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

In [2]:
```
df = pd.read_csv("C:\Users\Priyanshu\Downloads\UberDataset.csv")
```

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

Out[3]:

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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1156 entries, 0 to 1155
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   START_DATE  1156 non-null   object 
 1   END_DATE    1155 non-null   object 
 2   CATEGORY    1155 non-null   object 
 3   START       1155 non-null   object 
 4   STOP        1155 non-null   object 
 5   MILES       1156 non-null   float64
 6   PURPOSE     653 non-null    object 
dtypes: float64(1), object(6)
memory usage: 63.3+ KB
```

DATA PREPROCESSING

In [5]:
```
df['PURPOSE'].fillna("NOT",inplace = True)
```

```
C:\Users\Priyanshu\AppData\Local\Temp\ipykernel_16592\1204224265.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or 'df[col] = df[col].method(value)' instead, to perform the operation inplace on the original object.


  df['PURPOSE'].fillna("NOT",inplace = True)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1156 entries, 0 to 1155
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   START_DATE  1156 non-null   object 
 1   END_DATE    1155 non-null   object 
 2   CATEGORY    1155 non-null   object 
 3   START       1155 non-null   object 
 4   STOP        1155 non-null   object 
 5   MILES       1156 non-null   float64
 6   PURPOSE     1156 non-null   object 
dtypes: float64(1), object(6)
memory usage: 63.3+ KB
```

In [7]:
```
df['START_DATE'] = pd.to_datetime(df['START_DATE'],errors='coerce')
df['END_DATE'] = pd.to_datetime(df['END_DATE'],errors='coerce')
#For changing object datatype into "datetime"
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1156 entries, 0 to 1155
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   START_DATE  421 non-null    datetime64[ns]
 1   END_DATE    420 non-null    datetime64[ns]
 2   CATEGORY    1155 non-null   object        
 3   START       1155 non-null   object        
 4   STOP        1155 non-null   object        
 5   MILES       1156 non-null   float64       
 6   PURPOSE     1156 non-null   object        
dtypes: datetime64[ns](2), float64(1), object(4)
memory usage: 63.3+ KB
```

In [9]:
```
from datetime import datetime

df['DATE'] = pd.DatetimeIndex(df['START_DATE']).date
```

In [ ]:
```

```

In [11]:
```
df['TIME'] = pd.DatetimeIndex(df['END_DATE']).hour
```

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

Out[12]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	DATE	TIME
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	2016-01-01	21.0
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NOT	2016-01-02	1.0
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	2016-01-02	20.0
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	2016-01-05	17.0
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	2016-01-06	15.0

In [13]:
```
df['DAY-TYPE'] = pd.cut(x=df['TIME'],bins =[0,10,15,19,24],labels=['Morning','Afternoon','Evening','Night'])
```

In [14]:
```
df.head()
```

Out[14]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	DATE	TIME	DAY-TYPE
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	2016-01-01	21.0	Night
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NOT	2016-01-02	1.0	Morning
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	2016-01-02	20.0	Night
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	2016-01-05	17.0	Evening
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	2016-01-06	15.0	Afternoon

In [15]:
```
df.dropna(inplace=True)
```

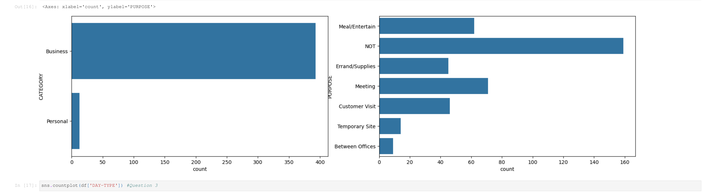
In [16]:
```
df.shape
```

Out[16]:
```
(406, 10)
```

DATA VISULIZATION

In [17]:
```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1) #Question 1
sns.countplot(df['CATEGORY'])
plt.subplot(1,2,2) # Question 2
sns.countplot(df['PURPOSE'])
```

Out[17]:
```
<Axes: xlabel='count', ylabel='PURPOSE'>
```



In [18]:
```
sns.countplot(df['DAY-TYPE']) #Question 3
```

Out[18]:
```
<Axes: xlabel='count', ylabel='DAY-TYPE'>
```

In [19]:
```
df['Month_Name'] = df.START_DATE.dt.month
month_label ={1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'Jul',8:'Aug',9:'Sep',10:'Oct',11:'Nov',12:'Dec'}

df['Month_Name']=df['Month_Name'].map(month_label)
```

In [20]:
```
day_label = df.Month_Name.value_counts()

