Question 3

In [1]:

```python
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt

```

In [2]:

```python
df=pd.read_csv("day.csv")
df
```

Out[2]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.3
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.3
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.1
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.2
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.2
...	...	...	...	...	...	...	...	...	...	...	...
726	727	2012-12-27	1	1	12	0	4	1	2	0.254167	0.2
727	728	2012-12-28	1	1	12	0	5	1	2	0.253333	0.2
728	729	2012-12-29	1	1	12	0	6	0	2	0.253333	0.2
729	730	2012-12-30	1	1	12	0	0	0	1	0.255833	0.2
730	731	2012-12-31	1	1	12	0	1	1	2	0.215833	0.2

731 rows × 16 columns

In [3]:

```python
#a)
len(df)

```

Out[3]:

```
731
```

```
#Based on day.csv, it has 731 observations and 16 attributes.
```

In [4]:

```
#b)
df.head(10)
```

Out[4]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	at
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229
5	6	2011-01-06	1	0	1	0	4	1	1	0.204348	0.233
6	7	2011-01-07	1	0	1	0	5	1	2	0.196522	0.208
7	8	2011-01-08	1	0	1	0	6	0	2	0.165000	0.162
8	9	2011-01-09	1	0	1	0	0	0	1	0.138333	0.116
9	10	2011-01-10	1	0	1	0	1	1	1	0.150833	0.150

In [5]:

```
#c)  Check what data types the pandas has inferred and if any of the featu
df.dtypes
```

Out[5]:

```
instant         int64
dteday         object
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
casual          int64
registered      int64
cnt             int64
dtype: object
```

the features that need data conversions are season, holiday, workingday, weekday, weathersit, mnth and yr as they are more suitable to be placed in category like for example season can be categorize into winter, summer, spring and autumn.

In [6]:

```python
#d)
nd = df.rename(columns = {'instant':'rec_id',
                                'dteday':'datetime',
                                'holiday':'is_holiday',
                                'workingday':'is_workingday',
                                'weathersit':'weather_condition',
                                'hum':'humidity',
                                'mnth':'month',
                                'cnt':'total_count',
                                'yr':'year'})
```

In [7]:

```python
#d Then, display the new top 10 rows to show the new attribute names.
nd.head(10)
```

Out[7]:

	rec_id	datetime	season	year	month	is_holiday	weekday	is_workingday	weather_conditio
0	1	2011-01-01	1	0	1	0	6	0	
1	2	2011-01-02	1	0	1	0	0	0	
2	3	2011-01-03	1	0	1	0	1	1	
3	4	2011-01-04	1	0	1	0	2	1	
4	5	2011-01-05	1	0	1	0	3	1	
5	6	2011-01-06	1	0	1	0	4	1	
6	7	2011-01-07	1	0	1	0	5	1	
7	8	2011-01-08	1	0	1	0	6	0	
8	9	2011-01-09	1	0	1	0	0	0	
9	10	2011-01-10	1	0	1	0	1	1	

In [8]:

```
#e)
```

Out[8]:

```
0      1
1      1
2      1
3      1
4      1
      ..
726    1
727    1
728    1
729    1
730    1
Name: season, Length: 731, dtype: category
Categories (4, int64): [1, 2, 3, 4]
```

In [9]:

```
nd['is_holiday'].astype('category')
```

Out[9]:

```
0      0
1      0
2      0
3      0
4      0
      ..
726    0
727    0
728    0
729    0
730    0
Name: is_holiday, Length: 731, dtype: category
Categories (2, int64): [0, 1]
```

In [10]:

```
nd['is_workingday'].astype('category')
```

Out[10]:

```
0      0
1      0
2      1
3      1
4      1
      ..
726    1
727    1
728    0
729    0
730    1
Name: is_workingday, Length: 731, dtype: category
Categories (2, int64): [0, 1]
```

In [11]:

```
nd['weekday'].astype('category')
```

Out[11]:

```
0      6
1      0
2      1
3      2
4      3
      ..
726    4
727    5
728    6
729    0
730    1
Name: weekday, Length: 731, dtype: category
Categories (7, int64): [0, 1, 2, 3, 4, 5, 6]
```

In [12]:

```
nd['weather_condition'].astype('category')
```

Out[12]:

```
0      2
1      2
2      1
3      1
4      1
      ..
726    2
727    2
728    2
729    1
730    2
Name: weather_condition, Length: 731, dtype: category
Categories (3, int64): [1, 2, 3]
```

In [13]:

```
nd['month'].astype('category')
```

Out[13]:

```
0       1
1       1
2       1
3       1
4       1
       ..
726    12
727    12
728    12
729    12
730    12
Name: month, Length: 731, dtype: category
Categories (12, int64): [1, 2, 3, 4, ..., 9, 10, 11, 12]
```

In [14]:

```python
nd['year'].astype('category')
```

Out[14]:

```
0      0
1      0
2      0
3      0
4      0
      ..
726    1
727    1
728    1
729    1
730    1
Name: year, Length: 731, dtype: category
Categories (2, int64): [0, 1]
```

In [15]:

```python
#f)
nd.describe(include='all')
```

Out[15]:

	rec_id	datetime	season	year	month	is_holiday	weekday	is_w
count	731.000000	731	731.000000	731.000000	731.000000	731.000000	731.000000	7
unique	NaN	731	NaN	NaN	NaN	NaN	NaN	
top	NaN	2011-01-01	NaN	NaN	NaN	NaN	NaN	
freq	NaN	1	NaN	NaN	NaN	NaN	NaN	
mean	366.000000	NaN	2.496580	0.500684	6.519836	0.028728	2.997264	
std	211.165812	NaN	1.110807	0.500342	3.451913	0.167155	2.004787	
min	1.000000	NaN	1.000000	0.000000	1.000000	0.000000	0.000000	
25%	183.500000	NaN	2.000000	0.000000	4.000000	0.000000	1.000000	
50%	366.000000	NaN	3.000000	1.000000	7.000000	0.000000	3.000000	
75%	548.500000	NaN	3.000000	1.000000	10.000000	0.000000	5.000000	
max	731.000000	NaN	4.000000	1.000000	12.000000	1.000000	6.000000	

In [16]:

```
#g)
nd.isnull
```

Out[16]:

```
<bound method DataFrame.isnull of      rec_id    datetime  season  year  mont
h  is_holiday  weekday  \
0         1  2011-01-01       1     0      1           0        6
1         2  2011-01-02       1     0      1           0        0
2         3  2011-01-03       1     0      1           0        1
3         4  2011-01-04       1     0      1           0        2
4         5  2011-01-05       1     0      1           0        3
..      ...         ...     ...   ...    ...         ...      ...
726     727  2012-12-27       1     1     12           0        4
727     728  2012-12-28       1     1     12           0        5
728     729  2012-12-29       1     1     12           0        6
729     730  2012-12-30       1     1     12           0        0
730     731  2012-12-31       1     1     12           0        1

     is_workingday  weather_condition      temp     atemp  humidity  \
0                0                  2  0.344167  0.363625  0.805833
1                0                  2  0.363478  0.353739  0.696087
2                1                  1  0.196364  0.189405  0.437273
3                1                  1  0.200000  0.212122  0.590435
4                1                  1  0.226957  0.229270  0.436957
..             ...                ...       ...       ...       ...
726              1                  2  0.254167  0.226642  0.652917
727              1                  2  0.253333  0.255046  0.590000
728              0                  2  0.253333  0.242400  0.752917
729              0                  1  0.255833  0.231700  0.483333
730              1                  2  0.215833  0.223487  0.577500

     windspeed  casual  registered  total_count
0     0.160446     331         654          985
1     0.248539     131         670          801
2     0.248309     120        1229         1349
3     0.160296     108        1454         1562
4     0.186900      82        1518         1600
..         ...     ...         ...          ...
726   0.350133     247        1867         2114
727   0.155471     644        2451         3095
728   0.124383     159        1182         1341
729   0.350754     364        1432         1796
730   0.154846     439        2290         2729

[731 rows x 16 columns]>
```

In [17]:

```python
#h)
nd.drop(['rec_id','datetime','casual','registered'],axis=1)
```

Out[17]:

	season	year	month	is_holiday	weekday	is_workingday	weather_condition	temp	
0	1	0	1	0	6	0	2	0.344167	0.3
1	1	0	1	0	0	0	2	0.363478	0.3
2	1	0	1	0	1	1	1	0.196364	0.1
3	1	0	1	0	2	1	1	0.200000	0.2
4	1	0	1	0	3	1	1	0.226957	0.2
...	...	...	...	...	...	...	...	...	
726	1	1	12	0	4	1	2	0.254167	0.2
727	1	1	12	0	5	1	2	0.253333	0.2
728	1	1	12	0	6	0	2	0.253333	0.2
729	1	1	12	0	0	0	1	0.255833	0.2
730	1	1	12	0	1	1	2	0.215833	0.2

731 rows × 12 columns

In [18]:

```python
#h)
nd.head(10)
```

Out[18]:

	rec_id	datetime	season	year	month	is_holiday	weekday	is_workingday	weather_conditio
0	1	2011-01-01	1	0	1	0	6	0	
1	2	2011-01-02	1	0	1	0	0	0	
2	3	2011-01-03	1	0	1	0	1	1	
3	4	2011-01-04	1	0	1	0	2	1	
4	5	2011-01-05	1	0	1	0	3	1	
5	6	2011-01-06	1	0	1	0	4	1	
6	7	2011-01-07	1	0	1	0	5	1	
7	8	2011-01-08	1	0	1	0	6	0	
8	9	2011-01-09	1	0	1	0	0	0	
9	10	2011-01-10	1	0	1	0	1	1	

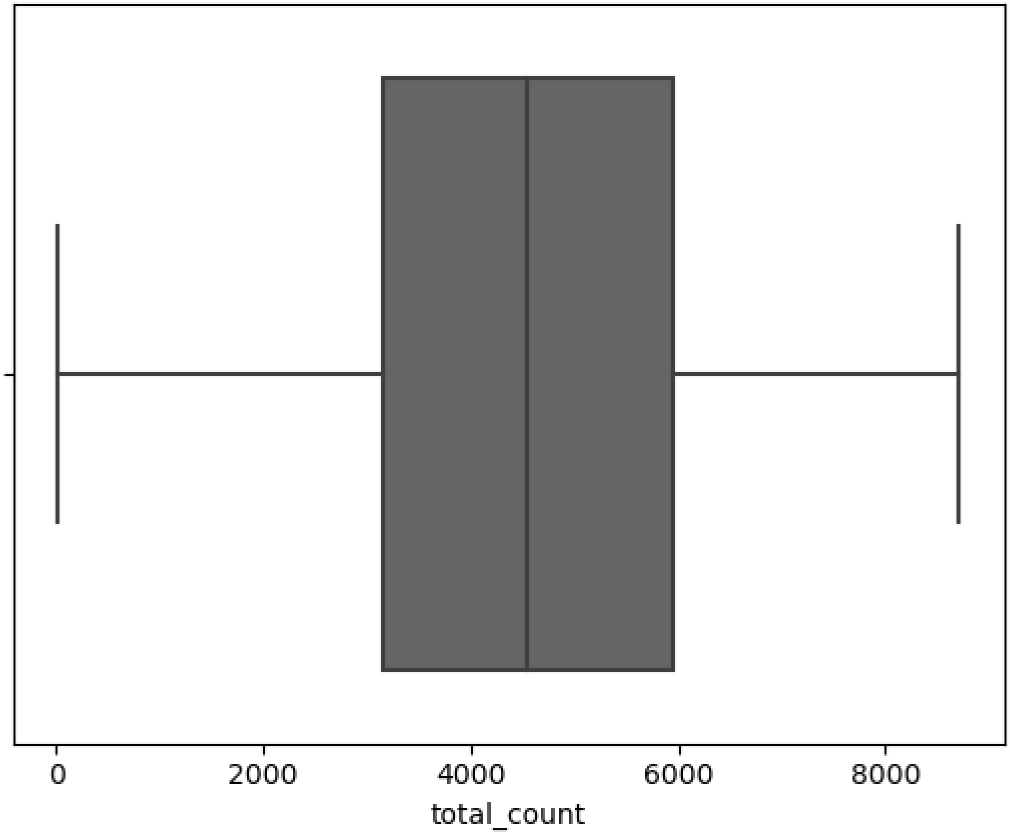
In [19]:

```
#i)
sns.boxplot(new_df['total_count'])
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, th
e 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[19]:

```
<AxesSubplot:xlabel='total_count'>
```



No outliers for total count.

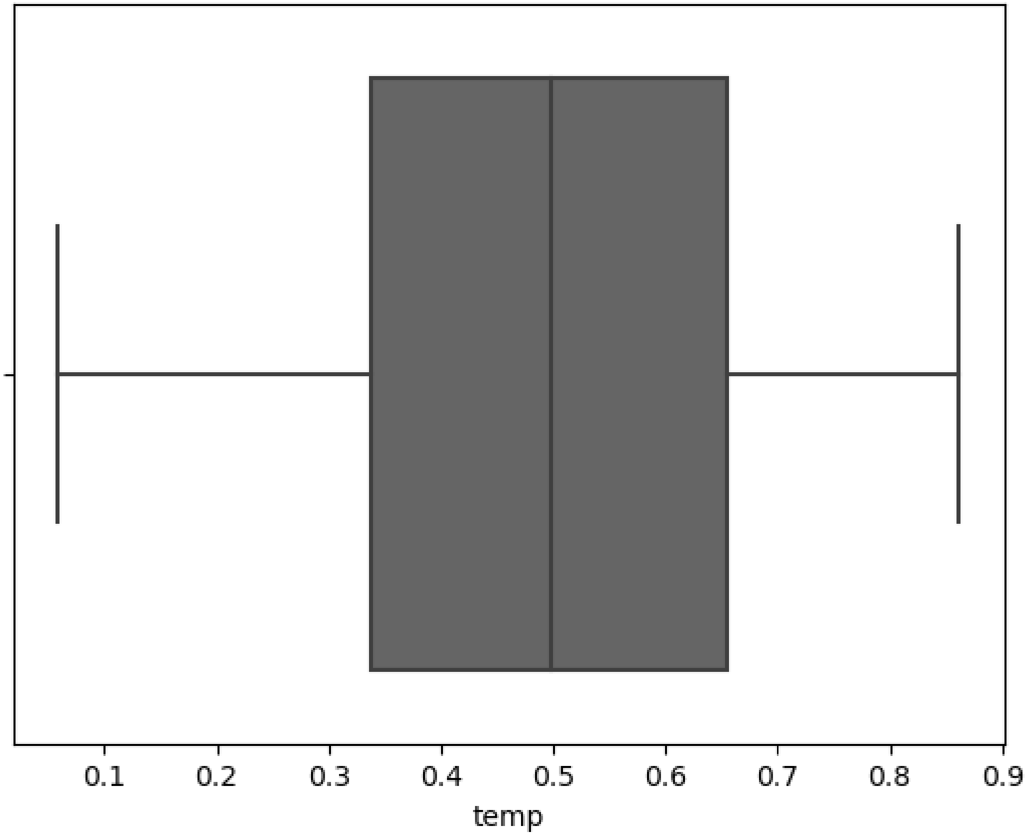
In [41]:

```
sns.boxplot(new_df['temp'])
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, th
e 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[41]:

```
<AxesSubplot:xlabel='temp'>
```



```
##No outliers for temperature
```

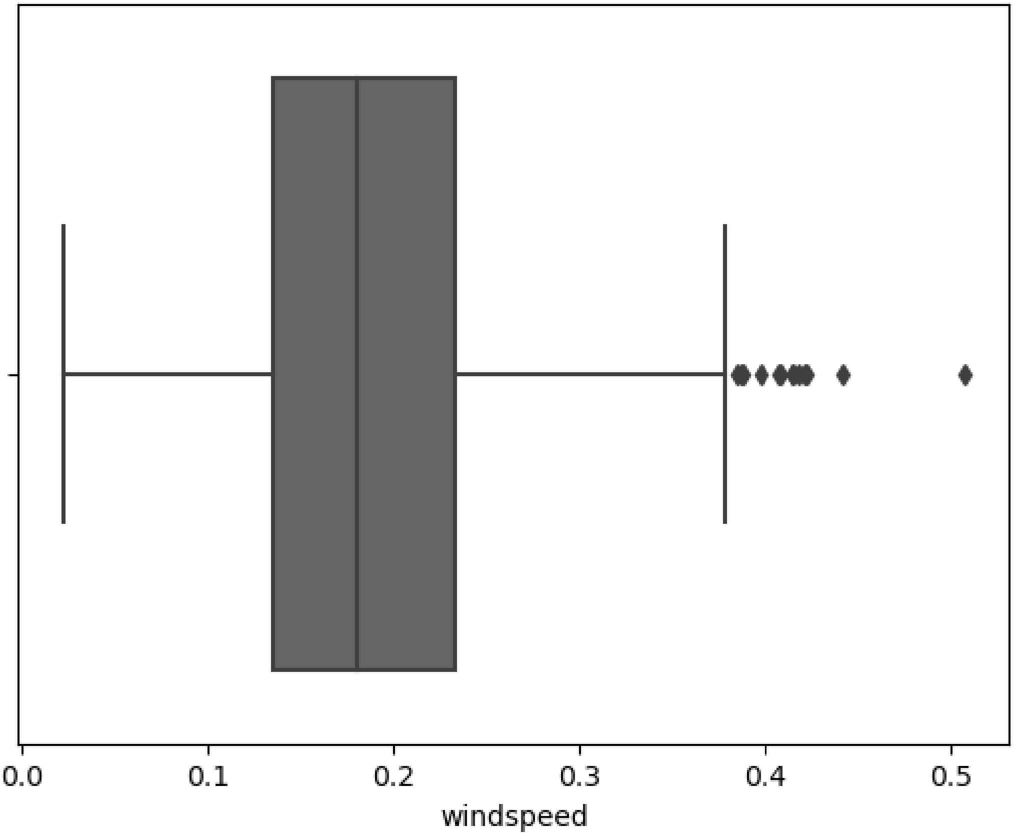
In [42]:

```
sns.boxplot(new_df['windspeed'])
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, th
e 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[42]:

```
<AxesSubplot:xlabel='windspeed'>
```

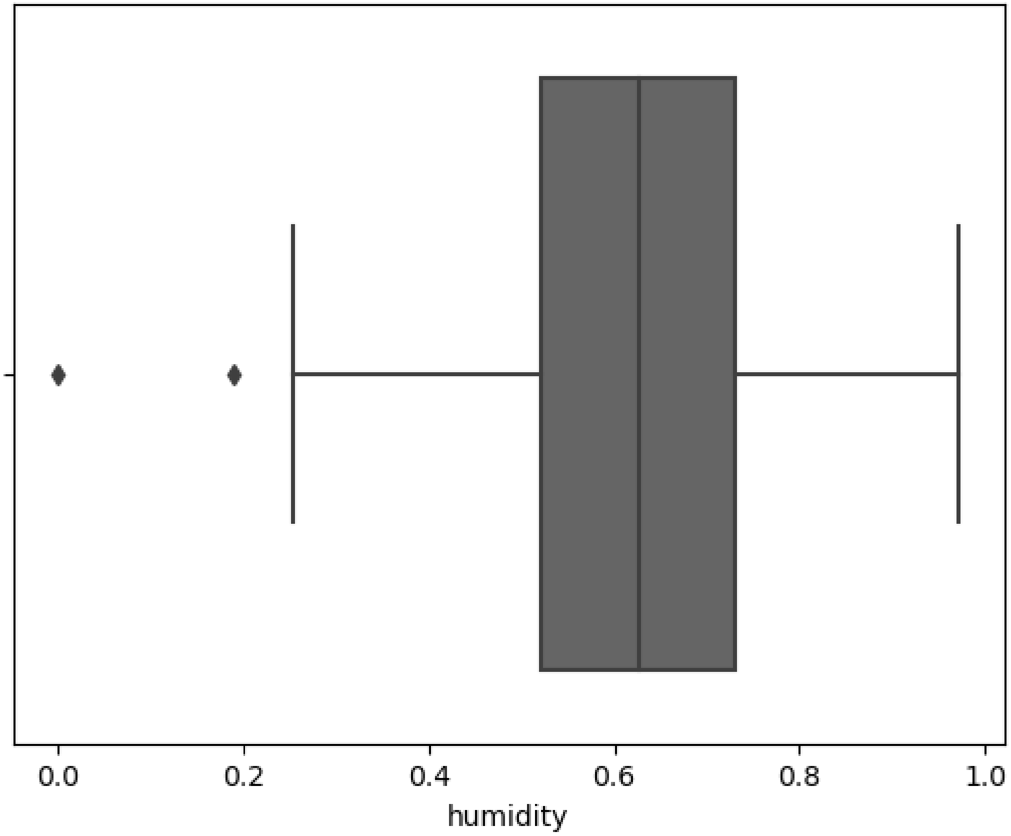


```
##Outliers exist for windspeed。
```

```
In [43]: sns.boxplot(new_df['humidity'])
```

```
C:\Users\USER\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

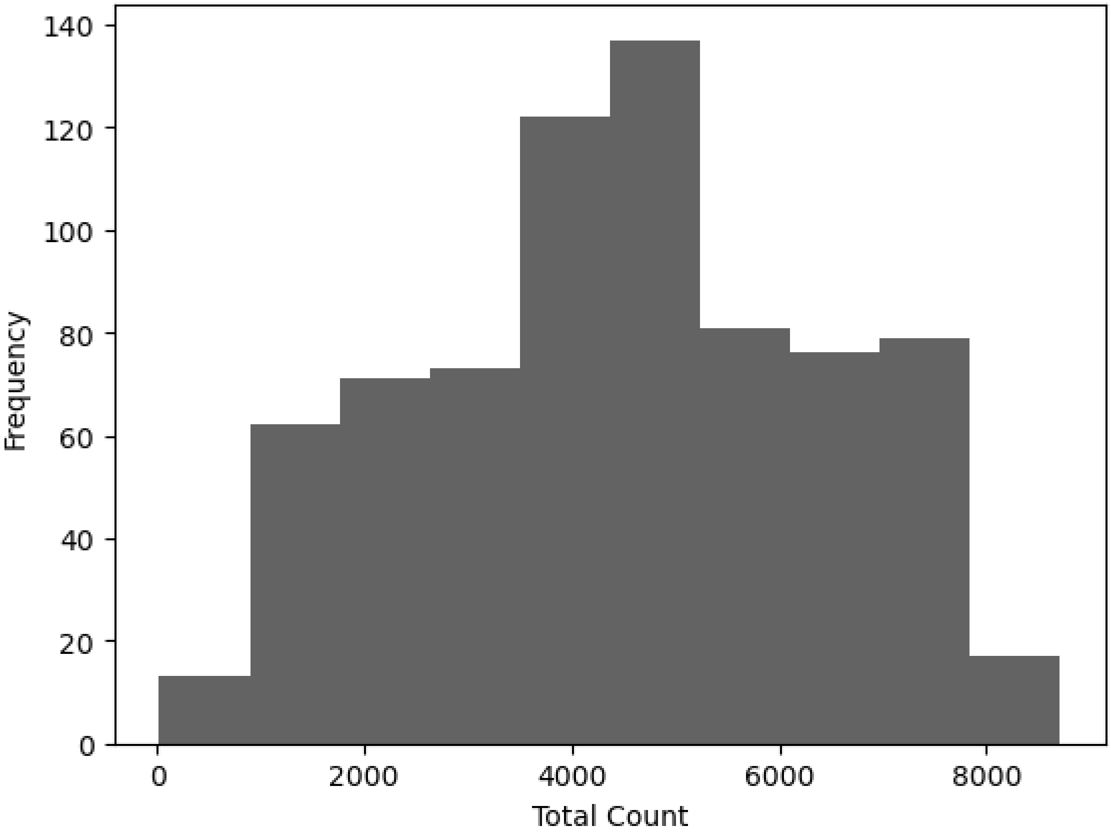
```
Out[43]: <AxesSubplot:xlabel='humidity'>
```



2 outliers for humidity

In [30]:

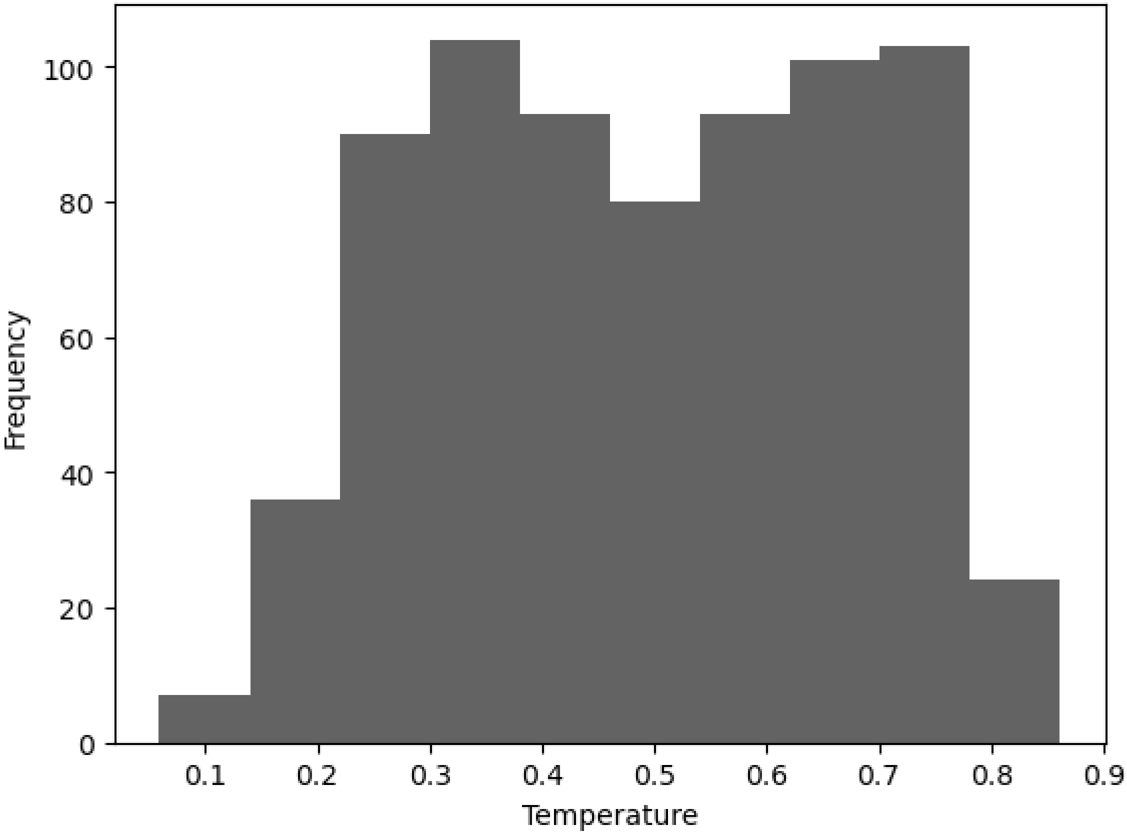
```
#j)
plt.hist(new_df['total_count'])
plt.xlabel('Total Count')
plt.ylabel('Frequency')
plt.show()
```



```
#The skewness of for total count distribution is left skewness.
```

In [31]:

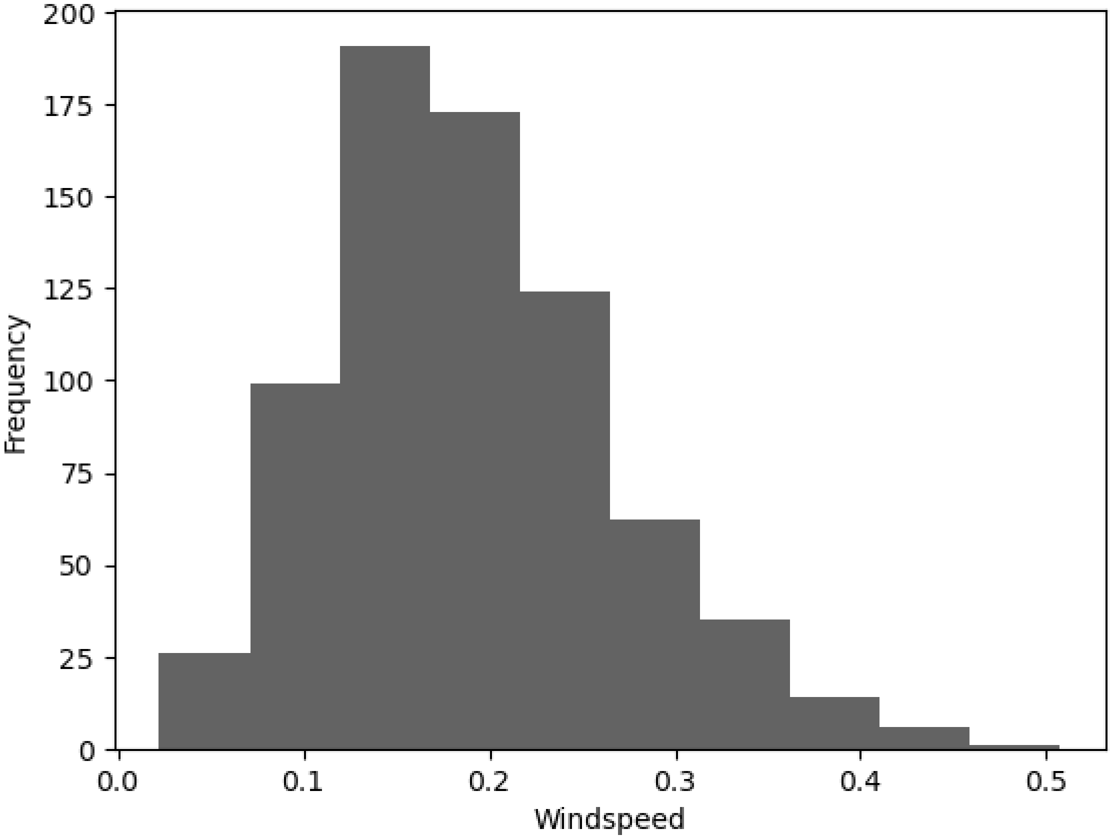
```
plt.hist(new_df['temp'])
plt.xlabel('Temperature')
plt.ylabel('Frequency')
plt.show()
```



```
#The distribution for temperature is bimodal.
```

In [32]:

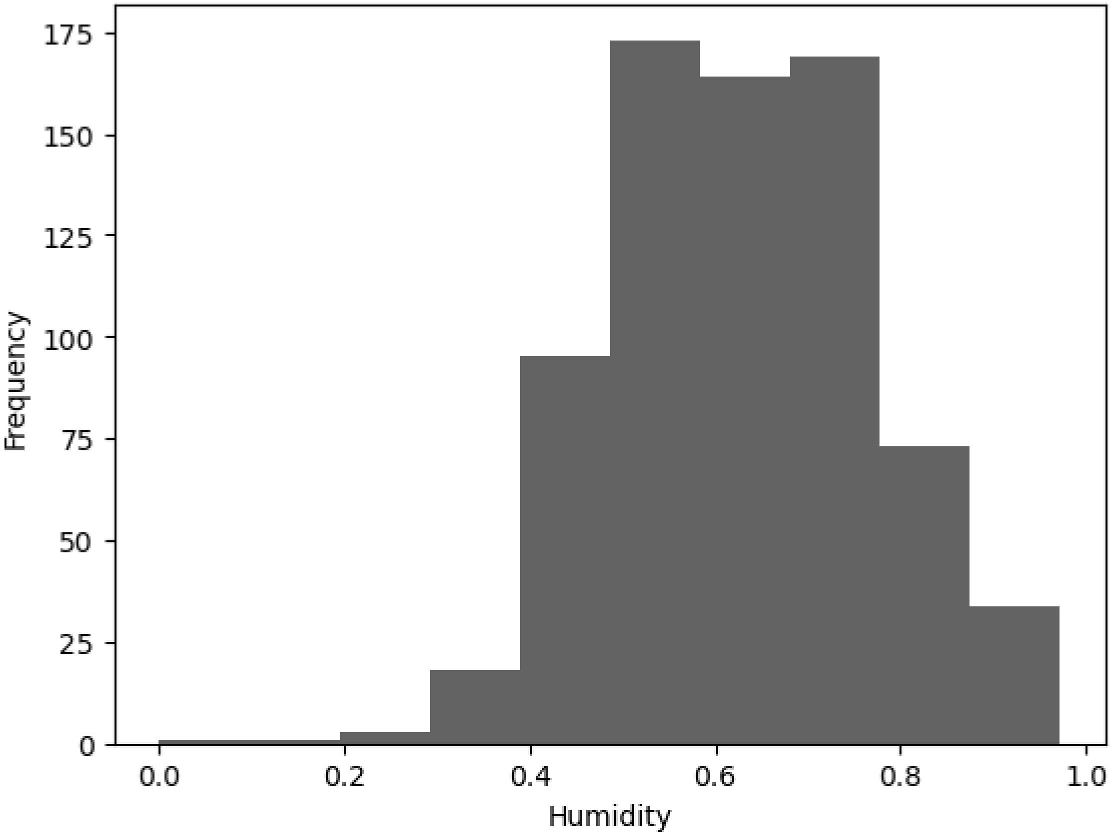
```
plt.hist(new_df['windspeed'])
plt.xlabel('Windspeed')
plt.ylabel('Frequency')
plt.show()
```



```
#The skewness for windspeed is positively skewed.
```

In [35]:

```
plt.hist(new_df['humidity'])
plt.xlabel('Humidity')
plt.ylabel('Frequency')
plt.show()
```



```
#The skewness for windspeed is bimodal.
```