r-rides-data-analysis-using-python

August 7, 2025

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

[ ]:
```
dataset = pd.read_csv("UberDataset.csv")
dataset.head()
```

[ ]:
```
         START_DATE          END_DATE  CATEGORY        START              STOP  \
0  01-01-2016 21:11  01-01-2016 21:17  Business  Fort Pierce       Fort Pierce
1  01-02-2016 01:25  01-02-2016 01:37  Business  Fort Pierce       Fort Pierce
2  01-02-2016 20:25  01-02-2016 20:38  Business  Fort Pierce       Fort Pierce
3  01-05-2016 17:31  01-05-2016 17:45  Business  Fort Pierce       Fort Pierce
4  01-06-2016 14:42  01-06-2016 15:49  Business  Fort Pierce  West Palm Beach

   MILES          PURPOSE
0    5.1   Meal/Entertain
1    5.0              NaN
2    4.8  Errand/Supplies
3    4.7          Meeting
4   63.7   Customer Visit
```

[ ]:
```
dataset.shape
```

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

[ ]:
```
dataset.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
```

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

[ ]:
```
dataset['START_DATE'] = pd.to_datetime(dataset['START_DATE'],
                                                                   errors='coerce')
dataset['END_DATE'] = pd.to_datetime(dataset['END_DATE'],
                                                                   errors='coerce')
```

[ ]:
```
from datetime import datetime

dataset['date'] = pd.DatetimeIndex(dataset['START_DATE']).date
dataset['time'] = pd.DatetimeIndex(dataset['START_DATE']).hour

#changing into categories of day and night
dataset['day-night'] = pd.cut(x=dataset['time'],
                                                        bins = [0,10,15,19,24],
                                                        labels =␣
 ↪['Morning','Afternoon','Evening','Night'])
```

[ ]:
```
dataset.dropna(inplace=True)
```

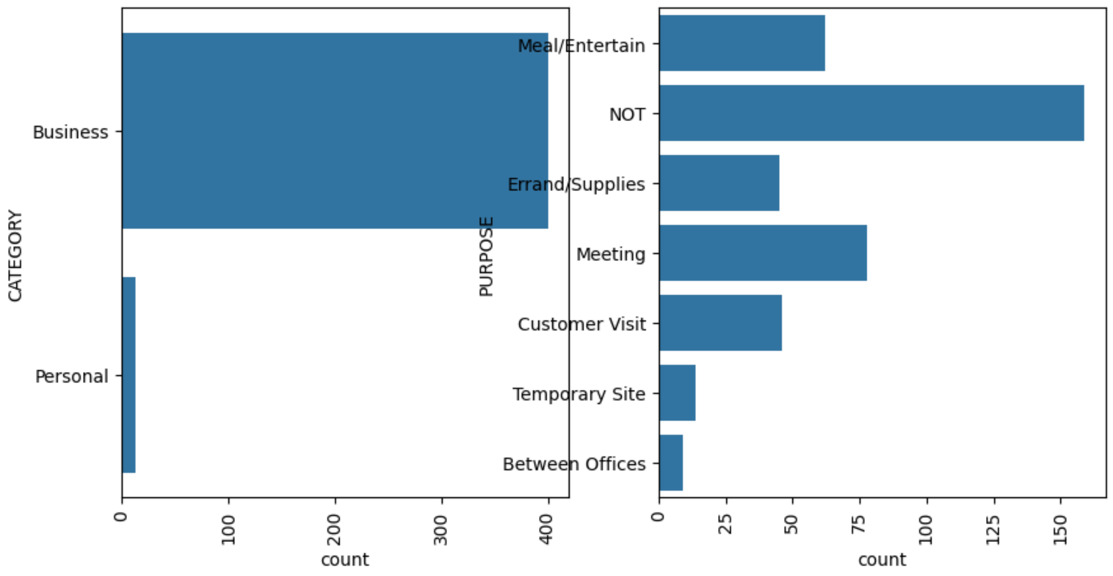
[ ]:
```
dataset.drop_duplicates(inplace=True)
```

[ ]:
```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)

unique_values = {}
for col in object_cols:
  unique_values[col] = dataset[col].unique().size
unique_values
```

[ ]:
```
{'CATEGORY': 2, 'START': 108, 'STOP': 112, 'PURPOSE': 7, 'date': 113}
```

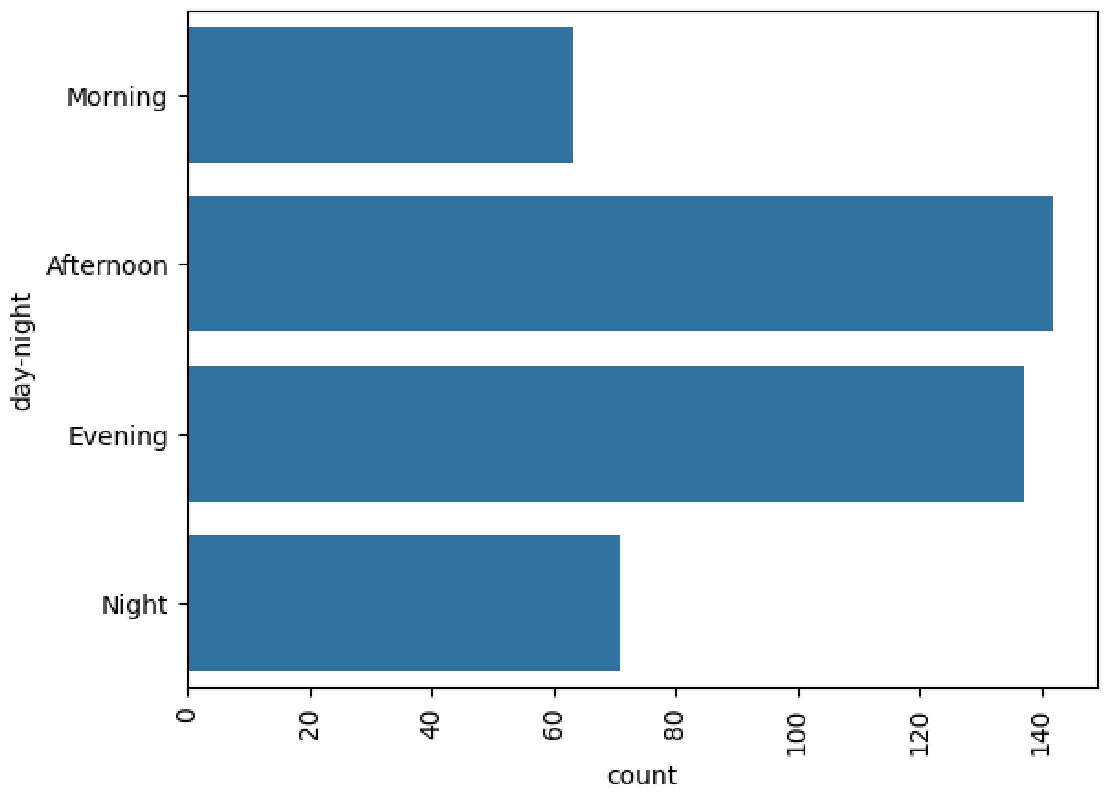
[ ]:
```
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
sns.countplot(dataset['CATEGORY'])
plt.xticks(rotation=90)

plt.subplot(1,2,2)
sns.countplot(dataset['PURPOSE'])
plt.xticks(rotation=90)
```

```
[ ]: (array([  0.,  25.,  50.,  75., 100., 125., 150., 175.]),
      [Text(0.0, 0, '0'),
       Text(25.0, 0, '25'),
       Text(50.0, 0, '50'),
       Text(75.0, 0, '75'),
       Text(100.0, 0, '100'),
       Text(125.0, 0, '125'),
       Text(150.0, 0, '150'),
       Text(175.0, 0, '175')])
```

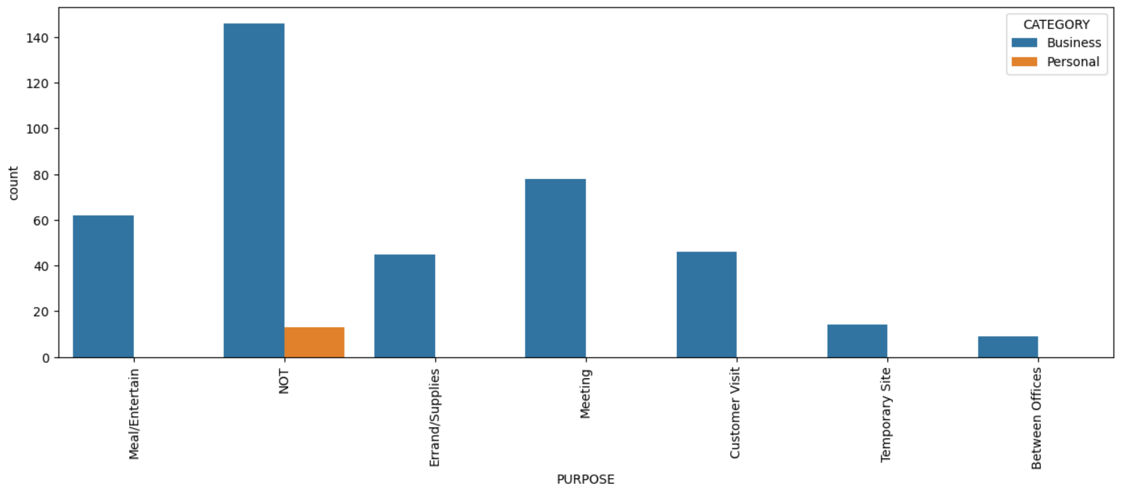


```
[ ]: sns.countplot(dataset['day-night'])
     plt.xticks(rotation=90)
```

```
[ ]: (array([  0.,  20.,  40.,  60.,  80., 100., 120., 140., 160.]),
      [Text(0.0, 0, '0'),
       Text(20.0, 0, '20'),
       Text(40.0, 0, '40'),
       Text(60.0, 0, '60'),
       Text(80.0, 0, '80'),
       Text(100.0, 0, '100'),
       Text(120.0, 0, '120'),
       Text(140.0, 0, '140'),
       Text(160.0, 0, '160')])
```



```
[ ]: plt.figure(figsize=(15, 5))
     sns.countplot(data=dataset, x='PURPOSE', hue='CATEGORY')
     plt.xticks(rotation=90)
     plt.show()
```

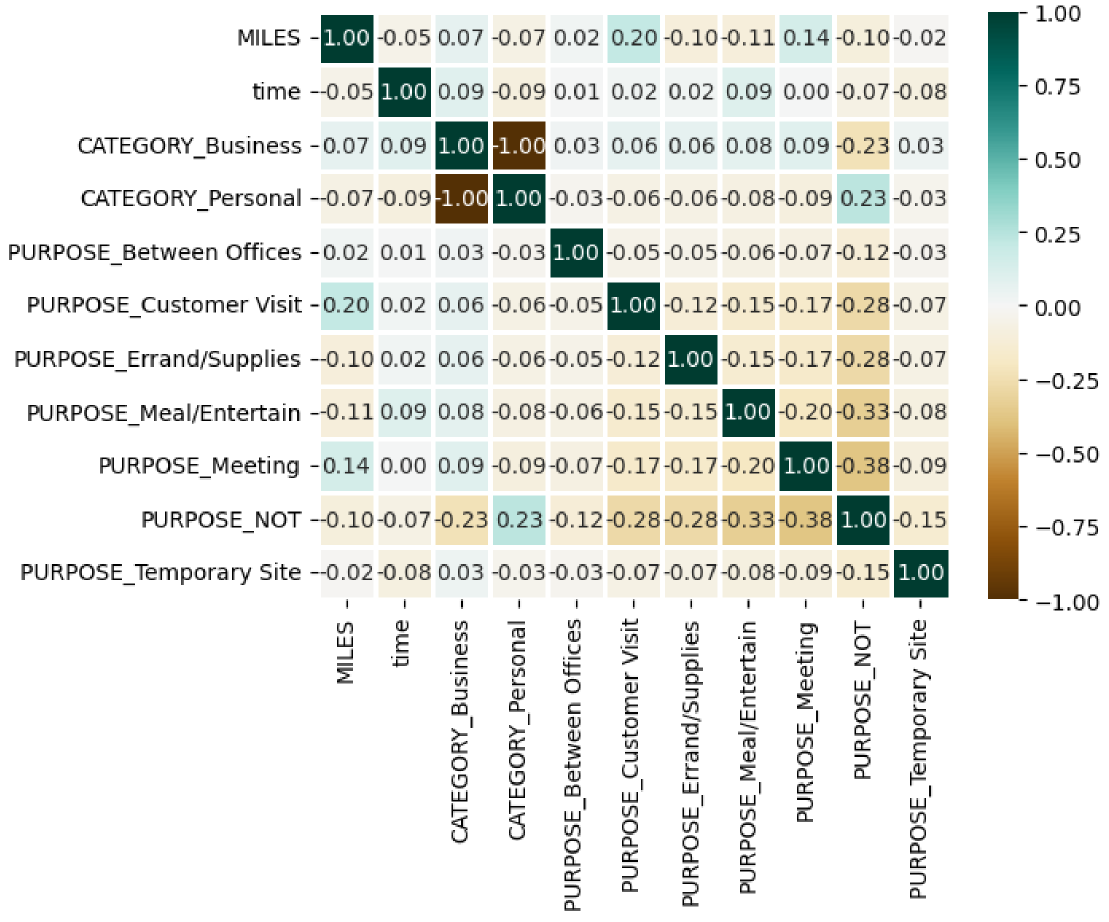


[ ]:
```
from sklearn.preprocessing import OneHotEncoder
object_cols = ['CATEGORY', 'PURPOSE']
OH_encoder = OneHotEncoder(sparse=False, handle_unknown='ignore')
OH_cols = pd.DataFrame(OH_encoder.fit_transform(dataset[object_cols]))
OH_cols.index = dataset.index
OH_cols.columns = OH_encoder.get_feature_names_out()
df_final = dataset.drop(object_cols, axis=1)
dataset = pd.concat([df_final, OH_cols], axis=1)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:975:
FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will
be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its
default value.
  warnings.warn(
```

[ ]:
```
# Select only numerical columns for correlation calculation
numeric_dataset = dataset.select_dtypes(include=['number'])

# Now you can create the heatmap
sns.heatmap(numeric_dataset.corr(),
            cmap='BrBG',
            fmt='.2f',
            linewidths=2,
            annot=True)
```

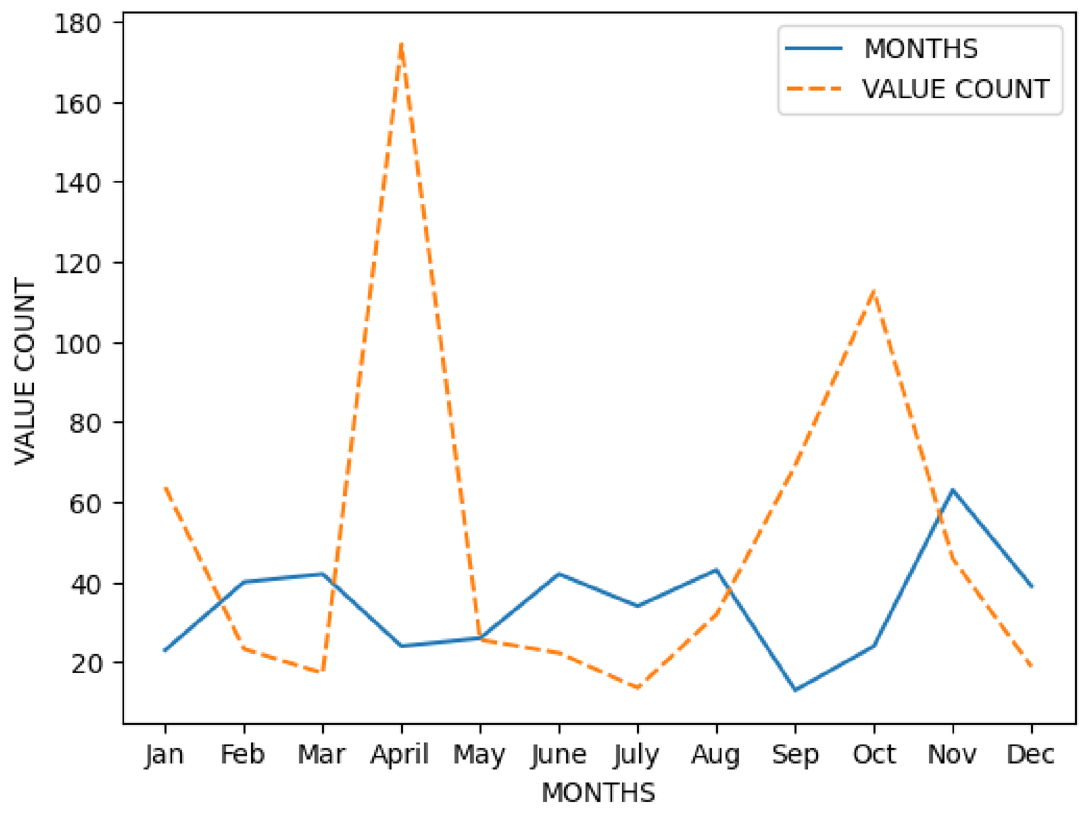
[ ]:
```
<Axes: >
```



	MILES	time	CATEGORY_Business	CATEGORY_Personal	PURPOSE_Between Offices	PURPOSE_Customer Visit	PURPOSE_Errand/Supplies	PURPOSE_Meal/Entertain	PURPOSE_Meeting	PURPOSE_NOT	PURPOSE_Temporary Site
MILES	1.00	-0.05	0.07	-0.07	0.02	0.20	-0.10	-0.11	0.14	-0.10	-0.02
time	-0.05	1.00	0.09	-0.09	0.01	0.02	0.02	0.09	0.00	-0.07	-0.08
CATEGORY_Business	0.07	0.09	1.00	-1.00	0.03	0.06	0.06	0.08	0.09	-0.23	0.03
CATEGORY_Personal	-0.07	-0.09	-1.00	1.00	-0.03	-0.06	-0.06	-0.08	-0.09	0.23	-0.03
PURPOSE_Between Offices	0.02	0.01	0.03	-0.03	1.00	-0.05	-0.05	-0.06	-0.07	-0.12	-0.03
PURPOSE_Customer Visit	0.20	0.02	0.06	-0.06	-0.05	1.00	-0.12	-0.15	-0.17	-0.28	-0.07
PURPOSE_Errand/Supplies	-0.10	0.02	0.06	-0.06	-0.05	-0.12	1.00	-0.15	-0.17	-0.28	-0.07
PURPOSE_Meal/Entertain	-0.11	0.09	0.08	-0.08	-0.06	-0.15	-0.15	1.00	-0.20	-0.33	-0.08
PURPOSE_Meeting	0.14	0.00	0.09	-0.09	-0.07	-0.17	-0.17	-0.20	1.00	-0.38	-0.09
PURPOSE_NOT	-0.10	-0.07	-0.23	0.23	-0.12	-0.28	-0.28	-0.33	-0.38	1.00	-0.15
PURPOSE_Temporary Site	-0.02	-0.08	0.03	-0.03	-0.03	-0.07	-0.07	-0.08	-0.09	-0.15	1.00

```
[ ]: dataset['MONTH'] = pd.DatetimeIndex(dataset['START_DATE']).month
     month_label = {1.0: 'Jan', 2.0: 'Feb', 3.0: 'Mar', 4.0: 'April',
                             5.0: 'May', 6.0: 'June', 7.0: 'July', 8.0: 'Aug',
                             9.0: 'Sep', 10.0: 'Oct', 11.0: 'Nov', 12.0: 'Dec'}
     dataset["MONTH"] = dataset.MONTH.map(month_label)

     mon = dataset.MONTH.value_counts(sort=False)

     # Month total rides count vs Month ride max count
     df = pd.DataFrame({"MONTHS": mon.values,
                                   "VALUE COUNT": dataset.groupby('MONTH',

       ↪max()})

     p = sns.lineplot(data=df)
     p.set(xlabel="MONTHS", ylabel="VALUE COUNT")
```

