_DATE.dt.weekday
```

In [33]:
```
day_label = {0: 'Mon', 1: 'Tue', 2: 'Wed', 3: 'Thu', 4: 'Fri', 5: 'Sat', 6: 'Sun'}
df['DAY'] = df['DAY'].map(day_label)
```

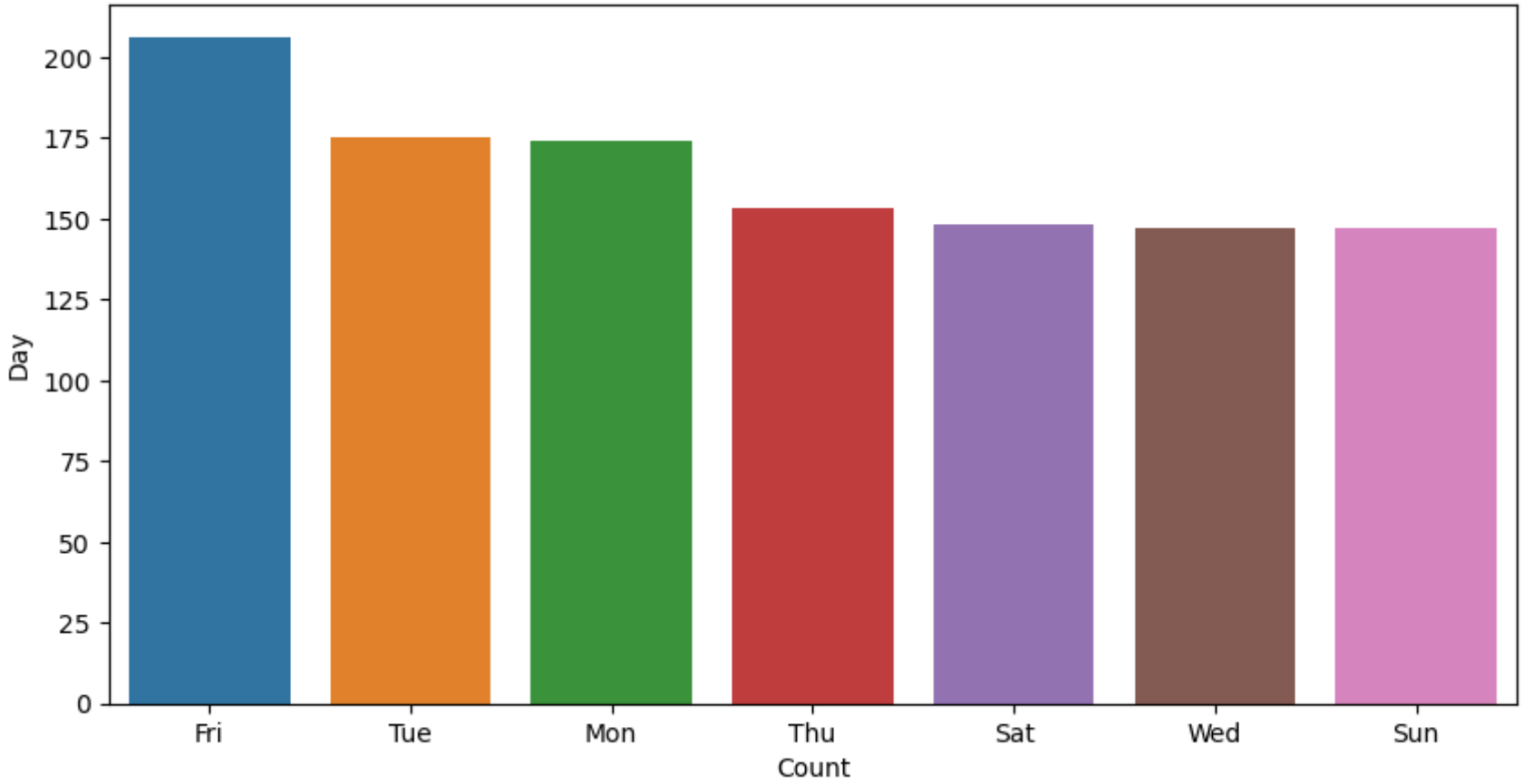
In [34]:
```
day_label = df.DAY.value_counts()
day_label
```

Out[34]:
```
Fri    206
Tue    175
Mon    174
Thu    153
Sat    148
Wed    147
Sun    147
Name: DAY, dtype: int64
```

In [35]:
```
#Bar Plot
plt.figure(figsize = (10, 5))
sns.barplot(day_label.index, day_label)