sns.barplot(x=day_label.index,y=day_label)
plt.xlabel("Day")
plt.ylabel("counts")
```

Out[20]:
```
Text(0, 0.5, 'counts')
```

In [21]:
```
df['DAY']= df.START_DATE.dt.weekday

day_label={0:'Mon',1:'Tue',2:'Wed',3:'Thu',4:'Fri',5:'Sat',6:'Sun'}

df['DAY']=df['DAY'].map(day_label)
```

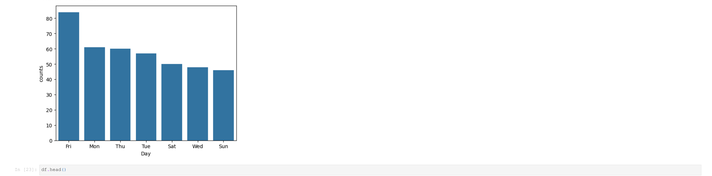
In [22]:
```
df.head()
```

Out[22]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	DATE	TIME	DAY-TYPE	Month_Name	DAY
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	2016-01-01	21.0	Night	Jan	Fri
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NOT	2016-01-02	1.0	Morning	Jan	Sat
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	2016-01-02	20.0	Night	Jan	Sat
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	2016-01-05	17.0	Evening	Jan	Tue
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	2016-01-06	15.0	Afternoon	Jan	Wed

In [23]:
```
day_label = df.DAY.value_counts()

sns.barplot(x=day_label.index,y=day_label)
plt.xlabel("Day")
plt.ylabel("counts")
```

Out[23]:
```
Text(0, 0.5, 'counts')
```



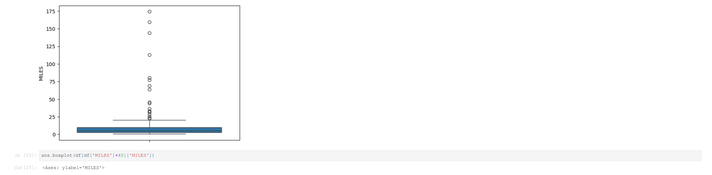
In [24]:
```
df.head()
```

Out[24]:

	START_DATE	END_DATE	CATEGORY	START	STOP	MILES	PURPOSE	DATE	TIME	DAY-TYPE	Month_Name	DAY
0	2016-01-01 21:11:00	2016-01-01 21:17:00	Business	Fort Pierce	Fort Pierce	5.1	Meal/Entertain	2016-01-01	21.0	Night	Jan	Fri
1	2016-01-02 01:25:00	2016-01-02 01:37:00	Business	Fort Pierce	Fort Pierce	5.0	NOT	2016-01-02	1.0	Morning	Jan	Sat
2	2016-01-02 20:25:00	2016-01-02 20:38:00	Business	Fort Pierce	Fort Pierce	4.8	Errand/Supplies	2016-01-02	20.0	Night	Jan	Sat
3	2016-01-05 17:31:00	2016-01-05 17:45:00	Business	Fort Pierce	Fort Pierce	4.7	Meeting	2016-01-05	17.0	Evening	Jan	Tue
4	2016-01-06 14:42:00	2016-01-06 15:49:00	Business	Fort Pierce	West Palm Beach	63.7	Customer Visit	2016-01-06	15.0	Afternoon	Jan	Wed

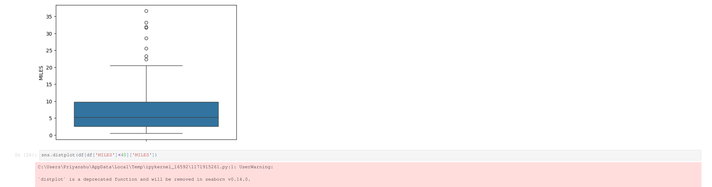
In [25]:
```
sns.boxplot(df['MILES'])
```

Out[25]:
```
<Axes: ylabel='MILES'>
```



In [27]:
```
sns.boxplot(df[df['MILES']<40]['MILES'])
```

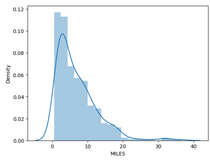
Out[27]:
```
<Axes: ylabel='MILES'>
```



In [26]:
```
sns.distplot(df[df['MILES']<40]['MILES'])
```

```
C:\Users\Priyanshu\AppData\Local\Temp\ipykernel_16592\1171915261.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[df['MILES']<40]['MILES'])
```

Out[26]:
```
<Axes: xlabel='MILES', ylabel='Density'>
```


0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
0
10
20
30
40
MILES