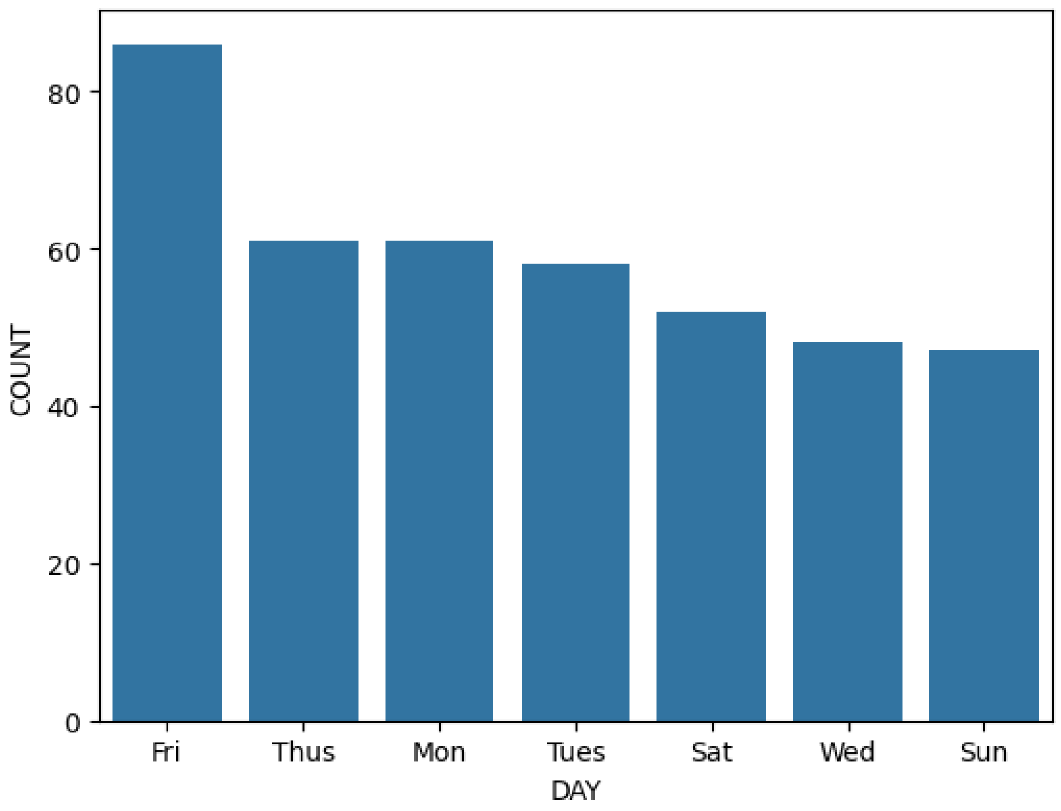
```
[ ]: [Text(0.5, 0, 'MONTHS'), Text(0, 0.5, 'VALUE COUNT')]
```



```
[ ]: dataset['DAY'] = dataset.START_DATE.dt.weekday
     day_label = {
             0: 'Mon', 1: 'Tues', 2: 'Wed', 3: 'Thus', 4: 'Fri', 5: 'Sat', 6: 'Sun'
     }
     dataset['DAY'] = dataset['DAY'].map(day_label)
```

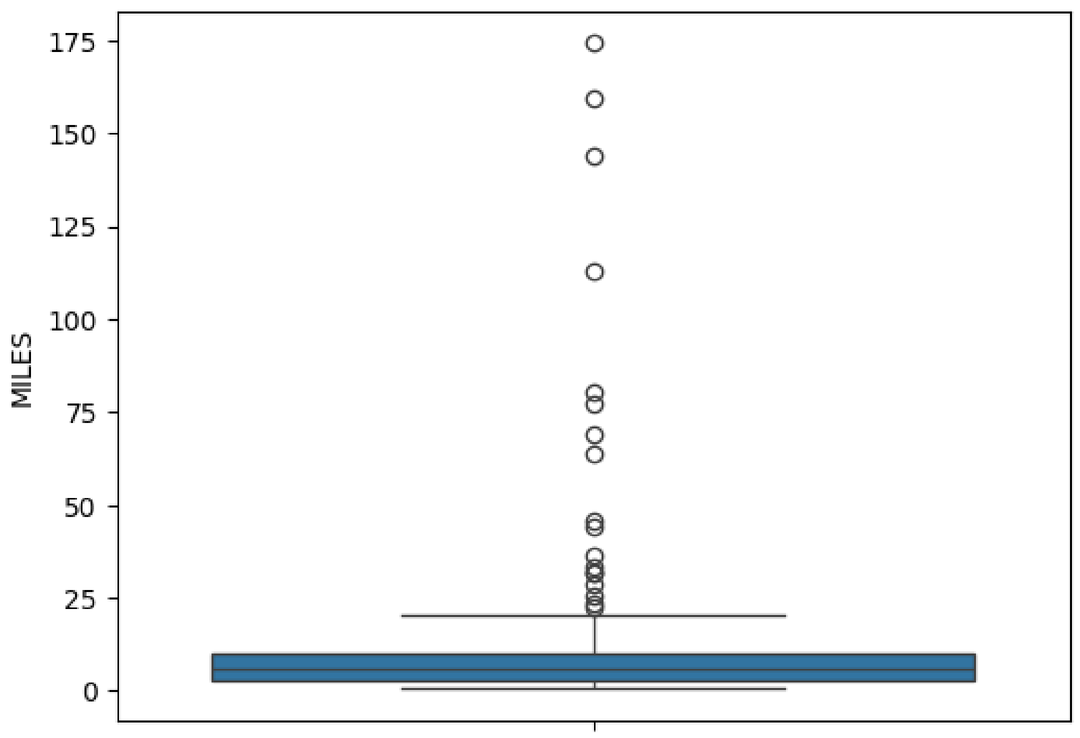
```
[ ]: day_label = dataset.DAY.value_counts()
     sns.barplot(x=day_label.index, y=day_label);
     plt.xlabel('DAY')
     plt.ylabel('COUNT')
```

```
[ ]: Text(0, 0.5, 'COUNT')
```



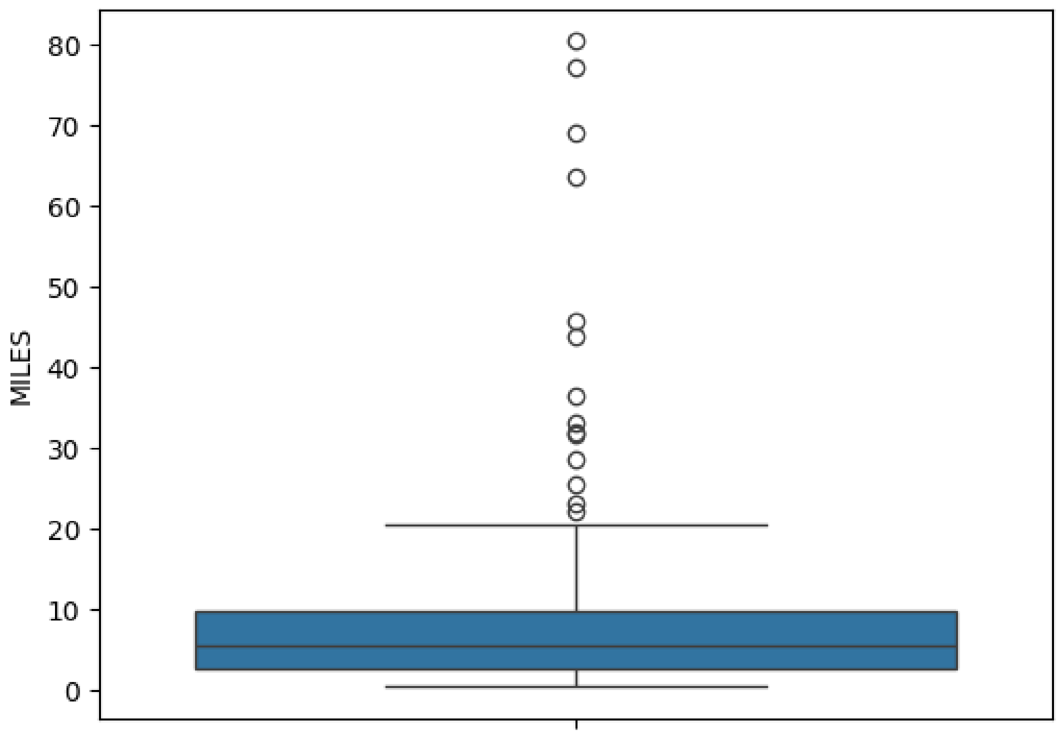
```
[ ]: sns.boxplot(dataset['MILES'])
```

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



```
[ ]: sns.boxplot(dataset[dataset['MILES']<100]['MILES'])
```

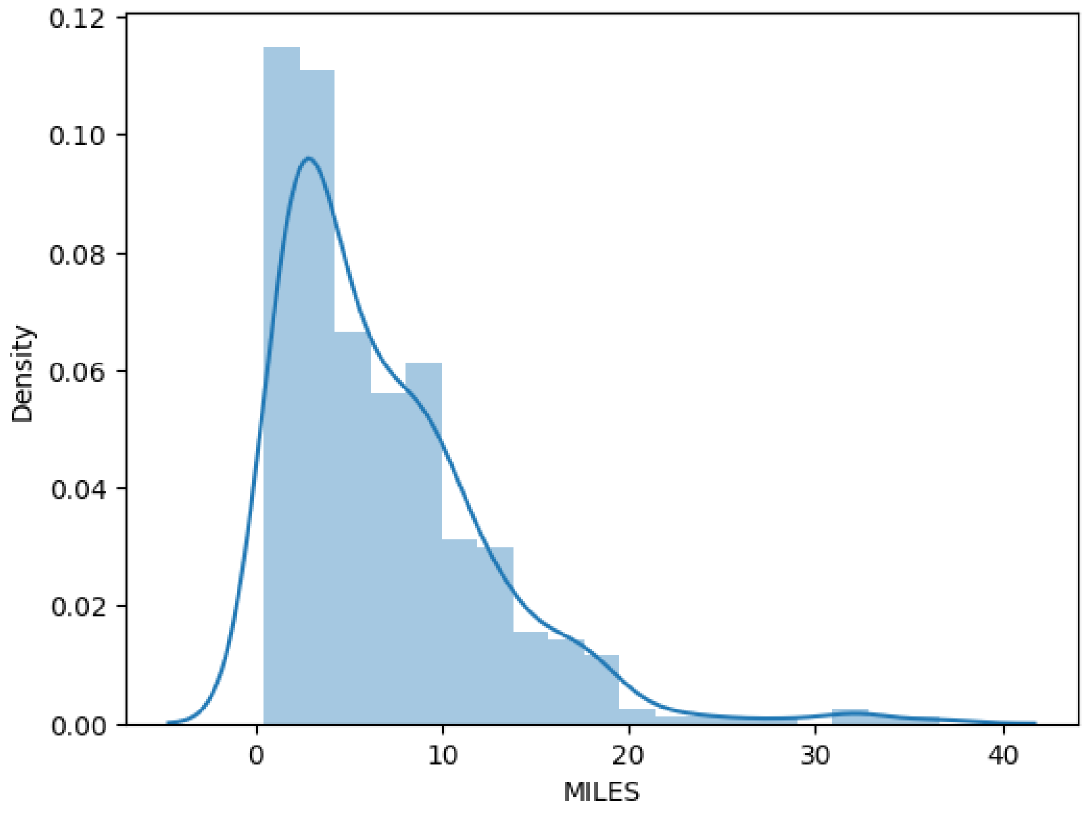
```
[ ]: <Axes: ylabel='MILES'>
```



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

```
<ipython-input-67-1d5904d4eb1d>: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(dataset[dataset['MILES']<40]['MILES'])
```

```
[ ]: <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