plt.ylabel('Day')
plt.xlabel('Count')
```

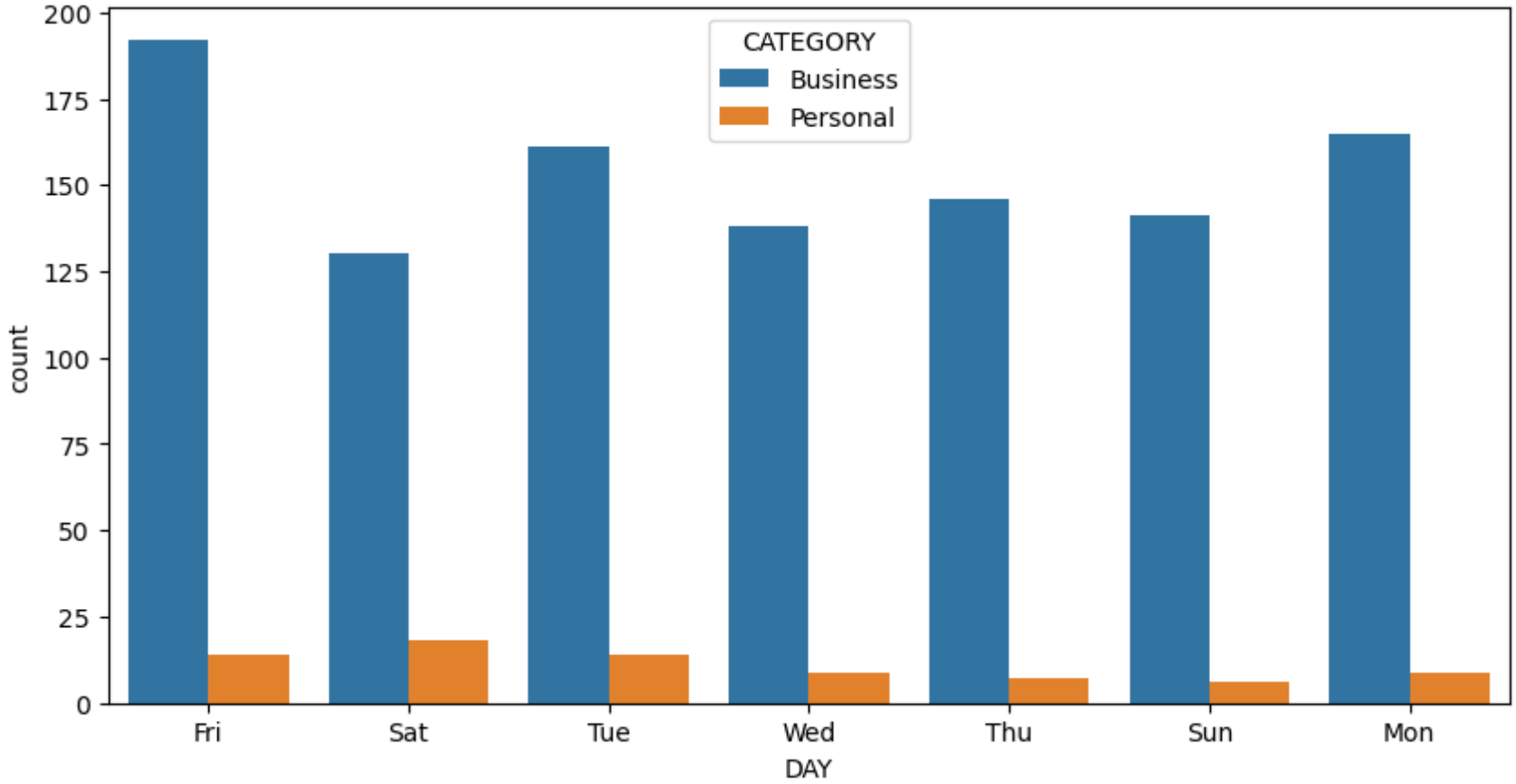
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[35]:
```
Text(0.5, 0, 'Count')
```



In [36]:
```
#Count plot as per Days
plt.figure(figsize = (10, 5))
sns.countplot(hue = 'CATEGORY', x = 'DAY', data = df)
```

Out[36]:
```
<AxesSubplot:xlabel='DAY', ylabel='count'>
```


CATEGORY
Business
Personal
count
200
175
150
125
100
75
50
25
0
Fri
Sat
Tue
Wed
Thu
Sun
Mon
DAY

In [ ]: